Numerická matematika pre informatika, Riešené príklady v programe MATHEMATICA

Book · May 2011

CITATIONS
0

READS
1,824

2 authors, including:

Roman Ďurikovič
Comenius University Bratislava
110 PUBLICATIONS **480** CITATIONS

SEE PROFILE

NUMERICKÁ MATEMATIKA PRE INFORMATIKA

NUMERICKÁ MATEMATIKA PRE INFORMATIKA

Riešené príklady v programe Mathematica

Roman Ďurikovič
Fakulta matematiky fyziky a informatiky, Univerzita Komenského,
Mlynská dolina, 842 48 Bratislava, Slovenská republika

Vladimír Ďurikovič
Fakulta prírodných vied, Univerzita sv. Cyrila a Metoda,
Nám. J. Herdu 2, 917 01 Trnava, Slovenská republika

Univerzita sv. Cyrila a Metoda v Trnave, Trnava 2011

Autori: doc. RNDr. Roman Ďurikovič, CSc., mim. prof.
doc. RNDr. Vladimír Ďurikovič, PhD., mim. prof.

Recenzenti: prof. RNDr. Rudolf Kodnár, DrSc.
Žilinská univerzita v Žiline
doc. RNDr. Angela Handlovičová, CSc.
Slovenská technická univerzita v Bratislave

Schválené edičnou radou Univerzity sv. Cyrila a Metoda v Trnave a vedením Fakulty prírodných vied Univerzity sv. Cyrila a Metoda v Trnave ako učebnica pre študentov vysokých škôl.

Prešlo jazykovou úpravou.

ISBN 978-80-8105-271-2

Knihu venujeme svojim manželkám Andreike a Márii za vytvorenie teplého zázemia a dobrých podmienok pri písaní tejto monografie.

Roman venuje aj svojim rodičom z vďaky za otvorenie cesty k poznaniu.

Obsah

PREDHOVOR

Hlavným cieľom predkladanej monografie je prezentovať rôzne metódy numerickej matematiky používané pri riešení praktických úloh využitím výpočtovej techniky, špeciálne programu *Mathematica* [15].

Riešenie úloh z praxe sa obyčajne realizuje v nasledujúcich fázach:

- Popis praktického problému, čo je úlohou fyzikálneho, technického, prírodovedeckého, ekonomického alebo dokonca humanitného spracovania.
- Pretlmočenie tohto popisu do spojitého alebo diskrétneho matematického modelu a skúmanie existencie jeho riešenia, jednoznačnosti, kvalitatívnych a kvantitatívnych vlastnosti, bez ktorých numerické a počítačové spracovávanie riešenia by sa mohlo stať iluzórnym.
- Nájdenie analytickej alebo aproximačnej metódy riešenia matematického modelu.
- Počítačové spracovanie navrhnutej metódy, ktoré v konečnom dôsledku dáva číselné vyjadrenie hľadaného riešenia s požadovanou presnosťou.

Druhá, tzv. teoretická fáza, zabezpečuje, že približné spracovanie riešenia má reálny základ. Hľadanie aproximačných metód riešenia je doménou nume-

rickej matematiky, najmä vtedy, keď nemáme možnosť získať jeho analytické vyjadrenie. Potrebná presnosť aproximácie číselného vyjadrenia sa dosahuje množstvom cyklicky sa opakujúcich operácií. Táto časť technicky náročného a zdĺhavého výpočtu sa v súčasnosti zvládla kombináciou numerických metód a počítačovej techniky.

Preto je prirodzené v našej monografii, že ďalší z jej cieľov je prepojiť tieto dve oblasti tak, aby používatelia numerických metód aj študenti informatiky i matematiky zvládli primerané množstvo kvalitných vedomostí z vyššej matematiky, ktoré vyžaduje numerická analýza a algoritmické myslenie nevyhnutné pre počítačové spracovanie numerických modelov. Literatúra s takouto symbiózou je netradičnou a vhodnou študijnou i praktickou pomôckou aj pre študentov PhD. i odborníkov z praxe. Ďalším cieľom predkladanej monografie je podať dostatočne široký a moderný výklad numerických metód pre riešenie rôznych druhov praktických úloh so stanovením odhadu výpočtových chýb a konvergencie použitých algoritmov riešenia.

Monografia je rozdelená do dvanástich kapitol, v ktorých sa postupne prezentujú témy:

- Programovacie jazyky, presnosť počítača a chyby numerickej analýzy.
- Riešenie systémov lineárnych algebraických rovníc, metódy riedkych matíc, inverzná matica a vlastné čísla matíc a riešenie nelineárnych algebraických rovníc.
- Interpolácia a aproximácia funkcie, kubické splajny a numerické derivovanie a integrovanie.
- Numerické riešenie začiatočných úloh pre obyčajné rovnice prvého a vyšších rádov.
- Numerické riešenie okrajových a zmiešaných úloh pre niektoré parciálne diferenciálne rovnice.

Každá kapitola je rozdelená na paragrafy, ktoré sú označené dvojčíslim. Prvá číslica označuje číslo kapitoly a druhá číselné poradie paragrafu v kapitole. Články v paragrafoch sú označované trojčíslím, z ktorého prvé dvojčíslie označuje číslo paragrafu. Napríklad článok 6.4.1 Jakobiho diagonalizácia označuje prvý článok z paragrafu 6.4 zo šiestej kapitoly. Základné algoritmy sú uvedené v programe *Mathematica* v rámčekoch. V každej kapitole sa uvádzajú riešené príklady v programe Mathematica a sú číslované v celej knihe priebežne. Na konci každej kapitoly prezentujeme dostatočné množstvo praktických a ilustratívnych úloh s riešeniami.

Kniha je napísaná po viacročných skúsenostiach autormi, ktorí prednášajú a vedú semináre na magisterskom i doktorandskom štúdiu na univerzitách z prezentovanej oblasti numerickej a výpočtovej matematiky i matematickej

analýzy. Je zaujímavá aj tým, že na relatívne malom priestore spracováva zrozumiteľným spôsobom potrebné kvantum numerických metód a algoritmov. Autori pôsobia pri výučbe rôznych druhov študijných programov a aktívne sa zapájajú do výskumnej a vedeckej práce na univerzitách i v praxi. Takto chcú prispieť ku skvalitneniu vzdelanosti a praktického využívania numerických metód v spojení s počítačovou technikou.

Pri štúdiu tejto monografie je potrebné mať základné vedomosti z algebry a maticového počtu, matematickej analýzy, diferenciálnych rovníc, z programovania a programu *Mathematica*.

Na záver nám pripadá veľmi milá úloha poďakovať sa recenzentom: prof. RNDr. Rudolfovi Kodnárovi, Dr.Sc. zo Žilinskej univerzity v Žiline a doc. RNDr. Angele Handlovičovej, CSc. zo Slovenskej technickej univerzity v Bratislave za dôsledné preštudovanie textu, cenné pripomienky a návrhy, ktoré významne prispeli k jeho skvalitneniu.

Roman Ďurikovič a Vladimír Ďurikovič

Bratislava, Slovenská republika
September, 2011

Kapitola 1

NUMERICKÁ MATEMATIKA

Numerická analýza je cesta k riešeniu zložitejších matematických problémov na počítači, je to technika často používaná vedcami a inžiniermi na riešenie ich problémov. Hlavnou výhodou numerickej analýzy je, že existujú numerické odpovede aj vtedy keď problém nemá analytické riešenie. Napríklad integrál

$$\int_0^{\pi} \sqrt{1 + cos^2(t)}\, dt,$$

ktorý určuje dĺžku jedného oblúka funkcie $y = sin(x)$, nemá riešenie v uzavretom tvare.

Numerická analýza umožňuje počítať veľkosť plochy pod týmto oblúkom štandardnými metódami, ktoré sú aplikované v podstate na ľubovoľný integrovateľný integrand; nepotrebujeme robiť špeciálne substitúcie alebo integrovať po častiach. Potrebujeme iba matematické operácie ako súčet, rozdiel, násobenie, delenie a navyše musíme vedieť robiť porovnávanie. Pretože sú tieto jednoduché operácie jediné, ktoré môže počítač vykonávať, tak počítač

a numerická analýza tvoria vhodnú kombináciu pre riešenie aj komplikovaných úloh.

Je potrebné si uvedomiť, že numericko-analytické riešenie je vždy numerické. Analytické metódy dávajú zvyčajne riešenia v tvare matematických výrazov, ktoré môžu byť následne vypočítané v požadovaných bodoch. Výhoda analytického riešenia je aj v tom, že vlastnosti funkcií a ich priebeh sú často zrejmé z analytického tvaru; túto výhodu žiaľ postrádame v prípade čisto numerických riešení. Avšak numerické výsledky môžu byť zobrazené a potom uvidíme aj niektoré vlastnosti riešenia.

Ďalšia dôležitá prednosť numerických riešení je, že hoci výsledok numerického postupu je iba aproximácia, môžeme ho vypočítať tak presne ako potrebujeme, ak nám to dovolí presnosť počítača. V počítači je presnosť výpočtu ohraničená vďaka tomu ako počítač spracováva aritmetické operácie. K dosiahnutiu veľkej presnosti musíme uskutočniť veľké množstvo aritmetických operácií, ale počítače ich vykonávajú rýchlo a preto v tom nie je podstatný problém. Analýza chýb generovaných počítačom a iných zdrojov chýb v numerických metódach je veľmi kritická časť metód numerickej analýzy.

Ďalej uvedieme niektoré numerické algoritmy na hľadanie približného riešenia menovaných úloh:

- Reprezentácia reálneho čísla a chyby pri počítaní s nimi.
- Nájdenie nulovej hodnoty funkcie.
- Úlohy lineárnej algebry, vektory, matice a riešenie lineárnych rovníc.
- Hľadanie inverznej matice a vlastných hodnôt matice.
- Metódy pre riedke matice.
- Interpolácia funkcii.
- Interpolácia kubických splajnov a použitie rozptýlených dát.
- Numerická diferencia a derivácia.
- Numerická integrácia.
- Numerické algoritmy riešenia úloh obyčajnej diferenciálnej rovnice.
- Numerické algoritmy riešenia úloh parciálnej diferenciálnej rovnice.

1.1 PROGRAMOVACIE JAZYKY

Pôvodné numerické výpočty boli robené v jazykoch FORTRAN a C. V posledných rokoch bolo využívane objektovo orientované programovanie (OOP)

pre jeho spoľahlivosť, ľahké debuggovanie a znovu použiteľnosť kódu. V dnešnej dobe najviac používané objektovo orientované programovacie jazyky sú C# a Java.

1.1.1 Dátové typy

Dostupné dátové typy závisia od architektúry počítača a od samotného kompilátora. Každý typ pozostáva z určitého počtu bitov a tým je daný interval (Minimum, Maximum), v ktorom môže daný typ reprezentovať reálne čísla.

Základné typy	**Veľkosť**	**Minimum**	**Maximum**
boolean	-	-	-
char	16-bit	0	$2^{16} - 1$
byte	8-bit	-128	+127
short	16-bit	-2^{15}	$2^{15} - 1$
int	32-bit	-2^{31}	$2^{31} - 1$
long	64-bit	-2^{63}	$2^{63} - 1$
float	32-bit	$\pm$ 1,1* 10^{-38}	$\pm$ 3,4* 10^{38}
double	64-bit	$\pm$ 4,9* 10^{-324}	$\pm$ 1,1* 10^{308}

Kapitola 2

PRESNOSŤ POČÍTAČA

V tejto kapitole sa budeme zaoberať reprezentáciou reálnych čísel v počítačoch. Množina reálnych čísiel, ktoré vieme presne reprezentovať je ohraničená najmenším a najväčším číslom a má konečný počet prvkov. Ďalej uvedieme numerickú presnosť, ktorá je daná vzdialenosťou prvkov od seba. Každý programovací jazyk na počítači môže mať iné konštanty reprezentácie, preto popisujeme aj algoritmy ako ich zistiť.

2.1 REPREZENTÁCIA REÁLNYCH ČÍSEL

Na dnešných počítačoch je reálne číslo reprezentované ako

$$m \cdot r^e$$

kde základ r je väčšinou rovný 2. Na niektorých počítačoch môže byť rovný 10 alebo 16. Takže každé reálne číslo je reprezentované dvoma číslami mantisou m a exponentom e. Toto vyjadrenie je známe ľuďom, ktorí počítajú buď s

veľmi malými, alebo veľmi veľkými číslami. Samozrejme prirodzený základ pre ľudí je 10. Napríklad priemerná vzdialenosť Zeme a Slnka vyjadrená v kilometroch sa zapíše $1,4959787 \cdot 10^8$.

V prípade, že je základ rovný 2, tak číslo 18446744073709551616 je reprezentované ako $1 \cdot 2^{64}$. Celkom krátka reprezentácia v porovnaní s vyjadrením v desiatkovej sústave. IEEE štandard reálneho čísla ([7] str. 45–51) používa 24-bitovú mantisu (čo je 8 číslic v zápise v desiatkovej sústave) pri obyčajnom type float; a 54-bitovú mantisu (čo je 15 číslic v zápise v desiatkovej sústave) pre typ double. Dôležitou vecou pri reprezentácii reálnych čísel je relatívna presnosť tejto reprezentácie, teda pomer medzi vyjadrením čísla a samotným číslom by mal byť rovnaký pre všetky čísla okrem nuly.

2.2 NÁJDENIE NUMERICKÉHO VYJADRENIA V POČÍTAČI

Niektoré numerické algoritmy pracujú dovtedy pokiaľ odhadovaná presnosť výsledku nie je menšia ako nejaká zadaná hodnota, ktorú voláme *požadovaná presnosť*.

Aplikácia môže byť spúšťaná na rôznom hardvéry preto je najlepšie určiť požadovanú presnosť počas výpočtu. Uveďme si teraz parametre reprezentácie reálnych čísel na konkrétnom počítači potrebných pre numerické výpočty:

Základ reprezentácie reálnych čísel je parameter r.

Presnosť počítača je najväčšie kladné číslo, ktoré keď prirátame k 1 bude pre počítač výsledok stále 1.

Záporná presnosť počítača je najväčšie kladné číslo, ktoré keď odrátame od 1 bude pre počítač výsledok stále 1.

Najmenšie číslo je najmenšie kladné číslo, ktoré dokážeme reprezentovať v počítači.

Najväčšie číslo je najväčšie kladné číslo, ktoré dokážeme reprezentovať v počítači.

Bežná numerická presnosť je presnosť, ktorú očakávame pre bežný numerický výpočet.

Malé číslo je číslo, ktoré môžeme pripočítať k nejakej hodnote bez zmeny výsledku.

2.2.1 Výpočet základu

Výpočet základu sa robí v dvoch krokoch. V prvom sa počíta číslo ekvivalentné s počítačovou presnosťou predpokladajúc, že základ je rovný 2. Potom,

pokračujeme pridávaním 1 pokiaľ sa výsledok nezmení. Počet pridaných 1 je potom základom.

2.2.2 Výpočet počítačovej presnosti

Počítačová presnosť je počítaná nájdením najväčšieho celého čísla n takého, že

$$(1 + r^{-n}) - 1 \neq 0.$$

Číslo n sa hľadá cyklom cez celé čísla. Číslo $\varepsilon_+ = r^{-(n+1)}$ je počítačová presnosť.

2.2.3 Záporná počítačová presnosť

Záporná počítačová presnosť sa počíta nájdením najväčšieho celého čísla n takého, že platí

$$(1 - r^{-n}) - 1 \neq 0.$$

Výpočet je analogický ako pri počítačovej presnosti. Číslo $\varepsilon_- = r^{-(n+1)}$ je záporná počítačová presnosť. Počítačová presnosť je väčšia ako záporná počítačová presnosť.

2.2.4 Najmenšie a najväčšie čísla

Na výpočet najmenších a najväčších čísel musíme v prvom kroku vypočítať číslo, ktorého mantisa je plná. Také číslo dostaneme výpočtom výrazu

$$f = 1 - r \cdot \varepsilon,$$

kde ε je bežná numerická presnosť, definovaná v odseku 2.2.5. Najmenšie číslo sa potom vyráta opakovaným delením tohto čísla f základom r, pokiaľ výsledná hodnota nepodtečie. Číslo podtečie, ak sa z malého čísla stane po operácii kladné číslo. Poslednú hodnotu dosiahnutú pred podtečením považujeme za najmenšie číslo. Podobne najväčšiu hodnotu počítame opakovaným násobením hodnoty f hodnotou základu pokiaľ nenastane pretečenie. Pretečenie nastane ak sa z veľkého čísla stane po operácii záporné číslo. Posledná hodnota dosiahnutá pred pretečením je najväčšie číslo.

Príklad 1. *Zistite presnosť počítača, najväčšie a najmenšie číslo v programe Mathematica [15].*

Riešenie. Presnosť počítača zistíme pomocou konštanty `$MachineEpsilon`, najväčšie číslo konštantou `$MaxMachineNumber` a najmenšie číslo konštantou `$MinMachineNumber`:

```
{ $MachineEpsilon, $MaxMachineNumber, $MinMachineNumber }
```

$\{2.22045 * 10^{-16}, 1.79769 * 10^{308}, 2.22507 * 10^{-308}\}$

Presnosť programu *Mathematica* je $2.22045*10^{-16}$, najväčšie číslo je $1.79769*10^{308}$, najmenšie číslo $2.22507 * 10^{-308}$. □

2.2.5 Bežná numerická presnosť

Bežná numerická presnosť, ktorú môžeme požadovať pre bežný numerický výpočet je definovaná ako odmocnina z počítačovej presnosti

$$\varepsilon = \sqrt{\varepsilon_+} \ .$$

Relatívna presnosť dvoch čísel a, b je daná váženým rozdielom

$$\frac{|a-b|}{max(|a|,|b|)}.$$

Potom môžeme napríklad uvažovať, že dve čísla a, b sú si rovné ak relatívny rozdiel medzi nimi je menší než bežná numerická presnosť ε

$$\frac{|a-b|}{max(|a|,|b|)} < \varepsilon.$$

2.2.6 Malé číslo

Malé číslo je hodnota, ktorá môže byť pripočítaná k nejakému číslu bez toho aby to zmenilo výsledok výpočtu. Hodnotu malého čísla definujeme ako odmocninu najmenšieho čísla, ktoré dokážeme reprezentovať v počítači. Malé číslo často využívame na dodefinovanie hodnoty výrazu, ktorý nie je matematicky definovaný. Napríklad, výraz typu $\frac{0}{0}$ nie je definovaný. V niektorých prípadoch môžeme definovať hodnotu takéhoto výrazu pomocou limity. Napríklad výraz $\frac{\sin x}{x}$ je rovný 1 pre $x = 0$. V takýchto prípadoch sa môžeme vyhnúť deleniu nulou pričítaním malého čísla k čitateľu a menovateľu a môžeme dostať správnu hodnotu.

2.3 ÚLOHY

1 Vypočítajte počítačovú presnosť Vášho počítača použitím krátkeho programu.

2 Určite najmenšie a najväčšie číslo na Vašom počítači.

3 Naprogramujte program na výpočet bežnej numerickej presnosti a určite ju pre Váš počítač.

4 Zistite malé číslo Vášho počítača.

Kapitola 3

CHYBY NUMERICKEJ ANALÝZY A VÝPOČET HODNÔT FUNKCIE

Žiaľ, bez chyby vieme počítačom reprezentovať iba reálne číslo vyjadrené konečným počtom číslic. To vyžaduje, aby vo všeobecnosti reálne čísla pri používaní počítača boli aproximované ich konečnými hodnotami, čo spôsobuje chybu pri výpočte. V tejto kapitole budeme túto chybu analyzovať a ukážeme si ako sa chyba šíri v počítačových operáciách. Popíšeme chybu zapríčiňujúcu stratu platnosti výsledku a ukážeme si niektoré postupy ako spomaliť stratu platnosti pri výpočte hodnôt funkcie, usporiadaním členov, rozvojom do Taylorovho radu ([2], str.129–185) alebo plynomickou aproximáciou.

3.1 ANALÝZA CHÝB

Pri praktických výpočtoch si musíme byť vedomí toho, že v numerickej analýze vypočítaný výsledok nie je zhodný s analytickým výsledkom. Presnosť numerického výsledku môže byť porušená viacerými spôsobmi.

Definícia 1. *Relatívna chyba. Predpokladajme, že $\tilde{p}$ je aproximáciou $p \neq 0$. Chyba je potom*

$$\varepsilon_p = \tilde{p} - p$$

a relatívna chyba je

$$R_p = (\tilde{p} - p)/p,$$

keď $p \neq 0$.

Definícia 2. *Platné číslice. Číslo $\tilde{p}$ aproximuje číslo p na d platných číslic, ak d je najväčšie kladné celé číslo, pre ktoré platí*

$$|p - \tilde{p}|/|p| < 0.5 * 10^{-d}.$$

Príklad 2. *Vypočítajte hodnotu* $\sin(2)$ *v programe Mathematica [15], zistite počet platných číslic výsledku a zmeňte počet platných číslic na* 30.

Riešenie. Výsledok je štandardne vypísaný na 6 platných číslic. Ak zistíme presnosť výsledku v tvare počtu platných číslic príkazom `Precision[%]`, tak vidíme, že výsledok bol vyrátaný na 15 platných číslic. Neboli vypísané všetky platné číslice:

```
N[Sin[2]]
```

0.909297

```
Precision[%] // N
```

15.9546

Počet platných číslic výsledku môžeme zmeniť. Nastavíme počet platných číslic výsledku na 30 príkazom `N[ ]` a nechajme si vypísať počet platných číslic výsledku.

```
N[Sin[2], 30]
```

0.909297426825681695396019865912

```
Precision[%]
```

30.

□

Možné spôsoby nastavenia počtu platných číslic vypočítaných výsledkov v programe *Mathematica* sú nasledovné:

`N[22/10, 20]`	`N` nám nastaví počet platných číslic, v tomto prípade 20.
`2.20000000000000000`	Vypíšeme všetky platné číslice, v tomto prípade 18.
`SetPrecision[2.2, 9]`	`SetPrecision` nám nastaví 9 platných číslic.

3.1.1 Nepresnosť zanedbaním

Nepresnosť zanedbaním nastáva, ak veľmi komplikovaný matematický výraz nahradíme jednoduchšou formou. Samotný termín pochádza z techniky keď funkciu nahradíme skráteným Taylorovým rozvojom ([2], str.129–185) a zanedbáme tak zvyšné členy rozvoja. Napríklad nekonečný Taylorov rozvoj

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} + \ldots$$

môžeme nahradiť prvými piatimi členmi, v niektorých konkrétnych úlohách, čím vznikne nepresnosť *zanedbaním*.

3.1.2 Chyba zaokrúhľovania

Počítačová reprezentácia reálnych čísel je ohraničená presnosťou mantisy. V mnohých prípadoch počítačová reprezentácia nezodpovedá skutočnej hodnote. Tento fakt nazývame chyba *zaokrúhľovania*.

Reálne číslo $0,1$ v desiatkovej sústave zodpovedá v dvojkovej sústave číslu $0,00110011$ čo zapíšeme ako $(0,1)_{10} = (0,00110011001100110011001100...)_2$. Pri ukladaní tejto reprezentácie do počítača je nekonečná perióda za desatinnou čiarkou skrátená. Číslo, ktoré je v skutočnosti v počítači môže mať useknutú alebo zaokrúhlenú poslednú číslicu. Počítač pracuje s konečným počtom číslic, a tak sa vo výpočtoch bude vyskytovať chyba zaokrúhľovania, tá sa bude ďalej šíriť vo výpočtoch.

Príklad 3. *Ukážte chybu zaokrúhľovania na súčte dvoch čísel v programe Mathematica.*

Riešenie. V programe *Mathematica* spočítajme

```
1.234567890123456 + 10.^-16
```

1.234567890123456

Ako vidieť, súčet je rovný prvému z čísel. To vzniklo preto, že výsledok nemôže byť správne reprezentovaný pomocou 16 číslic. Chybu zaokrúhľovania odstránime, keď nastavíme väčší počet platných číslic (napríklad 17) oboch sčítancov:

```
N[1.234567890123456, 17] + N[10., 17]^-16
```

1.2345678901234561

□

3.1.3 Strata platnosti

Uvažujme dve reálne čísla $p = 3,1415926536$ a $q = 3,1415957341$, ktoré sú skoro rovnaké a majú presnosť jedenástich platných číslic. Ak počítame ich rozdiel dostaneme $p - q = -0,0000030805$. Ako vidíme, rozdiel medzi p, q má iba päť platných desatinných cifier. Rozdielom sme stratili šesť platných číslic a táto chyba sa volá strata platnosti. Zníženie presnosti o tento rozdiel môže pozmeniť konečný výsledok.

3.1.4 Šírenie chyby

Vyšetrujme ako sa môže chyba šíriť pri operáciách. Uvažujme dve čísla p, q a ich aproximácie $\tilde{p}$ a $\tilde{q}$.

Sčítanie a odčítanie. Označme chybovú odchýlku čísel p a q ako ε_p a ε_q, potom dostaneme

$$p \pm q = (\tilde{p} + \varepsilon_p) \pm (\tilde{q} + \varepsilon_q) = (\tilde{p} \pm \tilde{q}) + (\varepsilon_p + \varepsilon_q).$$

Pre operácie sčítania a odčítania je chyba súčtom odchyliek operandov $\varepsilon_{p\pm q} = \varepsilon_p \pm \varepsilon_q$. Relatívna odchýlka rozdielu dvoch kladných čísel, môže byť veľmi veľká stratou platných číslic.

Príklad 4. *Relatívna chyba. Uvažujme dve čísla $p = 1,137$ a $q = 1,073$ s chybou $\varepsilon_p = 0,011$ a $\varepsilon_q = -0,011$. Určite relatívnu chybu rozdielu.*

Riešenie. Relatívna chyba ich rozdielu je

$$R_{p-q} = (\varepsilon_p - \varepsilon_q)/(p - q) = 0,34 = 34\%.$$

Vo výsledku nemáme platné číslice aj napriek tomu, že relatívne chyby vstupných čísel sú malé $R_p = R_q = 0,01 = 1\%$. □

Násobenie. Šírenie chyby pri násobení je zložitejšie. Súčin dvoch čísel môžeme písať ako

$$pq = (\tilde{p} + \varepsilon_p)(\tilde{q} + \varepsilon_q) = \tilde{p}\tilde{q} + \tilde{p}\varepsilon_q + \tilde{q}\varepsilon_p + \varepsilon_p\varepsilon_q.$$

Predpokladajme, že $p \neq 0$ a $q \neq 0$, potom z predchádzajúcej rovnice

$$\frac{pq - \tilde{p}\tilde{q}}{pq} = \frac{\tilde{p}\varepsilon_q}{pq} + \frac{\tilde{q}\varepsilon_p}{pq} + \frac{\varepsilon_p\varepsilon_q}{pq}.$$

Pretože $\tilde{p}/p \approx 1$, $\tilde{q}/q \approx 1$ a tiež pretože tretí člen je veľmi malý v porovnaní s prvými dvoma členmi, dostaneme vzťah pre relatívnu chybu násobenia

$$R_{pq} = R_p + R_q.$$

Relatívna chyba súčinu je približne súčet odchýliek operandov.

3.2 VÝPOČET HODNÔT FUNKCIE

Pozrime sa na to ako sa správajú čísla pri výpočte hodnôt funkcií. Treba si uvedomiť, že matematicky ekvivalentné úpravy môžu počas výpočtu vyprodukovať veľmi rozdielne zaokrúhľovanie odchýlky. Ak spôsob, ktorým rátame hodnoty funkcie dáva výsledky s veľkým zaokrúhlením, je tento spôsob prakticky nepoužiteľný. Potrebujeme sa naučiť ako vypočítať hodnoty funkcie použitím počítača a vyhnúť sa zaokrúhľovaniu pri výpočte.

Príklad 5. *Počítajme funkčné hodnoty funkcie* $f(x) = \frac{\log(1-x)+xe^{(\frac{x}{2})}}{x^3}$ *v blízkosti nuly pre* $x = 10^{-4}, 10^{-5}, \ldots, 10^{-8}$.

Riešenie. Výpočet po realizácii v programe *Mathematica* vykazuje veľké chyby:

```
f[x_]:=(Log[1-x] + x Exp[x/2]) / x^3
Table[f[10.^-n], {n, 4, 8}]
```

$\{-0.208345, -0.162823, -28.9641, 52635.4, -5.02476 * 10^7\}$

Použime 23 platných číslic pri zadávaní argumentov funkcie, aby sme dostali správne výsledky:

```
Table[f[N[10., 23]^-n], {n, 4, 8}]
```

$\{-0.20835625197412, -0.20833562502, -0.208333563, -0.2083334, -0.20833\}$

Všimnite si ako sa počet číslic vo vypočítaných funkčných hodnotách zmenšuje zľava doprava, vďaka rýchlej strate platnosti. Program *Mathematica* vypisuje iba platné číslice. □

Príklad 6. *Kvadratická rovnica. Vypočítajte korene kvadratickej rovnice* $ax^2 + bx + c = 0$ *podľa vzťahu*

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Nech naša kvadratická rovnica je $x^2 - 8000x + 1 = 0$.

Riešenie. Po vypočítaní pomocou predošlej formulky dostaneme korene

$$x_1 = 8000, x_2 = 2,44 * 10^{-4}.$$

Jeden z koreňov je však zlý. Hodnota x_2 mala by byť rovná $1,25 * 10^{-4}$. Ako sa môžeme vyhnúť strate presnosti?

Jednoduchý spôsob ako počítať korene v tomto prípade, ($|4ac| \ll b^2$) je vynechať člen ax^2 a spraviť nasledujúci odhad

$$x_2 = -c/b = 0,000125.$$

Riešme ešte raz tento príklad v programe *Mathematica* a hľadajme korene v blízkosti $x = 6000$ a $x = 0$ použitím príkazu `FindRoot[ ]`:

```
FindRoot[x^2 - 8000 x + 1, {x, 6000}]
FindRoot[x^2 - 8000 x + 1, {x, 0}]
```

$\{x \to 8000.\}$
$\{x \to 0.000125\}$

Metóda numerickej iterácie použitá v príkaze `FindRoot[ ]` našla oba korene správne. □

3.2.1 Preusporiadanie členov

Typické metódy preusporiadania členov funkcie si ukážeme na niekoľkých príkladoch. Podobné prístupy sa používajú pri výpočte výrazov, ktoré vznikajú pri aproximáciách derivácií funkcií.

Príklad 7. *Rozdiel odmocnín. Vypočítajte hodnotu funkcie* $f(x) = \sqrt{(x+1)} - \sqrt{(x)}$ *pre veľké* x.

Riešenie. Výraz vynásobíme (rozšírime) jednotkou

$$f(x) = \left(\sqrt{(x+1)} - \sqrt{(x)}\right)\left(\frac{\sqrt{(x+1)} + \sqrt{(x)}}{\sqrt{(x+1)} + \sqrt{(x)}}\right) = \frac{1}{\sqrt{(x+1)} + \sqrt{(x)}}.$$

Po tejto úprave už nedochádza k strate platnosti lebo sme odstránili rozdiel odmocnín. □

Príklad 8. *Rozdiel sínusov. Vyčíslite* $f(x) = \sin(x+\varepsilon) - \sin(x)$ *pre malé* ε.

Riešenie. Môžeme použiť trigonometrickú identitu $\sin a - \sin b = 2\cos\frac{a+b}{2} \times \sin\frac{a-b}{2}$. Potom dostávame

$$f(x) = 2\cos\left(x + \frac{\varepsilon}{2}\right)\sin\frac{\varepsilon}{2}$$

čo je už vyhovujúca forma pre výpočet, lebo nedochádza k strate platnosti. □

3.2.2 Rozvoj do postupnosti

V niektorých prípadoch preusporiadaním nie je možné odstrániť problém so stratou platných číslic, ale musíme nájsť nejaké iné približné vyjadrenie funkcie. Jedno z efektívnych priblížení je rozvoj funkcie pomocou Taylorovho radu okolo vhodného bodu ([2], str.129–185).

Ak je $f(x)$ spojitá funkcia a má spojité derivácie do rádu $n+1$ okolo $x=a$, potom v okolí $x=a$ môže byť funkcia reprezentovaná Taylorovým radom

$$f(x) = f(a)+(x-a)\frac{f'(a)}{1!}+(x-a)^2\frac{f''(a)}{2!}\cdots+(x-a)^n\frac{f^{(n)}(a)}{n!}+O((x-a)^{n+1}),$$

kde $O((x-a)^{n+1})$ označuje chybu spôsobenú zanedbaním členov rádu $(n+1)$ a vyššieho. Napríklad, majme $f(x)=\sin x$. Potom $f'(x)=\cos x$, $f''(x)=-\cos x$, $f'''(x)=-\sin x$ atď. Číselné hodnoty $f(0)=0$, $f'(0)=1$, $f''(0)=0$, $f'''(0)=-1$ ... musíme dosadiť do vzorca pre Taylorov rozvoj. Získame tak nasledujúci polynóm stupňa $n=9$, ktorý aproximuje $\sin x$ v bode $x=0$

$$\sin x = x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}.$$

Príklad 9. *Podiel. Počítajme $f(x)=\frac{\sin x}{x}$ pre malé x.*

Riešenie. Ak reprezentujeme $\sin x$ ako rad

$$\sin x = x-\frac{x^3}{6}+\dots$$

tak

$$f(x)=\frac{\sin x}{x}=1-\frac{x^2}{6}+\frac{x^4}{120}-\dots$$

čo je presný a jednoduchý výraz aj pre malé x. Preveďme túto úpravu v programe *Mathematica*, kde Taylorov rozvoj okolo bodu $x=0$ až do rádu 4 získame použitím `Series[%, {x, 0, 4}]`:

```
f[x_] = Series[Sin[x]/x, {x, 0, 4}]
```

$1-\frac{x^2}{6}+\frac{x^4}{120}+O[x]^5$

Výsledkom je funkcia definovaná rozvojom $f(x)\approx 1-\frac{x^2}{6}+\frac{x^4}{120}$ a funkčná hodnota v bode $x=0$ je potom $f(0)=1$. A naozaj $\lim_{x\to 0}\frac{\sin(x)}{x}=1$. □

Príklad 10. *Exponenciálna funkcia. Počítajme funkciu $f(x)=e^x-1$ pre malé x.*

Riešenie. Rozvojom Taylorovho radu funkcie e^x v okolí bodu $x=0$, dostaneme

$$e^x-1=1+x+\frac{x^2}{2}+\frac{x^3}{6}+\dots-1=x(1+\frac{x}{2}+\frac{x^2}{6}+\dots).$$

Tento výraz je už bez väčších ťažkostí vypočítateľný. □

3.3 VÝPOČET HODNOTY POLYNÓMOV

Polynómy hrajú vo výpočtoch hlavnú úlohu. Vyskytujú sa v rôznych výpočtových algoritmoch a sú tiež často používané pri aproximácii a interpolácii. Polynóm

$$P(x) = a_0 + a_1x^1 + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n$$

môžeme tiež zapísať aj v reťazovej forme, pomocou Hornerovej schémy ([14] str. 108–109), ktorá je výhodnejšia pre výpočet

$$P(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\dots)).$$

Hodnotu tohto polynomického výrazu v bode x dostaneme použitím n súčtov a n násobení. Algoritmus 1 používa reťazenie na výpočet hodnoty polynómu.

Algoritmus 1 Algoritmus výpočtu hodnoty polynómu

$f = a_n$
for $i = n - 1, 0$ **do**
  $f = f * x + a_i$
end for

Príklad 11. *Faktorizujte polynóm* $r(x) = 8^2 + 64x + 16x^2 + 16x^2 + 8x^3 + x^4$ *pomocou Hornerovej schémy v programe Mathematica.*

Riešenie. Najskôr definujeme polynóm, ktorý budeme faktorizovať príkazom `Horner[ ]` z knižnice `Algebra‘Horner‘`:

```
r = 8^2 + 64x + 16x^2 + 16x^2 + 8x^3 + x^4;
<< Algebra‘Horner‘
Horner[r]
```

$64 + x(64 + x(32 + x(8 + x)))$

Výsledkom je upravený polynóm do tvaru $r(x) = 64 + x(64 + x(32 + x(8 + x)))$. □

3.4 APROXIMÁCIE FUNKCIÍ

Niekedy sú funkcie definované tak, že nie je ľahké vypočítať ich funkčné hodnoty vzhľadom na spôsob ich definície. Toto je pravda napríklad pri funkciách, ktoré sú definované ako integrály na nekonečných intervaloch. Ak máme dobrú

aproximáciu takejto integrálnej funkcie potom je vhodnejšie takúto aproximáciu použiť namiesto výpočtov hodnôt funkcie pomocou numerickej integrácie.

Príklad 12. *Aproximácia chybovej funkcie. Chybová funkcia je integrál normálneho rozdelenia ([9] str. 65-66) a používa sa v štatistikách na určenie pravdepodobnosti nájdenia merania menšieho ako daná hodnota ak sú merania rozdelené podľa normálneho rozdelenia so stredom v bode* 0 *a odchýlkou* 1*. Chybová funkcia je vyjadrená nasledujúcim integrálom*

$$erf(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-t^2}{2}}\, dt.$$

Hodnoty funkcie ležia medzi 0 *a* 1*. Mohli by sme použiť numerickú integráciu na výpočet hodnôt funkcie ale existuje aj dobrá aproximácia tejto funkcie.*

Riešenie. Formula Abramovitza a Stegana je aproximácia chybovej funkcie

$$\operatorname{erf}(x) \approx \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \sum_{i=1}^{5} a_i r^i(x) \qquad \text{pre } x \geq 0,$$

kde

$$\begin{aligned} r(x) = 1/(1 - 0,2316419x) \quad & a_1 = 0,31938153 \\ a_2 = -0,356563782 \quad & a_3 = 1,7814779372 \\ a_4 = -1,821255978 \quad & a_5 = 1,330274429. \end{aligned}$$

Chyba tejto aproximácie je lepšia než $7,5 * 10^{-8}$ pre kladné x. Na počítanie hodnôt pre záporné x, je možné použiť vzťah $\operatorname{erf}(x) = 1 - \operatorname{erf}(-x)$, lebo chybová funkcia je symetrická a $\operatorname{erf}(\infty) = 1$. □

Príklad 13. *Počítajte hodnoty chybovej funkcie v programe Mathematica.*

Riešenie. V programe *Mathematica* postupujeme nasledovne:

```
{Erf[0.95], Erf[-0.95]}
```

$\{0.820891, -0.820891\}$

Výsledkom je hodnota chybovej funkcie $\operatorname{erf}(0.95) = 0.820891$ a $\operatorname{erf}(-0.95) = -0.820891$. Vyjadrime polynomickú aproximáciu v programe *Mathematica* príkazom `Series[ ]`:

```
Series[Erf[x], {x, 0, 10}]
```

$\frac{2x}{\sqrt{\pi}} - \frac{2x^3}{3\sqrt{\pi}} + \frac{x^5}{5\sqrt{\pi}} - \frac{x^7}{21\sqrt{\pi}} + \frac{x^9}{108\sqrt{\pi}} + O[x]^{11}$

Výsledkom je polynomická aproximácia $\operatorname{erf}(x) \approx \frac{2x}{\sqrt{\pi}} - \frac{2x^3}{3\sqrt{\pi}} + \frac{x^5}{5\sqrt{\pi}} - \frac{x^7}{21\sqrt{\pi}} + \frac{x^9}{108\sqrt{\pi}}$ □

3.5 ÚLOHY

1 Podľa definície 2 zistite na koľko platných číslic číslo $\tilde{p}$ aproximuje číslo p:

a) $p = 1,137$ kde $\varepsilon_p = \tilde{p} - p = 0,011$;
b) $p = 1,073$ kde $\varepsilon_p = \tilde{p} - p = -0,011$;
c) $p = 3,1415926535$ ak $\tilde{p} = 3,1415957341$;
d) $p = \sqrt{2}$ ak $\tilde{p} = 1,14156$;
e) $p = \sqrt{2} - 1,14156$ ak $\tilde{p} = \frac{1}{\sqrt{2}-1,14156}$;
f) $p = \sin(3,1415957341) - \sin(3,1415926536)$ ak
$\varepsilon = 3,1415957341 - 3,1415926536$ a $\tilde{p} = 2\cos(3,1415926536 + \frac{\varepsilon}{2})\sin\frac{\varepsilon}{2}$.

2 Vypočítajte hodnoty funkcie pre veľké x a y na 20 platných číslic tak, aby nedochádzalo k strate presnosti a určite kedy majú zmysel:

$$\text{a) } 4^{x+1} - 4^x; \quad \text{b) } \frac{\sqrt{x+y}-\sqrt{x-y}}{\sqrt{x+y}+\sqrt{x-y}}.$$

3 Vypočítajte hodnoty funkcie pre malé x, $y < 10^{-3}$ na 20 platných číslic, tak aby nedochádzalo k strate presnosti a určite kedy majú zmysel:

$$\text{a) } \sin 2x - \sin x; \quad \text{b) } \cos 2x - \cos x; \quad \text{c) } \frac{1}{\sin y - \sin x}.$$

4 Vypočítajte hodnoty funkcie pomocou rozvoja do Taylorovho radu:

a) $e^x - 1$ pre $x = 0,01$; b) $e^{\frac{-x^2}{2}}$ pre $x = 0,001$;
c) $\frac{\sin x}{x}$ pre $x = 0,0001$.

5 Vypočítajte hodnotu polynómu (podielu polynómov) v bode $x = 2$ pomocou reťazovej formy:

a) $x^2 - 5x + 7$; b) $x^2 - 3x + 43$;
c) $\frac{x^3-3x}{x^4+1}$; d) $x^5 - 1$.

Kapitola 4

RIEŠENIE SYSTÉMOV LINEÁRNYCH ALGEBRAICKÝCH ROVNÍC

V tejto kapitole uvedieme tie najdôležitejšie metódy, ktoré máme k dispozícii na numerické riešenie systému lineárnych algebraických rovníc. Uvedieme niekoľko príkladov aby sme upozornili na niektoré vlastnosti týchto metód.

4.1 MATICE A VEKTORY

Matica s rozmermi $m \times n$ je reprezentovaná obdĺžnikovou tabuľkou čísel s m riadkami a n stĺpcami. *Vektor* o rozmere m je stĺpcová matica s m riadkami. Uveďme si príklad matice $\mathbf{A}$ a vektora $\mathbf{v}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \qquad \mathbf{v} = \begin{Bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{Bmatrix}.$$

Najčastejšie operácie na maticiach a vektoroch sú zhrnuté v nasledujúcich rovniciach.

