NUMERICKÁ MATEMATIKA

Miroslav Vicher

7. ledna 2003

miroslav.vicher@mff.cuni.cz
http://mbox.troja.mff.cuni.cz/~vicher

Obsah

Kapitola 1

Numerická matematika

Celá řada úloh, které se vyskytují v matematice, není analyticky řešitelná nebo je nalezení přesného řešení příliš obtížné. Numerická matematika se snaží nalézt v těchto případech řešení přibližné.
Cílem těchto skriptech je vysvětlit základní algoritmy numerické matematiky, tak aby bylo možno základní algoritmy naprogramovat. Algoritmy lze také nalézt v knihovnách numerického softwaru pro správné používání těchto algoritmů je však třeba chápat jak fungují.

1.1 Reprezentace čísel

Čísla se kterými pracujeme v počítači se liší od čísel se kterými pracujeme v matematice. Hlavním rozdílem je omezený rozsah počítačových čísel a jejich omezená přesnost.

1.1.1 Dvojková soustava

Celá i reálná čísla jsou v počítači uložena ve dvojkové soustavě. Jednotlivé číslice c_i dvojkového zápisu čísla x se získají rozvojem

$$x = \sum x_i 2^i \tag{1.1}$$

Například číslo 26 je ve dvojkové soustavě 11010

$$26_{10} = 16 + 8 + 2 = 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 11010_2$$

Stejným způsobem se převedou do dvojkové soustavy i reálná čísla. Například číslo 6.75 je ve dvojkové soustavě 110.11

$$6.75_{10} = 4 + 2 + 0.5 + 0.25 = 1.2^2 + 1.2^1 + 0.2^0 + 1.2^{-1} + 1.2^{-2} = 110.11_2$$

Čísla která lze v desítkové soustavě zapsat pomocí konečného počtu číslic potřebují ve dvojkové soustavě nekonečný počet číslic, například

$$\frac{1}{3} = 0.33333333\ldots_{10} = 0.0101010101\ldots_2$$
$$\frac{1}{10} = 0.1_{10} = 0.000110011001100\ldots_2$$

1.1.2 Celá čísla

Při ukládání čísla do počítače se toto číslo nejprve převede do dvojkové soustavy. Jednotlivé číslice dvojkového zápisu se potom ukládají do jednotlivých bitů paměti počítače. Pro uložení čísla máme ale k dispozici pouze omezený počet bitů, který závisí na typu počítače a programovacím jazyku. Nejčastěji se setkáváme s 8, 16 a 32 bitovými čísly. Díky tomuto omezenému počtu je omezen i rozsah celých čísel která lze používat. Nejmenší číslo které lze uložit je 0 - všechny bity jsou nulové. Naopak největší číslo má ve všech bitech jedničku. Pokud máme k dispozici n bitů je toto číslo dáno součtem

$$x_{\max} = 111\ldots11_2 = \sum_{i=0}^{n} 2^i = 2^n - 1 \tag{1.2}$$

Největší hodnoty pro n bitová celá čísla

n	$x_{\max}$
8	255
16	65 535
32	4 294 967 295

Dosud popisovaná reprezentace čísel předpokládala, že všechna čísla jsou kladná. Pokud chceme používat i čísla záporná je třeba zvolit poněkud jinou reprezentaci, která dokáže rozlišit, zda dané číslo je kladné nebo záporné. Používá se metoda, která záporná čísla ukládá pomocí tzv. dvojkového doplňku.
V této metodě se kladná čísla ukládají stejně jako v předchozí metodě přímým uložením bitů. Reprezentaci záporného čísla získáme tak, že nejprve zapíšeme ve dvojkové soustavě jeho absolutní hodnotu. Poté zamění všechny jedničky za nuly a naopak. Nakonec přičteme jedničku. Výsledek po bitech uložíme do paměti. To, zda je uložené číslo kladné nebo záporné, se rozliší podle prvního bitu. Pokud je tento bit 1 je číslo záporné. Například pokud používáme 8 bitů bude reprezentace čísla -14 následující

$$\begin{aligned} -14 &\to 0000\,1100 && \text{číslo 14} \\ &\to 1111\,0011 && \text{záměna } 0\leftrightarrow 1 \\ &\to 1111\,0100 && \text{přičtení 1} \end{aligned}$$

Tento postup funguje stejně i obráceně, např.

$$\begin{aligned} & 1111\,0100 && \text{první je 1} \Rightarrow \text{záporné číslo} \\ \to\ & 0000\,1011 && \text{záměna } 0\leftrightarrow 1 \\ \to\ & 0000\,1100 = 14 && \text{přičtení 1} \\ \to\ & -14 \end{aligned}$$

