Jemný úvod do numerických metod

Matematické algoritmy (11MAG)

Jan Přikryl

8. přednáška 11MAG
pondělí 24. listopadu 2014

verze:2014-11-24 16:34

Obsah

1 Úvod **1**

1.1 Matematické modelování 2

1.2 Numerická matematika 2

1.3 Numerická úloha 4

2 Zobrazení čísel **5**

2.1 Celá čísla, pevná a pohyblivá řádová čárka 6

2.1.1 Reprezentace $\mathbb{R}$ v pevné řádové čárce 6

2.1.2 Reprezentace $\mathbb{R}$ v pohyblivé řádové čárce 6

3 Chyby **7**

3.1 Typy chyb v matematickém modelování 7

3.2 Relativní a absolutní chyba 7

3.3 Vliv zahokrouhlovacích chyb 9

3.4 Vliv aritmetických operací na relativní chybu 10

4 Typy úloh **10**

4.1 Matematická úloha a její formalizace 10

1 Úvod do numerické matematiky

Pro detailnější obeznámení s pojmy, uváděnými níže, doporučuji konzultovat monografii Michaela T. Heathe [1], případně nějaká z mnoha skript o numerické matematice, která v posledních letech vyšla – například [3] (části tohoto skripta jsou dostupné i on-line).

1.1 Matematické modelování

Termínem **matematické modelování** označujeme proces tvorby **matematického modelu**, jenž popisuje námi zkoumaný systém pomocí matematických pojmů a jim odpovídajícího zápisu. Připomeňme si, že jako **systém** chápeme část prostředí, kterou lze vnímat odděleně od jejího okolí. Systém od okolí odděluje nějaká hranice, ať už fyzická, či myšlenková.

Abychom mohli zkoumat chování nějakého systému, můžeme

- provádět **experimenty** anebo
- popsat systém matematicky – sestavit jeho **matematický model**.

Matematické modely mohou v závislosti na použitém matematickém aparátu nabývat mnoha různých forem. Nejčastěji se setkáme s diskrétními dynamickými systémy, statistickými modely, spojitými modely založenými na diferenciálních rovnicích, či s modely využívajícími poznatků teorie her. Na fakultě jste se s mnoha přístupy k matematickému modelování zajisté již setkali. Uveďme jeden příklad za všechny: V rámci předmětu *Modelování systémů a procesů* [4] jsme si ukazovali různé modely, popisující chování systémů ve spojitém či diskrétním čase a popis systémů těmito modely dělili na *vnější* a *vnitřní* (stavový) popis.

Příklad 1 (Závaží na pružině). Netlumené kmity závaží na pružině popisuje homogenní diferenciální rovnice harmonických kmitů

$$\frac{\mathrm{d}^2 y(t)}{\mathrm{d}t^2} + \omega^2 y(t) = 0.$$

V této rovnici označuje $y(t)$ polohu závaží v čase t a ω určuje úhlovou frekvenci kmitů. Počáteční podmínky této diferenciální rovnice potom udávají počáteční výchylku závaží od rovnovážné polohy a jeho počáteční zrychlení.

Příklad 2 (Model vývoje dluhu). Finanční model vývoje zadlužení může mít tvar diferenční rovnice

$$y[n+1] = (1 + \alpha[n]) \cdot y[n] - u[n].$$

Zde je $y[n]$ posloupnost výšky dluhu (hodnota $y[0]$ bude odpovídat výšce původní půjčky), $\alpha[n]$ je v čase proměnná úroková míra a $u[n]$ označuje posloupnost splátek dluhu.