Súčet vektorov:

$$\mathbf{w} = \mathbf{u} + \mathbf{v}, \qquad w_i = u_i + v_i, \quad i = 1, \ldots, n.$$

Násobenie vektora skalárom:

$$\mathbf{w} = \alpha\mathbf{v}, \qquad w_i = \alpha v_i, \quad i = 1, \ldots, n.$$

Skalárny súčin:

$$\mathbf{w} = \mathbf{u} \cdot \mathbf{v}, \qquad s = \sum u_i v_i, \quad i = 1, \ldots, n.$$

Norma vektora:

$$| \mathbf{v} | = \sqrt{\mathbf{v} \cdot \mathbf{v}}$$

Súčet matíc s rovnakou dimenziou:

$$\mathbf{C} = \mathbf{A} + \mathbf{B}, \qquad c_{ij} = a_{ij} + b_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Násobenie matice skalárom:

$$\mathbf{B} = \alpha\mathbf{A} \qquad b_{ij} = \alpha a_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Transpozícia matice $\mathbf{A}$ je matica $\mathbf{B}$:

$$\mathbf{B} = \mathbf{A}^\top \qquad b_{ij} = a_{ji}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Násobenie matice vektorom:

$$\mathbf{u} = \mathbf{A}\mathbf{v}, \qquad u_i = \sum a_{ij} v_j \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Súčin matice $m \times p$ s $p \times n$ maticou:

$$\mathbf{C} = \mathbf{A}\mathbf{B}, \qquad c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Identická matica $\mathbf{I}$ je štvorcová matica pre ktorú platí:

$$\mathbf{I}\mathbf{v} = \mathbf{v}, \qquad \mathbf{I}\mathbf{A} = \mathbf{A}\mathbf{I} = \mathbf{A}.$$

Symetrická matica je matica s vlastnosťou:

$$\mathbf{A}^\top = \mathbf{A}, \qquad a_{ij} = a_{ji}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Maticu reprezentujeme ako zoznam riadkov v programe *Mathematica* [15]. Zadajme maticu rozmerov 2×2 a vypíšme ju v maticovej forme príkazom `MatrixForm`

```
Q = {{3, 1}, {2, 5}}
Q // MatrixForm
```

$\{\{3,1\},\{2,5\}\}$

$$\begin{pmatrix} 3 & 1 \\ 2 & 5 \end{pmatrix}$$

Základné operácie s maticami sú jednoduché. Zadajme matice $\mathbf{A}$ a $\mathbf{B}$ a vypočítajme napríklad ich súčet.

```
A = {{3, 2}, {4, 1}};
B = {{2, 3}, {1, 2}};
A + B
```

$\{\{5,5\},\{5,3\}\}$

Násobenie matice $\mathbf{A}$ skalárom 3.

```
3 A
```

$\{\{9,6\},\{12,3\}\}$

Násobenie matíc $\mathbf{A}$ a $\mathbf{B}$.

```
A.B
```

$\{\{8,13\},\{9,14\}\}$

Násobenie matice $\mathbf{A}$ vektorom $v = \begin{Bmatrix} x \\ y \end{Bmatrix}$.

```
a = {{3, 2}, {4, 1}};
v = {x, y};
a.v
```

$\{3x+2y, 4x+y\}$

Spôsoby definície matíc a základné operácie s maticami v programe *Mathematica* zosumarizujeme v tabuľke:

`MatrixForm[A]`	Výpis matice $\mathbf{A}$ v maticovej forme.
`Array[f, m, n]`	Vytvor maticu $(m \times n)$ z prvkov `f[i, j]`.
`IdentityMatrix[n]`	Identická matica $(n \times n)$.
`ZeroMatrix[n]`	Nulová matica $(n \times n)$.
`A + B`	Súčet matíc $\mathbf{A}$ a $\mathbf{B}$.
`a A`	Násobenie matice $\mathbf{A}$ skalárom a.
`Transpose[A]`	Transpozícia matice $\mathbf{A}$.
`A . B`	Súčin matíc $\mathbf{A}$ a $\mathbf{B}$.

4.2 SYSTÉM LINEÁRNYCH ROVNÍC

Systém lineárnych algebraických rovníc

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

je možné reprezentovať v maticovo-vektorovom tvare ako

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

kde $\mathbf{A}$ je matica systému, $\mathbf{b} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{Bmatrix}$ je vektor pozostávajúci z pravých strán rovníc a $\mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}$ je riešenie systému lineárnych rovníc, resp. hľadaný vektor neznámych. V ďalších častiach kapitoly uvažujme matice, ktoré dostávame v praxi po diskretizácii spojitých problémov a po zahrnutí okrajových podmienok. Predpokladáme, že matica systému je regulárna, teda taká, že lineárny systém má práve jedno riešenie.

4.3 GAUSSOVA ELIMINÁCIA

Riešenie systému lineárnych rovníc pomocou Gaussovej eliminácie ([14] str. 31) spočíva v

1. eliminácii a

2. spätnej substitúcii.

Pre názornosť riešme systém bez nulových koeficientov $a_{ij} \neq 0$. V eliminačnom kroku použijeme nasledujúce ekvivalentné operácie:

a_{21}/a_{11} násobok prvej rovnice sa odčíta od druhej, čím sa z druhej rovnice eliminuje prvá neznáma x_1. Podobne sa x_i z i-tej rovnice odstráni odčítaním a_{i1}/a_{11} násobku prvej rovnice pre všetky $i > 2$. Po týchto ekvivalentných

operáciách sa pôvodný systém zmení na

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ &\ldots \\ a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n &= b_n^{(1)}, \end{aligned}$$

kde $a_{ij}^{(1)} = a_{ij} - (a_{i1}/a_{11})a_{1j}$. V ďalšom kroku sa eliminuje druhá neznáma z tretej až poslednej rovnice odčítaním $a_{i2}^{(1)}/a_{22}^{(1)}$ násobku druhej rovnice. Po $n-1$ takýchto krokoch proces eliminácie skončí a pôvodný systém bude mať trojuholníkový tvar

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ &\ldots \\ a_{nn}^{(n-1)}x_n &= b_n^{(n-1)}. \end{aligned}$$

Spätná substitúcia začína poslednou rovnicou. Riešenie pre x_n dostaneme ako $x_n = b_n^{(n-1)} \Big/ a_{nn}^{(n-1)}$. Následne zvyšné neznáme vypočítame z už známych hodnôt x_i v poradí

$$\begin{aligned} x_{n-1} &= (b_{(n-1)}^{n-2} - a_{n-1,n}^{(n-2)}x_n) \Big/ a_{n-1,n-1}^{(n-2)} \\ &\ldots \\ x_1 &= (b_1 - \sum_{j=2}^{n} a_{1j}x_j) \Big/ a_{11}. \end{aligned}$$

Takto sme našli všetky zložky riešenia x systému rovníc. Algoritmus Gaussovej eliminácie nájdete podrobne popísaný v pseudokóde algoritmu 2. Výpočtová zložitosť algoritmu Gaussovej eliminácie je odvodená od počtu modifikácií koeficientov a_{ik} v cykloch a je rovná $2n^3/3$ operáciám násobenia a sčítania.

4.3.1 Pivotizácia v Gaussovej eliminácii

Definícia 3. *Prvok a_{pp} , ktorý je použitý na elimináciu neznámej x_p v riadkoch $p+1$, $p+2$, ..., n v procese Gaussovej eliminácie sa nazýva p-ty pivot a p-ty riadok pivotným riadkom.*

Pivotizácia s nenulovým pivotom. Ak $a_{pp}^{(p)} = 0$ tak riadok p nemôže byť použitý na elimináciu prvkov v stĺpci p pod diagonálou. Je potrebné nájsť riadok k, pre ktorý $a_{kp}^{(p)} \neq 0$ a $k > p$ a potom zameniť riadky p a k aby sa získal nenulový pivot.

Algoritmus 2 Algoritmus Gaussovej eliminácie

Počíta riešenie systémov v tvare $\mathbf{Ax} = \mathbf{b}$.
Dopredná eliminácia
for $j = 1, n-1$ **do**
  for $i = j+1, n-1$ **do**
    $m_{ij} = a_{ij}/a_{jj}$
    $a_{ij} = 0$
    $b_i = b_i - m_{ij} * b_j$
    for $k = j+1, n$ **do**
      $a_{ik} = a_{ik} - m_{ij} * a_{jk}$
    end for
  end for
end for

Spätná substitúcia
for $i = n-1, 1$ **do**
  $x_i = b_i/a_{ii}$
  for $j = i+1, n$ **do**
    $x_i = (x_i – a_{ij} * x_j)/a_{ii}$
  end for
end for

Pivotizácia za účelom zníženia chyby. Kvôli zníženiu propagácie chyby je vhodné, aby sa z prvkov v stĺpci p ležiacich na diagonále a pod ňou vybral riadok k v ktorom leží prvok s najväčšou absolútnou hodnotou. Riadok k spravíme pivotným riadkom po zámene s riadkom p.

Príklad 14. *V programe Mathematica hľadajme riešenie sústavy* $\mathbf{Ax} = \mathbf{b}$, *kde*

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{b} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}.$$

Riešenie. Najskôr zadáme maticu `A` a pravú stranu sústavy `b`.

```
A = {{2, 1, 1}, {1, 1, 1}, {1, 0, 1}};
b = {1, 2, 3};
```

Na riešenie použijeme príkaz `LinearSolve`:

```
riesenie = LinearSolve[A, b]
```

$\{-1, -1, 4\}$

Riešenie $\mathbf{x} = \begin{Bmatrix} -1 \\ -1 \\ 4 \end{Bmatrix}$ sústavy sme našli presne.

Nájdime riešenie pomocou Gausovej eliminácie. Funkcia `RowReduce` zredukuje sústavu rovníc na jednotkový tvar použitím Gausovej eliminácie. Najskôr musíme zostrojiť maticu m, ktorú dostaneme pridaním pravej strany sústavy b k matici sústavy A.

```
m = Transpose[Append[Transpose[A], b]];
m // MatrixForm
```

$$\begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 3 \end{pmatrix}$$

Po zredukovaní matice m sa výsledok nachádza v poslednom stĺpci na mieste vektora b. Nakoniec vypíšeme zredukovanú maticu a výsledok.

```
r = RowReduce[m];
m // MatrixForm
Map[Last, r]
```

$$\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 4 \end{pmatrix}$$

$\{-1, -1, 4\}$

Presné riešenie sústavy je $\mathbf{x} = \begin{Bmatrix} -1 \\ -1 \\ 4 \end{Bmatrix}$. □

Príklad 15. *V programe Mathematica hľadajme riešenie sústavy so všeobecnou pravou stranou* $\mathbf{Ax} = \mathbf{b}$*, kde*

$$\mathbf{A} = \begin{bmatrix} 1 & 5 \\ 2 & 1 \end{bmatrix} \qquad \mathbf{b} = \begin{Bmatrix} a \\ b \end{Bmatrix}$$

a $a, b \in R$ *sú parametre.*

Riešenie. Zadajme maticu `A`.

```
A = {{1, 5}, {2, 1}};
```

Na riešenie použijeme príkaz `LinearSolve` s prvou stranou sústavy `{a, b}`:

```
riesenie = LinearSolve[A, {a, b}]
```

$\left\{\frac{1}{9}(-a + 5b), \frac{1}{9}(2a - b)\right\}$

Všeobecné riešenie s parametrami a a b je $\mathbf{x} = \begin{Bmatrix} \frac{1}{9}(-a + 5b) \\ \frac{1}{9}(2a - b) \end{Bmatrix}$. □

4.4 LU ROZKLAD

Regulárna matica $\mathbf{A}$ má trojuholníkovú faktorizáciu, ak sa dá vyjadriť ([14] str. 50–52) ako súčin dolnej trojuholníkovej $\mathbf{L}$ a hornej trojuholníkovej matice $\mathbf{U}$; $\mathbf{A} = \mathbf{LU}$. Trojuholníková faktorizácia pre maticu 4×4 je

$$\mathbf{A} = \begin{bmatrix} m_{11} & 0 & 0 & 0 \\ m_{21} & m_{22} & 0 & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4.4.1 Riešenie faktorizovaného lineárneho systému

Ak matica koeficientov $\mathbf{A}$ lineárneho systému $\mathbf{Ax} = \mathbf{b}$ má trojuholníkovú faktorizáciu $\mathbf{A} = \mathbf{LU}$, potom riešenie pre $\mathbf{LUx} = \mathbf{b}$ sa dá vypočítať položením $\mathbf{y}$=$\mathbf{Ux}$ a vyriešením dvoch systémov

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{Ux} &= \mathbf{y}. \end{aligned}$$

4.4.2 Trojuholníková faktorizácia

Vo všeobecnosti metóda výpočtu koeficientov $\mathbf{L}$ a $\mathbf{U}$ sa dá napísať ako ([14] str. 50-51)

$$\begin{aligned} m_{ij} &= a_{ij} - \sum_{k=1}^{j-1} m_{ik}u_{kj} && j \le i, \quad j = 1, 2, \ldots, n \\ u_{ij} &= \left(a_{ij} - \sum_{k=1}^{i-1} m_{ik}u_{kj} \right) \Big/ m_{ii} && i \le j, \quad j = 2, 3, \ldots, n. \end{aligned}$$

Je dobré si uvedomiť, že pre $j = 1$ sú koeficienty pre $\mathbf{L}$ rovné $m_{i1} = a_{i1}$, a pre $i = 1$ sú koeficienty pre $\mathbf{U}$ rovné $u_{1j} = a_{1j}/m_{11}$.

4.4.3 Redukcia pravej strany a spätná substitúcia

Všeobecná formula pre redukciu pravej strany $\mathbf{b}$ je daná

$$\begin{aligned} b'_1 &= b_1 \Big/ m_{11} \\ b'_i &= \left(b_i - \sum_{k=1}^{i-1} m_{ik}b'_k \right) \Big/ m_{ii} \qquad j = 2, 3, \ldots, n. \end{aligned}$$

Procedúra spätnej substitúcie je popísaná nasledujúcimi vzťahmi

$$x_n = b'_n$$
$$x_j = b'_j - \sum_{k=j+1}^{n} u_{jk} x_k \qquad j = n-1, n-2, \ldots, 1.$$

Celý proces výpočtu prvkov m_{ij} a u_{ij} matíc **L** a **U** je možné popísať pseudokódom uvedenom v algoritme 3.

Algoritmus 3 Algoritmus trojuholníkovej **LU** dekompozície

Rozloží maticu **A** na dolnú trojuholníkovú maticu **L** a hornú trojuholníkovú maticu **U**.

```
for i = 1, n do
  m_i1 = a_i1
end for
for j = 1, n do
  u_1j = a_1j / m_11
end for
for j = 2, n do
  for i = j, n do
    s = 0
    for k = 1, j − 1 do
      s = s + m_ik * u_kj
    end for
    m_ij = a_ij − s
  end for
  u_jj = 1
  for i = j + 1, n do
    s = 0
    for k = 1, j − 1 do
      s = s + m_jk * u_ki
    end for
    u_ji = (a_ji − s) / m_jj
  end for
end for
```

4.4.4 Implementácia LU rozkladu.

Dôvod, prečo je LU dekompozícia populárna v programovaní je ten, že dekomponovaná matica nemá väčšie pamäťové nároky ako matica pôvodná. Nulové prvky dolnej a hornej trojuholníkovej matice a tiež jednotky na diagonále matice **U** sa nemenia a preto nie je dôvod si ich explicitne pamätať. Tiež môžeme ukladať nenulové koeficienty matíc **L** a **U** na miesta koeficientov matice **A**. Je to možné preto, že keď je raz prvok a_{ij} použitý, tak sa viac nevyskytne

vo výpočtoch. Využitie prvkov pôvodnej matice na uloženie jej rozloženého tvaru ukazuje nasledujúca schéma

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \longrightarrow \begin{bmatrix} m_{11} & u_{12} & u_{13} & u_{14} \\ m_{21} & m_{22} & u_{23} & u_{24} \\ m_{31} & m_{32} & m_{33} & u_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

ktorá sa zvykne nazývať aj kompaktná schéma.

Príklad 16. *V programe Mathematica použite LU rozklad na nájdenie riešení sústav* $\mathbf{Ax} = \mathbf{b1}$ *a* $\mathbf{Ax} = \mathbf{b2}$*, kde*

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b1} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix}, \qquad \mathbf{b2} = \begin{Bmatrix} -4 \\ 5 \\ 6 \end{Bmatrix}.$$

Riešenie. Najskôr zadajme maticu `A` a pravé strany sústav `b1`, `b2`.

```
A = {{2, 1, 1}, {1, 1, 1}, {1, 0, 1}};
b1 = {1, 2, 3};
b2 = {-4, 5, 6};
```

Na riešenie použijeme LU rozklad, príkazom `LUDecomposition` a následne spätnú substitúciu príkazom `LUBackSubstitution`:

```
lu=LUDecomposition[A];
LUBackSubstitution[lu,b1];
```

$\{-1, -1, 4\}$

Riešenie sústavy $\mathbf{Ax} = \mathbf{b1}$ je $\mathbf{x} = \begin{Bmatrix} -1 \\ -1 \\ 4 \end{Bmatrix}$. Pri riešení sústavy s rovnakou maticou sústavy nemusíme rátať znovu LU rozklad ale použijeme len inú pravú stranu rovnice pri spätnej substitúcii.

```
LUBackSubstitution[lu,b2];
```

$\{-9, -1, 15\}$

Riešenie druhej sústavy $\mathbf{Ax} = \mathbf{b2}$ je $\mathbf{x} = \begin{Bmatrix} -9 \\ -1 \\ 15 \end{Bmatrix}$. □

Možné metódy riešenia sústav rovníc v programe *Mathematica* sú nasledujúce:

`LinearSolve[m,b]` Nájde vektor x, ktorý rieši maticovú rovnicu $\mathbf{mx} = \mathbf{b}$.
`NullSpace[m]` Nájde lineárne nezávislé vektory, ktorých lineárnou kombináciou dostaneme všetky riešenia maticovej rovnice $\mathbf{mx} = \mathbf{0}$.
`RowReduce[a]` Redukuje maticu na jednotkový tvar použitím Gausovej eliminácie.
`LUDecomposition[m]` Nájde LU rozklad štvorcovej matice $\mathbf{m}$.
`LUBackSubstitution[lu, b]` Rieši maticovú rovnicu $\mathbf{mx} = \mathbf{b}$ ak `lu` je LU rozklad matice $\mathbf{m}$.

4.5 ZLÁ PODMIENENOSŤ SYSTÉMU

Systém $\mathbf{Ax} = \mathbf{b}$ je zle podmienený, ak malé zmeny koeficientov $\mathbf{A}$ alebo $\mathbf{b}$ spôsobia veľké zmeny vo výsledku $\mathbf{x}$. V prípade zle podmienených systémov numerické metódy môžu dávať značne chybné riešenia alebo dokonca zlyhať úplne.

Systém $\mathbf{Ax} = \mathbf{b}$ je zle podmienený, ak $\mathbf{A}$ je „skoro singulárna“ teda determinant $|\mathbf{A}|$ (pozri sekciu 6.1) je blízky nule. Napríklad systém dvoch rovníc je zle podmienený, ak dve priamky reprezentované jeho rovnicami sú takmer rovnobežné.

Príklad 17. *V programe Mathematica zistite číslo podmienenosti matice ([14] str. 21) systému* $\mathbf{Ax} = \mathbf{b}$*, kde*

$$\mathbf{A} = \begin{bmatrix} 0.53 & 0.88 & 0.18 \\ 0.53 & 0.88 & 0.181 \\ 0.17 & 0.56 & 0.36 \end{bmatrix} \qquad \mathbf{b} = \begin{Bmatrix} 5 \\ 2 \\ 4 \end{Bmatrix}.$$

Riešenie. Najskôr zadajme maticu `A` a pravú stranu sústavy `b`.

```
A = {{0.53, 0.88, 0.18}, {0.53, 0.88, 0.181},
    {0.17, 0.56, 0.36}} ;
b = {5, 2, 4};
```

Príkaz `LUDecomposition` použijeme na zistenie čísla podmienenosti matice systému:

```
{LU, perm, cond} = LUDecomposition[A]
```

$\{\{\{0.53, 0.88, 0.18\}, \{0.320755, 0.277736, 0.302264\}, \{1., 0., 0.001\}\}, \{1, 3, 2\}, 4672.7\}$

Výsledkom LU rozkladu pomocou príkazu `LUDecomposition` nie sú dve matice ale jedna matica `LU` v kompaktnom tvare obsahujúca matice $\mathbf{L}$ a $\mathbf{U}$. Výsledok tiež obsahuje zoznam permutácii riadkov `perm` pre zvýšenie presnosti. Posledné číslo výsledku `cond` je odhad podmienenosti matice $\mathbf{A}$. Matica je zle podmienená s číslom podmienenosti $4672,7$, lebo dva jej riadky sú velmi podobné. Riešenie sústavy $\mathbf{Ax} = \mathbf{b}$ je

```
x = LUBackSubstitution[{LU, perm, cond}, {5, 2, 4}]
```

$\{-4407.07, 3273.57, -3000.\}$

Riešenie systému je $\mathbf{x} = \begin{Bmatrix} -4407,07 \\ 3273,57 \\ -3000 \end{Bmatrix}$. Na záver si môžeme vypísať matice LU rozkladu

```
<<LinearAlgebra`MatrixManipulation`
Map[MatrixForm, {L, U} = LUMatrices[LU]]
```

$$\left\{ \begin{pmatrix} 1. & 0. & 0. \\ 0.320755 & 1. & 0. \\ 1. & 0. & 1. \end{pmatrix}, \begin{pmatrix} 0.53 & 0.88 & 0.18 \\ 0 & 0.277736 & 0.302264 \\ 0 & 0 & 0.001 \end{pmatrix} \right\}$$

□

4.6 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Zistite či nasledujúce systémy rovníc majú riešenie a nájdite ho Gausovou eliminačnou metódou:

a)
$$\begin{aligned} 3x_1 + x_2 + x_3 &= 3, \\ 4x_1 - x_2 + 2x_3 &= 4, \\ x_1 - x_2 + x_3 &= 1, \\ 4x_1 - x_2 + 2x_3 &= 5; \end{aligned}$$

b)
$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 16, \\ 8x_1 + 7x_2 + 6x_3 &= 74, \\ 4x_1 + 5x_2 + 9x_3 &= 49, \\ 5x_1 + 4x_2 + 2x_3 &= 43, \\ x_1 + 4x_2 + x_3 &= 22. \end{aligned}$$

2 Riešte systém a nájdite riešenie:

a)
$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 5, \\ 2x_1 + 3x_2 + x_3 &= 8, \\ 3x_1 + x_2 + 2x_3 &= 5; \end{aligned}$$

b)
$$\begin{aligned} x_1 + 2x_2 - x_3 + 4x_5 &= 2, \\ x_1 + 4x_2 - 5x_3 + x_4 + 3x_5 &= 1, \\ 2x_1 - 2x_2 + 10x_3 + x_4 - x_5 &= 11, \\ 3x_1 + 2x_2 + 5x_3 + 2x_4 + 2x_5 &= 12. \end{aligned}$$

3 $\frac{3x+y}{z+1} = 2$, $\frac{3y+z}{x+1} = 2$, $\frac{3z+x}{y+1} = 2$.

4 $\frac{6}{x} + \frac{4}{y} + \frac{5}{z} = 4$, $\frac{3}{x} + \frac{8}{y} + \frac{5}{z} = 4$, $\frac{9}{x} + \frac{12}{y} - \frac{10}{z} = 4$.

5 Zdôvodnite, prečo nemajú riešenia systémy overením či je matica singulárna:

a)
$$\begin{aligned} x_1 + x_2 - x_3 &= 5, \\ 2x_1 + 2x_2 - 2x_3 &= 8, \\ 3x_1 + x_2 + 2x_3 &= 5; \end{aligned}$$

b)
$$\begin{aligned} x_1 + x_2 + x_3 &= 5, \\ 3x_1 - 2x_2 + x_3 &= 8, \\ 4x_1 - x_2 + 2x_3 &= 5. \end{aligned}$$

6 Homogénny systém linárnych rovníc $\mathbf{Ax} = \mathbf{0}$ môžeme riešiť uvedenými metódami tak, že položíme vektor pravých strán rovný nulovému vektoru $\mathbf{b} = \mathbf{0}$. Riešte homogénny systém linárnych rovníc s maticou:

a) $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 0 \\ 1 & 1 & 2 & 4 \\ 2 & 3 & 5 & 0 \end{bmatrix}$; b) $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 6 & 3 \\ 1 & 1 & 2 & 4 \\ 2 & 3 & 5 & 0 \end{bmatrix}$;

c) Nájdite riešenie nehomogenného systému $\mathbf{Ax} = \mathbf{b}$ s pravou stranou $\mathbf{b} = (0, 0, -2, 2)^\top$.

7 Vzhľadom na parameter λ vyšetrujte riešenie systému

$$\begin{aligned} 3x + 2y + z &= 0 \\ 6x + 4y + \lambda z &= 0 \end{aligned}$$

a zistite kedy:

a) nemá žiadne riešenie,
b) má práve jedno riešenie a nájdite ho.

8 Riešte systémy

a) $x + y = 4,\ y + z = 8,\ z + u = 12,\ u + x = 8;$
b) $x + y + z = 14,\ y + z + u = 22,\ z + u + x = 20,\ u + x + y = 16.$

9 Riešte systémy použitím LU rozkladu

a)
$$\begin{aligned} x + y + z + u &= 10, \\ x + y - z - u &= -4, \\ x - y - z + u &= 0, \\ -x + y + z + u &= 8; \end{aligned}$$

b)
$$\begin{aligned} x + y - z + u &= 11, \\ x - y + z + u &= 3, \\ x - u &= 4, \\ y + z &= 8; \end{aligned}$$

c)
$$\begin{aligned} x + 2y + 3z + 4u &= 10, \\ 2x + y - z + 3u &= 5, \\ 3x + 4y - z - u &= 5, \\ -x - 2y + 3z + u &= 1; \end{aligned}$$

d)
$$\begin{aligned} x + y - 2z + u &= 1, \\ 2x + 2y - z - u &= 2, \\ 3x + y + z + u &= 14, \\ x - y + z - u &= 0. \end{aligned}$$

10 Nájdite maticu $\mathbf{X}$, ak existuje, ktorá spĺňa rovnicu $\mathbf{AX} + \mathbf{B} = \mathbf{BC}$, kde

$$\mathbf{A} = \begin{bmatrix} 3 & 1 \\ 5 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & -3 \\ 4 & 4 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 6 & 4 \\ -2 & -1 \end{bmatrix}.$$

4.7 VÝSLEDKY

1 a) Nemá riešenie; b) $x_1 = 5$, $x_2 = 4$, $x_3 = 1$.

2 a) $x_1 = 1$, $x_2 = 2$, $x_3 = 0$; b) $x_1 = 4$, $x_2 = -1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$;.

3 $x = 1$, $y = 1$, $z = 1$.

4 Zaveďte nové neznáme $u = \frac{1}{x}$, $v = \frac{1}{y}$, $w = \frac{1}{z}$. $x = 3$, $y = 4$, $z = 5$.

6 a) Triviálne riešenie $(0, 0, 0, 0)^\top$; b) $(-9, 11, -3, 1)^\top$; c) pre maticu a) $(3, -3, 1, -1)^\top$; pre maticu b) $(-6, 8, -2, 0)^\top$.

8 a) $x = 8 - u$, $y = -4 + u$, $z = 12 - u$, u ľubovoľné číslo; a) $x = 2$, $y = 4$, $z = 8$, $u = 10$.

9 a) $x = 1$, $y = 2$, $z = 3$, $u = 4$; b) $x = \frac{11}{2}$, $y = 6$, $z = 2$, $u = \frac{3}{2}$; c) $x = 1$, $y = 1$, $z = 1$, $u = 1$; d) $x = 2$, $y = 2$, $z = 3$, $u = 3$.

10 $\mathbf{X} = \begin{bmatrix} 20 & 20 \\ -44 & -46 \end{bmatrix}$.

Kapitola 5

METÓDY RIEDKYCH MATÍC

Numerické metódy pre parciálne diferenciálne rovnice ako sú metóda konečných prvkov alebo metóda konečných diferencií často navrhujeme tak, aby systém rovníc mal maticu charakterizovanú ako *symetrickú*, *kladne definitnú* a *riedku*.

Symetrická matica. Štvorcová matica $\mathbf{A}$ je *symetrická* ak je symetrická podľa hlavnej diagonály, teda $a_{ij} = a_{ji}$ pre všetky i a j. Symetrická matica je rovnaká ako jej transponovaná matica $\mathbf{A}^\top = \mathbf{A}$. Pri symetrickej matici stačí, aby boli v pamäti počítača uchované len hodnoty na jednej strane od diagonály plus hodnoty na diagonále.

Kladne definitná matica. Štvorcová matica $\mathbf{A}$ je *kladne definitná*, ak jej kvadratická forma $\mathbf{x}^\top \mathbf{A}\mathbf{x}$ je kladná pre všetky reálne nenulové vektory $\mathbf{x}$. Riešenie sústav rovníc s kladne definitnými maticami je ľahšie, pretože obyčajne netreba používať pivota pri rozkladaní matice.

Riedka matica. Matica je *riedka* ak len malá časť jej prvkov je nenulová. V niektorých prípadoch môžeme uchovávať len nenulové prvky riedkej matice a používať ich počas riešenia sústavy, napríklad ak pri riešení sústavy používame iteračné metódy.

5.1 UKLADANIE SYMETRICKÉHO PÁSU

Príklad symetrickej matice s nenulovými koeficientami zoskupenými okolo hlavnej diagonály je ukázaný nižšie. Keďže je matica symetrická uvádzame len hornú polovicu matice s nenulovými koeficientami označenými ako x:

$$\begin{bmatrix} x & x & x & & & \\ & x & x & x & & \\ & & x & x & x & \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \end{bmatrix}.$$

Nenulové koeficienty matice môžu byť uložené ako dvojrozmerné alebo jednorozmerné pole podľa riadkov alebo stĺpcov. Ďalej si musíme pamätať aj šírku pásu, označíme si ju h. Je potrebné brať do úvahy, že posledné riadky alebo prvé stĺpce majú iný počet nenulových členov ako h. Na zjednodušenie ukladania podľa stĺpcov uložíme všetky stĺpce s rovnakou výškou pridaním potrebného počtu núl do prvých $h-1$ stĺpcov.

Napríklad, vyššie uvedená maticu uložíme ako nasledujúce pole obsahujúce stĺpce s konštantnou dĺžkou $h = 3$:

$$A = \{0; 0; a_{11}; 0; a_{12}; a_{22}; a_{13}; a_{23}; a_{33}; a_{24}; a_{34}; a_{44}; a_{35}; a_{45}; a_{55}; a_{46}; a_{56}; a_{66}\}.$$

5.2 UKLADANIE SYMETRICKÉHO PROFILU

V mnohých praktických prípadoch majú matice šírku pásu premennú. Šírka matice h je však veľká len pri malom počte riadkov alebo stĺpcov

$$\begin{bmatrix} x & & x & & x & \\ & x & x & & x & \\ & & x & x & x & \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \end{bmatrix}.$$

V takýchto prípadoch je efektívnejšie pre úsporu pamäte a počtu operácií ukladať *symetrický profil* matice. Symetrická matica $\mathbf{A} = a_{ij}$ stupňa n je

uložená podľa stĺpcov. Každý stĺpec začína od prvého horného nenulového prvku a končí diagonálnym prvkom. Symetrický profil matice reprezentujú dve polia:

1. Reálne pole A, obsahujúce prvky matice usporiadané po stĺpcoch.
2. Celočíselné pole smerníkov *pcol*.

I-ty prvok *pcol* obsahuje adresu prvého prvku v stĺpci mínus jedna. Dĺžka i-teho stĺpca je daná vzťahom $pcol[i+1]-pcol[i]$. Dĺžka poľa A je rovná $pcol[n+1]$ (ak indexy poľa začínajú od 1). Vhodné usporiadanie uzlov môže rapídne znížiť profil matice. Napríklad, vyššie uvedená matica môže byť uložená pomocou nasledujúcich dvoch polí:

$$A = \{a_{11}; a_{22}; a_{13}; a_{23}; a_{33}; a_{34}; a_{44}; a_{15}; a_{25}; a_{35}; a_{45}; a_{55}; a_{46}; a_{56}; a_{66}\}$$

$$pcol = \{0; 1; 2; 5; 7; 12; 15\}.$$

5.3 UKLADANIE RIEDKYCH RIADKOV

Formát založený na rozumnom ukladaní riedkych riadkov je užitočný pre iteratívne metódy riešenia sústav rovníc. V tejto schéme ukladáme hodnoty nenulových prvkov matice A podľa riadkov spolu s indexom stĺpca a prvým nenulovým prvkom. Matica v riedkom riadkovom formáte je reprezentovaná tromi poľami:

1. Pole $A[prow[n-1]]$ obsahujúce nenulové prvky matice.
2. Celočíselné pole $col[prow[n-1]]$ indexov stĺpcov pre nenulové prvky.
3. Pole smerníkov $prow[n-1]$ na začiatok každého riadku.

Napríklad, nasledujúca matica

$$\begin{bmatrix} x & & x & x & & \\ & x & & & & \\ x & & x & & x & \\ x & & & x & & \\ & & x & & x & \\ & & & & & x \end{bmatrix}$$

môže byť uložená v nasledujúcich troch poliach:

$$\begin{aligned} A &= \{a_{11}; a_{13}; a_{14}; a_{22}; a_{31}; a_{33}; a_{35}; a_{41}; a_{44}; a_{53}; a_{55}; a_{66}\}, \\ col &= \{0; 2; 3; 1; 0; 2; 4; 0; 3; 2; 4; 5\}, \\ prow &= \{0; 3; 4; 7; 9; 11\}. \end{aligned}$$

Skúsme spätne vypísať prvky tretieho riadku matice. Položme $n = 3$ potom $prow[3-1] = 4$. Stĺpce matice, ktoré chceme vypísať sú medzi smerníkmi $col[prow[3-1]] = col[4]$ a $col[prow[4-1]] = col[7]$ teda stĺpce $col[4] = 0$, $col[5] = 2$ a $col[6] = 4$. Prvky tretieho riadku sú $A[prow[3-1]] = A[4] = a_{31}$ v stĺpci 0, $A[prow[3-1]+1] = A[5] = a_{33}$ v stĺpci 2 a $A[prow[3-1]+2] = A[6] = a_{35}$ v stĺpci 4.

5.4 METÓDY PRIAMEHO RIEŠENIA

Metódy priameho riešenia sústavy lineárnych rovníc sú založené na algoritmoch, ktoré poskytujú riešenie sústav rovníc s vopred známym počtom krokov.

5.4.1 Metóda LDU rozkladu

LDU rozklad symetrickej matice je často používaný vo výpočtovej praxi. Riešenie symetrickej lineárnej algebraickej sústavy $\mathbf{Ax} = \mathbf{b}$ pozostáva z troch krokov:

1. Faktorizácia: $\mathbf{A} = \mathbf{U}^\top\mathbf{DU}$,
2. Dopredná redukcia: $\mathbf{y} = U^{-\top}\mathbf{b}$,
3. Spätná substitúcia: $\mathbf{x} = \mathbf{U}^{-1}\mathbf{D}^{-1}\mathbf{y}$.

Uvažujme dve implementácie LDU metódy na riešenie symetrickej sústavy rovníc s maticou uloženou vo formáte symetrického pásu a v profilovom formáte.

LDU metóda s ukladaním pásu podľa stĺpcov. Načrtnime si algoritmus LDU faktorizácie pre hornú symetrickú časť pásu nasledujúcim pseudokódom, pozri algoritmus 4. Všimnite si, že všetky výpočty sú robené vo vnútri pásu symetrickej časti matice, ale dolné indexy sú dané pre plnú maticu A_{ij}. Medzi dvojindexovým označením A_{ij} a prvkami jednorozmerného poľa a, do ktorého sa ukladajú stĺpce rovnakej výšky existuje nasledujúci vzťah:

$$A_{ij} \to a[i + (h-1)j],$$

kde h je šírka pásu matice. Dopredná redukcia a spätná substitúcia pre ľavú stranu sústavy môže byť opísaná algoritmom 5. Hľadané riešenie $\mathbf{x}$ sústavy po skončení algoritmu nájdeme v poli $\mathbf{b}$.

LDU metóda s ukladaním profilu. Matica v profilovom formáte je reprezentovaná dvomi poľami: reálne pole a, obsahujúce prvky matice a celočíselné pole smerníkov $pcol$. Predstavíme si algoritmus pre maticu A_{ij}. Preto potrebujeme

Algoritmus 4 LDU faktorizácia

$h1 = h - 1$
for $j = 2, n$ **do**
  for $i = max(j - h1, 1), j - 1$ **do**
    for $k = max(j - h1, 1), i - 1$ **do**
      $A_{ij} = A_{ij} - A_{ki}A_{kj}$
    end for
  end for
  for $i = max(j - h1, 1), j - 1$ **do**
    $w = A_{ij}$
    $A_{ij} = A_{ij}/A_{ii}$
    $A_{jj} = A_{ij} - wA_{ij}$
  end for
end for

Algoritmus 5 Dopredná redukcia a spätná substitúcia

for $j = 2, n$ **do**
  for $i = max(j - h1, 1), j - 1$ **do**
    $b_j = b_j - A_{ij}b_i$
  end for
end for
for $j = 1, n$ **do**
  $b_j = b_j/A_{jj}$
end for
for $j = n, 1, -1$ **do**
  for $i = max(j - h1, 1), j - 1$ **do**
    $b_i = b_i - A_{ij}b_j$
  end for
end for

mať vzťahy medzi dvojindexovým označením pevne danej matice A_{ij} a poľom a používaným v počítačových kódoch. Umiestnenie prvého nenulového prvku v i-tom stĺpci matice je dané nasledujúcou funkciou:

$$FN(i) = i - (pcol[i+1] - pcol[i] + 1).$$

Na prechod medzi dvojindexovým zápisom a zápisom jednorozmerného poľa môžeme použiť nasledujúci vzťah:

$$A_{ij} \to a[i + pcol[j+1] - j].$$

Pseudokód faktorizácie matice so symetrickým profilom s pozeraním doľava je uvedený v algoritme 6 a metódy doprednej redukcie a spätnej substitúcie sú uvedené v algoritme 7.

Algoritmus 6 LDU faktorizácia

```
for j = 2, n do
  for i = FN(j), j − 1 do
    for k = max(FN(i), FN(j)), i − 1 do
      A_ij = A_ij − A_ki A_kj
    end for
  end for
  for i = FN(j), j − 1 do
    w = A_ij
    A_ij = A_ij / A_ii
    A_jj = A_jj − w A_ij
  end for
end for
```

Algoritmus 7 Dopredná redukcia a spätná substitúcia

```
for j = 2, n do
  for i = FN(j), j − 1 do
    b_j = b_j − A_ij b_i
  end for
end for
for j = 1, n do
  b_j = b_j / A_jj
end for
for i = n, 1, −1 do
  for i = FN(j), j − 1 do
    b_i = b_i − A_ij b_j
  end for
end for
```

5.5 ITERAČNÉ METÓDY

5.5.1 Gaussova-Seidelova metóda

Nech je daný systém $\mathbf{Ax} = \mathbf{b}$ s n neznámymi. Pred začiatkom iteračného procesu zvolíme

$$x_i^{(0)} = b_i/a_{ii}$$

ako začiatočnú aproximáciu riešenia. Ak $x_i^{(k)}$ je hodnota neznámej x_i po k iteráciách, aproximácia x_i v nasledujúcom kroku je

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

V každom iteračnom kroku spočítame diferencie $r_i = x_i^{(k+1)} - x_i^{(k)}$ a iteračný proces ukončíme keď je diferenčná chyba malá $max|r_i| < \epsilon$. Niekedy môže byť rýchlosť iteračného procesu zvýšená použitím super relaxačnej metódy (SOR), ktorá sa od Gaussova-Seidelovej iterácie líši len vynásobením diferencie $x_i^{(k+1)} - x_i^{(k)}$ relaxačným faktorom ω, ktorý sa volí z intervalu $1 < \omega < 2$, tak dostaneme

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

$$x_i^{(k+1)} = x_i^{(k)} + \omega \left(y_i^{(k+1)} - x_i^{(k)} \right).$$

5.5.2 Konjugovaná gradientná metóda

Jednoduchá a účinná iteračná metóda široko používaná na riešenie riedkych sústav je konjugovaná gradientná (PCG) metóda. V mnohých prípadoch je v praxi priebeh konvergencie tejto metódy príliš pomalý. Zrýchlený môže byť použitím začiatočných podmienok v sústave rovníc:

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b},$$

kde $\mathbf{M}^{-1}$ je matica predpokladov ktorá v istom zmysle aproximuje $\mathbf{A}^{-1}$. Najjednoduchšie začiatočné podmienky sú diagonálne, kde $\mathbf{M}$ obsahuje len diagonálne prvky matice $\mathbf{A}$. Túto metódu popisujeme ako rad krokov v algoritme 8, nazvanom PCG algoritmus. V uvedenom PCG algoritme sa matica $\mathbf{A}$ počas výpočtov nemení. To znamená, že nepridávame počas výpočtu žiadne nenulové prvky do matice $\mathbf{A}$. Vďaka tomu je vhodný formát ukladania

Algoritmus 8 PCG algoritmus

Vypočítaj $[M]$
$\{x_0\} = 0$
$\{r_0\} = \{b\}$
for $i = 0, 1...$ **do**
  $\{w_i\} = [M]^{-1}\{r_i\}$
  $\gamma_i = \{r_i\}^T\{w_i\}$
  if $i = 0$ **then**
    $\{p_i\} = \{w_i\}$
  else
    $\{p_i\} = \{w_i\} + (\gamma_i/\gamma_{i-1})\{p_{i-1}\}$
  end if
  $\{w_i\} = [A]\{p_i\}$
  $\beta_i = \{p_i\}^T\{w_i\}$
  $\{x_i\} = \{x_{i-1}\} + (\gamma_i/\beta_i)\{p_i\}$
  $\{r_i\} = \{r_{i-1}\} - (\gamma_i/\beta_i)\{p_i\}$
  if $\gamma_i/\gamma_0 < \epsilon$ **then**
    exit
  end if
end for

Algoritmus 9 Násobenie riedkych matíc a vektora uchovaním iba nenulových prvkov

for $j = 1, N$ **do**
  $w[j] = 0$
  for $i = prow[j], prow[j+1] - 1$ **do**
    $w[j] = w[j] + A[i] * p[col[i]]$
  end for
end for

matice pre PCG iteračné metódy založený na rozumnom ukladaní riedkych matíc. Pseudokód uvedený v algoritme 9 vysvetľuje násobenie matice a vektora $\mathbf{w} = \mathbf{Ap}$, kde matica $\mathbf{A}$ uchováva iba nenulové prvky.

Príklad 18. *Ukladanie riedkych riadkov. Zapíšte štruktúru globálnej matice* $\mathbf{K}$ *tuhosti použitím rozumného ukladania riedkych riadkov. Nenulové prvky v matici sú označené* 1*.*

$$\mathbf{K} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Riešenie. Štruktúra riedkej matice podľa riadkov pozostáva z dvoch polí: $col[pcol[N+1]]$ obsahuje čísla stĺpcov s nenulovými prvkami matice; pole $pcol[N+1]$ obsahuje smerníky do pola $col[]$ na miesto, kde za nachádza prvý nenulový prvok každého riadku:

$col[\,] = 1,2,4,5,\ 1,2,3,4,5,6,\ 2,3,5,6,\ 1,2,4,5,\ 1,2,3,4,5,6,7,8,$
$2,3,5,6,7,8,\ \ 5,6,7,8,\ \ 5,6,7,8$
$pcol[\,] = 0,\ 4,\ 10,\ 14,\ 18,\ 26,\ 32,\ 36,\ 40$

□

Napríklad $pcol[0] = 0$ a tak na mieste $col[0] = 1$ sa nachádza prvý nenulový prvok prvého riadku. Podobne $pcol[4] = 18$, tak na mieste $col[18] = 1$ sa nachádza prvý nenulový prvok piateho riadku, ktorý má nenulové prvky na miestach $1,2,3,4,5,6,7,8,$.