Rozsah čísel, která lze v této reprezentaci používat se změnil. Největší číslo bude mít opět všechny bity 1, ale kromě prvního který musí být 0, aby číslo bylo kladné. Největší číslo tedy bude $x_{\max} = 2^{n-1} - 1$. Nejmenší záporné číslo má první bit 1 a ostatní bity 0, jeho velikost je $x_{\min} = -2^{n-1}$. Rozsah celých čísel se znaménkem v závislosti na počtu bitů n je

n	$x_{\min}$	$x_{\max}$
8	-128	-127
16	−32 768	32 767
32	−2 147 483 648	2 147 483 647

Při osmi bitových číslech dostáváme například následující reprezentace čísel

127	0111 1111
3	0000 0011
2	0000 0010
1	0000 0001
0	0000 0000
-1	1111 1111
-2	1111 1110
-3	1111 1101
-128	1000 0000

1.1.3 Reálná čísla

Reálné číslo x v počítači se skládá ze tří částí - znaménka $s = \pm 1$, mantisy m a exponenciální části b^e (b je pevně daný základ, nejčastěji $b = 2$).

$$x = s.m.b^e \tag{1.3}$$

Tento rozklad není jednoznačný, proto používáme normalizaci $1 \leq m < b$. Tři čísla s, m, e je potřeba uložit do daného počtu bitů. Nejčastěji se používá 32-bitové číslo (tzv. real, float, single) a 64-bitové číslo (tzv. double). U počítačů s procesory Intel se používá i 80-bitové číslo (tzv. long double, extended). Kompilátor Borland Pascal standardně používá 48-bitové číslo (tzv. real), tento formát však vybočuje z řady a nevyhovuje formátu popsanému dále.
Znaménko s se ukládá do jednoho bitu: 0 – kladné číslo, 1 – záporné číslo. Díky normalizaci má mantisa vždy tvar $m = 1.xxxxx\ldots$, úvodní jedničku je tedy zbytečné ukládat, ukládáme tedy pouze číslice za desetinnou čárkou. Exponent e může nabývat kladných i záporných hodnot. Přičtením předem dané hodnoty E převedeme exponent na kladné číslo, které uložíme do pevného počtu bitů. Počty bitů jednotlivých částí, konstanta E jsou uvedeny v následující tabulce, dále je zde uvedeno největší a nejmenší kladné číslo a strojové ε (viz. dále).

	single	double	extended
bitů	32	64	80
bitů e	8	11	15
bitů m	23	52	64
E	127	1023	16383
e	$-126\ldots127$	$-1022\ldots1023$	$-16382\ldots16383$
$e+E$	$1\ldots254$	$1\ldots2046$	$1\ldots32766$
x_{min}	1.2×10^{-38}	2.2×10^{-308}	3.4×10^{-4932}
x_{max}	3.4×10^{38}	1.8×10^{308}	1.2×10^{4932}
x_0	1.4×10^{-45}	4.9×10^{-324}	3.6×10^{-4954}
ε	1.2×10^{-7}	2.2×10^{-16}	5.4×10^{-20}

Číslo nula nelze ve tvaru (1.3) uložit. Krajní hodnoty exponentu mají proto speciální význam. To také umožňuje reprezentovat určité speciální hodnoty – zápornou nulu, kladné a záporné nekonečno, neexistující číslo (NaN - not a number), čísla se sníženou přesností viz následující tabulka

e+E	m	x
$1\ldots254$		$(-1)^s\ 2^e\ 1.m$
0	$m\neq0$	$(-1)^s\ 2^e\ 0.m$
	$m=0$	$(-1)^s\ 0$
255	$m=0$	$(-1)^s\ \infty$
	$m\neq0$	NaN

Příklady konkrétních čísel v 32-bitovém formátu jsou v následující tabulce

x	s	$e+E$	m
0	0	00000000	0000000 00000000 00000000
-0	1	00000000	0000000 00000000 00000000
1	0	01111111	0000000 00000000 00000000
2	0	10000000	0000000 00000000 00000000
3	0	10000000	1000000 00000000 00000000
$1+\varepsilon$	0	01111111	0000000 00000000 00000001
3.4×10^{38}	0	11111110	1111111 11111111 11111111
1.2×10^{-38}	0	00000001	0000000 00000000 00000000
1.4×10^{-45}	0	00000000	0000000 00000000 00000001
NAN	0	11111111	1101111 11111111 00000001
∞	0	11111111	0000000 00000000 00000000
$-\infty$	1	11111111	0000000 00000000 00000000