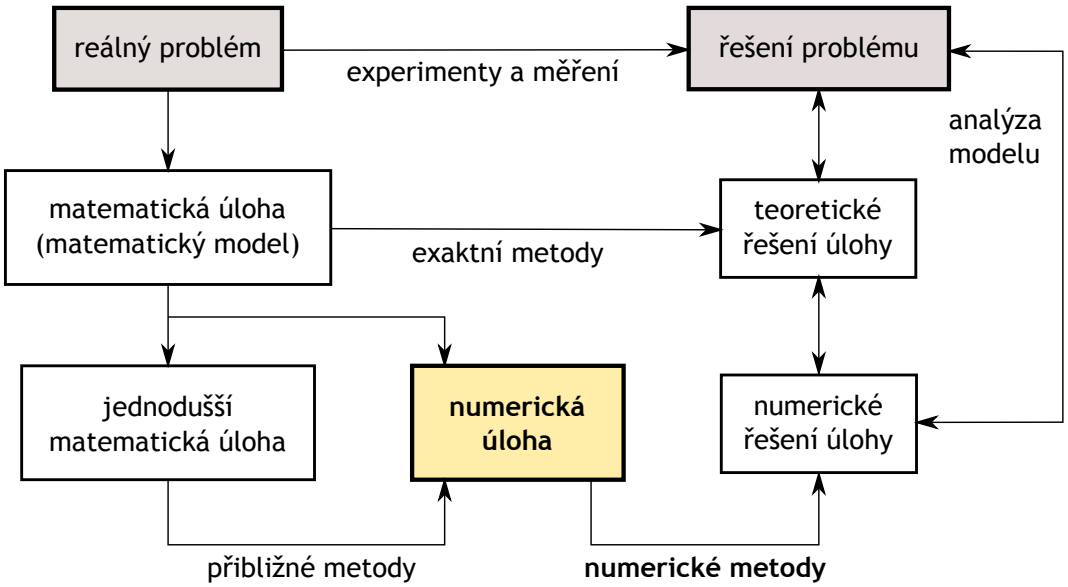
Ke zkoumání matematických modelů systémů jsme používali Matlab a Simulink.

V příštích přednáškách si stručně povíme

- co vlastně počítač musí umět, aby dokázal s dostatečnou přesností počítat s matematickými modely reálného světa,
- jaké matematické algoritmy se ve vybraných případech používají a
- proč není dobré počítači vždycky slepě věřit.

1.2 Numerická matematika

Reálný problém, který se snažíme vyřešit – například nastavení tuhosti odpružení podvozku nového modelu automobilu, nebo optimální alokaci skladů zásilkového obchodu –, můžeme zkoumat experimentální cestou, porovnáváním různých řešení a měřením výsledků bez detailní



Obrázek 1: Pozice numerické úlohy a numerických metod v kontextu matematického modelování.

znalosti jeho modelu. Pokud ale potřebujeme určité záruky, že jsme danou úlohu vyřešili korektně a chceme dodat našemu způsobu řešení důvěryhodný základ, použijeme na vyřešení problému nějaký **matematický model** (o něm jsme se už zmiňovali výše). Kdo je zdrojem tohoto modelu, není v tento okamžik až tak důležité: v případě fyzikálních úloh to patrně bude někdo, kdo dokáže matematicky popsat fyzikální stránku řešené úlohy, v případě chemických procesů pak to bude někdo znalý potřebné fyzikální chemie.

V procesu řešení matematického modelu zpravidla nemůžeme postupovat přímo, musíme se obrátit k jeho nahrazení jednodušší úlohou, tak zvaným **počítačovým modelem**. To je původní model upravený tak, že jej lze v konečném čase spočítat na počítači. Výsledky počítačového modelu jsou ale pouze přibližné (i když většinou dostatečně přesné) a slouží jako **přiblížení** neboli **aproximace** výsledků, které by byly přesným řešením uvažovaného matematického modelu. Dobré počítačové modely nám zároveň poskytují informaci o přesnosti získaných výsledků.

Takováto přibližná metoda řešení se v praxi používá velmi často, zapotřebí jí není spíše v malém počtu případů. Jako příklad nám můžou sloužit početné modely technických systémů založené na diferenciálních rovnicích. Úlohy pro diferenciální rovnice můžeme sice někdy řešit analyticky (tedy na papíře, užitím standardního matematického aparátu), ale u většiny diferenciálních rovnic takové analytické řešení není možné a musíme se spokojit s jeho aproximací.

Příklad 3 (Přibližná metoda řešení). Soustavu parciálních diferenciálních rovnic nelze zpravidla řešit přímo (podobně, jako jsme to zmiňovali u obyčejné diferenciální rovnice). Pokud ale nahradíme parciální derivace diferencemi a celou původní doménu úlohy rozdělíme na síť rovinných či objemových elementů (použijeme postup, jenž se nazývá *metoda konečných prvků*), lze původní soustavu převést na soustavu algebraických rovnic, lineárních nebo nelineárních – to je v našem případě ta jednodušší matematická úloha, uvedená na Obrázku 1. Soustavu lineárních rovnic můžeme být schopni vyřešit přesně například Gaussovou eliminací, ale tento proces je pro velké soustavy velmi pomalý a v mnoha případech se spokojíme s přibližným a mnohem rychlejším řešením některou iterační metodou.