Príklad 19. *Vytvorme maticu pozostávajúcu iba z niekoľkých nenulových prvkov v programe Mathematica [15]*

$$\mathbf{S} = \begin{bmatrix} 3 & 0 & 11 \\ 0 & 5 & 0 \\ 0 & 0 & 7 \end{bmatrix}.$$

Riešenie. Použijeme príkaz `SparseArray`

```
S = SparseArray[{{1, 1} -> 3, {2, 2} -> 5, {3, 3} -> 7,
  {1, 3} -> 11}]
MatrixForm[S]
```

$$\begin{pmatrix} 3 & 0 & 11 \\ 0 & 5 & 0 \\ 0 & 0 & 7 \end{pmatrix}$$

□

Na záver si pripomeňme, že s riedkymi maticami sa v programe *Mathematica* pracuje ako normálnymi maticami. Pri operáciách je zabezpečená optimalizácia pamäte.

`SparseArray[Subscript[pos, 1]->Subscript[val, 1],Subscript[pos, 2]->Subscript[val, 2],...]` slúži na zadanie riedkeho vektora s hodnotami val_i na pozicii pos_i.
`DiagonalMatrix[list,k]` slúži na zadanie pásovej matice s prvkami v zozname `list` do k-tej diagonály matice.

5.6 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Zapíšte štruktúru blokovej matice $\mathbf{A}$ použitím rozumného ukladania riedkych riadkov. Nenulové prvky v matici sú označené 1.

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

2 Riešte sústavu rovníc $\mathbf{Ax} = \mathbf{0}$ s maticou $\mathbf{A}$ z predchádzajúceho príkladu. Pri riešení použite $\mathbf{LU}$ dekompozíciu.

3 Zapíšte štruktúru päťdiagonálnej matice $\mathbf{A}$ použitím rozumného ukladania diagonálnych prvkov. Prvky na prvej diagonále označme a, na druhej diagonále b a na tretej diagonále c.

Kapitola 6

INVERZNÁ MATICA A VLASTNÉ ČÍSLA MATICE

V tejto kapitole sa budeme venovať výpočtu determinantu matice, výpočtu inverznej matice, aproximácii vlastných čísel a vlastných vektorov matice $\mathbf{A} \in \mathcal{R}^{n \times n}$. Uvádzame len tvrdenia, detaily a dôkazy nájdete v [1]. Ďalšie výsledky o vlastných číslach sa možno dočítať v ([14] str. 36–49), [6].

6.1 DETERMINANT MATICE

Štvorcová matica $\mathbf{A}$ rozmerov $n \times n$ má determinant, ktorý je daný nasledujúcim rekurzívnym vzťahom:

$$|\mathbf{A}| = a_{11}M_{11} - a_{12}M_{12} + a_{13}M_{13} - \ldots - (-1)^n a_{1n}M_{1n},$$

kde M_{ij} je determinat vytvorený z pôvodnej matice vynechaním i-teho riadku a j-teho stĺpca. Z čoho vidieť, že determinant 1×1 matice je rovný práve jej

jedinému prvku, ďalej dostávame

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

a

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + \\ + a_{13}(a_{21}a_{32} - a_{22}a_{31}).$$

Preusporiadaním riadkov v matici je možné odvodiť determinant nielen pomocou prvého riadku, ale i pomocou hociktorého iného riadku, či stĺpca. Z tohto dostávame, že determinant je rovný nule, ak nejaký riadok alebo stĺpec matice tvoria samé nuly. Tiež sa dá ukázať, že determinant je nulový, ak aspoň dva riadky alebo stĺpce matice sú lineárne závislé ([16] str. 76–79).

Pre veľké matice je priamy výpočet determinantu časovo veľmi náročný. Našťastie determinant matice môže byť vypočítaný z jej LU rozkladu. Táto možnosť vyplýva z nasledujúcich vlastností determinantov:

1. Determinant súčinu dvoch matíc sa rovná súčinu ich determinantov.
2. Determinant trojuholníkovej matice je súčinom prvkov na hlavnej diagonále.

Ak matica $\mathbf{A}$ má LU rozklad

$$\mathbf{A} = \begin{bmatrix} m_{11} & 0 & 0 & 0 \\ m_{21} & m_{22} & 0 & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

potom determinant môžeme vypočítať vzťahom $|A| = m_{12}m_{22}\dots m_{nn}$.

Príklad 20. *V programe Mathematica [15] vypočítajte determinanty*

$$|\mathbf{A}| = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} \qquad |\mathbf{B}| = \begin{vmatrix} b & c & 0 & 0 & 0 \\ a & b & c & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & 0 & a & b & c \\ 0 & 0 & 0 & a & b \end{vmatrix}.$$

Riešenie. Zadajme maticu $\mathbf{A}$ a na výpočet determinantu použijeme príkaz `Det[%]`.

```
Det[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}];
```

0

Vypočítaná hodnota determinantu je $|\mathbf{A}| = 0$.

Matica $\mathbf{B}$ je riedka preto je efektívnejšie zadávať jednotlivé nenulové prvky alebo ako riedku maticu s troma diagonálami, na ktorých sú prvky a, b a c. V programe *Mathematica* možeme zadať trojdiagonálnu maticu $n \times n$ príkazom `SparseArray[ ]`. Determinant trojdiagonálnej matice rátame nasledujúcimi príkazmi:

```
tridiagonal[n_] :=
SparseArray[{Band[{2, 1}] -> a, Band[{1, 1}] -> b,
  Band[{1, 2}] -> c}, {n, n}]
B = tridiagonal[5];
Det[B]
```

$b^5 - 4ab^3c + 3a^2bc^2$

Vypočítaná hodnota determinantu je $|\mathbf{B}| = b^5 - 4ab^3c + 3a^2bc^2$. □

6.2 INVERZNÁ MATICA

Inverznou maticou $\mathbf{A}^{-1}$ k štvorcovej regulárnej (ak $|\mathbf{A}| \neq 0$) matici $\mathbf{A}$ je štvorcová matica, pre ktorú platí

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I},$$

kde $\mathbf{I}$ je jednotková matica. Maticu $\mathbf{A}$ nazveme *ortogonálnou* ak $\mathbf{A}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{A} = \mathbf{I}$ teda $\mathbf{A}^{-1} = \mathbf{A}^\top$.

Pre určenie koeficientov inverznej matice je možné použiť rovnicu $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Na základe tohto vzťahu možno inverznú maticu nájsť pomocou riešenia n sústav rovníc

$$\mathbf{A}x_i = e_i,$$

kde e_i predstavuje i-ty stĺpec jednotkovej matice $\mathbf{I}$ a x_i je i-ty stĺpec hľadanej inverznej matice. Po vypočítaní riešení týchto rovníc, inverznú maticu možno zložiť z n stĺpcov x_i.

Výpočet inverznej matice je užitočný algebraický koncept, ale je časovo náročná operácia a v praxi sa používa iba pre matice malých rozmerov (2-4). V ostatných prípadoch sa snažíme preformulovať problém tak, aby sme používali riešenie systému lineárnych rovníc namiesto výpočtu inverznej matice.

Príklad 21. *V programe Mathematica vypočítajte inverznú maticu k maticiam*

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} b & c & 0 & 0 \\ a & b & c & 0 \\ 0 & a & b & c \\ 0 & 0 & a & b \end{pmatrix}.$$

Riešenie. Príkaz `Inverse[%]` použijeme na výpočet inverznej matice k matici **A**:

```
Inverse[{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}];
```

Inverse::sing: Matrix ... is singular.

Mathematica detekovala, že matica **A** je singulárna lebo $|\mathbf{A}| = 0$ a preto neexistuje jej inverzná matica.

Riedku maticu **B** s rozmermi 4×4 a troma diagonálami, na ktorých sú prvky a, b a c zadáme už spomenutou funkciou `tridiagonal[4]`. Inverznú maticu k riedkej matici vypočítame a vypíšeme v maticovom tvare programom *Mathematica* príkazom `MatrixForm[Inverse[B]]`:

```
tridiagonal[n_] :=
SparseArray[{Band[{2, 1}] -> a, Band[{1, 1}] -> b,
  Band[{1, 2}] -> c}, {n, n}]
B = tridiagonal[4];
MatrixForm[Inverse[B]]
```

$$\begin{pmatrix} \frac{b^3-2abc}{b^4-3ab^2c+a^2c^2} & \frac{-b^2c+ac^2}{b^4-3ab^2c+a^2c^2} & \frac{bc^2}{b^4-3ab^2c+a^2c^2} & -\frac{c^3}{b^4-3ab^2c+a^2c^2} \\ \frac{-ab^2+a^2c}{b^4-3ab^2c+a^2c^2} & \frac{b^3-abc}{b^4-3ab^2c+a^2c^2} & -\frac{b^2c}{b^4-3ab^2c+a^2c^2} & \frac{bc^2}{b^4-3ab^2c+a^2c^2} \\ \frac{a^2b}{b^4-3ab^2c+a^2c^2} & -\frac{ab^2}{b^4-3ab^2c+a^2c^2} & \frac{b^3-abc}{b^4-3ab^2c+a^2c^2} & \frac{-b^2c+ac^2}{b^4-3ab^2c+a^2c^2} \\ -\frac{a^3}{b^4-3ab^2c+a^2c^2} & \frac{a^2b}{b^4-3ab^2c+a^2c^2} & \frac{-ab^2+a^2c}{b^4-3ab^2c+a^2c^2} & \frac{b^3-2abc}{b^4-3ab^2c+a^2c^2} \end{pmatrix}$$

Uvedená matica je $\mathbf{B}^{-1}$. Všimnite si, že menovateľ prvkov $\mathbf{B}^{-1}$ je rovnaký a rovná sa determinantu $|\mathbf{B}| = b^4 - 3ab^2c + a^2c^2$, o ktorom predpokladáme, že je rôzny od nuly. □

6.3 VLASTNÉ ČÍSLA MATICE

Pre danú štvorcovú maticu **A** rozmerov $n \times n$ nazveme číslo $\lambda \in \mathcal{C}$ vlastným číslom matice **A**, ak existuje nenulový vektor $\mathbf{q} \in \mathcal{C}^n$ taký, že

$$\mathbf{A}\mathbf{q} = \lambda\mathbf{q}.$$

Vektor $\mathbf{q}$ nazývame vlastným vektorom pridruženým k vlastnému číslu λ [6]. Predchádzajúci vzťah sa dá upraviť do tvaru

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{q} = 0$$

a teda každé netriviálne (nenulové) riešenie $\mathbf{q}$ musí spĺňať charakteristickú rovnicu

$$\begin{vmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{vmatrix} = 0.$$

Vyriešením charakteristickej rovnice dostaneme vlastné čísla matice. Charakteristická rovnica je polynómom stupňa n s koeficientmi c_i a možno ju napísať v tvare [6]

$$\lambda^n + c_{n-1}\lambda^{n-1} + \ldots + c_1\lambda + c_0 = 0.$$

Korene charakteristického polynómu sú vlastné čísla matice preto existuje n vlastných čísel matice $\mathbf{A}$, nie nutne rôznych. Charakteristický polynóm môže mať komplexné korene a teda aj vlastné čísla matice môžu byť komplexné preto aj odpovedajúci vlastný vektor bude tiež komplexný.

Príklad 22. *Vlastné čísla matice. Uvažujme maticu*

$$\mathbf{A} = \begin{bmatrix} 16 & -24 & 18 \\ 3 & -2 & 0 \\ -9 & 18 & -17 \end{bmatrix}.$$

Ukážte, že má vlastné číslo 4*, zodpovedajúci vlastný vektor* $\begin{Bmatrix} 2 \\ 1 \\ 0 \end{Bmatrix}$*. Nájdite ďalšie vlastné čísla.*

Riešenie. Z definície vlastného čísla

$$\begin{bmatrix} 16 & -24 & 18 \\ 3 & -2 & 0 \\ -9 & 18 & -17 \end{bmatrix} \begin{Bmatrix} 2 \\ 1 \\ 0 \end{Bmatrix} = 4 \begin{Bmatrix} 2 \\ 1 \\ 0 \end{Bmatrix}$$

z čoho vyplýva, že má vlastné číslo 4, a zodpovedajúci vlastný vektor $\begin{Bmatrix} 2 \\ 1 \\ 0 \end{Bmatrix}$.

Charakteristickú rovnicu

$$\begin{vmatrix} 16 - \lambda & -24 & 18 \\ 3 & -2 - \lambda & 0 \\ -9 & 18 & -17 - \lambda \end{vmatrix} = 0$$

po vypočítaní determinantu možno zapísať v polynomickom tvare

$$\lambda^3 + 3\lambda^2 - 36\lambda + 32 = 0.$$

Túto rovnicu možno ďalej upraviť do tvaru $(\lambda-4)(\lambda-1)(\lambda+8) = 0$, z ktorého dostávame, že vlastnými číslami sú $\lambda_1 = 4$, $\lambda_2 = 1$ a $\lambda_3 = -8$. Ukážme si ako možno vypočítať charakteristický polynóm matice v programe *Mathematica* pomocou funkcie `CharacteristicPolynomial[%]`

```
A = {{16, -24, 18}, {3, -2, 0}, {-9, 18, -17}} ;
CharacteristicPolynomial[A, x]
```

$-32 + 36x - 3x^2 - x^3$

Našli sme charakteristický polynóm s neznámou x

$$-32 + 36x - 3x^2 - x^3 = 0,$$

čo je ten istý polynóm ako sme vyrátali vyššie. Ďalej postupujeme výpočtom determinantu $|\mathbf{A}-\lambda\mathbf{I}|$ podľa definície, pričom jednotkovú maticu 3×3 zadáme príkazom `IdentityMatrix[3]`:

```
A = {{16, -24, 18}, {3, -2, 0}, {-9, 18, -17}} ;
Det[A - x IdentityMatrix[3]]
```

$-32 + 36x - 3x^2 - x^3$

Hľadajme teraz korene charakteristickej rovnice príkazom `Solve[%]`:

```
A = {{16, -24, 18}, {3, -2, 0}, {-9, 18, -17}} ;
Solve[CharacteristicPolynomial[A, x] == 0, x]
```

$\{\{x \to -8\}, \{x \to 1\}, \{x \to 4\}\}$

Našli sme korene a sú to hľadané vlastné čísla -8, 4, 1 matice $\mathbf{A}$.

Príkaz `Eigenvalues[%]` v programe *Mathematica* nám umožňuje výpočet vlastných čísel a príkaz `Eigenvectors[%]` nám umožňuje výpočet vlastných vektorov matice.

```
A = {{16, -24, 18}, {3, -2, 0}, {-9, 18, -17}} ;
Eigenvalues[A]
Eigenvectors[A]
```

$\{-8, 4, 1\}$
$\{\{-2, 1, 4\}, \{2, 1, 0\}, \{2, 2, 1\}\}$

Našli sme tri vlastné čísla -8, 4, 1 a k nim odpovedajúce vlastné vektory $\begin{Bmatrix} -2 \\ 1 \\ 4 \end{Bmatrix}, \begin{Bmatrix} 2 \\ 1 \\ 0 \end{Bmatrix}, \begin{Bmatrix} 2 \\ 2 \\ 1 \end{Bmatrix}$. □

Metóda charakteristickej rovnice nie je efektívna pre numerický výpočet vlastných čísel. Počet násobení potrebných na získanie koeficientov charakteristickej rovnice je rádovo n^2 pre maticu s rozmermi $n \times n$. Preto sa odporúča použiť inú metódu ([7] kapitola 5, [14] str. 36–49).

6.3.1 Niektoré vlastnosti vlastných čísel

Niektoré z vlastností vlastných čísel matíc [1]:

1. Charakteristická rovnica môže byť upravená do tvaru

   $$(\lambda - \lambda_1)(\lambda - \lambda_2) \dots (\lambda - \lambda_n) = 0$$

   čím sa ukazuje, že matica rádu n má n vlastných čísel, pričom niektoré vlastné čísla môžu byť rovnaké.

2. Súčet vlastných čísel je rovný súčtu diagonálnych prvkov $tr(\mathbf{A})$ matice $\mathbf{A}$

   $$tr(\mathbf{A}) = a_{11} + a_{22} + \ldots + a_{nn} = \lambda_1 + \lambda_2 + \ldots \lambda_n.$$

3. Súčin vlastných čísel matice je rovný jej determinatu

   $$|\mathbf{A}| = \lambda_1 \lambda_2 \dots \lambda_n.$$

4. Kedže determinant matice je rovný determinantu matice k nej transponovanej, platí že matica má rovnaké vlastné čísla ako matica k nej transponovaná. Vo všeobecnosti nemá tie isté vlastné vektory.

5. Platí, že determinant trojuholníkovej matice je rovný súčinu jej diagonálnych prvkov. A preto, ak matica $\mathbf{A}$ je trojuholníková, tak

   $$|\mathbf{A} - \lambda \mathbf{I}| = (a_{11} - \lambda)(a_{22} - \lambda) \dots (a_{nn} - \lambda) = 0.$$

   Teda vlastné čísla trojuholníkovej a tiež diagonálnej matice sú rovné diagonálnym prvkom matice.

6.4 VLASTNÉ ČÍSLA A VEKTORY SYMETRICKEJ MATICE

Hľadanie vlastných čísel symetrickej matice je ľahšie, pretože všetky vlastné čísla sú reálne. Vieme transformovať maticu na inú maticu, ktorá bude mať tie isté vlastné čísla. Maticu budeme transformovať pokým nebude matica v tvare, v ktorom môže byť ľahšie analyzovaná inými metódami. Všeobecnou transformáciou, ktorá zachováva vlastné čísla je *podobnostná transformácia* matice $\mathbf{A}$ daná vzťahom

$$\bar{\mathbf{A}} = \mathbf{N}^{-1}\mathbf{A}\mathbf{N},$$

kde $\mathbf{N}$ môže byť ľubovoľná nesingulárna matica rovnakého rádu. Ak $\mathbf{N}$ je ortogonálna, potom $\mathbf{N}^{-1} = \mathbf{N}^{\top}$ a podobnostnú transformáciu možno zapísať ako

$$\bar{\mathbf{A}} = \mathbf{N}^{\top}\mathbf{A}\mathbf{N}.$$

Ak matica $\mathbf{A}$ je symetrická, potom i matica $\bar{\mathbf{A}}$ je symetrická. Teda použitie ortogonálnej transformácie zabezpečí zachovanie symetrie matice.

Ak symetrická matica $\mathbf{A}$ má vlastné čísla usporiadané v diagonálnom maticovom zápise $\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_n \end{bmatrix}$ a zodpovedajúce vlastné vektory sú usporiadané v matici $\mathbf{Q} = \begin{bmatrix} q_1 & q_2 & \dots & q_n \end{bmatrix}$, pričom $\mathbf{A}\mathbf{Q} = \mathbf{Q}\mathbf{\Lambda}$, potom platí

$$\bar{\mathbf{A}}(\mathbf{N}^{\top}\mathbf{Q}) = \mathbf{N}^{\top}\mathbf{A}\mathbf{N}\mathbf{N}^{\top}\mathbf{Q} = \mathbf{N}^{\top}\mathbf{A}\mathbf{Q} = (\mathbf{N}^{\top}\mathbf{Q})\mathbf{\Lambda}.$$

Teda vlastné čísla matice $\bar{\mathbf{A}}$ sú rovnaké ako vlastné čísla matice $\mathbf{A}$ a matica vlastných vektorov matice $\mathbf{A}$ je rovná $\bar{\mathbf{Q}} = \mathbf{N}^{\top}\mathbf{Q}$.

Symetrickú maticu možno transformovať (napr. Jacobiho diagonalizáciou), do tvaru, že všetky jej nediagonálne prvky sú nuly, potom diagonálne prvky po transformácii sú vlastnými číslami matice.

6.4.1 Jacobiho diagonalizácia

Každá Jacobiho transformácia eliminuje jeden pár nediagonálnych prvkov v symetrickej matici ([14] str. 52-53). Na elimináciu páru prvkov a_{pq} a a_{qp} sa používa ortogonálna transformačná matica $\mathbf{N}$ s nasledujúcimi vlastnosťami

$$\begin{aligned} n_{pp} &= n_{qq} = cos\,\alpha \\ n_{pq} &= sin\,\alpha \\ n_{qp} &= -sin\,\alpha \\ n_{ii} &= 1 \quad i \neq p, q \\ n_{ij} &= 0 \quad i \neq j \;\; i, j \neq p, q. \end{aligned}$$

α volíme tak, aby po transformácií $\bar{\mathbf{A}} = \mathbf{N}^{\top}\mathbf{A}\mathbf{N}$ platilo, že $\bar{a}_{pq} = \bar{a}_{qp} = 0$. Z podmienky odvodíme vzťah pre α

$$\bar{a}_{pq} = (-a_{pp} + a_{qq})cos\,\alpha\, sin\,\alpha + a_{pq}(cos^2\,\alpha - sin^2\,\alpha) = 0,$$

teda

$$tan2\alpha = \frac{2a_{pq}}{a_{pp} - a_{qq}}.$$

Jacobiho transformácia mení iba riadky p a q a stĺpce p a q. Zmenené dva prvky na diagonále budú mať po transformácii tvar

$$\begin{aligned} \bar{a}_{pp} &= a_{pp}\cos^2\alpha + a_{qq}\sin^2\alpha + 2a_{pq}\sin\alpha\cos\alpha \\ \bar{a}_{qq} &= a_{pp}\sin^2\alpha + a_{qq}\cos^2\alpha - 2a_{pq}\sin\alpha\cos\alpha. \end{aligned} \tag{6.1}$$

Ostatné prvky transformácie sú dané vzťahmi

$$\begin{aligned} \bar{a}_{ip} &= \bar{a}_{pi} = a_{ip}\,cos\,\alpha + a_{iq}\,sin\,\alpha \\ \bar{a}_{iq} &= \bar{a}_{qi} = -a_{ip}\,sin\,\alpha + a_{iq}\,cos\,\alpha. \end{aligned}$$

Výpočtová procedúra vykonáva sériu Jacobiho transformácií, pričom s každou transformáciou anuluje mimo diagonálny prvok s najväčšou absolútnou hodnotou. Prvky, ktoré boli anulované, nezostanú stále nulami, a preto má Jacobiho metóda iteratívny charakter.

Ak maticu ktorá vznikne po $k-1$ iteráciách označíme $\mathbf{A}^{(k)}$, potom k-tu transformáciu môžeme zapísať ako

$$\mathbf{A}^{(k+1)} = \mathbf{N}_k^\top \mathbf{A}^{(k)} \mathbf{N}_k$$

a matice vlastných vektorov matíc $\mathbf{A}^{(k)}$ a $\mathbf{A}^{(k+1)}$ spĺňajú vzťahy

$$\mathbf{A}^{(k)} = \mathbf{N}_{k-1}^\top \mathbf{Q}^{(k-1)}, \qquad \mathbf{A}^{(k+1)} = \mathbf{N}_k^\top \mathbf{Q}^{(k)}.$$

Ak bolo potrebných s transformácií $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_s$ na diagonalizáciu matice $\mathbf{A}$, potom vlastné vektory matice $\mathbf{A}$ tvoria stĺpce matice [1]

$$\mathbf{Q} = \mathbf{N}_1 \mathbf{N}_2 \ldots \mathbf{N}_s.$$

Vlastné čísla matice sú diagonálnymi prvkami matice $\mathbf{A}^{(s+1)}$, ktorá bola získaná po s transformáciách. V praxi sa postupnosť transformácií ukončí, keď najväčší nediagonálny prvok matice $\mathbf{A}^{(k)}$ je menší než stanovená prahová hodnota, $|a_{pq}| < \varepsilon$.

Algoritmus Jacobiho diagonalizácie môže byť realizovaný v jednotlivých krokoch, ktoré sú uvedené v algoritme 10.

Možné spôsoby výpočtu determinantu, hľadanie vlastných čísel a vlastných vektorov matice v programe *Mathematica* sú nasledujúce:

Algoritmus 10 Jacobiho algoritmus

Inicializuj maticu **A** prvkami matice pre ktorú hľadáme vlastné čísla
Inicializuj maticu **Q** prvkami jednotkovej matice **I**
while $|a_{pq}| > \varepsilon$ **do**
  Nájdi najväčší nediagonálny prvok a_{pg} matice **A**
  Vytvor transformáciu **N**, ktorá eliminuje prvok a_{pq}
  $\mathbf{A}_1 = \mathbf{N}^\top \mathbf{A}\mathbf{N}$
  $\mathbf{A} = \mathbf{A}_1$, $\mathbf{Q} = \mathbf{Q}\mathbf{N}$
end while
Vlastné čísla sú diagonálnymi prvkami matice **A**. Vlastné vektory sú stĺpcami matice **Q**.

`Det[A]` Nájde determinant matice **A**.
`Inverse[A]` Nájde inverznú maticu k matici **A**.
`tridiagonal[n]` Generuje trojdiagonálnu maticu s diagonálnymi prvkami b a mimo diagonálnymi prvkami a a c o rozmeroch $n \times n$.
`CharacteristicPolynomial[A, x]` Nájde charakteristický polygón premennej x matice **A**.
`Eigenvalues[A]` Nájde vlastné čísla matice **A**.
`Eigenvectors[A]` Nájde vlastné vektory matice **A** usporiadané v poradí vlastných čísel.

6.5 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Vypočítajte:

a) $\begin{vmatrix} a & b & p & q \\ c & d & r & s \\ 0 & 0 & e & f \\ 0 & 0 & g & h \end{vmatrix}$; b) $\begin{vmatrix} 1 & 1 & 1 \\ a^2 & b^2 & c^2 \\ a^3 & b^3 & c^4 \end{vmatrix}$; c) $\begin{vmatrix} a & b & b & b \\ b & a & b & b \\ b & b & a & b \\ b & b & b & a \end{vmatrix}$;

d) $\begin{vmatrix} a_1 + b_1 t & a_2 + b_2 t & a_3 + b3t \\ a^1 t + b_1 & a_2 t + b_2 & a_3 t + b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$; e) $\begin{vmatrix} 1 & -2 & 3 & 1 \\ 5 & -9 & 6 & 3 \\ -1 & 2 & -6 & -2 \\ 2 & 8 & 6 & 1 \end{vmatrix}$.

2 Vypočítajte: $\begin{vmatrix} b & c & 0 & \cdots & 0 \\ a & b & c & \cdots & 0 \\ 0 & a & b & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a & b & c \\ 0 & \cdots & 0 & a & b \end{vmatrix}$ s rozmermi 12×12.

3 Zvoľte si ľubovoľné reálne konštanty a_i, b_i, $c_i \in R$, $i = \{1, 2, 3\}$ a riešte sústavu:

$$\begin{aligned} a_1x + b_1y + c_1z &= 0, \\ a_2x + b_2y + c_2z &= 0, \\ a_3x + b_3y + c_3z &= 0. \end{aligned}$$

Nech $\mathbf{A}$ je matica systému 3×3. Vypočítajte súčiny $x\,|\mathbf{A}|$, $y\,|\mathbf{A}|$ a $z\,|\mathbf{A}|$.

4 Zvoľte si maticu $\mathbf{A}$ a celé číslo m tak, že m-tá mocnina matice, $\mathbf{A}^m$ je nulová matica. Vypočítajte vlastné čísla matice.

5 Vypočítajte vlastné čísla a vlastné vektory matíc

$$\text{a)} \begin{pmatrix} 3 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix}; \quad \text{b)} \begin{pmatrix} 4 & -5 & 7 \\ 1 & -4 & 9 \\ -4 & 0 & 5 \end{pmatrix}.$$

6 Zvoľte si maticu $\mathbf{A}$ tak, že budete vedieť nájsť jej inverznú maticu $\mathbf{A}^{-1}$ a štvorcovú maticu $\mathbf{B}$ s rovnakým rozmerom ako $\mathbf{A}$. Vypočítajte determinant $\left|\mathbf{A}^{-1}\mathbf{B}\mathbf{A}\right|$.

7 Pre vhodné k nájdite inverzné matice k maticiam:

$$\text{a)} \begin{pmatrix} 2 & 5 & 5 \\ -1 & -1 & 0 \\ 2 & 4 & 3 \end{pmatrix}; \quad \text{b)} \begin{pmatrix} 1 & 2 & 4 \\ 3 & 1 & 6 \\ k & 3 & 2 \end{pmatrix}; \quad \text{c)} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

8 Nájdite vlastné čísla a zodpovedajúce vlastné vektory matíc:

$$\text{a)} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad \text{b)} \begin{pmatrix} a & b & b & b \\ b & a & b & b \\ b & b & a & b \\ b & b & b & a \end{pmatrix}.$$

6.6 VÝSLEDKY

1 a) $(ad-bc)(eh-gf)$; b) $(b-c)(c-a)(a-b)(bc+ca+ab)$; c) $(a-b)^3(a+3b)$; d) $(1-t^2)\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$; e) 39.

2 $b^{12} - 11ab^{10}c + 45a^2b^8c^2 - 84a^3b^6c^3 + 70a^4b^4c^4 - 21a^5b^2c^5 + a^6c^6$.

3 $x\,|\mathbf{A}| = y\,|\mathbf{A}| = z\,|\mathbf{A}| = 0$.

4 Matica $\mathbf{A}^m$ má iba jediné vlastné číslo a to je nula.

5 a) Vlastné čísla 4, 2, 1, vlastné vektory $\{2,1,1\}$,$\{0,-1,1\}$,$\{-1,1,1\}$; b) Matica má jedno reálne a dve komplexné vlastné čísla 1, $2\pm 3i$, vlastné vektory $\{1,2,1\}$,$\left\{\frac{3}{4}-\frac{3i}{4},\frac{5}{4}-\frac{3i}{4},1\right\}$,$\left\{\frac{3}{4}+\frac{3i}{4},\frac{5}{4}+\frac{3i}{4},1\right\}$.

6 Determinant $\left|\mathbf{A}^{-1}\mathbf{B}\mathbf{A}\right| = |\mathbf{B}|$.

7 a) $\begin{pmatrix} 3 & -5 & -5 \\ -3 & 4 & 5 \\ 2 & -2 & -3 \end{pmatrix}$; b) $\begin{pmatrix} \frac{-16}{8+8k} & \frac{8}{8+8k} & \frac{8}{8+8k} \\ \frac{-6+6k}{8+8k} & \frac{2-4k}{8+8k} & \frac{6}{8+8k} \\ \frac{9-k}{8+8k} & \frac{-3+2k}{8+8k} & -\frac{5}{8+8k} \end{pmatrix}$ pre $k \neq -1$; c) $\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

8 a) Vlastné čísla $1, \cos(\theta)-\sqrt{-1+\cos(\theta)^2}, \cos(\theta)+\sqrt{-1+\cos(\theta)^2}$, vlastné vektory $\{0,0,1\}$,$\left\{\sqrt{-1+\cos(\theta)^2}\csc(\theta),1,0\right\}$, $\left\{-\sqrt{-1+\cos(\theta)^2}\csc(\theta),1,0\right\}$; b) Vlastné čísla $a-b$, $a-b$, $a-b$, $a+3b$, vlastné vektory $\{-1,0,0,1\}$, $\{-1,0,1,0\}$, $\{-1,1,0,0\}$, $\{1,1,1,1\}$.

Kapitola 7

RIEŠENIE NELINEÁRNYCH ROVNÍC

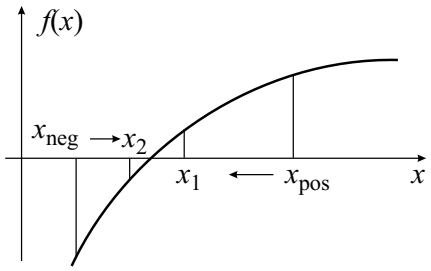
V tejto kapitole sa budeme venovať hľadaniu nulových bodov reálnej funkcie jednej premennej $f : (a, b) \subseteq \mathcal{R} \to \mathcal{R}$. Hľadáme $z \in (a, b)$ také, že $f(z) = 0$.

Problém hľadania bodov, v ktorých funkcia nadobúda svoje maximum alebo minimum, nazývame hľadanie extrémov. Ak existuje derivácia uvažovanej funkcie, potom tento problém sa dá pretransformovať na problém hľadania jej nulových bodov pretože extrémy funkcie sa nachádzajú v tých bodoch, kde je derivácia funkcie nulová.

Všetky techniky hľadania nulových bodov nelineárnych rovníc sú v podstate *iteratívne*. To znamená, že začínajú približnou hodnotou riešenia a postupne tento odhad spresňujú. Proces iterácie skončí, keď chyba odhadu riešenia klesne pod hraničnú úroveň. Tým sme dosiahli *konvergenciu* procesu. V určitých prípadoch sa môže stať, že táto chyba neobmedzene rastie a proces nazveme *divergentný*. Vtedy je zvykom celý proces spustiť s novým začiatočným odhadom riešenia.

7.1 METÓDA BISEKCIE

Metóda bisekcie ([7] str. 250–252) je založená na nasledujúcej vlastnosti: Nech je funkcia $f : (a, b) \subseteq \mathcal{R} \to \mathcal{R}$ spojitá na intervale $I = (x_{pos}, x_{neg})$ tak, že $f(x_{neg})f(x_{pos}) < 0$, tak existuje aspoň jedno $z \in I$, pre ktoré $f(z) = 0$. Bez ujmy na všeobecnosti ďalej predpokladajme, že existuje práve jedno také $z \in I$. Obrázok 7.1. ilustruje tento prípad.



Obr. 7.1. Metóda bisekcie.

Označme x_{pos} hodnotu, kde $f(x_{pos}) > 0$, a x_{neg}, pre ktorú $f(x_{neg}) < 0$. Algoritmus 11 popisuje implementáciu metódy bisekcie. V prvom kroku si zvolíme bod $x_1 = (x_{pos} - x_{neg})/2$ a vypočítame funkčnú hodnotu $f(x_1)$. Ak je táto hodnota kladná, tak v ďalšom kroku hodnotou x_1 nahradíme x_{pos} inak hodnotou x_1 nahradíme x_{neg}. Tento proces opakujeme až kým $|x_{pos} - x_{neg}| < \varepsilon$, kde ε je požadovaná presnosť riešenia. Približným riešením je buď x_{pos} alebo x_{neg}. Chybu riešenia je možné odhadnúť pomocou $|x_{pos} - x_{neg}|$, lebo hľadané riešenie je vždy v intervale s koncovými bodmi x_{pos} a x_{neg}.

Algoritmus 11 Algoritmus bisekcie

Nájde nulový bod spojitej funkcie $f(x)$.
repeat
  $x = (x_{pos} - x_{neg})/2$
  if $f(x) > 0$ **then**
    $x_{pos} = x$
  else
    $x_{neg} = x$
  end if
until $|x_{pos} - x_{reg}| > \epsilon$

Počet iterácii n potrebných na nájdenie riešenia s presnosťou ε, pri začiatočnom intervale (x_{pos}, x_{neg}) je

$$n = \left\lceil \log_2 \frac{|x_{pos} - x_{neg}|}{\varepsilon} \right\rceil.$$

Napríklad, ak má začiatočný interval dĺžku 1, a požadovaná presnosť je 10^{-8}, tak teoretický počet iterácií na nájdenie riešenia je 27. Neskôr si ukážeme, že metóda bisekcie patrí k tým pomalšie konvergujúcim.

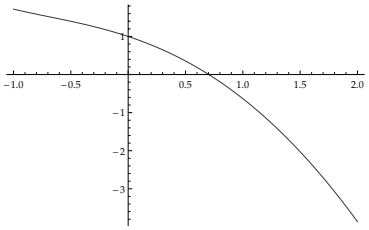
Ak funkcia f nie je na intervale I spojitá, tak prítomnosť nulového bodu nemožno zaručiť. A tak je spojitosť nutná podmienka fungovania metódy bisekcie.

Znalosť začiatočných hodnôt x_{pos} a x_{neg} je kľúčovým faktorom algoritmu. Pre všeobecné použitie algoritmu musí byť daný mechanizmus na ich určenie. Možným riešením je ich náhodný výber.

Príklad 23. *Hľadajme korene rovnice* $e^{-x} - x^2 = 0$.

Riešenie. Najskôr si formulujme funkciu a vykreslíme jej graf aby sme vedeli kde asi korene hľadať. Vyzerá to tak, že koreň bude v okolí $x = 0,7$. Použijeme

```
f = Exp[-x] - x^2;
Plot[f, {x, -1, 2}]
```



Obr. 7.2. Graf $f(x) = e^{-x} - x^2$.

metódu *bisekcie* s začiatočnými hodnotami $x_{pos} = 0$, $x_{neg} = 1$:

```
riesenie = FindRoot[f, {x, 0, 1}]
f /. riesenie
```

$\{x \to 0.703467\}$
$-1.4988 * 10^{-15}$

Koreň je $0,703467$ a hodnota funkcie je s veľkou presnosťou nulová až 10^{-15}.

Použijeme teraz metódu *sečníc* (pozri paragraf 7.2) so začiatočnými hodnotami $x_0 = 0$, $x_1 = 0.5$. Keďže sú obe znamienka rovnaké funkcia `FindRoot` použije automaticky metódu sečníc:

```
riesenie = FindRoot[f, {x, 0, 0.5}]
```

$\{x \to 0.703467\}$

Ak zadáme len jednu začiatočnú hodnotu, v tomto prípade $x_0 = 0$, tak bude automaticky použitá *Newtonova* metóda (pozri paragraf 7.4) na hľadanie koreňov:

```
riesenie = FindRoot[f, {x, 0}]
```

$\{x \to 0.703467\}$

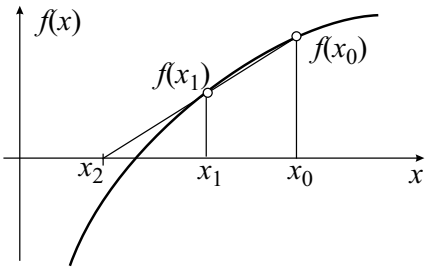
□

Možné metódy hľadania koreňov funkciou `FindRoot` v programe *Mathematica* [15] sú nasledujúce:

`FindRoot[eqn, x, x0]` Newtonova metóda nájde koreň so začiatkom $x0$.
`FindRoot[eqn, x, x0, x1]` Metódou bisekcie resp. sečníc nájde koreň so začiatkom $x0$ a $x1$.

7.2 METÓDA SEČNÍC

Metóda využíva tú vlastnosť, že každá spojitá funkcia sa na dostatočne malom intervale dá dostatočne presne aproximovať úsečkou. Začneme v bode x_0, o ktorom predpokladáme, že nie je príliš vzdialený od hľadaného nulového bodu z. Tým zvolíme bod $x_1 \neq x_0$, pre ktorý $|f(x_1)| < |f(x_0)|$ tým chceme dosiahnuť aby x_1 bolo bližšie k nulovému bodu z ako x_0. Bodmi $(x_0, f(x_0))$ a $(x_1, f(x_1)$ veďme priamku. Obrázok 7.3. znázorňuje túto situáciu. Ak by



Obr. 7.3. Metóda sečníc.

bola $f(x)$ lineárna, tak by priamka preťala os x v nulovom bode funkcie (tj. $x_2 = z$). Ak uvažovaná funkcia nie je lineárna, tak z podobnosti trojuholníkov dostávame

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

Z tejto rovnice vyriešime x_2:

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

ktoré sa nemusí nutne rovnať z, ale predpokladáme, že je bližšie k z ako niektorý z bodov x_0 alebo x_1 . Iterovaním tohto postupu a výberom vždy najbližších bodov na určenie priamky môžeme dospieť k stále presnejším odhadom nulového bodu z.

Po každej iterácii musíme nastaviť $x_0 = x_1$ a $x_1 = x_2$. Iteratívny proces zastavíme, ak $|x_0 - x_1| < \varepsilon$, kde ε je požadovaná presnosť. Približne nájdený nulový bod je potom $x_1 \approx z$. Popísaný postup zhrnieme v algoritme 12 a nazýva sa metódou sečníc ([7] str. 254). Konštanta ε určuje presnosť riešenia

Algoritmus 12 Algoritmus metódy sečníc

Nájde nulový bod spojitej funkcie $f(x)$.
if $|f(x0)| < |f(x1)|$ **then**
  zameň x_0 s x_1
end if
repeat
  $x_2 = x_0 - f(x_0) * (x_0 - x_1)/(f(x_0) - f(x_1))$
  $x_0 = x_1$
  $x_1 = x_2$
until $|x_0 - x_1| > \epsilon$

ale v niektorých prípadoch nevieme dosiahnuť danú presnosť a metóda sečníc nekonverguje. Môžeme si naprogramovať vlastnú funkciu na výpočet koreňov metódou sečníc v programe *Mathematica*:

```
secantSolve[f_, x_, x0_, x1_, eps_: 10^-10, d_: 1, n_: 20] :=
 Module[{y0 = x0, y1 = x1, f0 = N[f /. x -> x0], f1, newy,
   iters = {x0, x1}},
  Do[f1 = N[f /. x -> y1];
   If[Abs[f1] < eps, Break[]];
   newy = y1 - d f1 (y1 - y0)/(f1 - f0);
   iters = {iters, newy};
   y0 = y1; y1 = newy; f0 = f1, {n}];
  Flatten[iters]]
```

Program je iteratívnym procesom, keď je hodnota funkcie v nájdenom koreni menšia ako `eps`, štandardná hodnota je 10^{-10}, tak sa proces zastaví. Iterácia sa opakuje najviac `n` krát (štandardne 20 krát) s tlmiacim faktorom `d` ($d = 1$). Výsledok v jednotlivých krokoch sa ukladá v `iters`. Použime program na nájdenie koreňov funkcie $f(x) = e^{-x} - x^2$ nasledujúcim spôsobom:

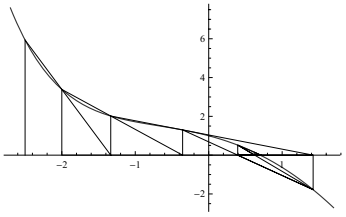
```
f = Exp[-x] - x^2;
```

```
secantSolve[f, x, 0.2, 0.5]
```

$\{0.2, 0.5, 0.753338, 0.699275, 0.703386, 0.703468, 0.703467\}$

Posledné číslo v zozname je náš koreň $0,703467$ nájdený s presnosťou 10^{-10}. Ukážeme si na príklade, postupnosť sečníc a priesečníky sečníc s osou x. Vykreslíme vynesené funkčné hodnoty pre postupnosť x_i, sečnicu vedenú bodmi $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$ a graf funkcie f:

```
it = secantSolve[f, x, -2.5, -2.0];
points = {};
Do[AppendTo[points, {it[[i]], 0}];
 AppendTo[points, {it[[i]],
  f /. x -> it[[i]]}];
 AppendTo[points, {it[[i + 1]],
  f /. x -> it[[i + 1]]}];
 AppendTo[points, {it[[i + 2]],0}],
 {i, Length[it] - 2}]
Plot[f, {x, -2.7, 1.7},
 PlotStyle->AbsoluteThickness[1],
 Epilog -> Line[points]]
```



Obr. 7.4. Metóda sečníc pre $f(x) = e^{-x} - x^2$.