■ doplnit další příklady? ■

1.3 Numerická úloha

Protože počítač je konečný automat pracující pouze s konečným počtem vstupních a výstupních dat, zavádí se někdy také pojem numerické úlohy.

Numerická úloha – jasný a jednoznačný popis funkčního vztahu mezi *konečným* počtem vstupních a výstupních dat. Data jsou tedy v případě numerické úlohy vyjádřitelná vždy konečným počtem čísel (například vektorem či maticí, případně nějakou komplikovanější datovou strukturou). Některé matematické úlohy jsou rovnou numerickými úlohami – například již zmíněné řešení soustav lineárních rovnic.

⇒ Počítačový model je taková aproximace matematického modelu, jež může být v konečném čase realizována na počítači.

Poznamenejme, že ne všechny numerické úlohy lze na počítači vyřešit v konečném počtu kroků. Takové řešení lze provést přesně například u řešení soustav lineárních rovnic (nebereme-li v úvahu zaokrouhlovací chyby), ale ne už obecně při hledání vlastních čísel matic, což je také numerická úloha. U řady numerických úloh se tedy spokojíme s tím, že stanovíme nějaké jejich (vhodně definované) přibližné řešení (aproximaci). Taková řešení se pak počítají pomocí vhodných **numerických metod**. **■ O aproximaci se mluví výše. ■**

Numerický algoritmus – postup, kterým se v konečném počtu kroků řeší daná numerická úloha. Při studiu vlastností numerických algoritmů nás zajímá především realizace aritmetických operací s čísly, nikoliv logické operace. Algoritmus numerické metody (Newtonova metoda) – teoretická záležitost, nemusí být konečný.

Realizace algoritmu na počítači (též **implementace**), směřuje k programu – třeba algoritmus ve Fortranu, realizovatelný v konečném čase (má už epsilon a maximální povolený počet iterací); hledím, abych se vyhnul třeba odčítání sobě blízkých čísel.

Konečná fáze je **program** – žádná numerická metoda mně nic nespočítá, vždycky je to až program.

Konstrukce a analýza metod a algoritmů pro realizaci numerických úloh na počítačích: **numerická matematika**.

Vstupní a výstupní data jsou vlastně danými a hledanými objekty numerické úlohy [3].

Příklad 4 (Numerická úloha). Přibližné řešení rovnice $x^4+a_1x^2+a_2x+a_3=0$ je možno počítat numericky pro konkrétní vstupní vektor $\mathbf{a}=[a_1,a_2,a_3]\in\mathbb{R}^3$.

Výstupem numerické metody řešení bude vektor $\mathbf{x}=[x_1,x_2,x_3,x_4]\in\mathbb{C}^4$. Řešení je v obecném případě garantováno pouze přibližné, při vhodné volbě vektoru parametrů $\mathbf{a}$ jej ale můžeme získat i přesně.

Příklad 5 (Co není numerická úloha). Řešení rovnice $y''(x)-y(x)^2=0$ za daných počátečních podmínek nelze vyjádřit konečným počtem čísel a nelze jej tedy hledat numericky (řešením rovnice je funkce, nikoliv množina číselných hodnot).

Numerický přístup lze použít pouze pro vyšetření hodnot ve vybraných bodech $x\in\{x_i\}_1^n$, kde typicky $x_i=x_0+i\cdot\Delta$, spojitý interval tedy aproximujeme diskrétním. Vyžaduje-li to přesnost výpočtu, nemusí být hodnoty x_i rozmístěny ekvidistantně – to uvidíme například při numerickém výpočtu integrálů.

V okamžiku, kdy máme v ruce výsledky, musí ještě proběhnout **analýza modelu**, respektive **analýza řešení**. Účelem těchto analýz je **verifikovat** a **validovat** použitý model a metodu řešení a obdržet odpověď na následující otázky:

validace – Počítáme se správným modelem? Pokud jsou přípustné teploty kladné, nepředpovídá model zápornou teplotu? Má-li modelovaná veličina monotónně růst nebo klesat, nedochází ve výsledku k oscilacím?

verifikace – Počítá náš model správně? Nedochází v něm k nepřípustným chybám?

Zdrojům možných chyb při použití validního modelu se budeme věnovat v jednom z následujících odstavců. Nejprve si ale musíme vysvětlit, jak vlastně počítač reprezentuje reálná čísla.

2 Zobrazení čísel v počítači

Dnešní počítače pracují výlučně v binárním kódu – veškerá informace je ukládána v operační paměti a na vnější paměťová média ve formě jedniček a nul. Mimo jiné to znamená, že namísto desítkové soustavy, používané pro počítání na papíře, jsou v počítačích čísla reprezentována ve dvojkové soustavě.

2.1 Celá čísla, pevná a pohyblivá řádová čárka

Celá čísla – ekvivalenty ve dvojkové soustavě, jeden (nejvyšší) bit na znaménko. Celá čísla reprezentují i čísla přirozená ve dvojnásobném rozsahu, musíme si proto dávat pozor, s čím zrovna počítáme (v ANSI C deklarace `int` versus `unsigned int`). ■ **Zmínit přímý a doplňkový kód** ■

Příklad 6. $66 = (01000010)_2$, $-126 = (11111110)_2$, ovšem také $(11111110)_2 = 254$

2.1.1 Reprezentace $\mathbb{R}$ v pevné řádové čárce

Pevná řádová čárka – pevný počet bitů pro celou a desetinnou část čísla. *Q notace* označuje počet bitů před a za desetinnou čárkou[5]: Q3.5 je osmibitové číslo, jež může nabývat hodnot $000{,}00000_2$ až $111{,}11111_2$, tedy $0{,}00000, 0{,}03125, \ldots 7{,}93750, 7{,}96875$.

Příklad 7. $5{,}3100_{10} \approx 10101010_2 (= 101{,}01010_2)$, $7{,}5625_{10} = 11110010_2$

2.1.2 Reprezentace $\mathbb{R}$ v pohyblivé řádové čárce

Pohyblivá řádová čárka – převod na tvar $a \cdot q^b$, kde $a \in \mathbb{R}$, $1 \leq a < 10$ je **mantisa**, $b \in \mathbb{Z}$ je **exponent** a $q \in \mathbb{N}$ je **základ**, typicky 2 (PC, IEEE 754) nebo 10 (kalkulačka).

Definice 8 (Semilogaritmický tvar)**.** Číslo x lze reprezentovat v **semilogaritmickém tvaru s normalizovanou mantisou** jako

$$x = \operatorname{sgn}(x) \cdot \left(\frac{a_1}{q} + \frac{a_2}{q^2} + \cdots + \frac{a_l}{q^l} \right) q^b,$$

kde $q \in \mathbb{N}$, $q > 1$ je *základ*, $a_i \in \{0, 1, \ldots, q-1\}$, $a_1 \geq 1$, jsou *číslice mantisy* a $b \in \{m_1, \ldots, m_2\}$ $m_1, m_2 \in \mathbb{Z}$, obvykle $m_1 < 0$ a $m_2 > 0$ je *exponent*. Předpokládá se, že b je v intervalu $\langle m_1, m_2 \rangle$ reprezentováno rovnoměrně. Stejně tak se uvažuje pouze normalizovaná mantisa $m \in \langle 1, q)$, protože m a b nejsou jinak určeny jednoznačně.

Všimněte si, že reprezentace x pokrývá pouze podmnožinu $\mathbb{R}$ – má pouze $2(q-1)q^{l-1}(m_2 - m_1 + 1) + 1$ prvků. Některá reálná čísla nelze tedy přesně reprezentovat. Jsou to například čísla s moc velkým a malým exponentem a čísla, která nemají konečný q rozvoj. Další limitující faktor je konečný počet desetinných míst mantisy – je jich pouze l.

Příklad 9 (Reprezentace 1/2 a 1/10). Budeme-li uvažovat $q = 2$, bude

$$\frac{1}{2} = \left(\frac{1}{2} + \frac{0}{4} + \frac{0}{8} \cdots \right) 2^0$$

ale

$$\frac{1}{10} = \left(\frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + \frac{1}{4096} + \frac{1}{8192} + \frac{1}{32768} \cdots \right) 2^{-4}$$

nelze reprezentovat konečným rozvojem.

Jedním z důsledků semilogaritmické reprezentace čísel v pohyblivé řádové čárce je i to, že v počítači existuje největší a nejmenší reprezentovatelné číslo.

Platí $a_1 \neq 0$, což při binární reprezentaci znamená ale $a_1 = 1$, a proto se a_1 vůbec při binární reprezntaci vůbec neukládá a získáme tak „zadarmo“ jeden bit přesnosti reprezentace navíc.