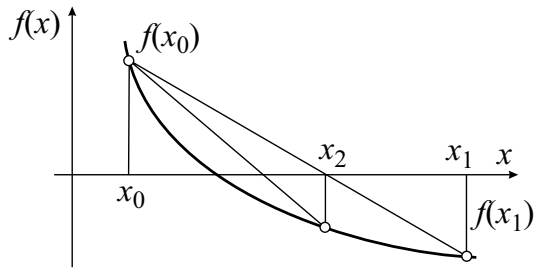
Konvergencia metódy sečníc vyplýva z nasledujúcej vlastnosti ([7] str. 254). Ak $f \in C^2(I)$, I je okolie nulového bodu z a nech $f'(z) \neq 0$. Potom, ak začiatočné hodnoty $x_0 \in I$, $x_1 \in I$ sú dostatočne blízko z postupnosť x_i konverguje k z.

7.3 METÓDA LINEÁRNEJ INTERPOLÁCIE

Obrázok 7.5. znázorňuje metódu lineárnej interpolácie, ktorá je podobná metóde bisekcie s tým rozdielom, že $f(x_0)$ a $f(x_1)$ majú opačné znamienka a nasledujúci odhad nulového bodu je priesečník sečnice $(x_0, f(x_0))$ a $(x_1, f(x_1))$ s osou x. Toto zabezpečuje rýchlejšiu konvergenciu ako v prípade metódy bisekcie. Algoritmus 13 popisuje metódu lineárnej interpolácie. Vstupné parametre sú rovnaké ako pri bisekcii teda interval (x_0, x_1), pričom hodnoty $f(x_0)$ a $f(x_1)$ majú opačné znamienka.

7.4 NEWTONOVA METÓDA

Jedným z najčastejšie použitých prístupov na riešenie nelineárnych rovníc je *Newtonova metóda* ([7] str. 255–258). Podobne ako predchádzajúce metódy, aj Newtonova metóda je založená na lineárnej aproximácii funkcie s tým rozdielom, že používa dotyčnice grafu funkcie ako znázorňuje obrázok 7.6. Základný



Obr. 7.5. Algoritmus lineárnej interpolácie.

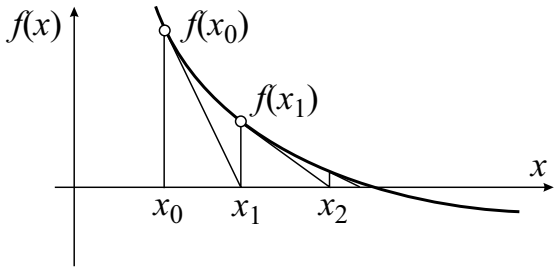
Algoritmus 13 Algoritmus lineárnej interpolácie

Nájde nulový bod spojitej funkcie $f(x)$.
$x = x_0$
if $|f(x0)| < |f(x1)|$ **then**
  zameň x_0 s x_1
end if
repeat
  $x_n = x$
  $x = x_0 - f(x_0) * (x_0 - x_1)/(f(x_0) - f(x_1))$
  if $f(x)$ má opačné znamienko ako $f(x_0)$ **then**
    $x_1 = x$
  else
    $x_0 = x$
  end if
until $|x - x_n| > \epsilon$

vzťah Newtonovej metódy dostaneme z Taylorovho rozkladu ([2], str.129–185) prvého rádu funkcie f v bode x_n v tvare $f(x) = f(x_n) + + f'(x_n)(x - x_n)$, kde $f'(x_n)$ je prvá derivácia funkcie f v bode x_n. Ak položíme $f(x) = 0$ môžeme odvodiť ďalšiu aproximáciu riešenia x_{n+1} z riešenia rovnice $f(x_n) + f'(x_n)(x - x_n) = 0$. Z tejto rovnice vyjadríme x a položíme $x = x_{n+1}$, čím vyjadríme všeobecný krok Newtonovho algoritmu v nasledujúcom tvare

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Počínajúc prvotným odhadom x_0, ktorý nie je príliš ďaleko od hľadaného riešenia, algoritmus nájde priesečník x-ovej osi s dotyčnicou k funkcii. Proces sa opakuje, kým sú nasledujúce x-ové hodnoty dostatočne blízko seba alebo prípadne, kým je hodnota funkcie v týchto bodoch dosť blízko nuly. Algoritmus 14 je pseudokódom Newtonovho algoritmu, ktorý hľadá koreň rovnice



Obr. 7.6. Newtonova metóda.

$f(x) = 0$ pri danom odhade x_0 hľadaného riešenia. Newtonov algoritmus je

Algoritmus 14 Newtonov algoritmus

Začiatočný odhad x_0 by mal byť čo najbližšie hľadanému riešeniu.
Spočítaj $f(x_0)$ a $f'(x_0)$
$x_1 = x_0$
if $|f(x0)| \neq 0$ a $|f'(x0)| \neq 0$ a $|f(x1)| \neq 0$ **then**
  repeat
    $x_0 = x_1$
    $x_1 = x_0 - f(x_0)/f'(x_0)$
  until $|x_0 - x_1| > \epsilon$
end if

široko používaný, lebo pre hodnoty blízke hľadanému riešeniu je jeho konvergencia v okolí riešenia rýchlejšia než u ostatných spomínaných algoritmov. Je možné ukázať, že pre chyby v dvoch po sebe nasledujúcich iteráciách platí $\varepsilon_{n+1} \approx K\varepsilon_n^2$. Newtonovu metódu môžeme naprogramovať podľa algoritmu 14 v programe *Mathematica* nasledujúcim spôsobom:

```
newton2[f_, x_, x0_, eps_, max_] :=
 Module[{xi = x0, fi, df = D[f, x], dfi, iters = {x0}},
  Do[{fi, dfi} = N[{f, df} /. x -> xi];
   If[Abs[fi] < eps, Break[]];
   xi = xi - fi / dfi;
   iters = {iters, xi}, {max}];
  Flatten[iters]]
```

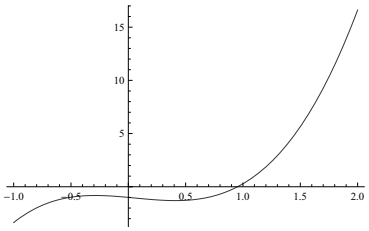
V uvedenej funkcii `newton2` je začiatočný bod metódy `x0`, ako náhle je hodnota funkcie menšia ako `eps`, iterácia skončí a po `Break[]` pokračuje program riadkom `Flatten[iters]`. Cyklus `Do[ ... , max]` zabezpečí to, že sa vykoná nanajvýš `max` iterácii. V každej iterácii sa presnejší odhad koreňu pripíše do `iters`. Hodnoty v `iters` sú postupne $\{x_0\}, \{\{x_0\}, x_1\}, \{\{\{x_0\}, x_1\}, x_2\}$, atd.

Nakoniec použijeme `Flatten`, aby sme dostali jednoduchý zoznam $\{x_0, x_1, x_2, \ldots\}$ ako výsledok programu.

Príklad 24. *Hľadajme korene rovnice* $f(x) = 3x^3 - e^x$ *v obore reálnych a komplexných čísel.*

Riešenie. V prvom riadku formulujeme funkciu a potom vykreslíme jej graf, aby sme vedeli kde hľadať reálne korene. Vyzerá to tak, že koreň bude v

```
f = 3x^3 - E^x;
Plot[f, {x, -1, 2}]
```



Obr. 7.7. Graf $f(x) = 3x^3 - e^x$.

okolí $x = 1$. Použijeme Newtonovu metódu `newton2` so začiatočnou hodnotou $x_0 = 2$, s požadovanou presnosťou $\epsilon = 10^{-14}$ a zároveň nastavíme presnosť výpočtu operácii 10^{-23}:

```
riesenie = SetAccuracy[newton2[f, x, 2, 10^-14, 20], 23]
```

$\{2.00000000000000000000, 1.41941992691654927938 32,$
$1.10192954448389030674 72, 0.97511673956742184188 33,$
$0.95308897864198982574 64, 0.95244621912288507381 56,$
$0.95244567942754665423 88, 0.95244567942716640285 28\}$

```
f /. x -> Last[%]
```

1.25258×10^{-16}

Nájdený koreň je $x = 0,952445679427166402852 8$. Hodnota funkcie v nájdenom koreni je s veľkou presnosťou nulová, až 10^{-16}, a je dokonca menšia ako požadovaná presnosť 10^{-14}.

Hľadajme teraz korene v obore komplexných čísel. Funkciu `newton2` môžeme použiť tak ako doteraz, len za začiatočný bod zvolíme bod komplexnej roviny $-0.5 + 0.5\imath$:

```
SetAccuracy[newton2[f, x, -0.5 + 0.5 I, 10^-14, 20], 23]
```

$\{-0.50000000000000000000 + 0.50000000000000000000\imath,$
$-0.40032765621782773202 14 + 0.46562868974999105375 900\imath,$
$-0.38408728773340949924 631 + 0.47330539886273542693 473\imath,$

$-0.384279996529262535265 30 + 0.473738818734479594230 18\imath$,
$-0.384279882597155841850 17 + 0.473738510757608088930 23\imath$,
$-0.384279882597120536757 98 + 0.473738510757454656108 22\imath\}$

Číslo $-0,384279882597120536757 98 + 0,473738510757454656108 22\imath$ je nájdený komplexný koreň. □

Množstvo iteratívnych alebo rekurzívnych matematických metód je vyjadrených v tvare $x_{i+1} = f(x_i)$. Pomocou iteratívneho procesu môžeme formulovať aj Newtonovu metódu ([7] str. 260–264). *Mathematica* má pre iteratívne počítanie špeciálne funkcionálne typy príkazov ako `Nest`, `FixedPoint` a `Fold`. Vysvetlime si použitie týchto príkazov a nakoniec naprogramujeme Newtonovu metódu.

`Nest` je základný iteratívny príkaz, ktorý vykonáva iteráciu $x_{i+1} = f(x_i)$ daný počet krát. Aplikujme iteráciu napríklad tri krát:

```
Nest[f, x0, 3]
```

$f[f[f[x0]]]$

`FixedPoint` aplikuje iteráciu pokiaľ sa x_{i+1} a x_i nemenia, zastavujúce kritérium je možné zadať. `Fold` taktiež aplikuje iteráciu na funkciu, ale v tomto prípade má funkcia dve premenné a v každej iterácii berie jeden prvok zo zadaného zoznamu ako druhý argument. K dispozícii máme tiež príkazy `NestList`, `FixedPointList` a `FoldList`, ktoré zároveň vypíšu všetky prechodné kroky. Použime napríklad iteráciu s počiatočnou hodnotou x_0 až pokiaľ sa x_i nemení s maximálnym počtom iterácii 3:

```
FixedPointList[f[#] &, x0, 3]
```

$x0, f[x0], f[f[x0]], f[f[f[x0]]]$

Zhrnieme možné spôsoby hľadania koreňov metódou `FixedPoint` v programe *Mathematica*:

`FixedPoint[f, x0]`	Iteruje $x_{i+1} = f(x_i)$ s počiatkom x_0 až pokiaľ sa x_i nemení.
`FixedPointList[f, x0]]`	Vypíše všetky postupné kroky.
`FixedPoint[f, x0, max]`	Iteruje pokiaľ sa x_i nemení, maximálne `max` iteracii.
`FixedPointList[f, x0, max]`	Vypíše všetky prechodné kroky.

Teraz si môžeme naprogramovať Newtonovu metódu aj použitím iterácii:

```
newton[f_, x_, x0_, max_, opts_] := With[{df = D[f, x]},
 FixedPointList[(x - f/df) /. x -> # &, N[x0], max, opts]]
```

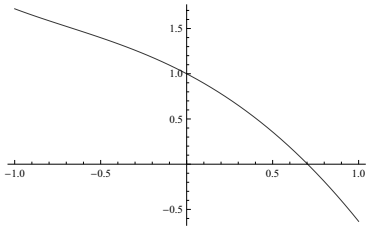
Prvou výhodou funkcionálnych iteratívnych príkazov je, že v porovnaní s procedurálnym programovaním skracujú kód. Porovnajte hore uvedený program `newton` s programom `newton2`. V programe `newton` sa nemusíme zaoberať

nastavovaním vhodných hodnôt premenných alebo implementáciou zastavenia iterácie. Jednoducho zadáme tri či štyri položky potrebné pre iteráciu: iteračnú funkciu, štartovací bod, maximálny počet krokov a (pokiaľ prednastavené zastavujúce kritérium nie je vhodné) zastavujúce kritérium. Druhou výhodou je, že funkcionálne iteratívne príkazy sú rýchle.

Príklad 25. *Hľadajme korene rovnice* $f(x) = e^{-x} - x^2$ *v obore reálnych čísel použitím iteratívnych pravidiel a vytvorte ilustráciu procesu Newtonovej metódy.*

Riešenie. Najskôr formulujme funkciu a vykreslíme jej graf, aby sme vedeli kde hľadať reálny koreň (pozri obrázok 7.8.). Koreň bude v okolí $x = 1$.

```
f = Exp[-x] - x^2;
Plot[f, {x, -1, 1}]
```



Obr. 7.8. Graf $f(x) = e^{-x} - x^2$.

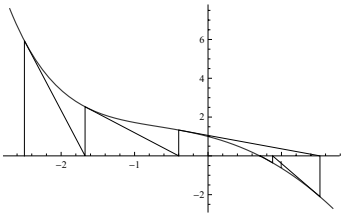
Použijeme Newtonovu metódu `newton` s začiatočnou hodnotou $x_0 = -2,5$, s maximálnym počtom iterácii 20:

```
newton[f, x, -2.5, 20]
```

$\{-2.5, -1.67403, -0.399213, 1.52398, 0.879521, 0.714585,$
$0.703516, 0.703467, 0.703467, 0.703467\}$

Posledné číslo v zozname je náš koreň $x = 0,703467$.

```
it = newton[f, x, -2.5, 20];
points = Flatten[Map[{{#, 0},
 {#, f /. x -> #}} &, it], 1];
Plot[f, {x, -2.7, 1.7},
 PlotStyle -> AbsoluteThickness[1],
 Epilog -> Line[points]]
```



Obr. 7.9. Newtonova metóda pre funkciu $f(x) = e^{-x} - x^2$.

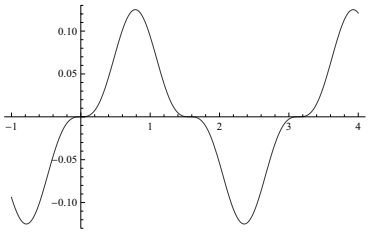
Ukážme si graficky ako postupuje Newtonova metóda pre tento príklad na obrázku. Z príkladu uvidíme, že Newtonova metóda je nepoužiteľná, ak bude

dotyčnica v bode rovnobežná s osou x. Na obrázku 7.9. pre každý bod na osi x vykreslíme vynesenú funkčnú hodnotu, dotyčnicu v tomto bode a graf funkcie f. □

Príklad 26. *Nájdime všetky korene rovnice* $f(x) = \cos(x)^3 \sin(x)^3$ *na intervale* $[-1, 4]$ *použitím Newtonovej metódy.*

Riešenie. Formulujme funkciu a pre ilustráciu si aj vykreslíme jej graf (pozri obrázok 7.10.), aby sme vedeli odhadnúť začiatočné hodnoty Newtonovej metódy. Odhadujeme, že korene budú v okolí $x = 0$, $x = 1,5$, $x = 3$.

```
f = Cos[x]^3 Sin[x]^3;
Plot[f, {x, -1, 4}]
```



Obr. 7.10. Graf $f(x) = e^{-x} - x^2$.

Použijeme Newtonovu metódu `newton2` s týmito začiatočnými hodnotami, s presnosťou $\epsilon = 10^{-14}$ a s maximálnym počtom iterácii 20:

```
newton2[f, x, 0, 10^-14, 20]
```

$\{0\}$

```
newton2[f, x, 1.5, 10^-14, 20]
```

$\{1.5, 1.52376, 1.53948, 1.54993, 1.55689, 1.56153, 1.56462, 1.56668,$
$1.56805, 1.56897, 1.56958, 1.56998, 1.57025, 1.57043, 1.57056,$
$1.57064, 1.57069, 1.57072, 1.57075, 1.57076, 1.57078\}$

```
newton2[f, x, 3, 10^-14, 20]
```

$\{3, 3.0485, 3.0799, 3.10057, 3.11427, 3.12339, 3.12946, 3.1335,$
$3.1362, 3.138, 3.1392, 3.14, 3.14053, 3.14088, 3.14112, 3.14128,$
$3.14138, 3.14145, 3.1415, 3.14153, 3.14155\}$

Posledné čísla v zoznamoch sú nájdené korene $x = 0$, $x = 1,57078$, $3,14155$. Spravme skúšku správnosti, aby sme videli s akou presnosťou sme našli korene:

```
f /. x -> 0
f /. x -> 1.57078
f /. x -> 3.14155
```

0
4.35214×10^{-15}
-7.76009×10^{-14}

Po dosadení do rovnice zistíme, že výsledky sú nájdené s požadovanou presnosťou. Najhoršia presnosť je 10^{-14} a jeden koreň $x = 0$ sme našli bez chyby. □

7.5 NEWTONOVA METÓDA PRE SYSTÉMY NELINEÁRNYCH ROVNÍC

So systémami nelineárnych rovníc sa v praxi môžeme stretnúť v mnohých oblastiach. Predpokladajme, že funkcie f_1, f_2, ..., f_m sú funkciami nezávislých premenných x_1, x_2, ..., x_m. Riešením systému:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_m) &= 0 \\ f_2(x_1, x_2, \ldots, x_m) &= 0 \\ &\cdots \\ f_m(x_1, x_2, \ldots, x_m) &= 0 \end{aligned}$$

rozumieme m-ticu (vektor) komplexných čísel $\{x_1, x_2, \ldots, x_m\}^\top$, ktorá spĺňa dané rovnice. Zaveďme nasledujúce vektorové označenia pre funkcie a pre neznáme

$$\mathbf{f} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{Bmatrix}, \qquad \mathbf{x} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{Bmatrix}.$$

V tomto označení sú vektory stĺpcové matice. Prepísaním systému rovníc do vektorového tvaru dostaneme $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Taylorov rozvoj ([2], str.129–185) prvého rádu v bode $\mathbf{x}^{(n)} = \{x_1^{(n)}, x_2^{(n)}, \ldots, x_m^{(n)}\}^\top$ môžeme zapísať vo forme

$$\begin{aligned} f_1(\mathbf{x}) &= f_1(\mathbf{x}^{(n)}) + \sum_{i=1}^{m} \frac{\partial f_1(\mathbf{x}^{(n)})}{\partial x_i}(x_i - x_i^{(n)}) \\ &\cdots \\ f_m(\mathbf{x}) &= f_m(\mathbf{x}^{(n)}) + \sum_{i=1}^{m} \frac{\partial f_m(\mathbf{x}^{(n)})}{\partial x_i}(x_i - x_i^{(n)}). \end{aligned}$$

Označenie (n) v exponente nám neskôr pomôže pri definovaní iteračného procesu. Ten istý rozvoj vo vektorovom tvare bude

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}^{(n)}) + \mathbf{J}(\mathbf{x}^{(n)}) \cdot (\mathbf{x} - \mathbf{x}^{(n)}).$$

$\mathbf{J}(\mathbf{x})$ značí Jacobiho maticu

$$\mathbf{J}(\mathbf{f},\mathbf{x}) \equiv \mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_m} \end{bmatrix}.$$

Položením $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ sa dá proces iterácie Newtonovej metódy pre systémy nelineárnych rovníc sformulovať ako ([7] str. 286–288)

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - (\mathbf{J}(\mathbf{x}^{(n)}))^{-1}\mathbf{f}(\mathbf{x}^{(n)}).$$

Výraz $(\mathbf{J}(\mathbf{x}^{(n)}))^{-1}$ označuje inverznú maticu k Jacobiho matici. Keďže výpočet inverznej matice je výpočtovo náročná operácia, namiesto jedného kroku Newtonovej iterácie nájdeme riešenie systému lineárnych rovníc

$$\mathbf{J}(\mathbf{x}^{(n)})\Delta\mathbf{x}^{(n)} = -\mathbf{f}(\mathbf{x}^{(n)}),$$

kde $\Delta\mathbf{x}^{(n)} = \mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}$. Lepší odhad riešenia potom dostaneme ako

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \Delta\mathbf{x}^{(n)}.$$

Pre náročnejších čitateľov uvedieme samostatne naprogramovanú funkciu `NewtonSolveSystem`, ktorá rieši systém nelineárnych rovníc spôsobom ako sme popísali v teórii. Kritériom na zastavenie iteratívneho procesu je druhá odmocnina súčtu štvorcov rozdielu medzi poslednými dvoma iteráciami, čo môžeme zapísať príkazom `SameTest->(Sqrt[(#1-#2).(#1-#2)] < eps &)`. Štandardne je presnosť `eps` rovná 10^{-6}. Potrebnú Jacobiho maticu vieme vyrátať príkazom `jac = N[Outer[D, f, x]]`.

```
newtonSolveSystem[f_List, x_List, x0_List, eps_:10^-6, n_:20] :=
 With[
  {jac = N[Outer[D, f, x]]},
  FixedPointList[(# +
   LinearSolve[jac/.Thread[x -> N[#]], -f/.Thread[x->N[#]]]) &,
   N[x0], n,
   SameTest->(Sqrt[(#1-#2).(#1-#2)]<eps &)]]
```

Príklad 27. *Použite program* `newtonSolveSystem` *na nájdenie koreňov sústavy rovníc* $f_1(x,y) = x^2 + y^2 - 1 = 0$, $f_2(x,y) = \sin(x) - y = 0$ *v obore reálnych čísel.*

Riešenie. Formulujme funkcie f_1, f_2 a zvolíme začiatočný bod iteračnej metódy $x = -1$, $y = 1$, presnosť hľadaného riešenia ponecháme štandardnú 10^{-6}.

```
f1 = x^2+y^2-1;
```

```
f2 = Sin[x]- y;
newtonSolveSystem[{f1, f2}, {x, y}, {-1, 1}]
```

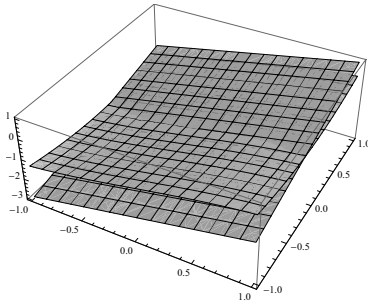
$\{\{-1.,1.\}, \{-3.91816,-2.41816\}, \{-2.75207,-0.130969\},$
$\{-1.48707,-1.54999\}, \{-0.900059,-0.947405\}, \{-0.766791,-0.700529\},$
$\{-0.739527,-0.674194\}, \{-0.739085,-0.673612\}, \{-0.739085,-0.673612\}\}$

Posledná dvojica čísel v zozname je naše riešenie sústavy $x = -0,739085$, $y = -0,673612$ nájdené s presnosťou 10^{-6}. □

Príklad 28. *Riešte sústavu rovníc* $\sin(x) - \cos(y) = 0$, $x + y - 1 = 0$ *v obore reálnych čísel.*

Riešenie. Najskôr si formulujme funkcie a vykreslíme ich grafy na intervale $x, y \in [-1, 1]$:

```
f = Sin[x] - Cos[y];
g = x + y - 1;
Plot3D[{f, g}, {x, -1, 1},
 {y, -1, 1}]
```



Obr. 7.11. Grafy $f(x,y) = \sin(x) - \cos(y)$ a $g(x,y) = x + y + 1$.

Začiatočné riešenie zvolíme $x = 1$ a $y = 1$ a použijeme jednoduchú funkciu `FindRoot`:

```
FindRoot[{Sin[x] == Cos[y], x + y == 1}, {{x, 1}, {y, 1}}]
```

$\{x \to -1.85619, y \to 2.85619\}$

Našli sme riešenie sústavy nelineárnych rovníc $x = -1,85619$, $y = 2,85619$. □

7.6 VÝBER METÓDY

Pre väčšinu derivovateľných funkcií je Newtonov algoritmus rýchlejší ako ostatné metódy. Avšak zlyhá ak je hodnota derivácie v niektorom bode príliš malá t.j. v prípadoch $f'(x) \approx 0$, $f''(x) \approx 0$. V takom prípade je nasledujúca

vybraná x-ová hodnota príliš ďaleko od predchádzajúcej a algoritmus „odskočí“ od riešenia.

Na druhej strane metóda bisekcie je veľmi spoľahlivá a vždy konverguje na intervale, kde funkcia nemá žiadnu singularitu. Preto použitie kombinácie Newtonovej metódy a metódy bisekcie môže byť rýchle a spoľahlivé.

7.7 ÚLOHY

1 Použite Newtonovu metódou na nájdenie všetkých koreňov nasledujúcich rovníc, v zátvorke uvádzame pomôcku ako voliť začiatočné hodnoty:

a) $x^3 - 19 = 0,\ (x_0 = 3)$;
b) $e^{2x} + 5x = 0,\ (x_0 = 0)$;
c) $\ln x - e^{-3x} = 0,\ (x_0 = 2)$;
d) $6x - \sqrt{1 - x^2} = 0,\ (x_0 = 0)$;
e) $\ln x - (x^2 - 4) = 0,\ (x_0^1 = 1,\ x_0^2 = 0,005)$;
f) $\tan x - (3x + 1) = 0,\ x \in (-\frac{\pi}{2}, \frac{\pi}{2}),\ (x_0^1 = -0,5,\ x_0^2 = -1,2,\ x_0^3 = 1,4)$;
g) $x^7 + 4x^3 + 12 = 0,\ (x_0 = -1,5)$;
h) $\cos(x) - x^2 = 0,\ (x_0^1 = 1,\ x_0^2 = -1)$;
i) $x^3 - 5x - 3 = 0,\ (x_0^1 = -2,\ x_0^2 = -0,5,\ x_0^3 = 2,5)$;
j) $e^{\sin(x)} + x = 0,\ (x_0 = -1)$;

2 Pre každú z nasledujúcich rovníc nájdite všetky korene metódou bisekcie:

a) $\frac{6x - 5\sqrt{x} + 1}{1 - \sqrt{x}} = 0$;
b) $4\cos^2 x + 4\cos x - 3 = 0$;
c) $2\sin^2 x + \sin x - 1 = 0$;
d) $\sin^2 x + \cos^2 x + \sin x = 0$;
e) $2\sin^2 x - 3\cos x = 0$.

3 Pre každú z nasledujúcich rovníc nájdite všetky korene v programe *Mathematica* Newtonovou metódou:

a) $\cos^2 x - \sin^2 x - \frac{1}{4} = 0$;
b) $\tan^2 x + 4\sin^2 x - 3 = 0$;
c) $\tan^3 x + \tan^2 x - 3\tan x - 3 = 0$;
d) $\frac{1 + \tan x}{1 - \tan x}\tan x - 1 = 0$;
e) $\sin^2 x - 2\sin x \cos x - \cos^2 x = 0$.

4 Pre každú z nasledujúcich rovníc nájdite najmenší kladný koreň pomocou programu *Mathematica*:

a) $\tan x + \tan 2x - \tan 3x = 0$;
b) $\cos 2x + \cos 4x + \cos 6x + \cos 8x = 0$;
c) $\sin 2x + \sin 3x - \sin x = 0$;
d) $\cos x - \cos 2x - \cos 4x = 0$.

5 Riešte nasledujúce systémy rovníc programom *Mathematica*:

a) $\frac{\sin x}{\sin y} - 2 = 0,\quad x + y - 120^\circ = 0$;
b) $\sin x \sin y = \frac{3}{4},\quad \tan x \tan y = 3$;
c) $x + y = 120^\circ,\quad \sin x + \sin y = 1,5$;
d) $x + y = 45^\circ,\quad \frac{\sin x}{\sin y} = \frac{\sqrt{3}+1}{\sqrt{2}}$;
e) $\sin x + \sin y = 1,\quad \cos x + \cos y = \sqrt{2}$.

6 Nájdite príklad funkcie a začiatočné hodnoty tak, aby:

a) Metóda bisekcie nekonvergovala; b) Newtonova metóda nekonvergovala.

7 Pomocou programu *Mathematica* nájdite korene nasledujúcej funkcie s presnosťou $\epsilon = 10^{-14}$ na príslušných intervaloch:

$$\text{a) } f(x) = \tfrac{x^3}{3} - x + \tfrac{2}{3}, \quad x \in (-3, 3).$$

8 Overte ako sa bude správať Newtonova metóda v nasledujúcich prípadoch:

a) Horizontálna dotyčnica, $f(x) = x^2 - 2$, pre $x_0 = 0$;
b) Začiatočná hodnota ďaleko od riešenia, $f(x) = 4\arctan(x)$, $x_0 = 1,5$;
c) Dvoj násobný koreň v bode $x = 1$, $f(x) = x^3 - 3x + 2$.

7.8 VÝSLEDKY

1 a) $x = 2,6684$; b) $x = -0,1486$; c) $x = 1,0445$; d) $x = 0,1690$, e) $x^1 = 2,187$, $x^2 = 0,0183$; f) $x^1 = -0,5275$, $x^2 = -1,206$, $x^3 = 1,3785$; g) $x = -1,2369$; h) $x^1 = 0,8241$, $x^2 = -0,8241$; i) $x^1 = -1,834$, $x^2 = -0,6566$, $x^3 = 2,491$; j) $x = -0.5787$.

2 a) $x_1 = 0,11111$, $x_2 = 0,25$; b) $\pm\frac{\pi}{3} + 2k\pi$ (k vždy celé číslo); c) $x_1 = -\frac{\pi}{2} + 2k\pi$, $x_2 = \frac{\pi}{6} + 2k\pi$, $x_3 = (2k+1)\pi - \frac{\pi}{6}$; d) ($x_1 = 30°$, $x_2 = 150°$, $x_3 = 270°$) $+ k360°$; e) ($x_1 = 60°$, $x_2 = 300°$) $+ k360°$.

3 a) $x_1 = \pm 37°46' + k180°$; b) ($x_1 = 45°$, $x_2 = 135°$) $+ k180°$; c) $(-\frac{\pi}{4}, \pm\frac{\pi}{3}) + k\pi$; d) $\arctan(-1 \pm \sqrt{2}) + k180°$; e) $\frac{\pi}{8}(4k-1)$.

4 a) 60°; b) 45°; c) 60°; d) 20°.

5 a) $x = 90°$, $y = 30°$; b) $x = 60°$, $y = 60°$; c) $x^1 = 90°$, $y^1 = 30°$, $x^1 = 30°$, $y^1 = 90°$; d) $x = 30°$, $x = 15°$; e) v radiánoch $x = 1,13908$, $x = 0,0918809$.

Kapitola 8

INTERPOLÁCIA A APROXIMÁCIA FUNKCIE

Táto kapitola sa venuje interpolácii a aproximácii funkcie, ktorú poznáme iba v niekoľkých tabuľkových hodnotách. Úlohou interpolácie z programovacieho hľadiska je výpočet funkčných hodnôt funcie v bodoch, ktoré nie sú v tabuľke.

Formulujme problém interpolácie. Majme $N + 1$ bodov (x_0, y_0), (x_1, y_1), ..., (x_N, y_N), problém interpolácie je nájsť funkciu $f = f(x)$ takú, že $f(x_i) = y_i$ pre $i = 0, \ldots, N$, kde y_i sú známe hodnoty neznámej funkcie. Potom hovoríme, že f interpoluje $\{y_i\}$ v uzloch $\{x_i\}$. Ak je f algebraický polynóm hovoríme o *polynomickej interpolácii*, ak je f trigonometrický polynóm hovoríme o *trigonometrickej interpolácii* a ak je f polynómom iba lokálne hovoríme o *splajnovej interpolácii*.

Kapitola sa zaoberá polynomickou interpoláciou pomocou Lagrangeovych a Newtonových polynómov. Posledné paragrafy sa venujú aproximácii lineárnou regresiou, nelineárnou regresiou a aproximáciou polynómom. Viac detailov a odvodení k interpolácii a aproximácii nájdete v ([14] str. 58–72, [7] str. 333–372).

8.1 LAGRANGEOVE POLYNÓMY

Interpolácia znamená odhad chýbajúcej hodnoty funkcie pomocou váženého priemeru známych funkčných hodnôt v susediacich bodoch. Lineárna interpolácia používa úsečku, ktorá prechádza dvoma bodmi

$$y = P(x) = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0}.$$

Lineárne polynómy môžeme zapísať v tvare Lagrangeovho polynómu

$$y = P_1(x) = y_0\frac{x - x_1}{x_0 - x_1} + y_1\frac{x - x_0}{x_1 - x_0}.$$

Zavedením Lagrangeovho polynomického koeficientu

$$L_{1,0} = \frac{x - x_1}{x_0 - x_1} \qquad L_{1,1} = \frac{x - x_0}{x_1 - x_0}$$

možno Lagrangeov polynóm prvého stupňa zapísať v tvare:

$$P_1(x) = \sum_{k=0}^{1} y_k L_{1,k}(x).$$

Hore uvedený vzťah zovšeobecníme do konštrukcie ([14] str. 60) polynómu $P_N(x)$ stupňa N, ktorý prechádza cez $N+1$ bodov $(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)$ a má tvar

$$P_N(x) = \sum_{k=0}^{N} y_k L_{N,k}(x),$$

kde $L_{N,k}(x)$ je Lagrangeov polynomický koeficient

$$L_{N,k} = \frac{(x - x_0)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_N)}{(x_k - x_0)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_N)}.$$

V koeficiente $L_{N,k}$ sa nenachádzajú členy $(x - x_k)$ a $(x_k - x_k)$. Pomocou znaku súčinu, $\prod$, zapíšeme Lagrangeov polynomický koeficient ako

$$L_{N,k} = \left(\prod_{j=0, j\neq k}^{N} (x - x_j)\right) / \left(\prod_{j=0, j\neq k}^{N} (x_k - x_j)\right).$$

Lagrangeov polynomický koeficient $L_{N,k}(x)$ má nasledujúce vlastnosti $L_{N,k}(x_j) = 1$, ak $j = k$ a $L_{N,k}(x_j) = 0$, ak $j \neq k$. To znamená, že polynomiálna funkcia $P_N(x)$ prechádza cez všetky body (x_j, y_j).

Vytvoríme program na výpočet Lagrangeovho polynómu v programe *Mathematica* [15]:

```
lagrangeInterpolation[xx_List, yy_List, x_] :=
 Sum[yy[[i]] Apply[Times, (x-Drop[xx, {i}])/(xx[[i]]-Drop[xx, {i}])],
  {i, Length[xx]}]
```

Vstupom pre program `lagrangeInterpolation[%]` sú dva zoznamy, zoznam x-ových hodnôt `xx_List` a zoznam y-ových hodnôt `yy_List`. Aby sme porozumeli programu pripomenieme, že funkcia `Drop[xx, i]` zmaže i-ty prvok zo zoznamu `xx`. Časť `Apply[Times,(...)]` vytvorí dva podiely a ich súčin, čím dostaneme Lagrangeov polynomický koeficient $L_{N,k}$. Nakoniec každý koeficient násobí `yy` a sčíta.

Príklad 29. *V programe Mathematica nájdite Lagrangeov polynóm daný troma bodmi* (a, f) , (b, g), (c, h).

Riešenie. Vytvorme zoznam x-ových, y-ových hodnôt a použijeme funkciu `lagrangeInterpolation`:

```
xx = {a, b, c}; yy = {f, g, h};
lagrangeInterpolation[xx, yy, x]
```

$$\frac{h(-a+x)(-b+x)}{(-a+c)(-b+c)} + \frac{g(-a+x)(-c+x)}{(-a+b)(b-c)} + \frac{f(-b+x)(-c+x)}{(a-b)(a-c)}$$

Hľadaný Lagrangeov polynóm má tvar $P(x) = \frac{h(-a+x)(-b+x)}{(-a+c)(-b+c)} + \frac{g(-a+x)(-c+x)}{(-a+b)(b-c)} + \frac{f(-b+x)(-c+x)}{(a-b)(a-c)}$. □

8.2 NEWTONOVE POLYNÓMY

Niekedy je užitočné nájsť niekoľko interpolačných polynómov $P_1(x), P_2(x), \ldots, P_N(x)$ a potom si zvoliť jeden, ktorý je najlepší. Ak použijeme Lagrangeove polynómy, nepoznáme rekurzívnu konštrukciu závislosti medzi $P_{N-1}(x)$ a $P_N(x)$. Každý Lagrangeov polynóm zostrojujeme jednotlivo. Ďalší spôsob interpolácie, ktorý odstraňuje tieto nevýhody používa Newtonove polynómy ([5] str. 60–62) definované nasledujúcim rekurzívnym vzťahom

$$\begin{aligned} P_1(x) &= a_0 + a_1(x - x_0) \\ P_2(x) &= P_1(x) + a_2(x - x_0)(x - x_1) \\ P_3(x) &= P_2(x) + a_3(x - x_0)(x - x_1)(x - x_2) \\ &\ldots \end{aligned}$$

Polynóm $P_N(x)$ je tak definovaný pomocou $P_{N-1}(x)$ rekurzívnym vzťahom

$$P_N(x) = P_{N-1}(x) + a_N(x - x_0)(x - x_1)\dots(x - x_{N-1}).$$

$P_N(x)$ je obyčajný polynóm N-tého stupňa. Problém je zistiť koeficienty a_k $(k = 0, 1, \dots, N)$ pre polynómy $P_1(x), P_2(x), \dots, P_N(x)$. Uvažujme polynóm prvého stupňa $P_1(x)$. V tom prípade chceme interpolovať prvé dve funkčné hodnoty $f(x_0)$ a $f(x_1)$, čo vyjadríme vzťahmi:

$$P_1(x_0) = f(x_0) \qquad P_1(x_1) = f(x_1).$$

Zistíme, že

$$f(x_0) = P_1(x_0) = a_0 + a_1(x_0 - x_0) = a_0.$$

Druhá podmienka v bode x_1 vedie k vzťahu

$$f(x_1) = P_1(x_1) = a_0 + a_1(x_1 - x_0) = f(x_0) + a_1(x_1 - x_0).$$

Z pravej a ľavej strany rovnosti je koeficient $a_1 = \frac{f(x_1)-f(x_0)}{x_1-x_0}$. Keď poznáme a_0 a a_1 je možné určiť a_2 atď.

Definícia 4. *Pomerná diferencia. Pomerné diferencie funkcie $f(x)$ v bode x_k definujeme:*

$$\begin{aligned}
f[x_k] &= f(x_k) \\
f[x_{k-1}, x_k] &= \frac{f[x_k] - f[x_{k-1}]}{x_k - x_{k-1}} \\
f[x_{k-2}, x_{k-1}, x_k] &= \frac{f[x_{k-1}, x_k] - f[x_{k-2}, x_{k-1}]}{x_k - x_{k-2}} \\
&\dots \\
f[x_{k-j}, x_{k-j+1}, \dots, x_{k-1}, x_k] &= \frac{f[x_{k-j+1}, \dots, x_k] - f[x_{k-j}, x_{k-1}]}{x_k - x_{k-j}}.
\end{aligned}$$

V prípade Newtonovho polynómu

$$P_N(x) = a_0 + a_1(x - x_0) + \dots + a_N(x - x_0)(x - x_1)\dots(x - x_N),$$

nám táto definícia umožňuje zapísať koeficienty pomernými diferenciami $a_k = f[x_0, x_1, \dots, x_k]$ $k = 0, 1, \dots, N$.

Príklad 30. *Polynomiálna interpolácia. Zostrojte tabuľku pomerných diferencií a Newtonov polynóm pre funkciu danú v šiestich uzloch (pozri prvý a druhý stĺpec v tabuľke).*

Riešenie.

x_k	$f[x_k]$	1. pd	2. pd	3. pd	4. pd	5. pd
$x_0 = 1$	$\boxed{-3}$					
$x_1 = 2$	0	$\boxed{3}$				
$x_2 = 3$	15	15	$\boxed{6}$			
$x_3 = 4$	48	33	9	$\boxed{1}$		
$x_4 = 5$	105	57	12	1	$\boxed{0}$	
$x_5 = 6$	192	87	15	1	0	$\boxed{0}$

Koeficienty Newtonovho polynómu sa nachádzajú v tabuľke na diagonále a sú rovné $a_0 = -3, a_1 = 3, a_2 = 6, a_3 = 1, a_4 = a_5 = 0$. Hodnoty danej funkcie sú presne interpolované Newtonovým polynómom tretieho stupňa $P_3(x) = -3 + 3(x-1) + 6(x-1)(x-2) + (x-1)(x-2)(x-3)$. □

Vytvorme program na výpočet Newtonovho polynómu v programe *Mathematica*:

```
div[z_List] := div[z] = (div[Drop[z,1]]-div[Drop[z,-1]])/
    (Last[z]-First[z])

newtonInterpolation[xx_List, yy_List, x_] :=
 With[{n = Length[xx]},
  Do[div[{xx[[i]]}] = yy[[i]], {i,n}];
  Sum[div[Take[xx, i]] Product[x - xx[[j]], {j, i-1}], {i, n}]]
```

Funkcia `div[ ]` počíta pomerné diferencie a funkcia `Drop[z, i]` zmaže i-ty prvok zo zoznamu `z`. Vstupom funckie `newtonInterpolation` sú dva zoznamy, zoznam x-ových hodnôt `xx_List` a zoznam y-ových hodnôt `yy_List`. V riadku s príkazom `Do[div[xx[[i]]] = yy[[i]], i,n];` rátame štartovacie hodnoty pomerných diferencií, čo sú vlastne požadované hodnoty funkcie zo zoznamu `yy`. Potom rátame druhé a ďalšie pomerné diferencie.

Príklad 31. *V programe Mathematica zostrojte tabuľku pomerných diferencií a Newtonov polynóm daný troma bodmi* (a, f) , (b, g), (c, h).