Jak zobrazení čísel, tak chování počítačové aritmetiky jsou dnes určovány všeobecně uznávanými normami, z nichž nejčastěji se užívá norma IEEE 754 [2]. Aritmetika podle této normy má tu vlastnost, že výsledek operace s dvěma strojovými čísly dává vždy nějaký strojově zobrazitelný výsledek. Tento požadavek vede k tomu, že v IEEE reprezentaci jsou potom speciální příznaky pro případy, kdy například dojde k podtečení a reálné číslo se reprezentuje v nižší přesnosti (n už nejde snížit) s hodnotou $a_1 = 0$ – tato čísla nazýváme subnormální (nebo denormalizovaná) ■ **ubírá se mantisa, jsou pod `realmin`, ale to definujeme až později** ■. Podobně jsou zde symboly pro překročení číselného rozsahu směrem k nekonečnu (v Matlabu výsledek `Inf`) nebo pro výsledek, který matematicky nedává smysl (`NaN`, například výpočet 0/0).

Hezký interaktivní příklad výpočtu reprezentace decimálně zapsaných čísel podle IEEE 754 lze nalézt na stránkách http://babbage.cs.qc.cuny.edu/IEEE-754/.

Když už nyní zhruba víme, jak se v počítači reprezentují čísla, podívejme se na možné zdroje chyb při matematickém výpočtu.

3 Typy chyb

V Odstavci 1 jsme se zmínili o tom, že při tvorbě matematického modelu problému z reálného světa jsme vždy nuceni provést určitá přiblížení a některé skutečnosti si idealizovat. Rozdíl řešení idealizovaného problému (matematické úlohy) a řešení reálného problému nazýváme **chybou matematického modelu** nebo také jenom **chybou modelu**. Z velikosti této chyby posuzujeme zpětně vhodnost či nevhodnost zvoleného modelu, tj. aproximace reality.

3.1 Typy chyb v matematickém modelování

Jestliže k řešení matematické úlohy použijeme metodu, která nám neposkytne přesné (teoretické) řešení dané úlohy, pak chybu, které se dopustíme, nazýváme **chybou metody**. Typickým příkladem je chyba, které se dopustíme, když za limitu nekonečné posloupnosti vezmeme některý její člen s dostatečně velkým indexem. Často řešíme matematickou úlohu tím, že pomocí jistých metod ji nahradíme (aproximujeme) úlohou jednodušší – obvykle již úlohou numerickou – a rozdíl řešení těchto dvou úloh nazýváme **chybou aproximace**. Tato chyba se často bere jako součást chyby metody.

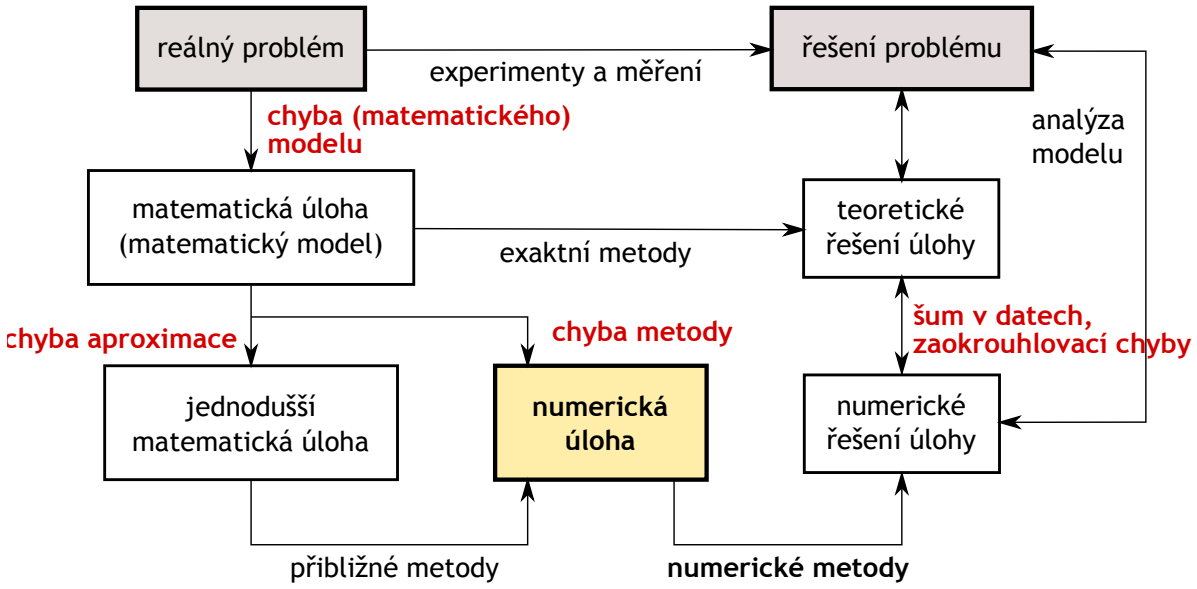
K posouzení přesnosti výsledku musíme ještě vzít v úvahu **chyby (šum) ve vstupních datech**, dané jednak chybami měření, jednak způsobené zobrazením vstupních dat do nesouvislé množiny reprezentující data v počítači. Poslední skupinou chyb jsou **chyby zaokrouhlovací**. Do této skupiny zahrnujeme všechny nepřesnosti způsobené realizací algoritmu v počítači včetně nepřesného provádění aritmetických operací.

3.2 Relativní a absolutní chyba

V dalším textu budeme používat označení, v němž číslo x v numerickém algoritmu je reprezentováno přiblížením $\tilde{x}$.

Definice 10 (Absolutní a relativní chyba)**. Absolutní chybou** $\mathcal{A}(x)$ aproximace čísla x číslem $\tilde{x}$ označujeme rozdíl

$$\mathcal{A}(x) = |x - \tilde{x}|$$



Obrázek 2: Typy chyb v matematickém modelování

Relativní chybou $\mathcal{R}(x)$ aproximace čísla x číslem $\tilde{x}$ označujeme podíl

$$\mathcal{R}(x) = \frac{\mathcal{A}(x)}{|x|} = \left|\frac{x - \tilde{x}}{x}\right|, \ x \neq 0$$

■ zmínit data naměřená s chybou? ■

Absolutní chyba tak není nazývána kvůli absolutní hodnotě, jedná se o absolutní odchylku mezi reprezentací čísla a jeho pravou hodnotou. Velikost absolutní chyby není postačujícím měřítkem pro posouzení přesnosti výpočtu, zde se nám naopak velmi hodí relativní chyba – ta bere v úvahu i velikost čísla, s nímž se počítá. Pro x blízko 0 se ovšem může stát, že relativní chyba bude moc přísná (uvidíme to v další přednášce při výpočtu kořenů nelineární funkce Newtonovou metodou) a v této oblasti je vhodnější měřit absolutní chybou.

Připomeňme si, co jsme zmínili již v předešlém odstavci, když jsme se bavili o reprezentaci reálných čísel v pohyblivé řádové čárce: *Reálná čísla nejsou v počítači většinou reprezentována přesně*. Použijeme-li již zmíněný standard IEEE 754 [2], čísla v pohyblivé řádové čárce reprezentujeme standardně ve **dvojnásobné přesnosti** (tj. *IEEE 754 binary64* s 53 bity mantisy, z nichž je uloženo 52)[1] – relativní chyba této reprezentace je o malinko větší, než 10^{-16} (mantisa má 15,95 platných dekadických číslic). Pokud je to třeba, lze na úkor vyšších zaokrouhlovacích chyb volit **jednoduchou přesnost** (tj. *IEEE 754 binary32* s 24 bity mantisy, z nichž je uloženo 23)[2], kde je relativní chyba reprezentace o malinko nižší, než 10^{-7} (mantisa má 7,22 platných dekadických číslic).

Jak vidíme, je-li relativní chyba reprezentace čísla x hodnotou $\tilde{x}$ rovna $\mathcal{R}(x) = 10^{-d}$, znamená to, že x je má ve zvolené reprezentaci d platných číslic. Absolutní chyba $\mathcal{A}(x) = 10^{-d}$ udává počet platných desetinných míst.

Přesnost aritmetiky v Matlabu: `eps` (relativní chyba reprezentace čísla), `realmax` (největší zobrazitelné číslo), `realmin` (nejmenší plně reprezentovatelné – tedy nikoliv subnormální – číslo).

3.3 Vliv zahokrouhlovacích chyb

Nahromadění zaokrouhlovacích chyb během postupu výpočtu může nakonec vést k situaci, kdy výsledek výpočtu je zcela odlišný od správné hodnoty – a to může vést k velmi závažným následkům. Jedním z tragických příkladů takové chyby je selhání baterie protiraketových střel Patriot, chránících americkou základnu Dhahran v Saudské Arábii během války v Perském zálivu dne 25. února 1991, jež stálo život 28 amerických vojáků, dalších 98 bylo zraněno [6]. Střely Patriot se přitom v době konfliktu považovaly za velmi úspěšné prostředky ochrany proti raketovým útokům protivníka. V tomto případě ale baterie vůbec nevystřelila. Co se vlastně stalo?