Riešenie. Vytvorme si zoznam x-ových, y-ových hodnôt v poliach `xx`, `yy` a použijeme funkciu `div[ ]`. Štartovacie hodnoty pomerných diferencií nastavíme v riadku `Do[div[xx[[i]]] = yy[[i]], i,3]`, teda pre $\{a\}, \{b\}, \{c\}$ nastavíme požadované hodnoty funkcie $\{f\}, \{g\}, \{h\}$ zo zoznamu `yy`. Vypíšme štartovacie hodnoty pomerných diferencií.

```
xx = {a, b, c}; yy = {f, g, h};
Do[div[{xx[[i]]}] = yy[[i]], {i, 3}]
{div[{a}], div[{b}], div[{c}]}
```

$\{f, g, h\}$

Teraz môžeme počítať prvé pomerné diferencie

```
{div[{a,b}],div[{b,c}]}
```

$\{\frac{-f+g}{-a+b}, \frac{-g+h}{-b+c}\}$

a druhé pomerné diferencie

```
div[{a,b,c}]
```

$$\frac{-\frac{-f+g}{-a+b}+\frac{-g+h}{-b+c}}{-a+c}$$

Keď chápeme princíp funkcie `div` môžeme zostrojiť tabuľku pomerných diferencií

```
TableForm[
Transpose[{{a, "", b, "", c}, {div[{a}], "", div[{b}], "",
   div[{c}]}, {"", div[{a, b}], "", div[{b, c}], ""}, {"", "",
   div[{a, b, c}], "", ""}}]]
```

a	f		
		$\frac{-f+g}{-a+b}$	
b	g		$\frac{-\frac{-f+g}{-a+b}+\frac{-g+h}{-b+c}}{-a+c}$
		$\frac{-g+h}{-b+c}$	
c	h		

Nakoniec použitím funkcie `newtonInterpolation[%]` zostrojíme Newtonov polynóm

```
xx = {a, b, c}; yy = {f, g, h};
newtonInterpolation[xx, yy, x]
```

$$f+\frac{(-f+g)(-a+x)}{-a+b}+\frac{\left(-\frac{-f+g}{-a+b}+\frac{-g+h}{-b+c}\right)(-a+x)(-b+x)}{-a+c}$$

Hľadaný Newtonov polynóm má druhý stupeň a tvar $P(x)=f+\frac{(-f+g)(-a+x)}{-a+b}+\frac{\left(-\frac{-f+g}{-a+b}+\frac{-g+h}{-b+c}\right)(-a+x)(-b+x)}{-a+c}$. □

Príklad 32. *Newtonova interpolácia. Zostrojte Newtonov polynóm pre funkciu danú v šiestich uzloch* $(1,-3)$, $(2,0)$, $(3,15)$, $(4,48)$, $(5,105)$, $(6,192)$ *a použijete funkciu* ***newtonInterpolation[%]****.*

Riešenie. Vytvorme zoznam x-ových, y-ových hodnôt v poliach `xx`, `yy` a použijeme funkciu `newtonInterpolation[ ]`.

```
xx = {1, 2, 3, 4, 5, 6}; yy = {-3, 0, 15, 48, 105, 192};
```

```
newtonInterpolation[xx, yy, x]
```

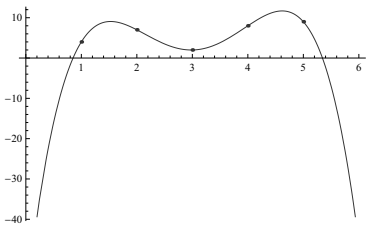
$$-3 + 3(-1 + x) + 6(-2 + x)(-1 + x) + (-3 + x)(-2 + x)(-1 + x)$$

Hodnoty danej funkcie sú presne interpolované Newtonovým polynómom tretieho stupňa $P_3(x) = -3+3(x-1)+6(x-1)(x-2)+(x-1)(x-2)(x-3)$. □

Príklad 33. *Nájdime interpolačný polynóm k bodom* $(1, 4)$, $(2, 7)$, $(3, 2)$, $(4, 8)$, $(5, 9)$ *a vykreslime jeho graf v programe Mathematica.*

Riešenie. Pred použitím funkcie `InterpolatingPolynomial[%]`, si najskôr vytvorme zoznam bodov `body`. Nakoniec vykreslíme graf polynómu a dáta, pozri obr. 8.1. Vstupné dáta sú interpolované polynómom štvrtého stupňa

```
body = {{1, 4}, {2, 7}, {3, 2},
 {4, 8}, {5, 9}};
InterpolatingPolynomial[body, x];
Show[Plot[%, {x, 0, 6}],
 ListPlot[body]]
```

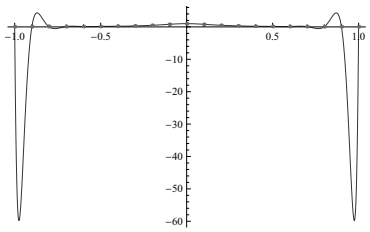


Obr. 8.1. Graf interpolačného polynómu.

$$P(x) = 4 + \left(3 + \left(-4 + \left(\frac{19}{6} - \frac{35}{24}(-4 + x)\right)(-3 + x)\right)(-2 + x)\right)(-1 + x).$$

□

```
f[x_] := 1/(1 + 25 x^2);
body = Table[{x, f[x]},
 {x, -1, 1, 0.1}];
Plot[Evaluate[
 InterpolatingPolynomial[body, x]],
 {x, -1, 1}, PlotRange -> All,
 Epilog -> {Red, Map[Point, body]}]
```

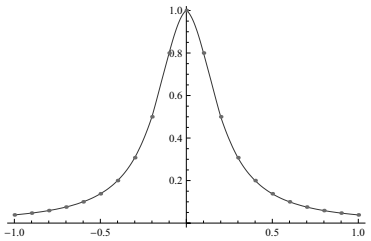


Obr. 8.2. Graf polynómu aproximujúci $f(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$.

Príklad 34. *Nájdime interpolačný polynóm Rungeho funkcie* $f(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$ *a vykreslime graf polynómu a interpolované hodnoty Rungeho funkcie v programe Mathematica.*

Riešenie. Najskôr si zapíšme Rungeho funkciu. Vypočítame tabuľku bodov $(x, f(x))$, ktoré majú byť interpolované a uložme ich do poľa `body`. Použime funkciu `InterpolatingPolynomial[%]`, nakoniec vykreslíme graf polynómu a interpolované body, pozri obr. 8.2. Polynóm, ktorý interpoluje body je

```
Interpolation[body];
Plot[%[x], {x, -1, 1},
 Epilog -> {Red, Map[Point, body]}]
```



Obr. 8.3. Aproximácia $f(x) = \frac{1}{1+25x^2}$, $x \in [-1, 1]$, po častiach polynómom s nízkym stupňom.

vyššieho stupňa a preto pozorujeme veľké oscilácie na krajoch intervalu. Tento problém je možné vyriešiť interpoláciou polynómom po častiach s nízkym stupňom namiesto interpolácie jedným polygónom veľkého stupňa. V programe *Mathematica* máme funkciu `Interpolation[%]`, ktorá používa interpoláciu po častiach polynómom s nízkym stupňom, pozri obr. 8.3. □

8.3 APROXIMÁCIA FUNKCIOU

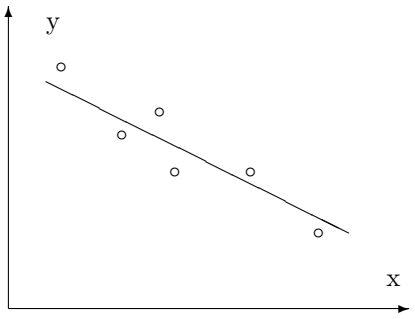
Experimenty často produkujú množiny meraní. Cieľom numerických metód je určiť vzťah $y = f(x)$, ktorého výsledkom je aproximácia experimentálnych hodnôt (x_1, y_1), (x_2, y_2),...,(x_N, y_N). Štandardne funkciu $f(x)$ vyberieme z triedy prípustných funkcii (vzťahov). Ostáva problém odhadnúť koeficienty funkcie $f(x)$. Zakreslené hodnoty x a y na obrázku 8.4. majú lineárnu tendenciu a preto je prirodzená aproximácia funkciou $y = f(x) = Ax + B$. Ako nájdeme najlepšiu priamku idúcu najbližšie k bodom? Bežný postup je minimalizovať súčet

$$E = \sum_{i=1}^{N} (y_k - f(x_k))^2,$$

kde hodnota E je suma štvorcových odchýlok, y_k sú namerané experimentálne hodnoty a $f(x_k)$ sú hodnoty funkcie, ktorou aproximujeme body (x_k, y_k).

8.3.1 Lineárna regresia

Aby sme mohli určiť koeficienty A a B aproximujúcej funkcie $y = f(x) = Ax + B$ je potrebné minimalizovať veličinu $E(A, B) = \sum_{k=1}^{N} (Ax_k + B - y_k)^2$. Nutná



Obr. 8.4. Lineárna regresia.

podmienka k tomu, aby hodnota $E(A, B)$ bola minimálna je aby parciálne derivácie $\frac{\partial E}{\partial A}$ a $\frac{\partial E}{\partial B}$ boli rovné nule:

$$\frac{\partial E}{\partial A} = 2\sum_{k=1}^{N}(Ax_k^2 + Bx_k - x_k y_k) = 0,$$
$$\frac{\partial E}{\partial B} = 2\sum_{k=1}^{N}(Ax_k + B - y_k) = 0.$$

Úpravou získame nasledujúci systém dvoch rovníc s neznámymi koeficientami A a B ([7] str. 440–443)

$$A\sum_{k=1}^{N} x_k^2 + B\sum_{k=1}^{N} x_k = \sum_{k=1}^{N} x_k y_k,$$
$$A\sum_{k=1}^{N} x_k + BN = \sum_{k=1}^{N} y_k.$$

Riešením tohto systému rovníc dostaneme koeficienty A a B regresnej priamky $y = Ax + B$.

Príklad 35. *V programe Mathematica nájdime lineárnu regresiu dát generovaných náhodne z kvadratickej rovnice nasledujúcim programom*

```
<<Statistics‘NormalDistribution‘
SeedRandom[2];
data = Table[{x,
   2 + x - 0.004 x^2 + 2 Random[NormalDistribution[0, 1]]},
   {x, 0, 50}];
```

nakoniec vykreslíme dáta a lineárnu funkciu.

Riešenie. Použijeme lineárnu regresiu aj keď podľa toho ako sú dáta generové na aproximáciu je lepšia kvadratická funkcia

```
fit = Fit[data, {1, x}, x]
```

$3.69809 + 0.801487x$

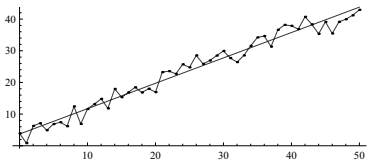
Našli sme lineárnu regresiu v tvare $f(x) = 3,69809 + 0,801487x$. V ďalšom spôsobe použijeme príkaz `FindFit`:

```
FindFit[data, a + b x, {a, b}, x]
fit = a + b x /. %
```

$\{a \to 3.69809, b \to 0.801487\}$
$3.69809 + 0.801487x$

Našli sme rovnakú lineárnu regresiu $f(x) = 3,69809 + 0,801487x$. Dáta môžeme jednoducho vykresliť príkazom `ListPlot[data]` ale aby sme lepšie videli priebeh dát spojíme ich lomenou čiarou a vykreslíme aproximujúcu funkciu. Z obrázku 8.5. vidíme, že nájdená funkcia dobre aproximuje dáta.

```
pdata = ListPlot[data,
 PlotJoined -> True,
 AspectRatio -> 0.4,
 Epilog -> {AbsolutePointSize[2],
 Map[Point, data]}];
pfit = Plot[fit, {x, 0, 50},
 DisplayFunction -> Identity];
Show[pdata, pfit]
```



Obr. 8.5. Lineárna regresia dát funkciou $f(x) = 3,69809 + 0,801487x$.

Nájdime chybu aproximácie ako súčet štvorcov rezíduí

```
{xx, ff} = Transpose[data];
```

Po transpozícii `xx` obsahuje x-ové hodnoty a `ff` obsahuje f hodnoty. Vypočítame funkčné hodnoty aproximácie pre x.

```
aproximacia = fit /. x -> xx;
```

Nakoniec vypočítame rezíduá a súčet štvorcov:

```
resf = ff - aproximacia;
resf.resf
```

239.337

Chyba aproximácie náhodne generovaných dát lineárnou regresiou $f(x) = 3,69809 + 0,801487x$ je $239,337$. □

8.3.2 Linearizácie dát

Predpokladajme, že sú dané body $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ a chceme ich aproximovať krivkou v exponenciálnom tvare

$$y = Ce^{Ax}.$$

Zlogaritmujme obe strany rovnice a dostaneme $\ln y = Ax + \ln C$. Použime transformáciu premenných $Y = \ln y$, $X = x$, $B = \ln C$ a výsledkom transformácie je lineárna závislosť medzi novými premennými X a Y; $Y = AX + B$. Teraz použijeme lineárnu regresiu pre koeficienty A a B, tak ako sme ukázali vyššie. Po nájdení A a B, parameter C vypočítame z rovnice $C = e^B$. Dostali sme tak hľadané koeficienty A a C.

Technika linearizácie dát je využívaná na aproximáciu dát známym tvarom krivky. Keď už je raz tvar aproximačnej funkcie zadaný, potom musíme nájsť vhodnú transformáciu premenných tak, aby sme získali lineárnu závislosť. Napríklad aproximácia funkciou $y = D/(x + C)$ sa transformuje na lineárnu úlohu $Y = AX + B$ použitím transformácie premenných a konštánt $X = xy,\ Y = y,\ C = -1/A$ a $D = -B/A$.

8.3.3 Metóda najmenších štvorcov

Predpokladajme, že máme N dátových bodov $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ hľadáme funkciu $f(x)$ v tvare

$$f(x) = \sum_{j=1}^{M} c_j f_j(x),$$

kde $f_1(x), \ldots, f_M(x)$ je súbor M lineárne nezávislých funkcií ([2] str.129–185). Úlohou metódy najmenších štvorcov je nájsť koeficienty $c_1(x), \ldots, c_M(x)$ tak, že funkcia je daná lineárnou kombináciou a výraz

$$E(c_1(x), \ldots, c_M(x)) = \sum_{k=1}^{N} (f(x_k) - y_k)^2 = \sum_{k=1}^{N} (\sum_{j=1}^{M} c_j f_j(x_k) - y_k)^2.$$

nadobúda minimálnu hodnotu. Pre minimum E je nutné, aby každá parciálna derivácia bola rovná nule ($\partial E/\partial c_i = 0,\ i = 1, 2, \ldots, M$), čo dáva systém

$$\sum_{k=1}^{N} \left[\sum_{j=1}^{M} c_j f_j(x_k) - y_k) \right] f_i(x_k) = 0, \quad i = 1, 2, \ldots, M.$$

Po zámene poradia sumácie získame systém M lineárnych rovníc s M neznámymi koeficientmi c_j

$$\sum_{j=1}^{M}\left[\sum_{k=1}^{N} f_i(x_k)f_j(x_k)\right] c_j = \sum_{k=1}^{N} f_i(x_k)y_k, \quad i = 1, 2, \ldots, M,$$

z ktorého určíme koeficienty c_j.

8.3.4 Aproximácia polynómom

Aplikujme vyššie uvedenú metódu najmenších štvorcov na aproximácie polynómami. V tomto prípade berieme funkcie v polynomickom tvare $f_j(x) = x^j, \quad j = 0, 1, \ldots, M$. Teda funkcia $f(x)$ bude polynomická napr. stupňa M; $f(x) = c_0 + c_1x^1 + \cdots + c_Mx^M$. Použitím všeobecného postupu pre metódu najmenších štvorcov dospejeme k nasledujúcemu systému rovníc, pre hľadané polynomické koeficienty $c_0, \ldots, c_M$, ktorý poskytuje najlepšiu vhodnú aproximáciu k dátam $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$:

$$\begin{aligned}
&c_0\sum_{k=1}^{N} x_k^{2M} + c_1\sum_{k=1}^{N} x_k^{2M-1} + \cdots + c_M\sum_{k=1}^{N} x_k^{M} = \sum_{k=1}^{N} y_k x_k^{M}\\
&c_0\sum_{k=1}^{N} x_k^{2M-1} + c_1\sum_{k=1}^{N} x_k^{2M-2} + \cdots + c_M\sum_{k=1}^{N} x_k^{M-1} = \sum_{k=1}^{N} y_k x_k^{M-1}\\
&\ldots\\
&c_0\sum_{k=1}^{N} x_k^{M} + c_1\sum_{k=1}^{N} x_k^{M-1} + \cdots + c_M N = \sum_{k=1}^{N} y_k.
\end{aligned}$$

Príklad 36. *Uvažujme nasledujúce dáta*

```
<<Statistics`NormalDistribution`
SeedRandom[2];
data = Table[{x,
  Exp[0.3 + 0.2 x] + 0.5Random[NormalDistribution[0, 1]]},
  {x, 0, 10, 0.2}];
```

V programe Mathematica nájdime aproximáciu dát v tvare $f(x) = e^{a+bx}$ a v polynomickom tvare $p(x) = a + bx + cx^2$. Nakoniec vykreslime dáta a funkciu.

Riešenie. Zadajme tvar hľadanej funkcie `Exp[a + b x]` a použijeme príkaz `FindFit[ ]`

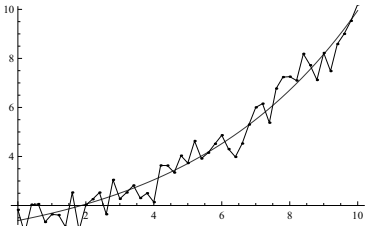
```
f = Exp[a + b x];
```

```
ab = FindFit[data, f, {a, b}, x]
expfit = f /. ab
```

$\{a \to 0.325637, b \to 0.197218\}$
$e^{0.325637+0.197218x}$

Našli sme nelineárnu regresiu v tvare $f(x) = e^{0,325637+0,197218x}$. Funkciu vykreslíme príkazom `Plot[ ]`.

```
Plot[expfit, {x, 0, 10},
 Epilog -> {AbsolutePointSize[2],
 Map[Point, data], Line[data]}]
```



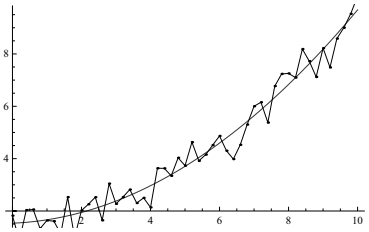
Obr. 8.6. Exponenciálna regresia dát funkciou $f(x) = e^{0,325637+0,197218x}$.

Aproximujme tie isté dáta funkciou v polynomickom tvare $p(x) = a + bx + cx^2$.

```
p = a + b x + c x^2;
abc = FindFit[data, p, {a, b, c}, x]
polfit = p /. abc
```

$\{a \to 1.53437, b \to 0.0531806, c \to 0.0761705\}$
$1.53437 + 0.0531806x + 0.0761705x^2$

```
Plot[polfit, {x, 0, 10},
Epilog -> {AbsolutePointSize[2],
Map[Point, data], Line[data]}]
```



Obr. 8.7. Regresia dát funkciou $f(x) = 1,53437 + 0,0531806x + 0,0761705x^2$.

Našli sme regresiu v tvare $p(x) = 1,53437 + 0,0531806x + 0,0761705x^2$. Funkciu môžeme jednoducho vykresliť príkazom `Plot`. □

Možné metódy aproximácie funkcie, lineárnej a nelineárnej regresie v programe *Mathematica* sú nasledujúce:

`lagrangeInterpolation[xx, yy, x]` Nájde Lagrangeov polynóm neznámej x interpolujúci hodnoty xx a yy. `newtonInterpolation[xx, yy, x]` Nájde Newtonov polynóm neznámej x interpolujúci hodnoty xx a yy. `InterpolatingPolynomial[data, x]` Nájde interpolačný polynóm neznámej x interpolujúci `data`. `Interpolating[data, x]` Nájde interpolačnú funkciu neznámej x, ktorá je po častiach polynóm tretieho stupňa interpolujúci `data`. `Fit[data, basis, var]` Nájde lineárnu kombináciu funkcii premennej `var` zo zoznamu v `basis`, ktorá sa zhoduje s dátami `data` v zmysle najmenších štvorcov. `FindFit[data, funct, params, var]` Nájde parametre `params` funkcie premennej `var` tak, že sa funkcia zhoduje s dátami `data` v zmysle najmenších štvorcov.

8.4 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Nech sú dané nasledujúce dáta:

```
<<Statistics‘NormalDistribution‘
SeedRandom[2];
data = Table[{x,
  Exp[0.3 + 0.1 x] + 0.5 Random[NormalDistribution[0, 1]]},
  {x, 0, 10, 0.2}];
```

a) Nájdite lineárnu regresiu dát, vykreslite funkciu, dáta a reziduálne hodnoty;

b) Nájdite aproximáciu dát polynómom tretieho stupňa, vykreslite polynóm, dáta a reziduálne hodnoty;

c) Nájdite funkciu v exponenciálnom tvare $t(x) = ae^{bx}$ aproximujúcu dáta, vykreslite funkciu, dáta a reziduálne hodnoty.

2 Nech sú namerané dáta rastu kvasiniek v hodinových intervaloch $0, 1, 2, \ldots, 18$ dané:

```
yeast = {9.6, 18.3,29.0,47.2,71.1,119.1,174.6,257.3,350.7,441.0,
   513.3,559.7,594.8,629.4,640.8,651.1,655.9,659.6,661.8};
```

a) Nájdite aproximáciu dát v tvare $l(x) = \frac{a}{1+be^{-cx}}$, vykreslite funkciu premennej x, a dáta;

b) Nájdite aproximáciu dát v tvare $m(t) = \frac{M}{1+\left(\frac{M}{y_0}-1\right)e^{-rMt}}$, vykreslite funkciu premennej t, a dáta;

c) Nájdite aproximáciu dát v tvare $t(x) = a+b\sin(x)+c\sin(2x)+d\cos(x)+e\cos(2x)$, vykreslite funkciu premennej x, a dáta.

3 Majme zoznam prvočísel daný nasledovným postupom:

```
primes = Table[Prime[x], {x, 20}]
```

a) Nájdite aproximáciu dát polynómom tretieho stupňa, vykreslite polynóm, dáta a reziduálne hodnoty;

b) Nájdite aproximáciu dát v tvare $l(x) = ax\log(b+cx)$, vykreslite funkciu premennej x, a zoznam prvočísel.

8.5 VÝSLEDKY

1 a) $f(x) = 1,19747+0,23025x$; b) $f(x) = 1,54363-0,11641x+0,0745422x^2-0,00440174x^3$; c) $t(x) = 1,37837e^{0,0983881x}$.

2 a) $a = 663,022$, $b = 71,5763$, $c = 0,546995$; b) $M = 663,022$, $y_0 = 9,13552$, $r = 0,000825002$; c) $t(x) = 372,912+34,9894\sin(x)+37,1333\sin(2x)-64,2355\cos(x)-17,5468\cos(2x)$.

3 a) $f(x) = 0,160784 + 1,14189x + 0,19826x^2 - 0,00392334x^3$; b) $a = 1,42076, b = 1,65558, c = 0,534645$.

Kapitola 9

KUBICKÉ SPLAJNY

Polynóm stupňa $(n-1)$ môže byť skonštruovaný tak, aby prechádzal cez n daných bodov. V mnohých prípadoch nie je výsledná krivka hladkou krivkou prechádzajúcou bodmi, pretože takáto funkcia zahŕňa okrem prípadných "šumov" v dátach aj oscilovanie medzi bodmi. V tejto kapitole sa venujeme problému ako získať hladkú krivku spájaním bodov po častiach. Tieto časti krivky spojíme tak, aby bola spojitá prvá a druhá derivácia celej krivky. Krivky, ktoré takto získame sú *splajny*. Konkrétne sa v kapitole venujeme kubickým splajnom, ich konštrukcií a kubickým B-splajnom. Kubické splajny sme zvolili preto, že sú priemyselným štandardom pri automatickej výrobe geometrických tvarov a ohýbaní plechov.

9.1 INTERPOLAČNÉ KUBICKÉ SPLAJNY

Predpokladajme, že $(x_0, y_0), (x_1, y_1), \ldots, (x_N, y_N)$ je $N + 1$ bodov, pre ktoré platí $x_0 < x_1 < \ldots < x_N$. Funkcia $S(x)$ sa nazýva *kubický splajn*, ak existuje N kubických polynómov $S_k(x)$ s vlastnosťami ([7] str. 356–358):

$S(x) = S_k(x)$ pre $k = 0, 1, \ldots, N - 1$.

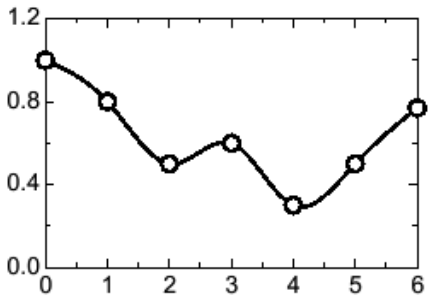
$S_k(x) = s_{k0} + s_{k1}(x - x_k) + s_{k2}(x - x_k)^2 + s_{k3}(x - x_k)^3$ pre $k = 0, 1, \ldots, N - 1$.

$S(x_k) = y_k$ pre $k = 0, 1, \ldots, N$. Splajn prechádza cez každý bod (x_k, y_k).

$S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ pre $k = 0, 1, \ldots, N - 2$. Splajn je spojitý.

$S'_k(x_{k+1}) = S'_{k+1}(x_{k+1})$ pre $k = 0, 1, \ldots, N - 2$. Splajn je hladký.

$S''_k(x_{k+1}) = S''_{k+1}(x_{k+1})$ pre $k = 0, 1, \ldots, N - 2$. Druhá derivácia je spojitá.



Obr. 9.1. Príklad splajnu prechádzajúceho cez zadané body.

Vzťahy kubických kriviek

Pretože $S(x)$ je po častiach kubická, jej druhá derivácia $S''(x)$ je po častiach lineárna na intervale $[x_0, x_N]$. Lineárny Lagrangeov interpolačný polynóm dáva nasledujúce vyjadrenie pre $S''(x) = S''_k(x)$ kde

$$S''_k(x) = S''(x_k)\frac{x - x_{k+1}}{x_k - x_{k+1}} + S''(x_{k+1})\frac{x - x_k}{x_{k+1} - x_k}.$$

Po zavedení označenia

$$m_k = S''(x_k), \qquad m_{k+1} = S''(x_{k+1}), \qquad h_k = x_{k+1} - x_k$$

je možné prepísať Lagrangeovu interpoláciu nasledujúcim spôsobom

$$S_k''(x) = \frac{m_k}{h_k}(x_{k+1} - x) + \frac{m_{k+1}}{h_k}(x - x_k) \qquad k = 0, 1, \ldots, N-1.$$

Dvojnásobným integrovaním tohto vzťahu získame dve integračné konštanty, označme ich p_k a q_k:

$$S_k(x) = \frac{m_k}{6h_k}(x_{k+1} - x)^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + p_k(x_{k+1} - x) + q_k(x - x_k).$$

Substitúciou x_k a x_{k+1} za x v predchádzajúcej rovnici a využitím hodnôt $y_k = S_k(x_k)$ a $y_{k+1} = S_k(x_{k+1})$ dostaneme nasledujúce rovnice pre integračné konštanty p_k a q_k:

$$y_k = \frac{m_k}{6}h_k^2 + p_k h_k \qquad y_{k+1} = \frac{m_{k+1}}{6}h_k^2 + q_k h_k.$$

Vyjadrime konštanty p_k, q_k a dosaďme do vyjadrenia $S_k(x)$. Tak dostávame rovnicu:

$$\begin{aligned} S_k(x) = \; & \tfrac{m_k}{6h_k}(x_{k+1} - x)^3 + \tfrac{m_{k+1}}{6h_k}(x - x_k)^3 + (\tfrac{y_k}{h_k} - \tfrac{m_k h_k}{6})(x_{k+1} - x) \\ & + (\tfrac{y_{k+1}}{h_k} - \tfrac{m_{k+1}h_k}{6})(x - x_k). \end{aligned}$$

Všimnite si, že v predchádzajúcom výraze krivky je použitá iba druhá derivácia $m_k = S''(x_k)$. Na nájdenie hodnôt druhej derivácie môžeme použiť prvú deriváciu funkcie krivky:

$$S_k'(x) = -\frac{m_k}{2h_k}(x_{k+1} - x)^2 + \frac{m_{k+1}}{2h_k}(x - x_k)^2 - (\frac{y_k}{h_k} - \frac{m_k h_k}{6}) + (\frac{y_{k+1}}{h_k} - \frac{m_{k+1}h_k}{6}).$$

Úpravou tohto výrazu v bode x_k dostaneme

$$\begin{aligned} S_k'(x_k) &= -\frac{m_k}{3}h_k - \frac{m_{k+1}}{6}h_k + d_k \\ d_k &= \frac{y_{k+1} - y_k}{h_k}. \end{aligned}$$

Aby sme odvodili deriváciu $S_{k-1}'(x_k)$ zameníme k za $k-1$ a vypočítame ju v bode x_k

$$S_{k-1}'(x_k) = \frac{m_k}{3}h_{k-1} + \frac{m_{k-1}}{6}h_{k-1} + d_{k-1}.$$

Z predchádzajúcich dvoch výrazov a zo spojitosti prvej derivácie $S_k'(x_k) = S_{k-1}'(x_k)$, získavame dôležitý vzťah dávajúci do súvisu m_{k-1}, m_k a m_{k+1}

$$\begin{aligned} h_{k-1}m_{k-1} + 2(h_{k-1} + h_k)m_k + h_k m_{k+1} &= u_k, \\ u_k = 6(d_k - d_{k-1}) \qquad & k = 1, 2, \ldots, N-1. \end{aligned}$$

9.2 KONŠTRUKCIA KUBICKÝCH KRIVIEK

Predchádzajúce vzťahy tvoria $N-1$ rovností pre $(N+1)$ neznámych derivácii m_k. Musíme ešte pridať dve rovnice, ktoré sú nevyhnutné pre odvodenie m_0 z prvej rovnice a odvodenie m_n z rovnice $(N-1)$. Môže byť použitých niekoľko podmienok pre koncové body, najčastejšie sú:

$$m_0 = m_N = 0.$$

Splajnová krivka s týmito podmienkami sa nazýva prirodzený kubický splajn. Pre prirodzený kubický splajn môžeme napísať nasledujúci systém rovníc pre neznáme parametre druhej derivácie

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_1 & b_2 & c_2 & & \\ & & \dots & & \\ & & a_{N-3} & b_{N-2} & c_{N-2} \\ & & & a_{N-2} & b_{N-1} \end{bmatrix} \begin{Bmatrix} m_1 \\ m_2 \\ \dots \\ m_{N-2} \\ m_{N-1} \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \\ \dots \\ u_{N-2} \\ u_{N-1} \end{Bmatrix}, \tag{9.1}$$

$$a_k = h_k, \qquad b_k = 2(h_{k-1} + h_k), \qquad c_k = h_k.$$

Nakoniec dostávame nasledujúce výrazy pre časti prirodzeného kubického splajnu

$$S_k(x) = s_{k0} + s_{k1}(x - x_k) + s_{k2}(x - x_k)^2 + s_{k3}(x - x_k)^3,$$

$$s_{k0} = y_k, \qquad s_{k1} = d_k - \frac{h_k(2m_k + m_{k+1})}{6},$$

$$s_{k2} = \frac{m_k}{2}, \qquad s_{k3} = \frac{m_{k+1} - m_k}{6h_k}.$$

Príklad 37. *Prirodzený kubický splajn. Nájdite prirodzený kubický splajn, ktorý prechádza bodmi* $(0,0), (1,0.5), (2,2)$ *a* $(3,1.5)$.

Riešenie. Vypočítajme hodnoty potrebné pre systém rovníc 9.1:

$$\begin{aligned} h_0 = h_1 = h_2 &= 1, \\ d_0 = (y1 - y0)/h0 &= 0,5, \\ d_1 = (y2 - y1)/h1 &= 1,5, \\ d_2 = (y3 - y2)/h2 &= -0,5, \\ u_1 = 6(d_1 - d_0) &= 6,0, \\ u_2 = 6(d_2 - d_1) &= -12,0. \end{aligned}$$

Systém rovníc určujúci druhé derivácie splajnovej krivky má nasledujúci tvar

$$\begin{bmatrix} 2(h_0+h_1) & h_1 \\ h_1 & 2(h_1+h_2) \end{bmatrix} \begin{Bmatrix} m_1 \\ m_2 \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix}.$$

Po substitúcii parametrov systém má nasledujúci tvar

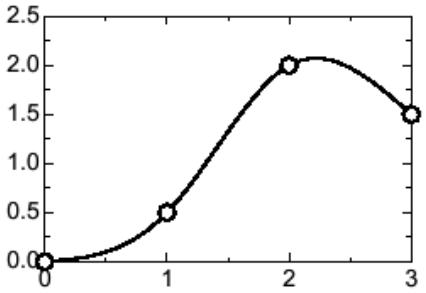
$$\begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{Bmatrix} m_1 \\ m_2 \end{Bmatrix} = \begin{Bmatrix} 6 \\ -12 \end{Bmatrix}.$$

Riešenie systému rovníc je:

$$m_1 = 2,4, \qquad m_2 = 3,6.$$

Využitím hodnôt druhej derivácie môžeme po častiach kubickú splajnovú krivku písať v tvare, (pozri obrázok 9.2.)

$$\begin{aligned} S_0(x) &= 0.4x^3 + 0.1x, && 0 \le x \le 1, \\ S_1(x) &= -(x-1)^3 + 1.2(x-1)^2 + 1.3(x-1) + 0.5, && 1 \le x \le 2, \\ S_2(x) &= 0.6(x-2)^3 - 1.8(x-2)^2 + 0.7(x-2) + 2, && 2 \le x \le 3. \end{aligned}$$



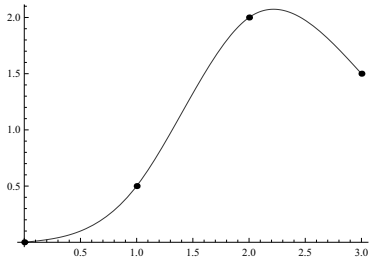
Obr. 9.2. Prirodzený kubický splajn z príkladu 37.

V programe *Mathematica* [15] vypočítajme kubický interpolačný splajn

```
data = {{0, 0}, {1, 0.5}, {2, 2}, {3, 1.5}};
cub = SplineFit[data, Cubic];
```

Použili sme funkciu `SplineFit[%]`, body máme zadané ako zoznam vrcholov v zozname `data`, s nastavením voľby výpočtu kubického splajnu parametrom `Cubic`. Výsledný splajn je funkcia (pozri obr. 9.3.). □

```
ParametricPlot[cub[t], {t, 0, 3},
   Compiled -> False,
   Epilog -> {AbsolutePointSize[5],
   Map[Point, data]}]
```



Obr. 9.3. Graf kubického splajnu.

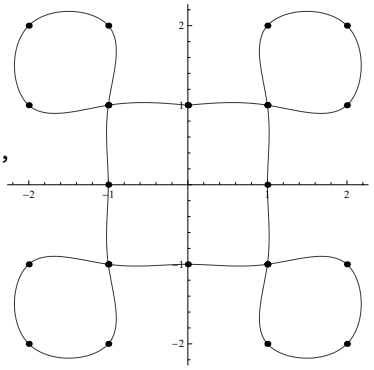
Príklad 38. *V programe Mathematica vypočítajte uzavretý kubický interpolačný splajn daný bodmi* $(0,1)$, $(1,1)$, $(2,1)$, $(2,2)$, $(1,2)$, $(1,1)$, $(1,0)$, $(1,-1)$, $(1,-2)$, $(2,-2)$, $(2,-1)$, $(1,-1)$, $(0,-1)$, $(-1,-1)$, $(-2,-1)$, $(-2,-2)$, $(-1,-2)$, $(-1,-1)$, $(-1,0)$, $(-1,1)$, $(-1,2)$, $(-2,2)$, $(-2,1)$, $(-1,1)$, $(0,1)$.

Riešenie. Zadajme zoznam vrcholov do zoznamu `data2`.

```
data2 = {{0,1},{1,1},{2,1},{2,2},{1,2},{1,1},{1,0},{1,-1},
  {1,-2},{2,-2},{2,-1},{1,-1},{0,-1},{-1,-1},{-2,-1},{-2,-2},
  {-1,-2},{-1,-1},{-1,0},{-1,1},{-1,2},{-2,2},{-2,1},{-1,1},
  {0,1}};
```

Všimnite si, že prvý a posledný bod sú zhodné to preto aby sme krivku uzavreli. Opäť použijeme príkaz `SplineFit`, pozri obr. 9.4. □

```
cub2 = SplineFit[data2, Cubic];
ParametricPlot[cub2[t], {t, 0, 24},
  Compiled -> False,
  Epilog -> {AbsolutePointSize[5],
  Map[Point, data2]}]
```



Obr. 9.4. Graf uzavretého kubického splajnu.

B-splajny

Kubické B-splajny sú podobné obyčajným interpolačným krivkám v tom, že samostatné kubické krivky sú použité pre každý interval a v tom, že B-splajny zaručujú spojitosť krivky ako aj spojitosť prvej a druhej derivácie. Avšak B-splajny nemusia prechádzať cez každý bod množiny, ktorá definuje krivku.

B-splajny môžu byť reprezentované vo forme parametrických rovníc s parametrom u. Dané sú body $p_i = (x_i, y_i), i = 0, 1, \ldots, N$, kubická B-splajnová krivka na intervale (p_i, p_{i+1}) $i = 0, 1, \ldots, N$ je ([7] str. 361–364):

$$\begin{aligned} B_i(u) &= \sum_{k=-1}^{2} b_k p_{i+k}, \\ b_{-1} &= \frac{(1-u)^3}{6}, \\ b_0 &= \frac{u^3}{2} - u^2 + \frac{2}{3}, \\ b_1 &= -\frac{u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6}, \\ b_2 &= \frac{u^3}{6}, \qquad 0 \le u \le 1. \end{aligned}$$

Koeficienty b_k sa nemenia pri prechode z jednej štvorice bodov k druhej. Môžu byť považované za váhové faktory použité na súradnice množiny štyroch bodov. Explicitné rovnice kubickej B-splajnovej krivky medzi bodmi $p_i = (x_i, y_i)$ a $p_{i+1} = (x_{i+1}, y_{i+1})$ v súradnicovom systéme x, y sú nasledujúce ([7] str. 361–364):

$$x(u) = \frac{1}{6}(1-u)^3 x_{i-1} + \frac{1}{6}(3u^3 - 6u^2 + 4)x_i + \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)x_{i+1} + \frac{1}{6}u^3 x_{i+2},$$

$$y(u) = \frac{1}{6}(1-u)^3 y_{i-1} + \frac{1}{6}(3u^3 - 6u^2 + 4)y_i + \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)y_{i+1} + \frac{1}{6}u^3 y_{i+2}.$$

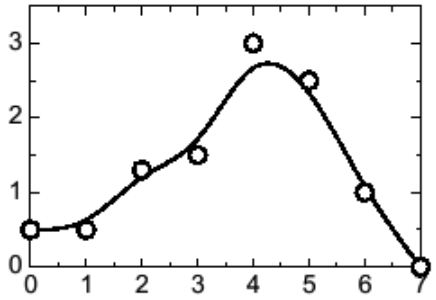
B-splajn zaručuje spojitosť funkcie, a jej prvej a druhej derivácie nasledujúcimi rovnosťami:

$$\begin{aligned} B_i(x) = B_{i+1}(0) &= \frac{p_i + 4p_{i+1} + p_{i+2}}{6}, \\ B_i'(x) = B_{i+1}'(0) &= \frac{-p_i + p_{i+2}}{2}, \\ B_i''(x) = B_{i+1}''(0) &= p_i - 2p_{i+1} + p_{i+2}. \end{aligned}$$

Jeden problém s B-splajnami je, že každá časť krivky vyžaduje štyri body pre jednoznačné určenie. Takýmto spôsobom môžeme zostrojiť B-splajny B_1

až B_{N-2}, ale potrebujeme aj B_0 a B_{N-1}. Jednoduchý spôsob ako doplniť chýbajúce B_0 a B_{N-1}, je pridať fiktívny bod na každý koniec (minimálne z každej strany jeden bod; dva body na každý koniec urobia aproximáciu B-splajnu hladšiu). Je vhodné vyjadriť kubické B-splajny pomocou matíc, lebo tak sa implementujú v programoch:

$$B_i(u) = \frac{1}{6}\left[u^3 u^2 u 1\right] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{Bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{Bmatrix}.$$



Obr. 9.5. B-splajnová aproximácia danými bodmi.

Príklad 39. *V programe Mathematica vypočítajte aproximáciu B-splajnom danými bodmi* $(0,0,5)$, $(1,0,5)$, $(2,1,5)$, $(3,1,6)$, $(4,3)$, $(5,2,5)$, $(6,1)$, $(7,0)$.

Riešenie. Zadajme si najskôr zoznam vrcholov do zoznamu `data`.

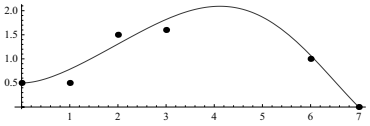
```
data = {{0, 0.5}, {1, 0.5}, {2, 1.5},
   {3, 1.6}, {4, 3}, {5, 2.5}, {6, 1}, {7, 0}};
```

Použijeme funkciu `SplineFit[%]`, body máme zadané v zozname vrcholov `data`, s nastavením voľby výpočtu B-splajnu parametrom `Bezier`. Výsledný splajn je parametrická funkcia prechádzajúca prvým a posledným bodom, (pozri obrázok 9.6.). □

Spôsoby tvorby splajnov v programe *Mathematica* sú nasledujúce:

`SplineFit[data, Cubic]` Kubický splajn interpolujúci body v zozname `data`.
`SplineFit[data, Bezier]` Bezierov splajn (B-splajn) aproximujúci body v zozname `data`.
`SplineFit[data, CompositeBezier]` Zložený B-splajn interpolujúci prvý, tretí, piaty, ... bod v zozname `data`.

```
bezier = SplineFit[data, Bezier];
ParametricPlot[cub2[t],
  {t, 0, 8},
  Compiled -> False,
  Epilog -> {AbsolutePointSize[5],
  Map[Point, data]}]
```



Obr. 9.6. Graf B-splajnu.

9.3 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Uvažujme body $(0,0)$, $(1,0)$, $(1,1)$, $(0,1)$.
a) Nájdite kubický splajn interpolujúci body;
b) Nájdite B-splajn aproximujúci body.

2 Uvažujme body $(0,0)$, $(0,0)$, $(1,0)$, $(1,0)$, $(1,1)$, $(1,1)$, $(0,1)$, $(0,1)$, kde každý z bodov je uvedený v zozname dva krát.
a) Nájdite kubický splajn interpolujúci body;
b) Nájdite B-splajn aproximujúci body.

3 Nájdite body tak, aby
a) Kubický splajn interpolujúci body tvoril kruh;
b) B-splajn aproximujúci body tvoril kruh.

Kapitola 10

NUMERICKÉ DERIVOVANIE A INTEGROVANIE

Vzorce na výpočet numerickej derivácie a vzorce na numerický výpočet integrálov nájdete v tejto kapitole za predpokladu, že hľadané derivácie a integrály existujú. Derivácie, diferenciálny počet reálnych funkcií reálnych premenných nájde čitateľ detailnejšie popísané v [4], [11]. V tejto kapitole uvádzame súbor techník. Venujeme sa hlavne jednodimenzionálnym integrálom na uzavretom intervale, uvedieme aj niekoľko príkladov výpočtu integrálov so singularitami a viacdimenzionálne prípady. Integrálnemu počtu, analytickému a numerickému výpočtu rôznych typov integrálov sa venuje naša predchádzajúca publikácia [3], ktorú odporúčame k detailnejšiemu štúdiu.

10.1 NUMERICKÁ APROXIMÁCIA DERIVÁCIE

Jednou z prirodzených ciest vyjadrenia aproximácie derivácie $f'(x)$ funkcie $f(x)$ je použitie definície jednostrannej derivácie ([4] str. 312-316)

$$f'_+(x) = \lim_{h\to 0^+} \frac{f(x+h)-f(x)}{h} = \frac{f(x+h)-f(x)}{h} + E(h).$$

Táto aproximácia sa nazýva *dopredná diferencia* pre $h > 0$. Prvá otázka, ktorú si zodpovieme je, ako presnosť závisí od h? Ak f je dvakrát diferencovateľná, potom z Taylorovej vety ([4] str. 351-352) je chyba rovná

$$\begin{aligned} E(h) &= (f(x) + hf^{'}(x) + h^2 f^{''}(\xi)/2 - f(x))/h - f^{'}(x) \\ &= hf^{''}(\xi)/2 = O(h) \end{aligned}$$

kde $x \le \xi \le x+h$. Symbol $O(h)$ označuje, že chyba je rádu h teda, že chyba je v tvare Ch, kde $C > 0$ je konštanta.