Příklad 11 (Proč Patriot netrefí Scud). Hlavním důvodem selhání byl nepřesný výpočet časové základny v zaměřovacím systému baterie Patriot: Systém počítal s hodnotami času v desetinách sekundy, jeho autoři proto systémový čas v sekundách získávali prostým vynásobením hodnotou 0,1. Jak jsme viděli v Příkladě 9, většina desetinných zlomků nemá ale v pohyblivé řádové čárce s binárním základem konečný rozvoj, platí

$$0{,}1 \approx (0{,}0001100110011001100110011001100110011\ldots)_2.$$

[1] V Matlabu i ANSI C/C++ `double`.
[2] V Matlabu `single`, v ANSI C/C++ `float`.

Interní registr v řídicím systému baterie pracoval ovšem pouze v jednoduché přesnosti, měl tedy 24 bitů mantisy a efektivně tak ukládal číslo

$$0{,}1 \approx (0{,}00011001100110011001100110011)_2 \approx 0{,}099999994,$$

tedy reprezentoval desetinu sekundy s absolutní chybou přibližně 0,000000096.

Systém, chránící spojeneckou základu, byl v provozu nepřetržitě po dobu 100 hodin, a během této doby odchylka časové základy od referenčního času dosáhla 0,34 sekundy. Scud letí (či spíše jeho zbytky včetně hlavice padají) rychlostí okolo 1700 m/s, posun časové základny potom způsobil, že řídicí systém baterie jej po prvotním radarovém kontaktu hledal v bodě cca 500 m za skutečnou polohou útočící střely. Vzhledem k tomu, že se v dané oblasti nic nevyskytovalo, prohlásil původní radarový kontakt za falešný poplach a proti blížícímu se Scudu nebyly odpáleny žádné rakety.

K příkladu je třeba pro úplnost doplnit dvě věci: (i) Patriot je původně mobilní systém protiletadlové obrany, upravený na ničení taktických balistických střel. Při návrhu se nepočítalo s dlouhodobým nasazením při obraně statických cílů – nikdo proto nepředpokládal, že by celý systém byl aktivován po dobu několika dní. (ii) Řešení problému přitom bylo jednoduché (a navržené izraelskými odborníky asi dva týdny před incidentem): stačilo vždy po pár hodinách restartovat řídicí systém baterie. Opravený software byl přitom výrobcem dodán 26. února 1991, tedy den po incidentu v Dhahranu.

3.4 Vliv aritmetických operací na relativní chybu

Aritmetické operace mohou mít na nepřesné reprezentace čísel devastující vliv (například podíl velkého a malého čísla, ale i odčítání dvou sobě blízkých čísel stejného znaménka).

Relativní chyba se může výrazně zvětšit při odčítání dvou blízkých čísel:

$$\mathcal{R}(x \pm y) = \frac{\mathcal{A}(x \pm y)}{|x \pm y|}$$

Podmínkou ale je, že čísla x a y nejsou v počítači reprezentována přesně. Násobení ani dělení nemají na $\mathcal{A}(x)$ a $\mathcal{R}(x)$ výraznější vliv.

Příklad 12. Mějme čísla $x_1 = 758320$, $x_2 = 757940$, a nechť jsou reprezentována jako $\tilde{x}_1 = 758330$ a $\tilde{x}_2 = 757930$. Platí $\mathcal{A}(x_1) = 10$, $\mathcal{A}(x_2) = 10$,

$$\mathcal{R}(x_1) = \frac{10}{758320} \leq 1{,}32 \cdot 10^{-5}, \mathcal{R}(x_2) = \frac{10}{757940} \leq 1{,}32 \cdot 10^{-5}.$$

Máme tedy $v = x_1 - x_2 = 380$ a je $\tilde{v} = \tilde{x}_1 - \tilde{x}_2 = 400$. Proto $\mathcal{A}(v) = |v - \tilde{v}| = 20$ a

$$\mathcal{R}(v) = \frac{\mathcal{A}(v)}{|v|} = \frac{20}{380} \leq 0{,}053.$$

Relativní chyba rozdílu $v = x_1 - x_2$ je tedy o tři řády vyšší než relativní chyby obou operandů.

4 Typy numerických úloh

4.1 Matematická úloha a její formalizace

Mějme dány dva vektorové prostory $\mathcal{B}_x$ (vstupní data) a $\mathcal{B}_y$ (výstupní data).

Definice 13 (Matematická úloha)**. Matematickou úlohou** rozumíme relaci

$$y = U(x),\ x \in \mathcal{B}_x, y \in \mathcal{B}_y$$

Definice neříká nic jiného, než že matematická úloha transformuje posloupnost vstupních dat na posloupnost výsledků. Ne všechny úlohy, odpovídající formální definici uvedené výše, lze na počítači numericky řešit. Možné to je pouze pro podmnožinu všech úloh, kterou si můžeme definovat takto:

Definice 14 (Korektní úloha)**.** Řekneme, že úloha je **korektní**, pokud

1. ke každému $x \in \mathcal{B}_x$ existuje právě jedno $y \in \mathcal{B}_y$,
2. řešení y spojitě závisí na datech, tedy pokud $x_n \to x$ a $U(x_n) = y_n$, pak také $y_n \to y = U(x)$.

Zbylé matematické úlohy označujeme jako **nekorektní**. Jde například o nejednoznačně řešitelné problémy, intervalové odhady, úlohy s nevhodnou formulací zadání.

Příklad 15 (Korektní úloha). Jako příklad korektní úlohy může sloužit například výpočet integrálu z dané spojité a ohraničené funkce přes nějaký interval.

Příklad 16 (Nekorektní úloha). Určete matici $\mathbf{A}$ splňující rovnici $\mathbf{Ax} = \mathbf{b}$ máte-li dány hodnoty $\mathbf{x}$ a $\mathbf{b}$

Příklad 17 (Jiná nekorektní úloha). Určete

$$y = \int_{-1}^{1} 1/x \,\mathrm{d}x.$$

Definice 18 (Číslo podmíněnosti)**.** Podíl

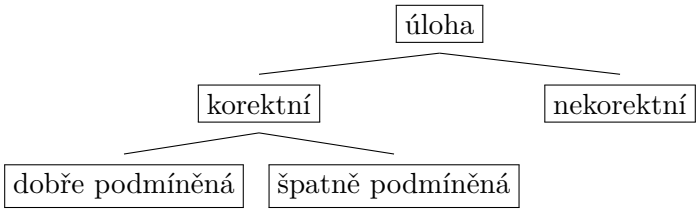
$$C_\mathrm{p} = \frac{\dfrac{|\Delta x|}{|x|}}{\dfrac{|\Delta y|}{|y|}}$$

se nazývá **číslo podmíněnosti** úlohy.

Číslo podmíněnosti nám udává, jak moc „silný“ je vliv změn ve vstupních datech úlohy na výstupní data. V případech, kdy $|x|$ nebo $|y|$ jsou malé, bývá namístě používat v definici čísla podmíněnosti v čitateli i jmenovateli nikoli relativní, ale absolutní chyby (mluvíme pak o **absolutním čísle podmíněnosti**).

Definice 19 (Dobře podmíněná úloha)**.** Budeme říkat, že korektní úloha je **dobře podmíněná**, jestliže malá změna ve vstupních datech vyvolá malou změnu řešení (resp. $C_\mathrm{p} \approx 1$).

Dobře podmíněné úlohy se numericky řeší „snadno“, tedy s malou očekávanou chybou a s relativně malými problémy se zaokrouhlovacími chybami. Úlohy špatně podmíněné díky svým vlastnostem způsobují numerické problémy, je třeba je řešit „opatrně“ (používat vhodné algoritmy, jež třeba nejsou tak efektivní, jsou ale numericky stabilní) a nebo se pokusit je transformovat na lépe podmíněné úlohy.



Obrázek 3: Taxonomie matematických úloh

Reference

[1] Michael T. Heath. *Scientific computing: an introductory survey.* McGraw-Hill, Boston, 2 edition, 2002.

[2] IEEE Std 754-2008. IEEE standard for binary floating-point arithmetic, 2008.

[3] Stanislav Míka and Marek Brandner. *Numerické metody I.* FAV ZČU, Plzeň, 2. edition, 2002.

[4] Modelování systémů a procesů.

[5] E. L. Oberstar. Fixed-point representation and fractional math, 2007.

[6] U.S. Government Accountability Office. Patriot missile defense: Software problem led to system failure at dhahran, saudi arabia., 1992.