10.1.1 Vzorec centrálnej diferencie

Ak vieme vypočítať hodnoty funkcie $f(x)$ zľava a sprava okolo bodu x, potom najlepší dvojbodový vzorec vyžaduje funkčné hodnoty v symetrických bodoch

$$f^{'}(x) = \frac{f(x+h)-f(x-h)}{2h} + E(h),$$

nazývame ho *centrálnou diferenciou*. Aby sme odvodili chybu centrálnej diferencie napíšeme Taylorov rozvoj pre $f(x+h)$ a $f(x-h)$

$$\begin{aligned} f(x+h) &= f(x) + hf^{'}(x) + \frac{h^2}{2!}f^{''}(x) + \frac{h^3}{3!}f^{(3)}(\xi) \\ f(x-h) &= f(x) - hf^{'}(x) + \frac{h^2}{2!}f^{''}(x) - \frac{h^3}{3!}f^{(3)}(\xi). \end{aligned}$$

Odčítaním týchto dvoch rovníc získame chybu aproximácie

$$E(h) = -\frac{h^2}{3!}f^{(3)}(\xi) = O(h^2).$$

10.1.2 Druhá derivácia

Druhú deriváciu funkcie $f^{''}(x)$ môžeme aproximovať s chybou $O(h^2)$ ak sčítame Taylorov rozvoj funkčných hodnôt $f(x+h)$ a $f(x-h)$ dostaneme

$$f^{''}(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h^2).$$

Príklad 40. *V programe Mathematica [15] vyrátajte prvú, druhú a tretiu deriváciu funkcie* $f(x) = e^{-x^2}$ *v bode* 1 *numericky príkazom* `ND[%]`*, potom analyticky príkazom* `D[%]` *a nakoniec porovnajme výsledky.*

Riešenie. Najskôr zadajme funkciu `f`. Príkaz `ND[f,x,i,1]` numericky vyráta i-tu deriváciu podľa x v bode 1. Podobne príkaz `Table[D[f,x,i],i,3]` vyráta analyticky i-tu deriváciu podľa x ako funkciu premennej x. Deriváciu v bode 1 získame dosadením, `/.x->1.0`.

```
Needs["NumericalCalculus‘"]
f = Exp[-x^2];
a = Table[ND[f, {x, i}, 1], {i, 3}]
b = Table[D[f, {x, i}], {i, 3}] /. x -> 1.0
```

$\{-0.735759, 0.73576, 1.47147\}$
$\{-0.735759, 0.735759, 1.47152\}$

Numericky vyrátané derivácie sú $f^{'}(1) = -0,735759$, $f^{''}(1) = 0,73576$, $f^{'''}(1) = 1,47147$ s presnosťou $1,76167{\cdot}10^{-9}$, $7,28006{\cdot}10^{-7}$, $-0,00004$. Rozdiel medzi numerickým algoritmom a analytickým výpočtom dostaneme:

```
a - b
```

$1,76167 \times 10^{-9}$, $7,28006 \times 10^{-7}$, $-0,0000482498$

□

10.2 NUMERICKÁ INTEGRÁCIA

Numerický výpočet integrálov je odvodený od integrácie interpolačných polynómov prechádzajúcich cez uzlové body. Metódy založené na integrácií Newtonových interpolačných polynómov nazývame Newton-Cotesove integračné vzorce.

10.2.1 Lichobežníkové pravidlo

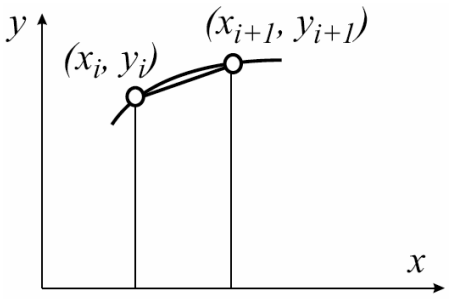
Lichobežníkové pravidlo je metóda numerickej integrácie odvodená z integrácie lineárnej interpolácie medzi dvoma uzlovými bodmi [3]. Zapísané matematicky

$$I = \int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{f(x_i) + f(x_{i+1})}{2}\Delta x = \frac{h}{2}(f_i + f_{i+1}),$$

kde $\Delta x \equiv h = x_{i+1} - x_i$. Ak rozdelíme uzavretý interval $[a, b]$ na n podintervalov dĺžky h dostaneme zložené lichobežníkové pravidlo ([14] str. 91)

$$I = \int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{h}{2}(f_i + f_{i+1}) = \frac{h}{2}(f_1 + 2f_2 + 2f_3 + \ldots + 2f_n + f_{n+1}).$$

Obrázok 10.1. znázorňuje funkciu f a jej lineárnu aproximáciu nad dvoma uzlami. Chyba lichobežníkovho pravidla ([7] str. 379–380), pre jeden krok



Obr. 10.1. Lichobežníkové pravidlo nad bodmi (x_i, y_i) a (x_{i+1}, y_{i+1}) kde $y_i = f(x_i)$.

integrácie o dĺžke h je

$$e(h) = -\frac{1}{12}h^3 f''(\xi) = O(h^3), \qquad x_i < \xi < x_{i+1}.$$

Celková chyba zloženého lichobežníkoveho pravidla je súčtom chýb všetkých intervalov

$$e(h) = -\frac{1}{12}h^3 \sum_{i=1}^{n} f''(\xi_i) = O(h^3).$$

Vytvorme príkaz na výpočet zloženého lichobežníkoveho pravidla *lichobeznik[ ]* v programe *Mathematica*:

```
lichobeznik[f_, x_, a_, b_, n_] :=
 With[{h = (b-a)/n},
  h/2 (N[f /. x -> a] + 2 Sum[N[f /. x -> a + i h], {i, 1, n-1}]
   + N[ f/. x -> b])]
```

V tomto programe rátame funkčnú hodnotu v bode $a+ih$ numericky príkazom `N[f /. x -> a + i h]`. Uzatvorenie príkazom `With[%]` definuje h ako lokálnu konštantu. Počet pásov potrebných k výpočtu integrálu je daný vstupnou konštantou `n`. Použime uvedený príkaz na výpočet $I = \int_0^2 x \sin x dx$:

```
f = x Sin[x];
NInt = lichobeznik[f, x, 0, 2, 50]
```

1.7416

Výsledok numerického integrovania je $I = 1,7416$, pri počte 50 deliacich uzlov. Presnosť nájdeného riešenia, $0,0000102739$, zistíme pomocou rozdielu od analytického riešenia:

```
f = x Sin[x];
{ NInt = lichobeznik[f, x, 0, 2, 50],
  Int = Integrate[f, {x, 0., 2}] };
NInt - Int
```

0.0000102739

10.3 ROMBERGOVO PRAVIDLO

Zlepšiť presnosť lichobežníkoveho pravidla môžeme technikou, ktorá je známa ako Rombergova integrácia ([14] str. 94). Predpokladajme, že výsledok lichobežníkoveho pravidla s intervalom dĺžky $h = (b - a)/n$ je I_h a výsledok s intervalom $2h$ je I_{2h}. Vzhľadom na to , že chyba lichobežníkoveho pravidla je úmerná h^2, chyby z intervalov h a $2h$ môžeme zapísať ako:

$$E(h) = ch^2 \qquad E(2h) = c(2h)^2$$

kde c je neznáma konštanta. Na druhej strane, presnú hodnotu integrálu vieme zapísať ako

$$I = I_h + E(h) = I_{2h} + E(2h)$$

alebo

$$E(h) - E(2h) = I_{2h} - I_h.$$

Vyriešme z tejto rovnice c

$$c = \frac{1}{3h^2}(I_h - I_{2h})$$

a substitúciou za $E(h)$ dostaneme nasledujúce Rombergovo integračné pravidlo

$$I = I_h + \frac{1}{3}(I_h - I_{2h}),$$

ktoré nám dáva zlepšenie odhadu integrálu, založeného na hodnotách integrálov s dĺžkami h a $2h$.

10.3.1 Simpsonovo $1/3$ pravidlo

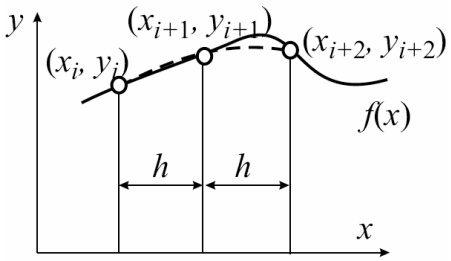
Simpsonovo 1/3-vé pravidlo je založené na kvadratických interpolačných polynómoch [13]. Všeobecný tvar paraboly druhého stupňa prechádzajúcej troma bodmi je

$$y = ax^2 + bx + c.$$

Integráciou tohto polynómu v intervale $\langle -h, h\rangle$ dostávame oblasť zloženú z dvoch pásov integrácie (pozri obrázok 10.2.)

$$I_{2pasy} = \int_{-h}^{h} (ax^2 + bx + c)\, dx = \frac{2}{3}ah^3 + 2ch.$$

Konštanty a a c môžeme nájsť z vlastnosti, že parabola má prechádzať cez



Obr. 10.2. Simpsonovo 1/3-vé pravidlo použitím bodov (x_i, y_i), (x_{i+1}, y_{i+1}) a (x_{i+2}, y_{i+2}), kde $y_i = f(x_i)$.

body $(-h, y_i)$, $(0, y_{i+1})$ a (h, y_{i+2}). Odtiaľ

$$\begin{aligned} y_i &= a(-h)^2 + b(-h) + c, \\ y_{i+1} &= c, \\ y_{i+2} &= ah^2 + bh + c. \end{aligned}$$

Riešením týchto rovníc a nahradením konštánt dostaneme

$$I_{2pasy} = \frac{h}{3}(f_i + 4f_{i+1} + f_{i+2}).$$

Ak rozdelíme oblasť pod krivkou na n rovnakých pásov (n je párne číslo) dostávame nasledujúce Simpsonovo 1/3 zložené pravidlo:

$$I = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \ldots + 2f_{n-2} + 4f_{n-1} + f_n) + O(h^4).$$

10.3.2 Simpsonovo $3/8$ pravidlo

Zložené pravidlo používajúce interpoláciu štyroch bodov polynóm tretieho stupňa vedie k Simpsonovmu 3/8 pravidlu [13]. Začneme priamo s pravidlom integrácie pre tri pásy

$$I_{3pasy} = \frac{3h}{8}(f_i + 3f_{i+1} + 3f_{i+2} + f_{i+3}).$$

Použitím tohto vzťahu na interval pozostávajúci z trojíc pásov získame zložené pravidlo:

$$\begin{aligned} I &= \frac{3h}{8}(f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + 3f_5 + 2f_6 + \\ &\ldots \quad +2f_{n-3} + 3f_{n-2} + 3f_{n-1} + f_n) + O(h^4). \end{aligned}$$

10.3.3 Gaussova kvadratúra

Naše predchádzajúce vzorce numerickej integrácie boli založené na rovnomerne rozložených uzloch na x-ovej osi, čím sme dostali rovnaké podintervaly. Vzorec s troma parametrami (troma bodmi), používa polynóm druhého stupňa. Gaussove kvadaratúrne vzorce používajú n parametrov pre x-ové hodnoty, n parametrov pre váhy a výsledný interpolačný polynóm rádu $2n + 1$. Gaussove kvadratúrne vzorce vieme použiť, ak dokážeme $f(x)$ vypočítať v ľubovoľnej požadovanej hodnote x.

Ukážme si ako sa táto myšlienka použije pri odvodení parametrov Gausovej integrácie v jednoduchom prípade dvoch uzlov s celkovým počtom štyroch parametrov. Vypočítajme nasledujúci integrál

$$I = \int_{-1}^{1} f(x)\,dx$$

a zapíšeme integračný vzorec použitím dvoch uzlov

$$I = w_1 f(x_1) + w_2 f(x_2),$$

kde w_k sú váhy a x_k sú neurčené uzly. Náš vzorec by mal byť presný pre každý polynóm tretieho stupňa. Preto musí platiť presne aj pre funkcie $f(x) = 1$,

$f(x) = x$, $f(x) = x^2$ a $f(x) = x^3$. Zapísaním týchto podmienok dostaneme

$$\begin{aligned}
\int_{-1}^{1} dx &= \quad 2 \quad &= w_1 + w_2,\\
\int_{-1}^{1} x\,dx &= \quad 0 \quad &= w_1x_1 + w_2x_2,\\
\int_{-1}^{1} x^2\,dx &= \quad \tfrac{2}{3} \quad &= w_1x_1^2 + w_2x_2^2,\\
\int_{-1}^{1} x^3\,dx &= \quad 0 \quad &= w_1x_1^3 + w_2x_2^3.
\end{aligned}$$

Hranice určitých integrálov $\langle -1, 1\rangle$ sú symetrické okolo uzla $x = 0$, preto požadujeme, aby uzly x_1, x_2 boli symetrické a položíme $x_2 = -x_1$. Z prvej a druhej rovnice po dosadení dostaneme $w_1 = w_2 = 1$. S týmito hodnotami je štvrtá rovnica automaticky splnená. Z tretej rovnice dostávame $\frac{1}{3} = x_1^2$ a teda

$$x_1 = \frac{1}{\sqrt{3}} = 0,57735026918962576450914878050196.$$

V tomto kroku sme našli všetky koeficienty integračnej formulky a tak sú nám známe váhy w_1, w_2 a uzlové body x_1, x_2, v ktorých budú spočítané funkčné hodnoty. Odvodili sme najjednoduchšie Gausovo integračné pravidlo

$$I = \int_{-1}^{1} f(x)dx = f(0,577) + f(-0,577).$$

Všeobecná Gaussova kvadratúra je vyjadrená nasledovným spôsobom ([14] str. 81–84)

$$\int_{-1}^{1} f(t)\,dt = \sum_{k=1}^{n} w_k f(t_k),$$

kde n je počet integračných bodov, t_k sú uzlové súradnice a w_i sú integračné váhy. V tabuľke 10.1. uvádzame Gaussove integračné pravidlá s použitými váhami w_i a súradnice uzlov t_i pre počty integračných uzlov od 2 po 5.

Zatiaľ vieme počítať integrál funkcie numericky pomocou Gaussových integračných vzorcov na intervale $[-1, 1]$. Gaussovu integráciu môžeme aplikovať aj na ľubovoľný interval $[a, b]$ použitím transformácie integračných hraníc

$$x = \frac{1}{2}(b + a) + \frac{1}{2}(b - a)t.$$

Gaussova kvadratúra potom bude mať nasledujúci tvar

$$\int_a^b f(x)\,dx = \frac{1}{2}(b - a)\sum_{k=1}^{n} w_k f(x_k).$$

n	t_i	w_i
2	$(-/+)0.577350269189626$	1
3	$(-/+)0.774596669241483$	0.555555555555556
	0	0.888888888888889
4	$(-/+)0.861136311594053$	0.347854845137454
	$(-/+)0.339981043584856$	0.652145154862546
5	$(-/+)0.906179845938664$	0.236926885056189
	$(-/+)0.538469310105683$	0.478628670499366
	0	0.568888888888889

Tabuľka 10.1. Tabuľka Gaussových integračných pravidiel s uzlami a váhami pre počet uzlov 2, 3, 4 a 5.

Vytvorme funkciu Gaussovej kvadratúry v programe *Mathematica*. Gaussova kvadratúra na intervale $[-1, 1]$ používa uzly x_i, ktoré sú koreňmi Legendreovho polynómu n-tého stupňa $P_n(x)$ a váhy $w_i = 2/[P_n'(x_i)(1 - x_i^2)]$. Príkazy na výpočet uzlov a váh sú nasledujúce:

```
gaussianPoints[n_] := x /. NSolve[LegendreP[n, x] == 0, x];
gaussianWeights[n_] := With[{points = gaussianPoints[n]},
 2/((D[LegendreP[n, x], x] /. x -> points)^2(1 - points^2))]
```

Porovnáme napríklad uzly a váhy Gaussovho integračného pravidla pre počet uzlov 3 s tabuľkou 10.1.

```
gaussianPoints[3]
gaussianWeights[3]
```

$\{-0.774597, 0., 0.774597\}$
$0.555556, 0.888889, 0.555556$

Gaussovu kvadratúru na intervale $[a, b]$ vypočítame transformáciou $c+dt$, kde $c = (a + b)/2$ a $d = (b - a)/2$ pomocou algoritmu:

```
gaussianQuadrature[f_, x_, a_, b_, n_] :=
  With[{points = gaussianPoints[n], weights = gaussianWeights[n],
    c = (a + b)/2, d = (b - a)/2},
    d weights.(f /. x -> c + d points)];
```

Príklad 41. *V programe Mathematica pomocou Gaussovho integračného pravidla vypočítajte integrál* $I = \int_0^2 \sin(x)dx$, *použite tri uzly.*

Riešenie. Použijeme algoritmus `gaussianQuadrature[%]` a potom vypočítame chybu ako odchýlku od analytického riešenia.

```
Int = gaussianQuadrature[Sin[x], x, 0, 2, 3]
chyba = Integrate[Sin[x], {x, 0, 2}] - Int
```

1.4162
-0.0000518162

Numerický výsledok kvadratúry je $I = 1,4162$ chyba vypočítaného riešenia je $-0,00005$. □

Príklad 42. *Vypočítajte integrály z nespojitých funkcií ([3] str.252–253)*
a) $\iint\limits_A \operatorname{sgn}(x^2 - y^2 + 2)\, dx\, dy$, *kde* $A = \{(x, y) \in R^2; x^2 + y^2 \leq 4\}$.
b) $\iint\limits_A [y - x^2]\, dx\, dy$, *kde* $A = \{(x, y) \in R^2; x^2 \leq y \leq 4\}$ *a* $[f]$ *označuje celú časť funkcie* f.

Riešenie. Základné nespojité funkcie sú reprezentované v programe *Mathematica* delením na podoblasti v bodoch nespojitosti a môžu byť použité tiež v algebraických výrazoch.

a) Integračnú oblasť zadáme príkazom `Boole[ ]` a numerickú integráciu vypočítame príkazom `Integrate[ ]` ([15])

```
<< Calculus‘Integration‘
Integrate[Sign[x^2 - y^2 + 2] Boole[ x^2 + y^2 <= 4],
 {x, -Infinity, Infinity}, {y, -Infinity, Infinity}]
N[%]
```

$\frac{4}{3}\left(\pi + 6\text{ArcSinh}\left[\frac{1}{\sqrt{2}}\right]\right)$

Výsledok integrálu je výraz $\frac{4}{3}\left(\pi + 6\text{ArcSinh}\left[\frac{1}{\sqrt{2}}\right]\right) = 9,45662$.

b) Celá časť je daná funkciou `Floor[ ]`. V programe *Mathematica* vypočítame integrál efektívne príkazom `Integrate[ ]`:

```
<< Calculus‘Integration‘
Integrate[Floor[y - x^2 + 2]Boole[x^2 <= y && y <= 4],
 {x, -Infinity, Infinity}, {y, -Infinity, Infinity}]
N[%]
```

$\frac{4}{3}\left(17 + 2\sqrt{2} + 3\sqrt{3}\right)$

Výsledok integrálu po vyčíslení výsledku je $33,3661$. □

Príklad 43. *V programe Mathematica vieme vypočítať aj nevlastné násobné integrály. Vypočítajte oba integrály I_1 a I_2 v programe Mathematica a porovnajme výsledky.*

$$I_1 = \int_1^\infty dx \int_1^\infty \frac{x^2 - y^2}{(x^2 + y^2)^2}\, dy, \qquad I_2 = \int_1^\infty dy \int_1^\infty \frac{x^2 - y^2}{(x^2 + y^2)^2}\, dx.$$

Riešenie. Použijeme príkaz `Integrate[ ]` dva krát s príslušnými hranicami na výpočet I_1 a nakoniec vyčíslime výsledok numericky príkazom `N[%]`.

```
<< Calculus‘Integration‘
I1 = Integrate[
 Integrate[ (x^2 - y^2)/(x^2 + y^2)^2, {y, 1, Infinity}],
 {x, 1, Infinity}]
N[%]
```

-0.785398

Výsledok prvého integrálu je $I_1 = -0,785398$.

Teraz rátajme integrál I_2 s vymeneným poradím premenných

```
<< Calculus‘Integration‘
I2 = Integrate[
 Integrate[ (x^2 - y^2)/(x^2 + y^2)^2, {x, 1, Infinity}],
 {y, 1, Infinity}]
N[%]
```

0.785398

Výsledok druhého integrálu je $I_2 = 0,785398$. Porovnaním výsledkov $I_1 \neq I_2$. Všimnime si, že išlo o zámenu poradia integrácie. Týmto príkladom sme ukázali, že nemôžeme ľubovoľne zamieňať poradie integrácie v prípade nevlastných násobných integrálov ([3] str. 121-123). □

Príklad 44. *V programe Mathematica vypočítajte nevlastný integrál*

$$a) \int_A \frac{dx\,dy}{\sqrt{1 - x^2 - y^2}},$$

kde $A = \{(x, y) \in R^2;\ x^2 + y^2 \leq 1\}$*;*

$$b) \int_A \frac{dx\,dy}{x^2 + y^2},$$

kde $A = \{(x, y) \in R^2;\ x^2 + y^2 \leq 1\}$.

Riešenie. a) Integrál vypočítame príkazom `Integrate[ ]` v programe *Mathematica*, singularitu ošetrí program automaticky a výsledok je 2π. Príkaz `Boole[%]` slúži na definíciu integračnej oblasti.

```
<< Calculus‘Integration‘
Integrate[1/Sqrt[1 - x^2 - y^2] Boole[ x^2 + y^2 <= 1 ],
 {x, -Infinity, Infinity}, {y, -Infinity, Infinity}]
```

2π

b) Ak sa snažíme vypočítať integrál priamo príkazom `Integrate[ ]`, program *Mathematica* zistí, že integrál nekonverguje na oblasti A a tento výsledok aj oznámi správou.

```
<< Calculus‘Integration‘
Integrate[1/(x^2 + y^2) Boole[ x^2 + y^2 <= 1 ],
 {x, -Infinity, Infinity}, {y, -Infinity, Infinity}]
```

Integral does not converge.

□

Príklad 45. *Vypočítajte nevlastný integrál*

$$\int_0^{\infty} dy \int_0^{y} e^{-(x+y)} dx = \frac{1}{2}.$$

Riešenie. Tento integrál vieme vypočítať príkazom `Integrate[ ]` pomocou programu *Mathematica* nasledovným spôsobom:

```
<< Calculus‘Integration‘
Integrate[ Integrate[ Exp[-(x + y)], {x, 0, y}],
 {y, 0, Infinity}]
```

$\frac{1}{2}$

Numerický výsledok integrálu je $\frac{1}{2}$. □

Zhrnieme metódy hľadania numerickej derivácie a integrácie funkcie v programe *Mathematica*.

`ND[f, x, a]` Prvá derivácia f podľa x v bode a.
`ND[f, x, n, a]` n-tá derivácia f podľa x v bode a.
`D[f, x]` Prvá derivácia f podľa x, $\frac{\partial f}{\partial x}$.
`D[f, x, x]` Druhá derivácia f podľa x, $\frac{\partial^2 f}{\partial x^2}$.
`D[f, x, y]` Druhá zmiešaná derivácia f podľa x a y, $\frac{\partial^2 f}{\partial x \partial y}$.
`Integrate[f, x, a, b]` Analytický alebo numerický výpočet $\int_a^b f dx$.
`NIntegrate[f, x, a, b]` Numerický výpočet $\int_a^b f dx$.
`lichobeznik[f, x, a, b, n]` Lichobežníková metóda $\int_a^b f dx$ delením na $3n$ uzlov.
`gaussianPoints[n]` Výpočet n uzlových bodov Gaussovej metódy.
`gaussianWeights[n]` Výpočet n váh Gaussovej metódy.
`gaussianQuadrature[f, x, a, b, n]` Gaussova metóda $\int_a^b f dx$ pomocou n uzlov.

10.4 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Vypočítajte numericky prvé derivácie funkcií v bodoch:

a) $f'(8)$, ak $f(x) = \sqrt[3]{x}$;
b) $f'(0), f'(1), f'(2)$, ak $f(x) = x(x-1)^2(x-2)^3$;
c) $f'(1)$, ak $f(x) = \sqrt[5]{x-1}$;
d) $f'(1)$, ak $f(x) = \sqrt{xe^x + x}$;
e) $f'(\pi/2)$, ak $f(x) = \sin(3x) + \cos(\frac{x}{5}) + \tan\sqrt{x}$;
f) $f'(4)$, ak $f(x) = \frac{x^3}{3\sqrt{(1+x^2)^3}}$;
g) $f'(2)$, ak $f(x) = \frac{4}{3}\sqrt[4]{\frac{x-1}{x+2}}$;
h) $f'(\pi/3)$, ak $f(x) = e^{\sin^2 x}$;
i) $f'(2)$, ak $f(x) = \ln(x + \sqrt{1+x^2})$;
j) $f'(0,2)$, ak $f(x) = \arctan\ln\frac{1}{x}$.

2 Vypočítajte numericky prvé derivácie funkcií v bodoch:

a) $f'(0)$, ak $f(x) = e^{-x}\cos 3x$;
b) $f'(1)$, ak $f(x) = \ln(1+x) + \arcsin\frac{x}{2}$.

3 Vypočítajte numericky druhé derivácie funkcií v bodoch:

a) $f''(e)$, ak $f(x) = \frac{\ln x}{x}$;
b) $f''(\pm\sqrt{3})$, ak $f(x) = \frac{x}{\sqrt[3]{x^2-1}}$;
c) $f''(\pm\frac{1}{\sqrt{2}})$, ak $f(x) = e^{-x^2}$;
d) $f''(-2)$, ak $f(x) = \sqrt[3]{x+2}$;
e) $f''(-2)$, ak $f(x) = a\cosh\frac{x}{a}$, zvoľte si $(a > 0)$.

4 Vypočítajte numericky prvé derivácie funkcií v bode x, ktorý si vhodne zvoľte

a) $F(x) = \int_1^x \ln t dt$, $(x > 0)$;
b) $F(x) = \int_x^0 \sqrt{1+t^4} dt$.

5 Vypočítajte numericky Gaussovou kvadratúrou alebo lichobežníkovým pravidlom nasledujúce integrály

a) $\int_0^1 \frac{dx}{1+x}$; b) $\int_2^5 \frac{dx}{\sqrt{5+4x-x^2}}$; c) $\int_1^4 \frac{1+\sqrt{y}}{y^2} dy$;
d) $\int_e^{e^2} \frac{dx}{x\ln x}$; e) $\int_1^e \frac{\sin(\ln x)}{x} dx$; f) $\int_1^2 (x^2 - 2x + 3) dx$;

6 Vypočítajte nasledujúce nevlastné integrály

a) $\int_0^\infty \frac{dx}{1+x^2}$; b) $\int_0^\infty e^{-x^2} dx$; c) $\int_0^{\frac{1}{2}} \frac{dx}{x\ln x}$;
d) $\int_0^{\frac{1}{2}} \frac{dx}{x\ln^2 x}$; e) $\int_0^\infty \frac{dx}{(x^2-1)^2}$;

10.5 VÝSLEDKY

1 a) $f\frac{1}{12}$; b) $f'(0) = -8$, $f'(1) = 0$; c) $\pm\infty$; d) $f'(x) = \frac{e^x+xe^x+1}{2\sqrt{xe^x+x}}$; f) $f'(x) = \frac{x^2}{\sqrt{(1+x^2)^5}}$; g) $f'(x) = \frac{1}{\sqrt[4]{(x-1)^3(x+2)^5}}$; h) $f'(x) = \sin 2x e^{\sin^2 x}$; i) $f'(x) = \frac{1}{\sqrt{a^2+x^2}}$; j) $f'(x) = -\frac{1}{x(1+\ln^2 x)}$.

2 a) $f'(0) = -1$; b) $f'(1) = \frac{1}{2} + \frac{\sqrt{3}}{3}$.

3 a) $f''(x) = \frac{2\ln x - 3}{x^3}$; b) $f''(x) = \frac{2x(9-x^2)}{9\sqrt{(x^2-1)^7}}$; c) $f''(x) = (4x^2 - 2)e^{-x^2}$; d) $f''(x) = \frac{-2}{9\sqrt[3]{(x+2)^5}}$, $f''(-2)$ neexistuje; e) $f''(x) = \frac{1}{a}\cosh\frac{x}{a}$;

4 a) $F'(x) = \ln(x)$; b) $F'(x) = -\sqrt{1+x^4}$.

5 a) $\ln 2$; b) $\frac{\pi}{2}$; c) $\frac{7}{4}$; d) $\ln 2$; e) $1 - \cos 1$; f) $\frac{7}{3}$.

6 a) $\frac{\pi}{2}$; b) e^{-1}; c) diverguje; d) $\frac{1}{\ln 2}$; e) $\frac{1}{3} + \frac{1}{4}\ln 3$.

Kapitola 11

OBYČAJNÉ DIFERENCIÁLNE ROVNICE A METÓDA KONEČNÝCH DIFERENCIÍ

V tejto kapitole popíšeme niektoré základné numerické metódy riešenia obyčajných diferenciálnych rovníc so začiatočnou podmienkou [12]. Uvedieme Eulerovu metódu, prediktor-korektor metódu, Runge-Kutta metódy, viackrokové metódy a metódy na riešenie systému diferenciálnych rovníc. Pri použití týchto metód predpokladáme, že existuje riešenie daných úloh.

11.1 PROBLÉM SO ZAČIATOČNOU HODNOTOU

Nech I je interval $[a, b] \subseteq \mathcal{R}$. Hľadajme reálne riešenie začiatočnej úlohy pre rovnicu prvého rádu

$$y'(x) = f(x, y), \text{ so začiatočnou podmienkou } y(a) = y_a,$$

kde $f(x, y)$ je reálna a spojitá funkcia na $I \times (-\infty, +\infty)$ v premenných x a y. Reálnym riešením predchádzajúcej, tzv. *Cauchyho úlohy*, nazývame reálnu

funkciu $u(x) \in C^1(I)$, ktorá spĺňa rovnicu

$$u'(x) = f(x, u(x)), \text{ a podmienku } u(a) = y_a$$

pre všetky $x \in I$.

11.1.1 Eulerova metóda

Najjednoduchšia metóda je odvodená z rozvoja Taylorovho radu ([2] str.129–185)

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(\xi_n), \\ h &= x_{n+1} - x_n, \qquad x_n < \xi_n < x_n + h. \end{aligned}$$

Algoritmus Eulerovej metódy ([7] str. 482–484) je nasledujúci:

$$y_{n+1} = y_n + hf(x_n, y_n).$$

Dostali sme tak rekurzívny vzťah, ktorý nám umožní vypočítať v nasledujúcom kroku riešenie $y_{n+1} = y(x_{n+1})$ na základe riešenia známeho v bode x_n, $y_n = y(x_n)$. Z Taylorovho rozvoja sa dá odvodiť chyba jedného kroku Eulerovej metódy

$$\epsilon_{n+1} = \frac{h^2}{2}y''(\xi_n) = O(h^2).$$

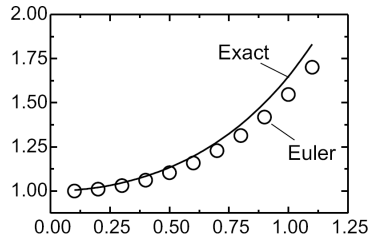
Celková chyba po N rekurzívnych krokoch je súčet lokálnych chýb

$$E(h) = \sum_{n=1}^{N} \epsilon_{n+1} = NCh^2 = \frac{b-a}{h}Ch^2 = O(h),$$

kde N je počet členov x_n a $C > 0$ je konštanta. Obrázok 11.1. znázorňuje bodmi aproximované riešenie $y(x_n)$ presného riešenia daného spojitou krivkou. Pozorujeme hlavný nedostatok Eulerovej metódy v tom, že numerické riešenie horšie odhaduje presné riešenie pre posledné členy, ktoré zahŕňajú chyby z predchádzajúcich členov.

Môžeme naprogramovať vlastný program `euler[ ]` na riešenie podľa popisu Eulerovej metódy v programe *Mathematica* [15]. Program ráta riešenie na intervale $[a, b]$ s n krokmi a začiatočnou podmienkou v bode $y(a) = y_a$ uloženou v premennej `ya`. Pravú stranu rovnice zadáme v premennej `f`. Príkaz `Prepend` pridá začiatoný bod do vyrátanej tabuľky riešení.

```
euler[f_, x_, y_, a_, ya_, b_, n_] :=
 Module[{xi = a, yi = ya, h = N[(b - a)/n]},
  Prepend[Table[{xi, yi} =
     N[{x + h, y + h f} /. {x -> xi, y -> yi}], {n}], {a, ya}]]
```



Obr. 11.1. Porovnanie presného riešenia `Exact` s riešením podľa Eulerovej metódy `Euler`.

K dispozícii máme aj príkaz `NDSolve`, ktorý použije presnejšiu numerickú metódu na intervale $[a, b]$. Diferenciálna rovnica sa definuje vo vstupnej premennej `eqn`, začiatočné podmienky v `conds`. Riešenie je vyrátané v niekoľkých uzlových bodoch a nakoniec interpolované B-splajnom. Štandardné použitie je nasledovné:

```
sol = y[t] /. NDSolve[{eqn, conds}, y[t], {t, a, b}]
```

Príklad 46. *Použite program* `euler` *na nájdenie riešenia Cauchyho úlohy* $y' = x - y^2$, $y(0) = 1$ *na intervale* $[0, 1]$ *pre* 10 *krokov. Vyriešte problém aj pomocou príkazu* `NDSolve`.

Riešenie. Zadajme pravú stranu rovnice `x - y^2` a začiatočnú podmienku $y(0) = 1$ ako bod $(x, y) = (0, 1)$.

```
eu = euler[x - y^2, x, y, 0, 1, 1, 10]
```

$\{\{0, 1\}, \{0.1, 0.9\}, \{0.2, 0.829\}, \{0.3, 0.780276\}, \{0.4, 0.749393\},$
$\{0.5, 0.733234\}, \{0.6, 0.729471\}, \{0.7, 0.736258\}, \{0.8, 0.75205\},$
$\{0.9, 0.775492\}, \{1., 0.805354\}\}$

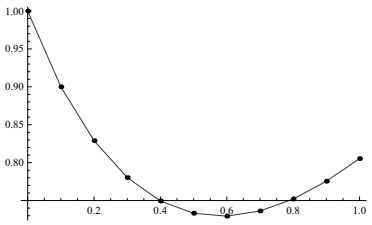
Našli sme riešenie pre jednotlivé kroky. Napríklad pre $x = 1$ nadobúda riešenie hodnotu $y(1) = 0,805354$. Vykreslime si graf riešenia pospájaním vypočítaných hodnôt z pola `eu` úsečkami.

Použime príkaz `NDSolve` [15] na riešenie zadanej úlohy. Vstupom príkazu sú diferenciálna rovnica, začiatočná podmienka, hľadaná funkcia `y` a interval `{x, 0, 1}` zadané príkazom `{y'[x] == {x - y[x]^2, y[0] == 1}`:

```
riesenie =
 y[x] /. NDSolve[{y'[x] == x - y[x]^2, y[0] == 1}, y, {x, 0, 1}];
```

{InterpolatingFunction[{{0., 1.}}, <>][x]}

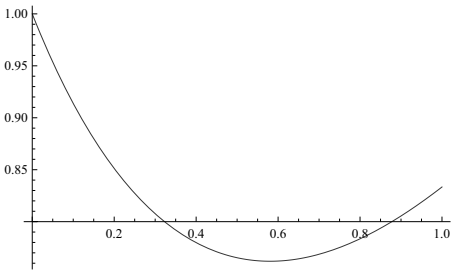
```
ListPlot[eu, PlotJoined -> True,
Epilog -> {AbsolutePointSize[4],
 Map[Point, eu]}]
```



Obr. 11.2. Graf riešenia $y' = x - y^2$, $y(0) = 1$ na intervale $x \in [0, 1]$.

Program *Mathematica* nám oznámil, že našiel riešenie v tvare interpolačnej funkcie na intervale $[0, 1]$. Nájdená krivka je uložená v premennej `riesenie`. Vykreslime graf riešenia príkazom `Plot[ ]` pozri obrázok 11.3. □

```
Plot[riesenie, {x, 0, 1},
 PlotRange -> All]
```



Obr. 11.3. Graf riešenia $y' = x - y^2$, $y(0) = 1$ na intervale $x \in [0, 1]$ príkazom `NDSolve`.

11.1.2 Metóda prediktor-korektor

Metóda prediktor-korektor ([14] str. 168–170, [7] str. 516) sa skladá z dvoch krokov.

1. V prvom kroku, odhadujeme (predict) riešenie v bode a následne zistíme deriváciu riešenia v tomto bode.

2. V druhom kroku, použitím sklonu riešenia, sa vrátime o jeden krok naspäť a vyrátame ďalší o niečo správnejší (correct) odhad riešenia.

Obe metódy prediktor aj korektor by mali mať lokálne chyby rovnakého rádu. Použime vzorec centrálnej diferencie pre krok prediktora

$$y_{n+1} = y_{n-1} + 2hy'_n.$$

Vypočítame sklon y'_{n+1} potom zvolíme priemer dvoch koncových sklonov ako rozumnejšiu voľbu na opravu hodnoty lichobežníkovým pravidlom numerickej integrácie. Náš korektor má tvar

$$y_{n+1} = y_n + h\left(\frac{y'_{n+1} + y'_n}{2}\right).$$

Lokálna chyba zanedbania pre prediktor je

$$E_p(h) = \frac{h^3}{3}y'''(\xi_p)$$

a lokálna chyba zanedbania pre korektor je

$$E_c(h) = -\frac{h^3}{12}y'''(\xi_c).$$

Prvá otázka, ktorú si položíme je ako začať prediktor-korektor metódu. Naša prvá predikcia požaduje, aby sme poznali jeden predošlý bod a aktuálny sklon. Jedna odpoveď je použitie diferenciálnej rovnice a jej derivácie po dosadení do Taylorovho rozvoja

$$y_1 = y_0 + hy'_0 + \frac{h^2}{2!}y''_0 + \frac{h^3}{3!}y'''_0 + \dots$$

v začiatočnom bode y_0. Z rozvoja vypočítame prvý bod riešenia y_1, ktorý použijeme ako štartovacie hodnoty prediktor-korektor metódy. Pretože lokálne chyby prediktora a korektora závisia na tretej derivácii je vhodné použiť Taylorov rozvoj až po člen y'''_0.

11.2 RUNGE-KUTTA METÓDY

Aby sme dosiahli presnosť vyššieho stupňa Taylorovým radom museli by sme nájsť rôzne derivácie $y'(x)$, $y''(x)$, ..., čo je ťažká a často nemožná úloha. Našťastie môžeme nahradiť derivácie vyčíslením $f(x, y)$ v pomocných bodoch tak, aby dosiahli tú istú požadovanú presnosť. Metódy pracujúce touto cestou sa nazývajú Runge-Kutta metódy ([7] str. 518–520, [14] str. 134–145).

11.2.1 Runge-Kutta metóda druhého rádu

Metódu Runge-Kutta odvodíme z rozvoja

$$\begin{aligned} y(x+h) &= y(x) + hy'(x) + \frac{h^2}{2}y''(x) + O(h^3) = \qquad (11.1)\\ &= y(x) + hf(x,y) + \frac{h^2}{2}(f_x(x,y) + f_y(x,y)f(x,y)) + O(h^3) \end{aligned}$$

čo môžeme skrátene zapísať

$$y(x+h) = y(x) + w_1 k_1 + w_2 k_2,$$

kde $k_1 = hf(x,y)$, $k_2 = hf(x+\alpha h, y+\beta k_1)$ a w_1, w_2, α a β sú vybrané tak, aby zachovávali stupeň aproximácie. Motiváciou je, že k_1, k_2 sú aproximácie k Δy. Z rozvoja k_2 v bode (x,y) máme

$$k_2 = h(f(x,y) + \alpha h f_x(x,y) + \beta k_1 f_y(x,y)) + O(h^2)$$

tak, že po dosadení za k_1 a k_2

$$\begin{aligned} y(x+h) &= y(x) + w_1 h f(x,y) + \\ &+ w_2 h[f(x,y) + \alpha h f_x(x,y) + \beta h f(x,y) f_y(x,y)] + O(h^3) = \\ &= y(x) + (w_1 + w_2) h f(x,y) + \\ &+ w_2 h^2 [\alpha f_x(x,y) + \beta f(x,y) f_y(x,y)] + O(h^3). \end{aligned} \tag{11.2}$$

Porovnaním posledných dvoch rovníc 11.1 a 11.2 pre $y(x+h)$ dostaneme tri rovnice

$$w_1 + w_2 = 1, \qquad w_2\alpha = 1/2, \qquad w_2\beta = 1/2.$$

pre štyri neznáme. Obyčajne je najlepšie uvažovať váhy w_1 a w_2 rovnaké. Potom zvolíme $w_1 = w_2 = 1/2$ a $\alpha = \beta = 1$.

Algoritmus Runge-Kutta druhého rádu je teda nasledujúci

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2 \\ k_1 &= hf(x_n, y_n), \qquad k_2 = hf(x_{n+1}, y_n + k_1), \end{aligned}$$

a lokálna chyba riešenia je rádu $O(h^3)$.

11.2.2 Runge-Kutta metóda štvrtého rádu

Metóda Runge-Kutta štvrtého rádu ([5] str. 290) je nasledujúca

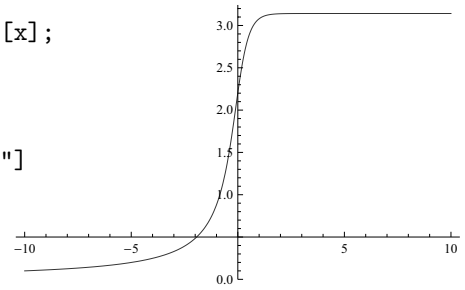
$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= hf(x_n, y_n), \\ k_2 &= hf(xn + \frac{h}{2}, y_n + \frac{k_1}{2}), \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}), \\ k_4 &= hf(x_{n+1}, y_n + k_3). \end{aligned}$$

Metóda má lokálnu chybu riešenia rádu $O(h^5)$.

Príklad 47. *Runge-Kutta metóda. Riešte metódou Runge-Kutta štvrtého rádu obyčajnú diferenciálnu úlohu* $y' = \sin(y(x))y(x)$, $y(-10) = 0,1$ *na intervale* $[-10, 10]$ *a vykreslite graf riešenia. Vykreslite parametrickú krivku* $(y(x), y'(x))$ *pre* $x \in [-10, 10]$.

Riešenie. Použijeme príkaz `NDSolve`, počet krokov metódy bude zvolený automaticky preto ho neuvádzame. Diferenciálnu rovnicu zadáme v premennej `rovnica`. Nájdené riešenie uložené v premennej `riesenie` vykreslíme príkazom `Plot[ ]`. Vykreslíme riešenie $y(x)$ a uveďme program na výpočet riešenia, pozri obrázok 11.4.

```
rovnica = y'[x] == Sin[y[x]] y[x];
riesenie = NDSolve[rovnica,
 y[-10] == 0.1}, y,
 {x, -10, 10},
 Method -> "ExplicitRungeKutta"]
Plot[y[x] /. riesenie ,
 {x, -10, 10},
 PlotRange -> All]
```



Obr. 11.4. Graf riešenia $y' = \sin(y(x))y(x)$, $y(-10) = 0,1$ na intervale $[-10, 10]$.

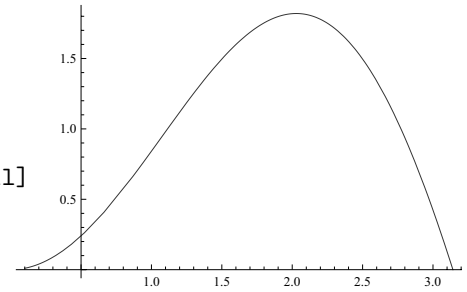
Vypíšme si riešenie a jeho deriváciu $y(0)$, $y'(0)$:

```
{y[0], y'[0]} /. riesenie
```

$\{\{2.2037, 1.77688\}\}$

Vo výpise sú uvedené funkčná hodnota riešenia $y(0) = 2,2037$ a derivácia $y'(0) = 1,77688$. Vykreslime parametrickú krivku $(y(x), y'(x))$ pre $x \in [-10, 10]$, pozri obrázok 11.5. □

```
ParametricPlot[
 {y[x], y'[x]} /. riesenie,
 {x, -10, 10}, PlotRange -> All]
```



Obr. 11.5. Graf parametrickej krivky $(y(x), y'(x))$ riešenia Cauchyho úlohy $y' = \sin(y(x))y(x)$, $y(-10) = 0,1$ na intervale $[-10, 10]$.

11.3 VIACKROKOVÉ METÓDY

Viackroková metóda s výhodou využíva skutočnosť, že je dostupná informácia o funkcii, ktorá bola získaná z predošlého kroku. Základom viackrokovej metódy je použiť minulé hodnoty, y alebo y', zostrojiť mnohočlen, ktorý aproximuje deriváciu a extrapoluje hodnoty do ďalšieho intervalu.

11.3.1 Adamsova metóda

Odvodíme si vzťahy pre Adamsovu metódu ([7] str. 501). Začneme prepisom diferenciálnej rovnice $dy/dx = f(x, y)$ do tvaru

$$dy = f(x, y)dx$$

a integrujeme v hraniciach medzi x_n a x_{n+1}

$$\int_{x_n}^{x_{n+1}} dy = y_{n+1} - y_n = \int_{x_n}^{x_{n+1}} f(x, y)dx.$$

Preložme, teraz interpolačný polynóm druhého stupňa tak, aby prechádzal cez posledné 3 body (x_n, f_n), (x_{n-1}, f_{n-1}), (x_{n-2}, f_{n-2}). Kvadratická aproximácia neznámej funkcie $f(x, y(x))$ môže byť daná v tvare

$$f(x, y) = \frac{1}{2}h^2(f_n - 2f_{n-1} + f_{n-2})x^2 + \frac{1}{2}h(3f_n - 4f_{n-1} + f_{n-2})x + f_n.$$

Po integrácii dostaneme Adamsovu metódu pre výpočet riešenia y_{n+1} v nasledujúcom bode:

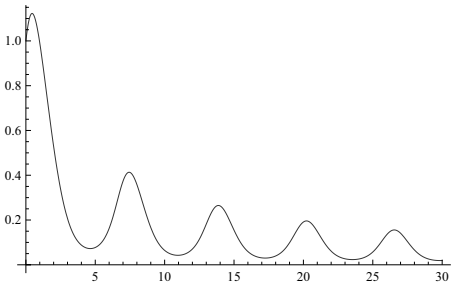
$$y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}) + O(h^4).$$

Hlavnou výhodou viackrokových metód je, že môžeme dosiahnuť vyššiu presnosť s menšou prácou než pre jednokrokové metódy. Má však niekoľko nevýhod. Jednou je, že viackrokové metódy nie sú samospúšťacie metódy. Pri postupe viackrokovou metódou potrebujeme informáciu o niekoľkých predošlých krokoch, avšak na začiatku výpočtu táto informácia nie je dostupná.

Príklad 48. *Metóda prediktor-korektor. Riešte Cauchyho úlohu* $y' = y(x)\cos(x + y(x))$, $y(0) = 1$ *na intervale* $[0, 30]$ *Adamsovou metódou prediktor-korektor ([14] str. 174). Vyrátajte hodnotu* $y(10, 5)$.

Riešenie. Príkaz `NDSolve` použijeme s voľbou Adamsovej metódy `Method -> "Adams"`. Diferenciálnu rovnicu a začiatočnú podmienku zadáme príkazom `{y'[x] == y[x] Cos[x + y[x]], y[0] == 1}`, hľadáme funkciu `y` nakoniec zadáme interval `{x, 0, 30}`. Nájdené riešenie je uložené v premennej `riesenie`.

```
riesenie =
y[x] /. NDSolve[{
 y'[x] == y[x] Cos[x + y[x]],
 y[0] == 1},
 y, {x, 0, 30},
 Method -> "Adams"]
Plot[riesenie, {x, 0, 30},
 PlotRange -> All]
```



Obr. 11.6. Graf riešenia $y' = y(x)\cos(x + y(x))$, $y(0) = 1$ na intervale $[0, 30]$.

Riešenie vykreslíme graficky. Hodnotu riešenia $y(10,5)$ vyrátame nasledovne:

```
riesenie /. x -> 10.5
```

$\{0.0469302\}$

Riešenie $y(10,5) = 0,0469302$. Vypíšme si tabuľku funkčných hodnôt riešenia pre celočíselné hodnoty $x = \{1, \ldots, 5\}$. Prebytočných zátvoriek sa zbavíme tak, že vypíšeme prvý prvok `[[1]]` lebo máme aj tak len jedno riešenie.

```
Table[{x, riesenie[[1]]}, {x, 0, 5}]
```

$\{\{0, 1.\}, \{1, 0.991387\}, \{2, 0.528358\}, \{3, 0.206604\}, \{4, 0.0880718\},$
$\{5, 0.0773121\}\}$

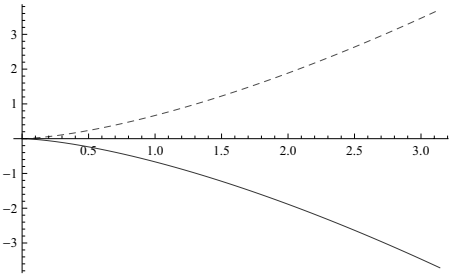
Vypísané funkčné hodnoty riešenia sú $y(0) = 1$, $y(1) = 0,991387$, $y(2) = 0,528358$, $y(3) = 0,206604$, $y(4) = 0,0880718$, $y(5) = 0,0773121$. □

Príklad 49. *Dve riešenia diferenciálnej rovnice. Riešte obyčajnú diferenciálnu rovnicu prvého rádu $y'(x)^2 = x$, so začiatočnou podmienkou $y(0) = 0$ na intervale $x \in [0, \pi]$ príkazom `NDSolve`. Vykreslite grafy riešení.*

Riešenie. Diferenciálna rovnica má dve riešenia, čo môžeme pochopiť aj intuitívne z druhej mocniny $y'(x)^2$. Najskôr definujme rovnicu, ktorú ideme riešiť v premennej `rovnica`, definujme začiatočnú podmienku v premennej `podmienka`. Nakoniec vykreslíme všetky riešenia, ktoré nájdeme príkazom `NDSolve` v jednom súradnicovom systéme. Tenkou čiarou znázorníme prvé riešenie $y_1(x)$, a prerušovanou čiarou druhé riešenie $y_2(x)$, pozri obrázok 11.7.

□

```
rovnica = y'[x]^2 == x;
podmienka = y[0] == 0;
riesenie = y[x] /. NDSolve[
 {rovnica, podmienka},
 y[x], {x, 0, \[Pi]}]
Plot[riesenie, {t, 0, \[Pi]},
 PlotStyle -> {Thin, Dashed}]
```



Obr. 11.7. Dve riešenia rovnice $y'(x)^2 = x$, so začiatočnou podmienkou $y(0) = 0$ na intervale $x \in [0, \pi]$.

11.3.2 Rovnice vyšších rádov

Bežné diferenciálne rovnice pre fyzikálne úlohy sú vyššieho rádu. Čo znamená, že v rovnici sa nachádzajú derivácie rádu 2 a vyššieho. Napríklad, obyčajná diferenciálna rovnica druhého rádu

$$m\frac{d^2y}{dx^2} + b\frac{dy}{dx} + ky = f(x, y)$$

znázorňuje kmitanie lineárnej pružiny, s konštantou pružnosti k, pôsobiacou na teleso s hmotnosťou m s tlmením b. Funkcia $f(x, y)$ je vonkajšia sila pôsobiaca na teleso.

Diferenciálnu rovnicu druhého rádu môže previesť na systém rovníc prvého rádu. Majme všeobecnú začiatočnú úlohu (alebo Cauchyho úlohu) pre diferenciálnu rovnicu druhého rádu

$$\frac{d^2y}{dx^2} = f(x, y, \frac{dy}{dx}),$$

so začiatočnými podmienkami

$$y(x_0) = y_0, \qquad y'(x_0) = y'_0.$$

Označme $y \equiv y_1(x)$, deriváciu označme tiež ako novú funkciu $\frac{dy}{dx} = \frac{dy_1}{dx} \equiv y_2(x)$. Potom preto, že $d^2y/dx^2 = (d/dx)(dy/dx)$ dostávame systém rovníc prvého rádu

$$\begin{aligned} \frac{dy_1}{dx} &= y_2, \qquad y_1(x_0) = y_0, \\ \frac{dy_2}{dx} &= f(x, y_1, y_2), \qquad y_2(x_0) = y'_0 \end{aligned}$$

so začiatočnými podmienkami pre neznáme funkcie $y_1(x)$ a $y_2(x)$. Výsledný pár diferenciálnych rovníc je ekvivalentný pôvodnej rovnici druhého rádu.

Rovnaký postup môžeme použiť na prevod diferenciálnej rovnice rádu m do systému diferenciálnych rovníc prvého rádu. Diferenciálnu úlohu m-tého rádu

$$\begin{aligned} y^{(m)} &= f(x, y, y', \ldots, y^{(m-1)}), \\ y(x_0) &= A_1, \quad y'(x_0) = A_2, \quad \ldots, \quad y^{(m-1)}(x_0) = A_n, \end{aligned}$$

prevedieme do systému m diferenciálnych rovníc prvého rádu označením $y_1 \equiv y$:

$$y_1' = y_2, \quad y_2' = y_3, \quad ..., \quad y_{m-1}' = y_m, \quad y_m' = f(x, y_1, y_2, ..., y_m)$$

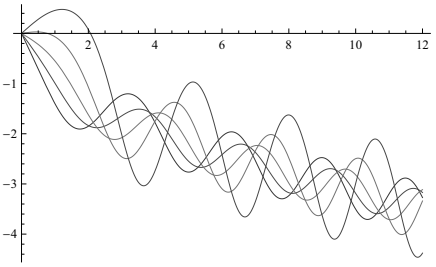
so začiatočnými podmienkami

$$y_1(x_0) = A_1, \quad y_2(x_0) = A_2, \ldots, \quad y_m(x_0) = A_n.$$

Príklad 50. *Zmena začiatočnej podmienky. Riešte obyčajnú diferenciálnu rovnicu druhého rádu* $y'' = y(x)^2 - x$, *so začiatočnými podmienkami* $y(0) = 0$, $y'(0) = d_0$ *na intervale* $x \in [0, 12]$ *príkazom* `NDSolve`. *Začiatočná podmienka* $y'(0) = d_0$ *sa mení s krokom* $0,5$ *na intervale* $d_0 \in [-1, 4,\ 1]$. *Vykreslite grafy riešení pre rôzne začiatočné podmienky.*

Riešenie. Najskôr zadajme rovnicu, ktorú ideme riešiť v premennej `rovnica`, zadajme podmienky `y[0] == 0, y'[0] == d0`, odstránime prebytočné zátvorky `[[1]]` lebo úloha má len jedno riešenie ([12]) a meníme začiatočnú podmienku v cykle `d0, -1.4, 1, 0.5`. Nakoniec vykreslíme všetky riešenia, uložené v premennej `MnozinaRieseni`, v jednom súradnicovom systéme príkazom `Plot[ ]`, pozri obrázok 11.8. □

```
rovnica = y''[x] == y[x]^2 - x;
MnozinaRieseni =
 Table[y[x] /. NDSolve[
  {rovnica, y[0] == 0,
   y'[0] == d0},
   y[x], {x, 0, 12}][[1]],
   {d0, -1.4, 1, 0.5}];
Plot[MnozinaRieseni, {x, 0, 12}]
```

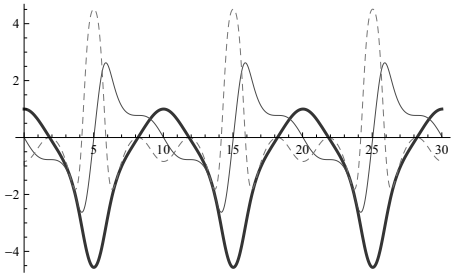


Obr. 11.8. Riešenia $y'' = y(x)^2 - x$, $y(0) = 0$, $y'(0) = d_0$ na intervale $x \in [0, 12]$ pre $d_0 = \{-1,4;\ -0,9;\ -0,4;\ 0,1;\ 0,6\}$.

Príklad 51. *Riešte obyčajnú diferenciálnu rovnicu druhého rádu* $y''(x) + \sin(y(x))y(x) = 0$*, so začiatočnými podmienkami* $y(0) = 1$*,* $y'(0) = 0$ *na intervale* $x \in [0, 30]$ *Adamsovou metódou prediktor-korektor a príkazom* `NDSolve`*. Vykreslite graf riešenia* $y(x)$*, prvú deriváciu* $y'(x)$ *a druhú deriváciu* $y''(x)$*.*

Riešenie. Naskôr definujme rovnicu, ktorú ideme riešiť v premennej `rovnica`, definujme podmienky v premennej `podmienky`. Nakoniec vykreslíme všetky riešenia a derivácie v jednom súradnicovom systéme. Hrubou čiarou znázorníme riešenie $y(x)$, tenkou čiarou $y'(x)$, a prerušovanou $y''(x)$, pozri obrázok 11.9.

```
rovnica =
 y''[x] + Sin[y[x]] y[x] == 0;
podmienky =
 {y[0] == 1, y'[0] == 0};
riesenie = NDSolve[
 {rovnica, podmienky},
  y, {x, 0, 30},
  Method -> "Adams"];
Plot[
 Evaluate[
 {y[x], y'[x], y''[x]}
 /. riesenie],
 {x, 0, 30},
 PlotStyle -> {Thick, Thin, Dashed}]
```



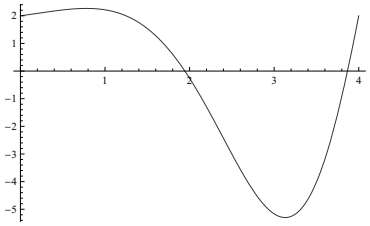
Obr. 11.9. Riešenie úlohy $y''(x) + \sin(y(x))y(x) = 0$, $y(0) = 1$, $y'(0) = 0$, $x \in [0, 30]$ je znázornené hrubou čiarou, $y'(x)$ tenkou čiarou, a $y''(x)$ prerušovanou čiarou.

□

Príklad 52. *Okrajová úloha. Riešte okrajovú úlohu danú obyčajnou diferenciálnou rovnicou druhého rádu* $y''(x) = y'(x) - xy(x)$*, s okrajovými podmienkami* $y(0) = 2$*,* $y(4) = 2$ *na intervale* $x \in [0, 4]$ *príkazom* `NDSolve`*. Vykreslite graf riešenia* $y(x)$*.*

Riešenie. Zadanú úlohu nazývame okrajovou, lebo sú dané podmienky na okraji (hranici) intervalu $[0, 4]$, $y(0) = 2$, $y(4) = 2$. Okrajové úlohy sa riešia veľmi ťažko a preto príkaz `NDSolve` vie nájsť riešenia len pre niektoré lineárne okrajové úlohy. Naskôr zadajme rovnicu, ktorú ideme riešiť v premennej `rovnica`, zadajme podmienky v premennej `podmienky`. Nakoniec vykreslíme riešenie, pozri obrázok 11.10. □

```
rovnica = y''[x] == y'[x] - x y[x];
podmienky = {y[0] == 2, y[4] == 2};
riesenie = y[x] /. NDSolve[
 {rovnica, podmienky},
  y[x], {x, 0, 4}];
Plot[riesenie, {x, 0, 4}]
```



Obr. 11.10. Riešenie okrajovej úlohy $y''(x) = y'(x) - xy(x)$, s okrajovými podmienkami $y(0) = 2$, $y(4) = 2$ na intervale $x \in [0, 4]$.

11.4 SYSTÉMY ROVNÍC PRVÉHO RÁDU

Uvažujme začiatočnú (Cauchyovu) úlohu pre systém obyčajných diferenciálnych rovníc prvého rádu. Zapíšeme ju v tvare

$$\mathbf{Y}'(x) = \mathbf{F}(x, \mathbf{Y}), \quad \mathbf{Y}(a) = \mathbf{Y}_a,$$

kde $\mathbf{Y}$ a $\mathbf{F}$ sú stĺpcové vektory s m zložkami. Vezmime do úvahy nezávislú premennú x na intervale $[a, b]$. Namiesto jednej závislej premennej teraz máme m závislých premenných funkcií $y_1(x), y_2(x), \ldots, y_m(x)$, ktoré sú zložkami vektora $\mathbf{Y}(x)$. Predpokladáme, že existuje m reálnych funkcií $f_1(x, \mathbf{Y})$, $f_2(x, \mathbf{Y})$, $\ldots$, $f_m(x, \mathbf{Y})$ pre každú dvojicu $(x, \mathbf{Y})$. Vektory $\mathbf{Y}$ a $\mathbf{F}$ rozpísané po zložkách majú tvar

$$\mathbf{Y}(x) = \left\{ \begin{array}{c} y_1(x) \\ y_2(x) \\ \ldots \\ y_m(x) \end{array} \right\}, \qquad \mathbf{F}(x, \mathbf{Y}) = \left\{ \begin{array}{c} f_1(x, \mathbf{Y}) \\ f_2(x, \mathbf{Y}) \\ \ldots \\ f_m(x, \mathbf{Y}) \end{array} \right\}.$$

Metódy a analýza chýb riešenia systému diferenciálnych rovníc prvého rádu sú analogické metódam použitým pre jednu diferenciálnu rovnicu prvého rádu.

11.4.1 Runge-Kutta metóda štvrtého rádu pre systémy

Aby sme mohli použiť Runge-Kutta metódu štvrtého rádu aj pre systém m diferenciálnych rovníc prvého rádu musíme ju modifikovať nasledovne ([8] str.

730–735)

$$\begin{aligned}
k_{1i} &= hf_i(x_n, y_{1n}, y_{2n}, \ldots, y_{mn}),\\
k_{2i} &= hf_i(x_n + \frac{h}{2}, y_{1n} + \frac{k_{11}}{2}, \ldots, y_{mn} + \frac{k_{1m}}{2}),\\
k_{3i} &= hf_i(x_n + \frac{h}{2}, y_{1n} + \frac{k_{21}}{2}, \ldots, y_{mn} + \frac{k_{2m}}{2}),\\
k_{4i} &= hf_i(x_n + h, y_{1n} + k_{31}, \ldots, y_{mn} + k_{3m}),\\
y_{in+1} &= y_{in} + \frac{1}{6}(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}),
\end{aligned}$$

pre každé $i = 1, 2, \ldots, m$. To znamená, že predtým než vypočítame napríklad k_{21} musíme vypočítať všetky $k_{11}, k_{12}, \ldots, k_{1m}$.

Koeficienty Runge-Kutta metódy štvrtého rádu pre systém dvoch diferenciálnych rovníc dostaneme z predchádzajúcej metódy tak, že položíme $m = 2$:

$$\begin{aligned}
k_{11} &= hf_1(x_n, y_{1n}, y_{2n}),\\
k_{12} &= hf_2(x_n, y_{1n}, y_{2n}),\\
k_{21} &= hf_1(x_n + \frac{h}{2}, y_{1n} + \frac{k_{11}}{2}, y_{2n} + \frac{k_{12}}{2}),\\
k_{22} &= hf_2(x_n + \frac{h}{2}, y_{1n} + \frac{k_{11}}{2}, y_{2n} + \frac{k_{12}}{2}),\\
k_{31} &= hf_1(x_n + \frac{h}{2}, y_{1n} + \frac{k_{21}}{2}, y_{2n} + \frac{k_{22}}{2}),\\
k_{32} &= hf_2(x_n + \frac{h}{2}, y_{1n} + \frac{k_{21}}{2}, y_{2n} + \frac{k_{22}}{2}),\\
k_{41} &= hf_1(x_n + h, y_{1n} + k_{31}, y_{2n} + k_{32}),\\
k_{42} &= hf_2(x_n + h, y_{1n} + k_{31}, y_{2n} + k_{32}).
\end{aligned}$$

Výpočet hodnôt riešení $y_1(x)$ a $y_2(x)$ je daný dvoma rovnicami

$$\begin{aligned}
y_{1n+1} &= y_{1n} + \frac{1}{6}(k_{11} + 2k_{21} + 2k_{31} + k_{41}),\\
y_{2n+1} &= y_{2n} + \frac{1}{6}(k_{12} + 2k_{22} + 2k_{32} + k_{42}).
\end{aligned}$$

Príklad 53. *Systém dvoch rovníc. Riešte nasledujúcu začiatočnú úlohu pre systém rovníc prvého rádu*

$$\begin{aligned}
y_1'(x) &= -y_2(x) - y_1(x)^3,\\
y_2'(x) &= 2y_1(x) - y_2(x)^2,
\end{aligned}$$

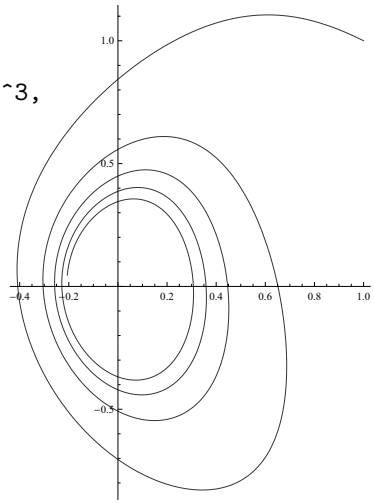
so začiatočnými podmienkami

$$y_1(0) = y_2(0) = 1,$$

Runge-Kutta metódou pre systémy príkazom `NDSolve` *na intervale* $x \in [0, 20]$. *Vykreslite graf riešenia ako krivku* $(y_1(x), y_2(x))$.

Riešenie. Najskôr zadajme rovnice, ktoré ideme riešiť v premennej `rovnice`, zadajme začiatočné podmienky v premennej `podmienky`, použité hľadané funkcie v premennej `premenne`. V príkaze `NDSolve` nastavíme použitú metódu Runge-Kutta príkazom `ExplicitRungeKutta`. Nakoniec vykreslíme parametrickú krivku $(y_1(x), y_2(x))$ príkazom `ParametricPlot[ ]`, pozri obrázok 11.11. □

```
rovnice = {y1'[x] == -y2[x] - y1[x]^3,
 y2'[x] == 2 y1[x] - y2[x]^2};
podmienky = {y1[0] == y2[0] == 1};
premenne = {y1, y2};
riesenie =
 NDSolve[{rovnice, podmienky},
  premenne, {x, 20},
  Method -> "ExplicitRungeKutta"];
ParametricPlot[
 {y1[x], y2[x]} /. riesenie,
 {x, 0, 20}]
```



Obr. 11.11. Riešenie je krivka $(y1(x), y2(x))$ systému dvoch rovníc $y1'(x) = -y2(x) - y1(x)^3$, $y2'(x) = 2y1(x) - y2(x)^2$ s začiatočnými podmienkami $y1(0) = y2(0) = 1$ na intervale $x \in [0, 20]$.

Príklad 54. *Systém troch rovníc. Riešte systém troch obyčajných diferenciálnych rovníc prvého rádu*

$$\begin{aligned} x'(t) &= -3(x(t) - y(t)), \\ y'(t) &= -x(t)z(t) + 26,5x(t) - y(t), \\ z'(t) &= x(t)y(t) - z(t), \end{aligned}$$

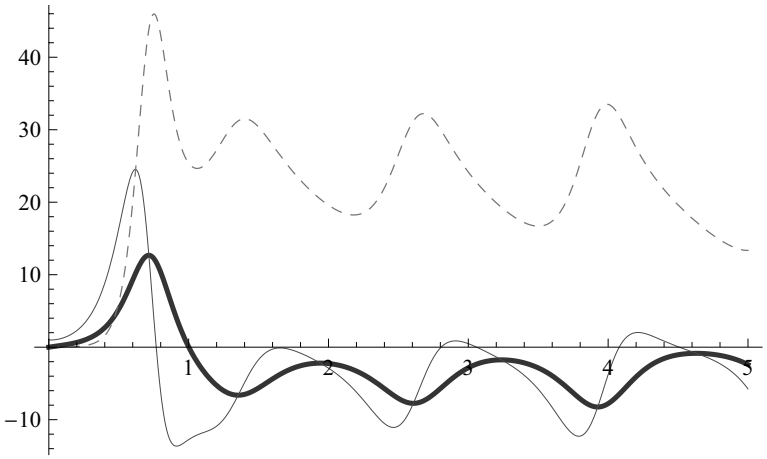
so začiatočnými podmienkami

$$x(0) = z(0) = 0, \quad y(0) = 1,$$

príkazom `NDSolve` *na intervale* $t \in [0,5]$. *Vykreslite grafy riešenia ako krivky* $x(t)$,$y(t)$ *a* $z(t)$.

Riešenie. Naskôr zadajme rovnice, ktoré ideme riešiť v premennej `rovnice`, zadajme začiatočné podmienky v premennej `podmienky`, použité hľadané funkcie v premennej `premenne`. Nakoniec vykreslíme sústavu parametrických kriviek, pozri obrázok 11.12. Na ukážku si vykreslime graf riešenia ako paramet-

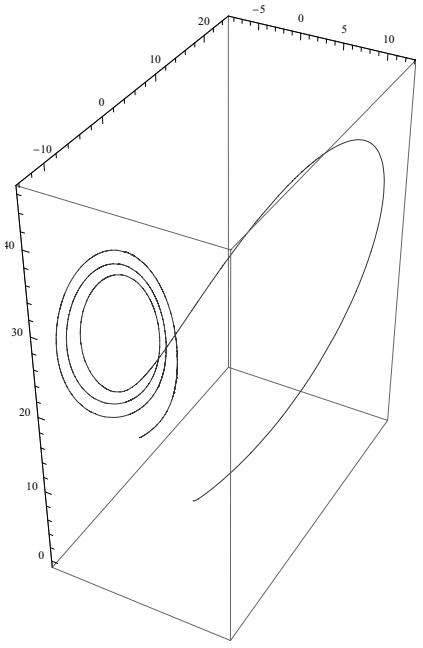
```
rovnice = {
 x'[t] == -3 (x[t] - y[t]),
 y'[t] == -x[t] z[t] + 26.5 x[t] - y[t],
 z'[t] == x[t] y[t] - z[t]};
podmienky = {x[0] == z[0] == 0, y[0] == 1};
premenne = {x, y, z};
riesenie = NDSolve[{rovnice, podmienky}, premenne, {t, 0, 5}];
Plot[ Evaluate[{x[t], y[t], z[t]} /. riesenie], {t, 0, 5},
PlotStyle -> {Thick, Thin, Dashed}]
```



Obr. 11.12. Riešenie systému troch rovníc $x'(t) = -3(x(t) - y(t))$, $y'(t) = -x(t)z(t) + 26,5x(t) - y(t)$, $z'(t) = x(t)y(t) - z(t)$ s začiatočnými podmienkami $x(0) = z(0) = 0$, $y(0) = 1$ na intervale $t \in [0,5]$. $x(t)$ je zobrazené hrubou čiarou, $y(t)$ tenkou čiarou a $z(t)$ prerušovanou čiarou.

rickú krivku $(x(t),y(t),z(t))$ v priestore, pozri obrázok 11.13. □

```
ParametricPlot3D[
 {x[t], y[t], z[t]} /. riesenie,
 {t, 0, 5}]
```



Obr. 11.13. Riešenie systému troch rovníc $x'(t) = -3(x(t) - y(t))$, $y'(t) = -x(t)z(t) + 26,5x(t) - y(t)$, $z'(t) = x(t)y(t) - z(t)$ s začiatočnými podmienkami $x(0) = z(0) = 0$, $y(0) = 1$ na intervale $t \in [0, 5]$ zobrazené priestorovou parametrickou krivkou.

Príklad 55. *Lotka-Volterrova rovnica dravec a korisť. V jednej oblasti žijú spolu dravce a ich korisť. Dravce jedia korisť, a obe skupiny majú obmedzene dlhý život. Populácia dravcov závisí od množstva potravy. Takýto biologický systém môžeme simulovať Lotka-Volterrovou rovnicou. Označme si počet dravcov a ich koristí v čase t funkciami $x(t)$ a $y(t)$. Riešte začiatočnú úlohu pre Lotka-Volterrov systém obyčajných diferenciálnych rovníc*

$$\begin{aligned} f(x,y) &= x(t)(p - qy(t)) \\ g(x,y) &= y(t)(-P + Qx(t)) \\ x'(t) &= f(x,y), \\ y'(t) &= g(x,y), \end{aligned}$$

so začiatočnými podmienkami

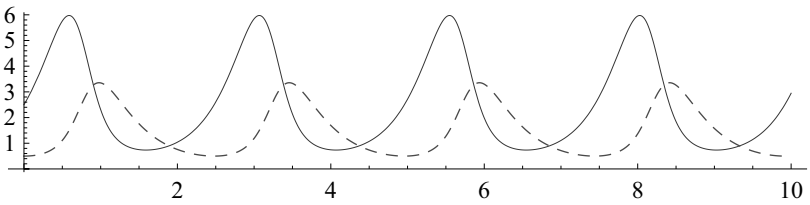
$$x(0) = 2, \quad y(0) = 0,$$

prvého rádu na časovom intervale 0 *až* 10 *rokov,* $t \in [0, 10]$. *Vykreslite graficky počet dravcov a korisť* $(x(t), y(t))$ *pre rôzne začiatočné podmienky* $x(0) = x_0$ *a* $y(0) = y_0$. *Reprodukcia dravcov a koristí je daná parametrami* p, P, q, Q.

Riešenie. Zvoľme si parametre biologického systému $p = 3$; $q = 2$; $P = 2,5$; $Q = 1$. Začiatočné podmienky si zvolíme napríklad $x(0) = 2,5$ a $y(0) = 0,5$.

V programe *Mathematica* zadajme rovnice, ktoré ideme riešiť v premennej `rovnice`, zadajme začiatočné podmienky v premennej `podmienky`, použité hľadané funkcie v premennej `premenne`. Plnou čiarou znázorníme graf riešenia $x(t)$ a prerušovanou graf $y(t)$.

```
f = x[t] (p - q y[t]);
g = y[t] (-P + Q x[t]);
rovnice = {x'[t] == f, y'[t] == g};
podmienky = {x[0] == 2.5, y[0] == 0.5};
premenne = {x[t], y[t]};
p = 3.0; q = 2; P = 2.5; Q = 1;
riesenie = premenne /. NDSolve[{rovnice, podmienky}, premenne,
 {t, 0, 10}][[1]];
Plot[riesenie, {t, 0, 10}, PlotStyle -> {Thin, Dashed},
 AspectRatio -> 0.2]
```

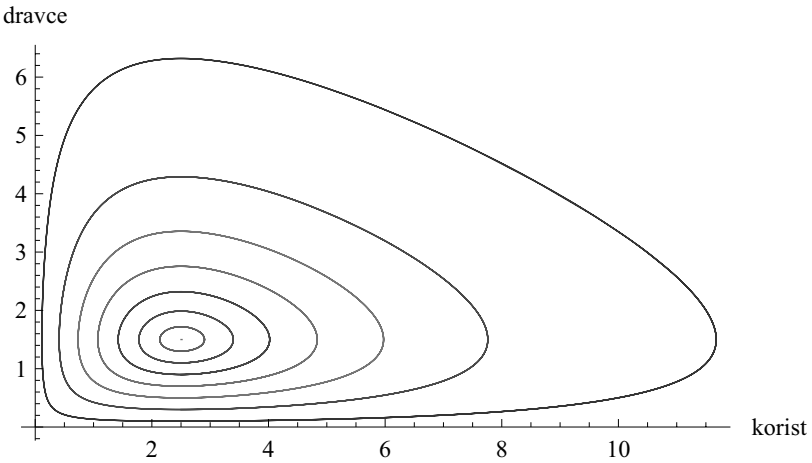


Obr. 11.14. Riešenie Lotka-Volterrovej rovnice dravec a korisť. Počet dravcov $x(t)$ je zobrazený plnou čiarou a počet koristí $y(t)$ je zobrazený prerušovanou čiarou.

Ilustrujme zmenu riešenia vzhľadom na množstvo koristi v biologickom systéme. Budeme meniť začiatočnú podmienku $y(0) = y_0$, $y_0 \in \{0,1;\ 0,3;\ 0,5;\ 0,7;\ 0,9;\ 1,1;\ 1,3;\ 1,5\}$. Uzavreté krivky sú grafom riešenia diferenciálnej rovnice pre rôzne začiatočné podmienky. Najväčšia krivka spĺňa začiatočnú podmienku $y(0) = 0,1$, ..., najmenšia krivka spĺňa začiatočnú podmienku $y(0) = 1,3$. Riešením diferenciálnej rovnice pre podmienku $y(0) = 1,5$ je bod, teda počet dravcov a koristi sa nemení v čase. Vtedy hovoríme, že biologický systém je rovnovážny.

```
f = x[t] (p - q y[t]);
g = y[t] (-P + Q x[t]);
```

```
rovnice = {x'[t] == f, y'[t] == g};
podmienky = {x[0] == 2.5, y[0] == y0};
premenne = {x[t], y[t]};
p = 3.0; q = 2; P = 2.5; Q = 1;
MnozinaRieseni = Table[premenne /. NDSolve[{rovnice, podmienky},
 premenne, {t, 0, 10}], {y0, 0.1, 1.5, 0.2}];
ParametricPlot[MnozinaRieseni, {t, 0, 10}, PlotRange -> All,
 AxesLabel -> {"korist", "dravce"}]
```



Obr. 11.15. Riešenie Lotka-Volterrovej rovnice dravec a korisť. Systém sa stane rovnovážny keď $y(0) = 1,5$ vtedy sa množstvo dravcov a koristi nemení.

□

Uzatvorme túto kapitolu a zhrňme si možné spôsoby numerického riešenia obyčajných diferenciálnych rovníc v programe *Mathematica*:

`NDSolve[eqn, conds, y[t], t, a, b]` Numerické riešenie diferenciálnej rovnice definovanej v `eqn` so začiatočnými podmienkami v `conds` na intervale $[a, b]$.
`NDSolve[eqn, conds, y[t], t, a, b, Method ->` ExplicitEuler"]Explicitná Eulerova metóda.
`NDSolve[eqn, conds, y[t], t, a, b, Method ->` Adams"]"Metóda prediktor-korektor.
`NDSolve[eqn, conds, y[t], t, a, b, Method ->` ExplicitRunge-Kutta"]Explicitná metóda Runge-Kutta.
`NDSolve[eqn, conds, vars, t, a, b]` Rieši systém diferenciálnych rovníc `eqn` s podmienkami `conds` a neznamimi funkciami `vars` s jedným parametrom na intervale $[a, b]$.
`euler[f, x, y, x0, y0, x1, n]` Eulerova metóda riešenia rovnice $y'(x) = f[x, y(x)]$, $y(x_0) = y_0$ na intervale $[x_0, x_1]$ v n uzloch.

11.5 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Použitím Eulerovej metódy nájdite riešenia diferenciálnych rovníc so začiatočnou podmienkou a krokom h:

a) $y' = y,\ y(0) = 1,$ nájdite $y(1),\ (h = 0,1)$;
b) $y' = x + y,\ y(1) = 1,$ nájdite $y(2),\ (h = 0,1)$;
c) $y' = -\frac{y}{1+x},\ y(0) = 2,$ nájdite $y(1),\ (h = 0,1)$;
d) $y' = y - \frac{2x}{y},\ y(0) = 1,$ nájdite $y(1),\ (h = 0,2)$.

2 Numericky nájdite hodnoty riešenia diferenciálnych rovníc so začiatočnou podmienkou a vykreslite riešenie:

a) $(1 + e^x)yy' = e^x,\ y(0) = 1$;
b) $(x^2y - y)y' = (xy^2 + x),\ y(0) = 1$;
c) $y' \sin x = y \ln y,\ y(\frac{\pi}{2}) = 1$.

3 Numericky nájdite hodnoty riešenia diferenciálnych rovníc so začiatočnou podmienkou a vykreslite sústavu riešení (sústavu integrálnych kriviek) pri zmene začiatočnej podmienky:

a) $xy' + y - e^x = 0,\ y(a) = b$;
b) $y' - \frac{y}{1-x^2} - 1 - x = 0,\ y(0) = 0$;
c) $y' - \frac{y}{1-x^2} - 1 - x = 0,\ y(0) = a$;
d) $y' - y \tan x = \frac{1}{\cos x},\ y(0) = 0$;
e) $y' - y \tan x = \frac{1}{\cos x},\ y(0) = a$.

4 Zvoľte si začiatočnú podmienku $y(a) = b$. Numericky vykreslite sústavu riešení (sústavu integrálnych kriviek) pri zmene začiatočnej podmienky diferenciálnych rovníc:

a) $y'^2 - \frac{2y}{x}y' + 1 = 0$;
b) $4y'^2 - 9x = 0$;
c) $yy'^2 - (xy + 1)y' + x = 0$;
d) $yy'^2 - 2xy' + y = 0$;
e) $y'^2 + y^2 = 1,\ y(0) = \frac{1}{2}$.

5 Numericky nájdite hodnoty riešenia diferenciálnych rovníc so začiatočnou podmienkou a vykreslite sústavu riešení:

a) $xy' = y,\ y(1) = 1$;
b) $xy' = y,\ y(0) = 0$;
c) $2xy' = y,\ y(1) = 1$;
d) $2xy' = y,\ y(0) = 0$;
e) $2xyy' + x^2 - y^2 = 0,\ y(0) = 0$;
f) $2xyy' + x^2 - y^2 = 0,\ y(0) = 1$;
g) $2xyy' + x^2 - y^2 = 0,\ y(1) = 0$.

6 Numericky nájdite hodnoty riešenia diferenciálnych rovníc druhého rádu so začiatočnou podmienkou a vykreslite sústavu riešení:

a) $(1 + x^2)y'' - 2xy' = 0,\ y(0) = 0, y'(0) = 3$;
b) $1 + y'^2 = 2yy'',\ y(1) = 1, y'(1) = 1$;
c) $yy'' + y'^2 = y'^3,\ y(0) = 1, y'(0) = 1$;
d) $xy'' = y',\ y(0) = 0, y'(0) = 0$.

7 Numericky nájdite hodnoty riešenia diferenciálnej rovnice tretieho rádu so začiatočnou podmienkou a vykreslite sústavu riešení:

$$y''' + 2y'' + 2y' + y = x,\ y(0) = y'(0) = y''(0) = 0.$$

8 Zvoľte si vhodnú začiatočnú podmienku, ak nie je uvedená, napríklad $y(0) = 1$, $z(0) = 0$. Numericky vykreslite riešenie (sústavu integrálnych kri-

viek) sústavy diferenciálnych rovníc:

a) $\begin{aligned} y'(x) &= z(x), \\ z'(x) &= -y(x); \end{aligned}$ b) $\begin{aligned} y'(x) &= -3y(x) - z(x), \\ z'(x) &= y(x) - z(x); \end{aligned}$

c) $\begin{aligned} y'(x) + 3y(x) + 4z(x) &= 2x, \\ z'(x) - y(x) - z(x) &= x, \\ y(0) = 0,\ z(0) &= 0; \end{aligned}$

d) $\begin{aligned} x'(t) - 4x(t) - y(t) + 36t &= 0, \\ y'(t) + 2x(t) - y(t) + 2e^t &= 0, \\ x(0) = 0,\ y(0) &= 1; \end{aligned}$

e) $\begin{aligned} x'(t) &= y(t), \\ y'(t) &= z(t), \\ z'(t) &= x(t); \end{aligned}$ e) $\begin{aligned} x'(t) &= y(t) + z(t), \\ y'(t) &= x(t) + z(t), \\ z'(t) &= x(t) + y(t). \end{aligned}$

11.6 VÝSLEDKY

1 a) $y(1) = e \simeq 2,593$; b) $y(2) = 3(e-1) \simeq 4,780$; c) $y(1) = 1 \simeq 0,946$; d) $y(1) = \sqrt{3} \simeq 1,826$.

2 a) $2e^{\frac{y^2}{2}} = \sqrt{e}(1+e^x)$; b) $1 + y^2 = \frac{2}{1-x^2}$; c) $y = 1$.

3 a) $y = \frac{e^x}{x} + \frac{ab-e^a}{x}$; b) $y = \frac{1}{2}(x\sqrt{1-x^2} + \arcsin x)\sqrt{\frac{1+x}{1-x}}$; d) $y = \frac{x}{\cos x}$.

4 Nech $C \in R$, krivky pre (x, y) spĺňajú a) $(x^2C^2+1-2Cy)(x^2+C^2-2Cy) = 0$; b) $(y+C)^2 = x^3$; c) $(\frac{x^2}{2} - y + C)(x - \frac{y^2}{2} + C) = 0$; d) $y^2 + C^2 = 2Cx$; e) $y = \frac{1}{2}\cos x \pm \frac{\sqrt{3}}{2}\sin x$.

5 Nech $C \in R$, a) $y = x$; b) $y = Cx$; c) $y^2 = x$; d) $y^2 = 2px$; e) $(x-C)^2+y^2 = C^2$; f) nemá riešenie; g) $x^2 + y^2 = x$.

6 a) $y = x^3 + 3x$; b) $y = \frac{1}{2}(x^2+1)$; c) $y = x + 1$; d) $y = Cx^2$.

7 $y = e^{-x} + e^{-\frac{x}{2}}\left(\cos\frac{\sqrt{3}}{2}x + \frac{1}{\sqrt{3}\sin\frac{\sqrt{3}}{2}x}\right) + x - 2$.

8 Nech $C_1, C_2 \in R$ a) $y(x) = C_1\cos x + C_2\sin x$, $z(x) = C_2\cos x - C_1\sin x$; b) $y(x) = (C_1 - C_2 - C_1x)e^{-2x}$, $z(x) = (C_1x + C_2)e^{-2x}$; c) $y(x) = (C_2 - 2C_1 - 2C_2x)e^{-x} - 6x + 14$, $z(x) = (C_1 + C_2x)e^{-x} + 5x - 9$, zo začiatočných podmienok $C_1 = 9$, $C_2 = 4$, $y(x) = 14(1 - e^{-x}) - 2x(3 + 4e^{-x})$, $z(x) = -9(1 - e^{-x}) + x(5 + 4e^{-x})$; d) $x(t) = 10e^{2t} - 8e^{3t} - e^t + 6t - 1$, $y(t) =$

$-20e^{2t}+8e^{3t}+3e^{t}+12t+10$. e) $x(t)=C_1e^{t}+e^{-\frac{t}{2}}\left(C_2\cos\frac{\sqrt{3}}{2}t+C_3\sin\frac{\sqrt{3}}{2}t\right)$, $y(t)=C_1e^{t}+e^{-\frac{t}{2}}\left(\frac{C_3\sqrt{3}-C_2}{2}\cos\frac{\sqrt{3}}{2}t-\frac{C_2\sqrt{3}+C_3}{2}\sin\frac{\sqrt{3}}{2}t\right)$, $z(t)=C_1e^{t}+e^{-\frac{t}{2}}\left(\frac{-C_3\sqrt{3}-C_2}{2}\cos\frac{\sqrt{3}}{2}t+\frac{C_2\sqrt{3}+C_3}{2}\sin\frac{\sqrt{3}}{2}t\right)$; f) $x(t)=C_1e^{-t}+C_2e^{2t}$, $y(t)=C_3e^{-t}+C_2e^{2t}$, $z(t)=-(C_1+C_3)e^{-t}+C_2e^{2t}$.

Kapitola 12

NIEKTORÉ PARCIÁLNE DIFERENCIÁLNE ROVNICE A METÓDA KONEČNÝCH DIFERENCIÍ

V poslednej kapitole tejto knihy sa venujeme aproximácii riešení parciálnych diferenciálnych rovníc závislých na čase. Zameriame sa na parabolické a hyperbolické začiatočné úlohy a pri diskretizácii použijeme metódu konečných diferencii. Uvádzame príklady riešenia úloh šírenia tepla, Laplaceovej rovnice, Poissonovej rovnice, úlohy šírenia vĺn a systém parciálnych diferenciálnych rovníc závislých aj nezávislých na čase. Avšak, Laplaceove a Poissonove rovnice nezávisia na čase. Všetky uvedené úlohy pre parciálne diferenciálne rovnice majú jediné riešenie, ako je uvedené v citovanej literatúre. K detailnejšiemu štúdiu numerických metód pre parciálne diferenciálne rovnice odporúčame ([14] kap. 6,7,8), [10], ([7] kap. 13).

12.1 KLASIFIKÁCIA PARCIÁLNYCH DIFERENCIÁLNYCH ROVNÍC

Mnoho fyzikálnych problémov vedie k parciálnej diferenciálnej rovnici druhého rádu v tvare

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu + G = 0$$

kde A, B, C, D, E, F a G sú dané funkcie premenných x a y, spojité v oblasti Ω roviny Oxy. Parciálne diferenciálne rovnice klasifikujeme do jedného z troch typov:
Nech $B^2 - 4AC < 0$ v oblasti Ω, potom nazveme rovnicu *eliptickou*.
Nech $B^2 - 4AC = 0$ v oblasti Ω, potom nazveme rovnicu *parabolickou*.
Nech $B^2 - 4AC > 0$ v oblasti Ω, potom nazveme rovnicu *hyperbolickou*.

12.2 METÓDA KONEČNÝCH DIFERENCIÍ

Metóda konečných diferencií je založená na aproximácii parciálnych derivácií konečnými diferenciami ([7] str. 541). Väčšinou sú konečné diferenčné rovnice pevne určené obdĺžnikovou sieťou bodov, v ktorej sa hľadajú neznáme hodnoty funkcií.

Konečno-diferenčné vzťahy pre prvé a druhé derivácie môžeme odvodiť z Taylorovho rozvoja funkcií ([2], str.129–185),

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3!}u^{(3)}(x) + \dots$$

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u^{(3)}(x) + \dots$$

Prvú deriváciu aproximujeme centrálnou diferenciou

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} + O(h^2)$$

a pre druhú deriváciu použijeme centrálnu diferenciu

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2).$$

Prvá derivácia môže byť tiež aproximovaná doprednou alebo spätnou diferenciou, ale takáto aproximácia bude s chybou rádu $O(h)$. Ak $u(x, y)$ je funkcia dvoch premenných x a y, druhú parciálnu deriváciu $\frac{\partial^2 u}{\partial x^2}$ dostaneme ak y považujeme za konštantu, a funkciu vyčíslime v troch bodoch x, $x + h$ a $x - h$. Parciálnu deriváciu $\frac{\partial^2 u}{\partial y^2}$ vypočítame analogicky tak, že za konštantu považujeme x. Potom

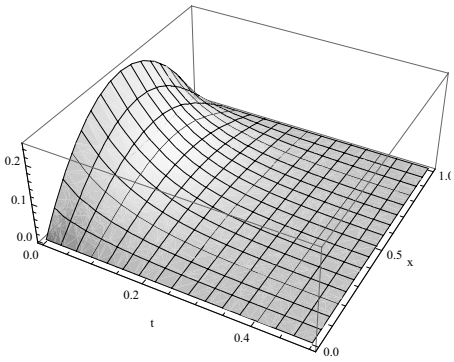
$$\frac{\partial^2 u}{\partial x^2}(x, y) = \frac{u(x+h, y) - 2u(x, y) + u(x-h, y)}{h^2} + O(h^2),$$

$$\frac{\partial^2 u}{\partial y^2}(x,y) = \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} + O(h^2).$$

V programe *Mathematica* [15] môžeme riešiť rovnice parabolického a hyperbolického typu príkazom `NDSolve`, ktorý použije numerickú metódu na intervale $x \in [a,b]$ a $y \in [c,d]$. Diferenciálnu rovnicu zadávame v `eqn`, začiatočné a okrajové podmienky v `conds`. Riešenie je vyrátané v niekoľkých uzlových bodoch, nakoniec interpolované B-splajnom a uložené v premennej `sol` pre ďalšie použitie alebo vizualizáciu. Štandardné použitie je nasledovné:

```
sol = u[x, y] /. NDSolve[{eqn, conds}, u[x, t], {x, a, b},
      {y, c, d}][[1]]
```

```
RovnicaTepla =
 {D[u[x, t], t] - D[u[x, t], x, x] == 0};
podmienky =
 {u[x, 0] == x (1 - x),
  u[0, t] == 0, u[1, t] == 0};
premenne = u[x, t];
riesenie =
 premenne /.
  NDSolve[
   {RovnicaTepla, podmienky},
    premenne, {x, 0, 1},
    {t, 0, 0.5}][[1]];
Plot3D[riesenie, {x, 0, 1},
 {t, 0, 0.5}, PlotRange -> All,
 AxesLabel -> {"x", "t", ""}]
```



Obr. 12.1. Riešenie $u(x,t)$ rovnice šírenia tepla.

Príklad 56. *Diferenciálna rovnica šírenia tepla. Riešte začiatočno-okrajovú úlohu pre parabolickú parciálnu diferenciálnu rovnicu šírenia tepla*

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0,$$

so začiatočnou podmienkou

$$u(x,0) = x(1-x),$$

a okrajovými podmienkami

$$u(0,t) = 0, \quad u(1,t) = 0,$$

na intervale $t \in [0; 0,5]$, $x \in [0,1]$ *príkazom* `NDSolve`. *Vykreslite grafy riešení.*

Riešenie. Vytvorme návod riešenia v programe *Mathematica*. Naskôr zadajme rovnicu, ktorú ideme riešiť v premennej `RovnicaTepla`, zadajme začiatočné podmienky v premennej `podmienky`. V zozname `premenne` uvedieme hľadané funkcie. Druhú parciálnu deriváciu u podľa x vypočítame `D[u[x, t], x, x]`. Riešenie $u(x,t)$, ktoré nájdeme príkazom `NDSolve` v súradnicovom systéme (x,t) uložíme do premennej `riesenie`. Nakoniec vykreslíme riešenie, (pozri obrázok 12.1.).

□

Príklad 57. *Systém diferenciálnych rovníc zmiešaného typu. Riešte nasledujúci systém zmiešaných parciálnych diferenciálnych rovníc o neznámych* $u(x,t)$ *a* $v(x,t)$

$$\begin{aligned} \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} &= v(x,t) \\ \frac{\partial^2 v}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} &= 0 \end{aligned}$$

s vedľajšími podmienkami

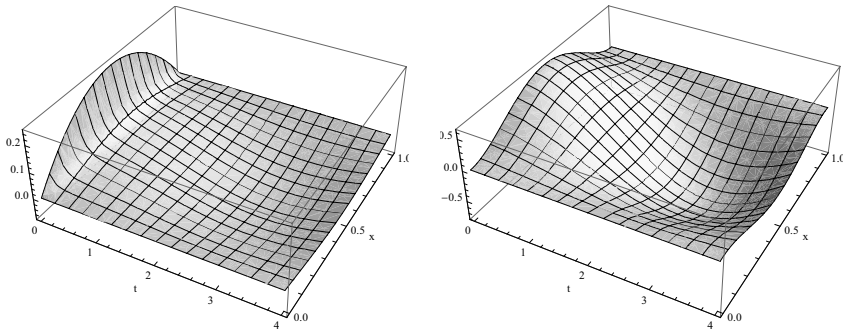
$$\begin{aligned} u(x,0) &= x(1-x), \\ u(0,t) &= 0, \\ u(1,t) &= 0, \\ v(x,0) &= 10x^2(1-x)^2, \\ \frac{\partial v}{\partial t}(x,0) &= 0 \\ v(0,t) &= 0, \\ v(1,t) &= 0, \end{aligned}$$

na intervaloch $x \in [0,1]$, $t \in [0,4]$ *príkazom* `NDSolve`. *Vykreslite grafy riešení.*

Riešenie. Vytvorme návod riešenia v programe *Mathematica*. V premennej `SustavaRovnic` zadajme rovnice, ktoré ideme riešiť, zadajme začiatočné podmienky v premennej `podmienky`. Hľadané funkcie uvedieme v zozname `premenne`. Prvú parciálnu deriváciu v začiatočnej podmienke $\frac{\partial v}{\partial t}(x,0) = 0$ zadáme príkazom `Derivative[0, 1][v][x, 0] == 0`. Nakoniec vykreslíme riešenia $u(x,t)$ a $v(x,t)$ ako pole dvoch grafov príkazom `GraphicsArray`.

```
SustavaRovnic = {D[u[x, t], t] - D[u[x, t], x, x] == v[x, t],
 D[v[x, t], t, t] - D[u[x, t], x, x] == 0};
podmienky = {u[x, 0] == x (1 - x), u[0, t] == 0, u[1, t] == 0,
 v[x, 0] == 10 x^2 (1 - x)^2, Derivative[0, 1][v][x, 0] == 0,
```

```
 v[0, t] == 0, v[1, t] == 0};
premenne = {u[x, t], v[x, t]};
riesenie = premenne /. NDSolve[{SustavaRovnic, podmienky},
 premenne, {x, 0, 1}, {t, 0, 4}][[1]];
Show[GraphicsArray[Map[Plot3D[#, {x, 0, 1}, {t, 0, 4},
PlotRange -> All, AxesLabel -> {"x", "t", ""}] &, riesenie]]]
```



Obr. 12.2. Riešenie sústavy zmiešaných diferenciálnych rovníc. Riešenie $u(x,t)$ vľavo a riešenie $v(x,t)$ vpravo na intervaloch $x \in [0,1]$, $t \in [0,4]$.

Vypíšeme si niekoľko funkčných hodnôt riešenia $u(0,5;t)$ a $v(0,5;t)$ pre $t \in \{0;0,4;0,8;\ 1,2;\ 1,6;\ 2;\ 2,4\}$ ako usporiadané dvojice $\{u,v\}$ príkazom

```
Table[riesenie /. x -> 0.5, {t, 0, 2.4, 0.4}]
```

$\{\{0.25, 0.625\}, \{0.0541024, 0.518659\}, \{0.0318357, 0.305753\},$
$\{0.00728594, 0.035611\}, \{-0.0193654, -0.250439\}, \{-0.0439917, -0.507252\},$
$\{-0.0625896, -0.692619\}\}$

Nájdené približné hodnoty riešenia systému diferenciálnych rovníc sú $u(0,5;0) = 0,25$, $v(0,5;0) = 0,625$, $u(0,5;0,4) = 0,0541024$, $v(0,5;0,4) = 0,518659$, $u(0,5;0,8) = 0,0318357$, $v(0,5;0,8) = 0,305753$, $u(0,5;1,2) = 0,00728594$, $v(0,5;1,2) = 0,035611$, $u(0,5;1,6) = -0,0193654$, $v(0,5;1,6) = -0,250439$, $u(0,5;2) = -0,0439917$, $v(0,5;2) = -0,507252$, $u(0,5;2,4) = -0,0625896$, $v(0,5;2,4) = -0,692619$. □

12.3 LAPLACEOVA ROVNICA

Laplaceova rovnica

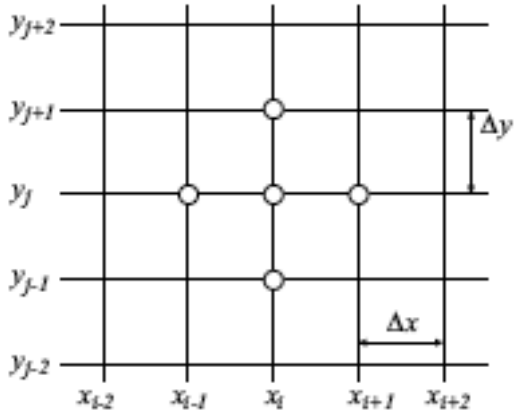
$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

s okrajovými podmienkami

$$\begin{aligned} u(a,y) &= u_a, \\ u(b,y) &= u_b, \\ u(x,c) &= u_c, \\ u(x,d) &= u_d, \end{aligned}$$

v obdĺžnikovej oblasti $(x,y) \in [a,b] \times [c,d]$ je eliptickou parciálnou diferenciálnou rovnicou. u_a, u_b, u_c a u_d sú konštanty.

Na riešenie eliptického typu rovníc je vhodná metóda sietí, ktorú si odvodíme pomocou konečných diferencií. Rozdeľme obdĺžnikovú oblasť $[a,b] \times [c,d]$ rovnomerne priamkami rovnobežnými s osami x a y tak ako znázorňuje obrázok 12.3. Ak nahradíme derivácie diferenčnými aproximáciami v bode



Obr. 12.3. Rovnomerné delenie obdĺžnikovej oblasti a označenie uzlov.

$(x_i; y_j)$, dostaneme diferenčnú rovnicu ([7] str. 541)

$$\begin{aligned} \nabla^2 u(x_i,y_j) &= \frac{u(x_{i+1},y_j) - 2u(x_i,y_j) + u(x_{i-1},y_j)}{(\Delta x)^2} + \\ &+ \frac{u(x_i,y_{j+1}) - 2u(x_i,y_j) + u(x_i,y_{j-1})}{(\Delta y)^2} = 0. \end{aligned}$$

Prepíšme rovnicu použitím dvojitého indexovania uzlov u, jeden index pre x-ové hodnoty a druhý pre y-ové hodnoty:

$$\nabla^2 u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = 0.$$

Ak uvažujeme ekvidištančný krok $\Delta x = \Delta y = h$, výsledkom je značné zjednodušenie, čiže

$$\nabla^2 u_{i,j} = \frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) = 0. \qquad (12.1)$$

Daný vzťah môžeme reprezentovať symbolicky, kde v prvom riadku je $u_{i-1,j}$ v druhom riadku sú $u_{i,j+1}$, $u_{i,j-1}$, $-4u_{i,j}$ v treťom riadku je $u_{i+1,j}$

$$\nabla^2 u_{i,j} = \frac{1}{h^2} \left\{ \begin{array}{ccc} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{array} \right\} u_{i,j} = 0.$$

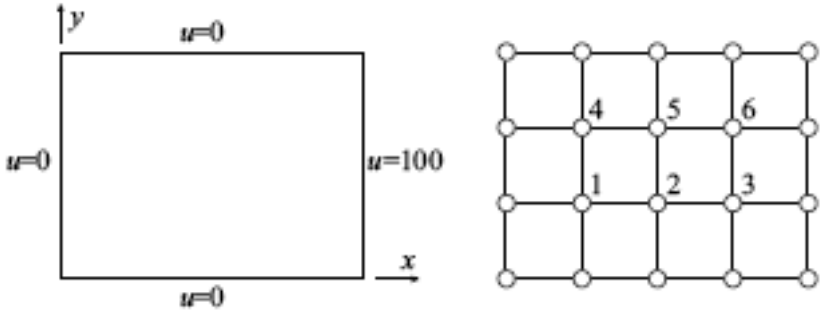
Táto aproximácia bola robená s chybou $O(h^2)$. Takéto symbolické značenie voláme päť-bodová alebo päť-bodová hviezdicová formulácia lebo je použitých iba päť uzlov vo vzorci (pozri. obr. 12.3.).

Príklad 58. *Laplaceova rovnica na obdĺžnikovej oblasti. Uvažujme problém prúdenia tepla na tenkej kovovej doštičke obdĺžnikového tvaru* $4\ m \times 3\ m$. *Jednu zo strán obdĺžnika udržujeme na* $100°C$ *a ostatné tri na* $0°C$. *Aké sú teploty vo vnútorných bodoch?*

Riešenie. Za predpokladu, že teplo prúdi len v smere x a y môžme daný problém vyjadriť matematicky tak, že hľadáme $u(x, y)$, pre ktoré

$$\nabla^2 u = 0, \quad \text{kde } u(x,0) = u(x,3) = u(0,y) = 0,\ u(4,y) = 100.$$

Ak poznáme hodnoty funkcie pozdĺž celej hranice teda na okraji obdĺžnikovej



Obr. 12.4. Dirichletove okrajové podmienky a rozdelená oblasť sieťou.

oblasti hovoríme, že daný problém spĺňa *Dirichletove okrajové podmienky*. Za účelom vyriešenia problému vložíme uzly 1 až 6 do obdĺžnikovej siete s $h = 1$ (pozri obr.12.4.). Pre každý z týchto šiestich uzlov vieme napísať diferenčnú

rovnicu, ktorá je kombináciou teploty v danom uzle a teplôt v štyroch susedných uzloch. Diferenčná rovnica 12.1 pre uzol 1 vyzerá nasledovne:

$$u_4 + 0 + u_2 + 0 - 4u_1 = 0.$$

Dve nuly na ľavej strane rovnice znamenajú len toľko, že teplotu v dvoch susedných uzloch poznáme $u(0,1) = u(1,0) = 0$. Diferenčná rovnica 12.1 pre uzol 2 vyzerá nasledovne:

$$u_5 + 0 + u_1 + u_3 - 4u_2 = 0.$$

Rovnica pre uzol 3 bude

$$u_6 + 0 + u_2 + 100 - 4u_3 = 0.$$

Rovnice pre všetky vnútorné uzly poskladáme do systému lineárnych rovníc:

$$\begin{bmatrix} 4 & -1 & & -1 & & \\ -1 & 4 & -1 & & -1 & \\ & -1 & 4 & & & -1 \\ -1 & & & 4 & -1 & \\ & -1 & & -1 & 4 & -1 \\ & & -1 & & -1 & 4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 100 \end{Bmatrix}.$$

Pozrime sa bližšie na systém a všimneme si, že je symetrický. Vďaka symetrii môžeme redukovať pamäťové nároky a počet operácií. Samozrejme so sieťou, ktorú sme použili ($h = 1$) nedosiahneme dostatočnú presnosť. V praxi sa využíva omnoho menšia hodnota parametra h, čo vedie k systémom rovníc s tisíc alebo viac rovnicami. □

12.4 LIEBMANNOVA METÓDA NA RIEŠENIE LAPLACEOVEJ ROVNICE

Liebmannova metóda je jedna z najjednoduchších metód na riešenie diferenčných rovníc, ktorá nevyžaduje vytvorenie systému rovníc ([14] kap. 6). Podľa tejto metódy získame riešenie nasledujúcimi krokmi:

1. Vyber začiatočnú aproximáciu $u_{i;j}^{(0)}$ pre vnútorné uzly. Začiatočná aproximácia môže byť vypočítaná z lineárnej interpolácie okrajových podmienok.

2. Vypočítaj aproximácie $u_{i;j}^{(k+1)}$ takto:

$$u_{i,j}^{(k+1)} = \frac{1}{4}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}).$$

Predpokladáme že uzly v sieti sú očíslované v poradí indexmi i a j. Horný index (k) znamená, že $u_{i,j}^{(k)}$ je k-ta iterácia riešenia v uzle (i,j). Dá sa ukázať, že proces konverguje. Počet iterácií potrebných k dosiahnutiu danej presnosti riešenia nie je dopredu známy. Iteračná metóda pokračuje až kým nie je splnený test konvergencie. Napríklad, iteračný proces môžeme zastaviť, ak $\left|\left(u_{i,j}^{(k+1)} - u_{i,j}^{(k)}\right)/u_{i,j}^{(k+1)}\right| < \varepsilon$ je splnená vo všetkých uzloch siete, kde ε je predpísaná chybová tolerancia.

12.5 SOR METÓDA

Hlavný nedostatok Liebmannovej metódy je jej pomalá konvergencia. Metóda SOR *(successive overrelaxation)* môže konvergenciu značne urýchliť. Prírastok neznámej funkcie sa násobí *relaxačným faktorom* ω:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + \frac{\omega}{4}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - 4u_{i,j}^{(k)}).$$

Maximálne zrýchlenie konvergencie sa získa s nejakou optimálnou hodnotou. Táto hodnota bude pre Laplaceovu rovnicu vždy niekde medzi 1.0 a 2.0.

12.6 POISSONOVA ROVNICA

Metódy spomenuté v predchádzajúcich sekciách sú ihneď aplikovateľné na Poissonovu rovnicu. Ilustrujeme si príklad analýzy pnutia v hranole, ktorý postupne skrúcame. Skúmajme pnutie v priereze hranola, čo nám dáva obdĺžnikovú oblasť $[a,b] \times [c,d]$. Pnutie v bode $(x,y) \in [a,b] \times [c,d]$, označíme funkciou $u(x,y)$. Funkcia u vyhovuje Poissonovej rovnici:

$$\nabla^2 u + 2 = 0$$

s okrajovými podmienkami

$$\begin{aligned} u(a,y) &= 0, \\ u(b,y) &= 0, \\ u(x,c) &= 0, \\ u(x,d) &= 0, \end{aligned}$$

v obdĺžnikovej oblasti $(x,y) \in [a,b] \times [c,d]$. Ide o eliptickú parciálnu diferenciálnu rovnicu.

Použime metódu sietí a rozdeľme prierez hranola štvorcovou sieťou s veľkosťou kroku h. Diferenčná rovnica pre vnútorný bod s indexovým označením ij je ([7] str. 581)

$$\frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) + 2 = 0.$$

Iteračný vzťah Liebmannovej metódy vyzerá nasledovne:

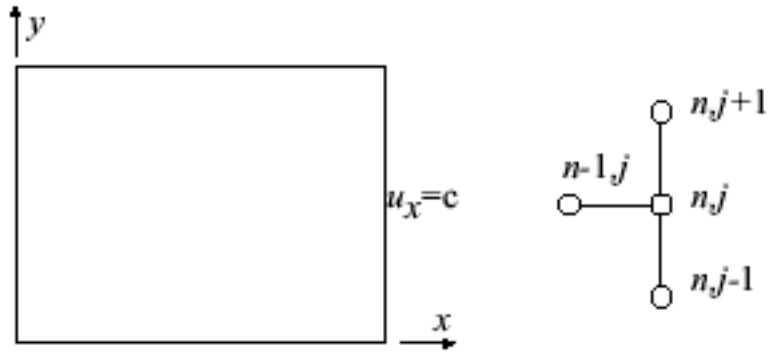
$$u_{i,j}^{(k+1)} = \frac{1}{4}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} + 2h^2).$$

Analogicky môžeme SOR metódu aplikovať na riešenie Poissonovej diferenčnej rovnice v tvare

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + \frac{\omega}{4}(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - 4u_{i,j}^{(k)} + 2h^2).$$

12.7 OKRAJOVÉ PODMIENKY S DERIVÁCIOU

Neumannove okrajové podmienky s deriváciu $u(x, y)$ na hranici oblasti určujú deriváciu. Pre problémy vedenia tepla to znamená že je definovaný tepelný tok cez hranu. Názorne si ukážeme implementáciu okrajových podmienok s deriváciou pre obdĺžnikovú oblasť, pozri obr. 12.5. Predpokladajme, že poznáme deriváciu $\frac{\partial u}{\partial x} = u_x = c$ pre pravú hranicu, kde $x = x_n$. Laplaceova rovnica pre



Obr. 12.5. Uzly na hranici oblasti.

uzol (x_n, y_j) je daná diferenčným vzťahom

$$u_{n+1,j} + u_{n-1,j} + u_{n,j+1} + u_{n,j-1} - 4u_{n,j} = 0. \tag{12.2}$$

Hodnota $u_{n+1,j}$ neexistuje, pretože daný bod je umiestnený mimo oblasti. Avšak, pre tento fiktívny uzol môžeme použiť vzťah centrálnej diferencie

$$u_x(x_n, y_j) = \frac{u_{n+1,j} - u_{n-1,j}}{2h} = c.$$

Ak vyjadríme $u_{n+1,j}$ z vyššie uvedeného vzťahu a dosadíme do diferenčnej rovnice 12.2 dostaneme rovnicu, ktorá spĺňa okrajovú podmienku s deriváciou na pravej hranici:

$$2u_{n-1,j} + u_{n,j+1} + u_{n,j-1} - 4u_{n,j} + 2hc = 0$$

Vzťahy pre okrajové podmienky s deriváciou na ostatných hraniciach oblasti vypočítame analogicky.

12.8 HYPERBOLICKÉ ROVNICE

Príklad 59. *Vlnová rovnica. Hľadajme riešenie, $u(t,x)$ v oblasti $t \in [0,10] \times [-10,10]$, Wolframovej hyperbolickej parciálnej diferenciálnej rovnice šírenia vlny*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + (1 - u(t,x)^2)(1 + 2u(t,x)),$$

s okrajovou podmienkou

$$u(t,-10) = u(t,10),$$

a začiatočnými podmienkami

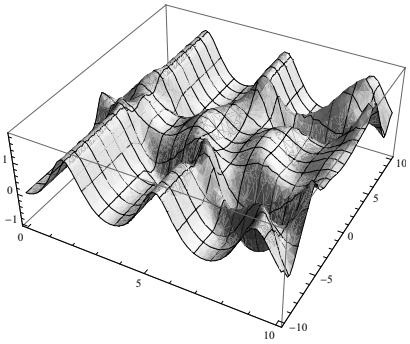
$$\begin{aligned} u(0,x) &= e^{-x^2}, \\ \frac{\partial u}{\partial t}(0,x) &= 0, \end{aligned}$$

na intervale $t \in [0,10]$, $x \in [-10,10]$ príkazom `NDSolve`*. Vykreslite graf riešenia.*

Riešenie. Vytvorme návod riešenia v programe *Mathematica*. V premennej `VlnovaRovnica` si najskôr zadajme rovnicu, ktorú ideme riešiť, v premennej `podmienky` zadajme začiatočné a okrajové podmienky. Hľadanú funkciu uvedieme v zozname `premenne`. Prvú parciálnu deriváciu $\frac{\partial u}{\partial t}(0,x) = 0$ vypočítame príkazom `Derivative[1, 0][u][0, x] == 0`. Nakoniec vykreslíme riešenie uložené v premennej `riesenie`, pozri obrázok 12.6.

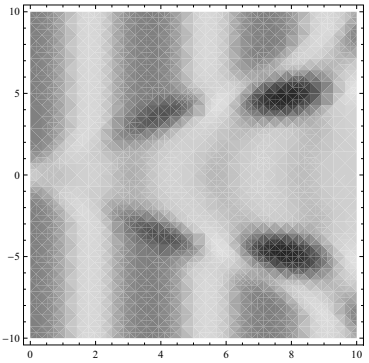
```
VlnovaRovnica = {D[u[t, x], t, t] == D[u[t, x], x, x] +
 (1 - u[t, x]^2) (1 + 2 u[t, x])};
podmienky = {u[0, x] == E^-x^2, Derivative[1, 0][u][0, x] == 0,
 u[t, -10] == u[t, 10]};
premenne = u[t, x];
riesenie = NDSolve[{VlnovaRovnica, podmienky}, premenne,
 {t, 0, 10}, {x, -10, 10}];
Plot3D[Evaluate[u[t, x] /. riesenie], {t, 0, 10}, {x, -10, 10}]
```

Keď zobrazíme riešenie pomocou intenzít, tak ľahko pozorujeme vlny z horného pohľadu, pozri obrázok 12.7.



Obr. 12.6. Riešenie vlnovej rovnice $u(t, x)$.

```
DensityPlot[Evaluate[u[t, x] /. s],
 {t, 0, 10}, {x, -10, 10}]
```



Obr. 12.7. Riešenie vlnovej rovnice $u(t, x)$ zobrazené zmenou intenzít.

□

Príklad 60. *Vlnová rovnica. Hľadajte riešenie $u(x, t)$ hyperbolickej parciálnej diferenciálnej rovnice šírenia vlny*

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = -9,80665,$$

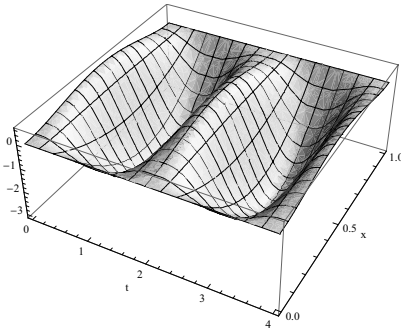
s vedľajšími podmienkami

$$\begin{aligned} u(0,t) &= u(1,t) = 0, \\ u(x,0) &= 10x^2(1-x)^2, \\ \frac{\partial u}{\partial t}(x,0) &= 0, \end{aligned}$$

na intervale $x \in [0,1]$, $t \in [0,4]$ *príkazom* `NDSolve`. *Vykreslite grafy zložiek* u, v *riešenia* (u,v).

Riešenie. Napíšeme návod pre program *Mathematica*. Najskôr v premennej `VlnovaRovnica` zadajme rovnicu, ktorú ideme riešiť, v premennej `podmienky` zadajme vedľajšie podmienky. Hľadané funkcie uvedieme v zozname `premenne`. Prvú parciálnu deriváciu $\frac{\partial u}{\partial t}(x,0) = 0$ vypočítame príkazom `Derivative[0, 1][u][x, 0] == 0`. Nakoniec vykreslime zložky riešenia $u(x,t)$.

```
VlnovaRovnica = {D[u[x, t], t, t] - D[u[x, t], x, x] == -9.80665};
podmienky = {u[x, 0] == 10 x^2 (1 - x)^2, u[0, t] == 0, u[1, t] == 0,
 Derivative[0, 1][u][x, 0] == 0};
premenne = u[x, t];
riesenie = premenne /. NDSolve[{VlnovaRovnica, podmienky}, premenne,
  {x, 0, 1}, {t, 0, 4}][[1]]
Plot3D[riesenie, {x, 0, 1}, {t, 0, 4}, PlotRange -> All,
 AxesLabel -> {"x", "t", ""}]
```



Obr. 12.8. Riešenie vlnovej rovnice $u(x,t)$ v oblasti $[0,1] \times [0,4]$.

Vypíšeme si niekoľko funkčných hodnôt riešenia $u(0,5;t)$, pre $t \in \{0;0,4;0,8;\ 1,2;\ 1,6;\ 2;\ 2,4\}$

```
Table[riesenie /. x -> 0.5, {t, 0, 2.4, 0.4}]
```

$\{0.625, -0.703437, -2.6978, -2.69672, -0.704929, 0.624796, -0.701898\}$

Nájdené približné hodnoty riešenia vlnovej rovnice sú $u(0,5;0) = 0,625$, $u(0,5;0,4) = -0,703437$, $u(0,5;0,8) = -2,6978$, $u(0,5;1,2) = -2,69672$, $u(0,5;1,6) = -0,704929$, $u(0,5;2) = 0.624796$, $u(0,5;2,4) = -0,701898$. □

Príklad 61. *Nelineárna rovnica v troch dimenziach. Hľadajme riešenie* $u(t,x,y)$ *trojdimenzionálnej nelineárnej parciálnej diferenciálnej rovnice*

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}\frac{\partial^2 u}{\partial y^2} + (1 - u(t,x,y)^2)(1 + 2u(t,x,y)),$$

s vedľajšími podmienkami

$$\begin{aligned} u(0,x,y) &= e^{-(x^2+y^2)}, \\ u(t,-5,y) &= u(t,5,y), \\ u(t,x,-5) &= u(t,x,5), \\ \frac{\partial u}{\partial t}(0,x,y) &= 0, \end{aligned}$$

na intervaloch $t \in [0,4]$, $x \in [-5,5]$ *a* $y \in [-5,5]$ *príkazom* `NDSolve`*. Vykreslite graf riešenia.*

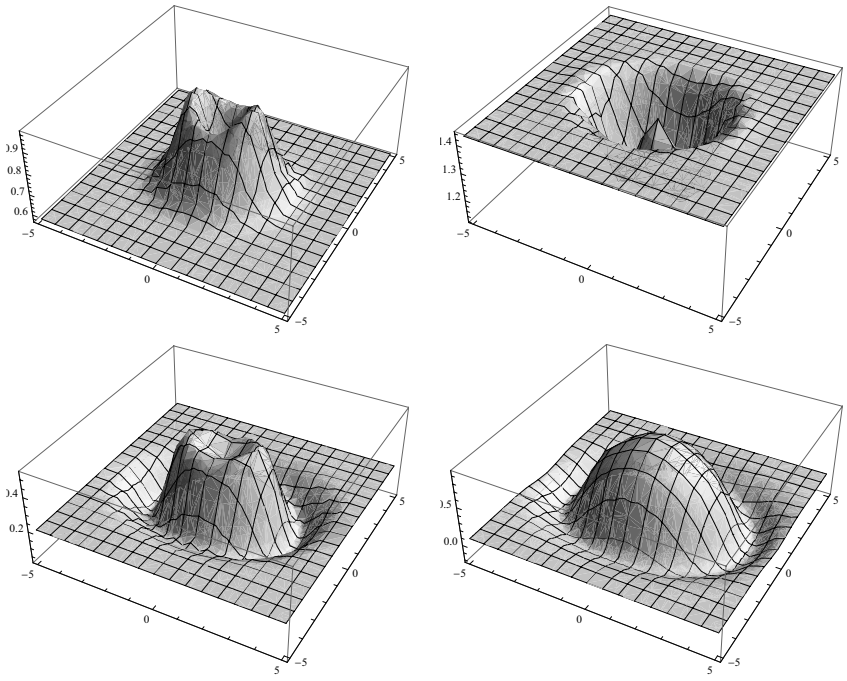
Riešenie. Vytvorme návod riešenia v programe *Mathematica*. Najskôr zadajme rovnicu, ktorú ideme riešiť v premennej `rovnica`, zadajme vedľajšie podmienky v premennej `podmienky`. Prvú parciálnu deriváciu $\frac{\partial u}{\partial t}(0,x,y) = 0$ zadáme príkazom `Derivative[1, 0, 0][u][0, x, y] == 0`. Hľadanú funkciu uvedieme v zozname `premenne`. Nakoniec vykreslíme riešenie $u(t,x,y)$ ako štyri grafy funkcii pre parameter $t = \{1,2,3,4\}$: $u(1,x,y)$, $u(2,x,y)$, $u(3,x,y)$ a $u(4,x,y)$ (pozri obrázok 12.9.).

```
rovnica = {D[u[t, x, y], t, t] == D[u[t, x, y], x, x] +
 D[u[t, x, y], y, y]/2 + (1 - u[t, x, y]^2) (1 + 2 u[t, x, y])};
podmienky = {u[0, x, y] == E^-(x^2 + y^2), u[t, -5, y] == u[t, 5, y],
 u[t, x, -5] == u[t, x, 5], Derivative[1, 0, 0][u][0, x, y] == 0};
premenne = u[t, x, y];
riesenie = NDSolve[{rovnica, podmienky}, premenne, {t, 0, 4},
 {x, -5, 5}, {y, -5, 5}]
Table[Plot3D[u[t, x, y] /. riesenie, {x, -5, 5}, {y, -5, 5},
 PlotRange -> All], {t, 1, 4}]
```

□

12.9 OBLASTI S KRIVOU HRANICOU

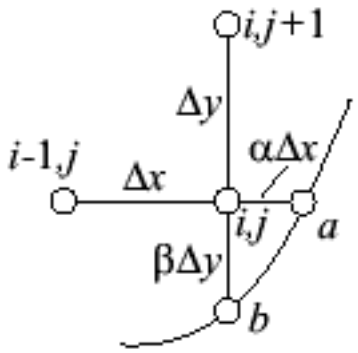
Predpokladali sme, že oblasti v ktorých počítame úlohy pre parciálne diferenciálne rovnice sú obdĺžnikové. V praxi majú však často oblasti zakrivené hranice. Aby sme sa vysporiadali so zakrivenými hranicami použijeme obdĺžnikovú mriežku ale prispôsobením diferenčnej rovnice v blízkosti hranice



Obr. 12.9. Riešenie nelineárnej rovnice v troch dimenziách. Grafy funkcii: hore vľavo $u(1,x,y)$, hore vpravo $u(2,x,y)$, dole vľavo $u(3,x,y)$ a dole vpravo $u(4,x,y)$.

krivým hraniciam. Zaveďme si špeciálne uzly v mriežke na prieniku čiar klasickej mriežky a zakrivenej hranice a označme ich a a b, v zhode s obr. 12.10. Vzdialenosť uzla (x_i, y_j) od uzla a je $\alpha\Delta x$ a vzdialenosť od uzla b je $\beta\Delta y$. Laplaceova diferenčná rovnica 12.1 v uzle (x_i, y_j) bude mať potom nasledovný tvar:

$$\begin{aligned}\left(\frac{u_a - u_{i,j}}{\alpha\Delta x} - \frac{u_{i,j} - u_{i-1,j}}{\Delta x}\right) \Big/ \left(\frac{\Delta x}{2}(1+\alpha)\right) \quad &+ \\ \left(\frac{u_{i,j+1} - u_{i,j}}{\Delta y} - \frac{u_{i,j} - u_b}{\beta\Delta y}\right) \Big/ \left(\frac{\Delta y}{2}(1+\beta)\right) \quad &= \quad 0.\end{aligned}$$



Obr. 12.10. Uzly na zakrivenej hranici.

Zhrňme dostupné príkazy numerického riešenia parciálnych diferenciálnych rovníc v programe *Mathematica*:

`NDSolve[eqn, conds, u[x, t], x, a, b, t, c, d]` Numerické riešenie diferenciálnej rovnice zadanej v `eqn` so začiatočnými podmienkami v `conds`. Hľadaná funkcia `u[x t]` je funkciou dvoch premenných na intervaloch $x \in [a, b]$ a $t \in [c, d]$.
`NDSolve[eqn, conds, u[t, x, y], t, a, b, x, c, d, y, e,f]` Numerické riešenie diferenciálnej rovnice zadanej v `eqn` so začiatočnými podmienkami v `conds`. Hľadaná funkcia `u[t, x, y]` je funkcia na intervaloch $t \in [a, b]$, $x \in [c, d]$ a $y \in [e, f]$.
`Derivative[1, 0][u][0, x] == 0` je príkaz na zadanie Neumannovej okrajovej podmienku $\frac{\partial u}{\partial t}(0, x) = 0$.

12.10 ÚLOHY

Úlohy riešte pomocou programu *Mathematica*.

1 Riešte parciálnu diferenciálnu rovnicu šírenia tepla

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0,$$

s vedľajšími podmienkami

$$\begin{aligned} u(x, 0) &= \sin x, \\ u(0, t) &= 0, \\ u(5, t) &= 0. \end{aligned}$$

na intervale $x \in [0, 5]$, $t \in [0, 10]$ príkazom `NDSolve`. Vykreslite graf riešenia $u(x, t)$.

2 Riešte nelineárnu parciálnu diferenciálnu rovnicu

$$\frac{\partial^2 z}{\partial t^2} = \frac{z(t,s)}{(z(t,s)^2 + (\frac{1}{2}(1 + 0,1\sin(2\pi t)))^2)^{3/2}},$$

s vedľajšími podmienkami

$$\begin{aligned} z(0,s) &= s, \\ \frac{\partial z}{\partial t}(0,s) &= 0. \end{aligned}$$

na intervale $t \in [0,10]$, $s \in [1,2]$ príkazom `NDSolve`. Vykreslite grafy riešenia $z(t,s)$.

LITERATÚRA

1. Bradley, G.: *A Primer of Linear Algebra*, Prentice-Hall, Englewood Cliffs, New York, 1975.
2. Ďurikovič, V., Ďurikovič, D.: *Matematická analýza 2*, ISBN 978-80-89220-47-9, Univerzita sv. Cyrila a Metoda, Trnava, 2006.
3. Ďurikovič, V., Ďurikovič, R.: *Matematická analýza 3*, ISBN 978-80-89220-83-0, Univerzita sv. Cyrila a Metoda, Trnava, 2008.
4. Gera, M., Ďurikovič, V.: *Matematická analýza 1*, Alfa, Bratislava 1990.
5. Hildebrand, F. B.: *Introduction to Numerical Analysis*, Second Edition, ISBN-0-486-65363-3, Dover Publications, New York, 1987.
6. Householder, A.: *The Theory of Matrices in Numerical Analysis*, Dover Publications, New York, 1975.
7. Quarteroni, A., Sacco R., Saleri F.: *Numerical Mathematics*, Second Edition, ISBN-10 3-540-34658-9, Springer Press, Berlin Heidelberg, 2007.
8. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T.: *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, Second Edition, ISBN-0-521-43064-X, Cambridge, England: Cambridge University Press, 1992.
9. Roussas, George G.: *A Course in Mathematical Statistics*, Second Edition, ISBN 0-12-599315-3, Academic Press, London, UK, 1997.

10. Schäfer, M.: *Computational Engineering – Introduction to Numerical Methods*, ISBN-10 3-540-30685-4, Springer Press, Berlin Heidelberg, 2006.

11. Kluvánek, I., Mišík, L, Švec, M.: *Matematika I 1. vyd.*, SVTL, Bratislava 1959.

12. Kluvánek, I., Mišík, L, Švec, M.: *Matematika II 1. vyd.*, Alfa, Bratislava 1970.

13. Ralston, A.: *Základy numerické matematiky.*, Praha, Academia 1978.

14. Vitásek, E.: *Numerické metódy*, SNTL Nakladatelství technické literatury, Praha 1987.

15. Wolfram, S.: *The Mathematica Book*, Fifth Edition, Wolfram Media-Cambridge University Press, Chanpaign, Illinois 2003.

16. van der Waerden, B.L.: *Algebra Volume I*, ISBN 0-387-40624-7, Springer-Verlag Press, New York, USA, 2003.

Register

Numerická matematika pre informatika

Riešené príklady v programe Mathematica

Autori:
Roman Ďurikovič
Vladimír Ďurikovič

Recenzenti:
prof. RNDr. Rudolf Kodnár, DrSc.
Žilinská univerzita v Žiline,
doc. RNDr. Angela Handlovičová, CSc.
Slovenská technická univerzita v Bratislave

1.vydanie
Vydala Univerzita sv. Cyrila a Metoda v Trnave
v roku 2011
Náklad 150 ks
Tlač: Michal Vaško - Vydavateľstvo, Prešov

ISBN 978-80-8105-271-2