Pro charakteristiku formátu reálného čísla důležitý pojem *strojové epsilon* ε. Toto číslo je definováno jako nejmenší číslo, které přičtené k jedničce dá výsledek různý od jedné. Hodnoty ε pro různé formáty můžeme nalézt v předchozí tabulce. Strojové epsilon vystihuje přesnost reálného čísla, např. $\varepsilon = 1.2\ 10^{-7}$ znamená, že přesnost tohoto formátu je 7 desetinných míst.
Je důležité si uvědomit, že reálná čísla v počítači mají pouze omezenou přesnost, i čísla, která lze v desítkové soustavě zapsat přesně mají ve dvojkové soustavě nekonečný desetinný rozvoj. Takto například vypadá dopadne reprezetace čísel v 4 bytových reálných číslech.

$$1/10 \to 0.10000000149011611938\ldots$$
$$1/3 \to 0.33333334326744079590\ldots$$

1.2 Počítačová aritmetika

Počítačová aritmetika se v řadě věcí odlišuje od aritmetiky kterou známe z matematiky. Díky zaokrouhlovacím chybám neplatí například následující vztahy

$$x.(1/x) = 1 \qquad (1+x)-1 = x \qquad (x+y)+z = x+(y+z) \tag{1.4}$$

Řada vztahů však zůstává v platnosti, například

$$1.x = x \qquad x.y = y.x \qquad x+x = 2.x \tag{1.5}$$

Představme si, že máme počítačovou aritmetiku s pouhými čtyřmi platnými číslicemi. Mějme dvě čísla $a = 1001$, $b = 1003$. Možná chyba těchto čísel je v této aritmetice rovna 1. Je tedy správně $a = 1001 \pm 1$, $b = 1003 \pm 1$. Relativní chyba těchto čísel je přibližně 0.1%. Pokud čísla sečteme dostaneme $c = a+b = 2004 \pm 2$, relativní chyba je opět zhruba 0.1%. Pokud čísla vynásobíme přibližně dostaneme $c = a.b = 1004000 \pm 2000$, relativní chyba 0.2% je opět přijatelná. Problém nastane pokud se čísla pokusíme odečíst $c = b-a = 2 \pm 2$, relativní chyba je 100%, výsledek je nepoužitelný.
V reálném případě nebývá výsledek tak špatný, ale je třeba v numerické matematice této situaci věnovat zvýšenou pozornost. **Nesmíme odečítat čísla která mají přibližně stejnou absolutní hodnotu**. Pokud tak učiníme bude mít výsledek sníženou přesnost.
V následující tabulce jsou rozdíly čísel, které se shodují v různém počtu číslic. Výpočty bylo provedeny v jednoduché přesnosti.

a	b	$a-b$
0.1	0.0	0.100000001
1.1	1.0	0.100000024
11.1	11.0	0.100000381
111.1	111.0	0.099998474
1111.1	1111.0	0.099975586
11111.1	11111.0	0.099609375
111111.1	111111.0	0.101562500
1111111.1	1111111.0	0.125000000
11111111.1	11111111.0	0.000000000

Z tabulky je vidět, že pokud roste počet shodných číslic, přesnost výsledku klesá. Pokud relativní rozdíl klesne pod strojové epsilon, je výsledný rozdíl již nula.

1.2.1 Srovnávání reálných čísel

Při tvorbě numerického softwaru je třeba mít neustále na paměti, že reálná čísla jsou pouze přibližná, vždy jsou do určité míry zatíženy zaokrouhlovací chybou. Velice často se chybuje při srovnávání reálných čísel. V programu nikdy nesmíme použít test rovnosti reálných čísel, jako například

```
real a,b
if (a==b) ...
```

Pokud totiž čísla a, b vnikla jako výsledek výpočtu, nebudou se kvůli zaokrouhlovacím chybám rovnat, i když při použití přesné aritmetiky by byla stejná. Místo testu rovnosti je třeba testovat zda rozdíl těchto čísel je dostatečně malý, tj. předchozí test má správně být

```
if (abs(a-b)<E) ...
```

Konstanta E určuje co považuje za možnou hranici zaokrouhlovací chyby. Tato chyba je při jedné matematické operaci úměrná hodnotě a a strojovému ε. S rostoucím počtem operací tato chyba poroste, lze tedy např. použít

```
if (abs(a-b)<10*eps*abs(a)) ...
```

kde `eps` je strojové ε.
Je tedy vidět, že testování rovnosti reálných čísel je obtížné, a pokud je to možné měli bychom se mu vyhýbat.
Chyba vzniklá srovnáním reálných čísel se často vyskytuje při procházení intervalu. Mějme zadán interval $\langle a, b\rangle$. Tento interval chceme rozdělit na n intervalů a provést výpočet se všemi hraničními body. Jedním z potenciálních způsobů výpočtu je nejprve vypočítat velikost dílčího intervalu h a k proměnné x postupně tuto délku přičítat, dokud nedosáhneme konce intervalu

```
h = (b-a)/n
x = a
do {
    x = x + h
    vypocet(x)
    if (x==b) exit
}
```

Takovýto výpočet je však zcela špatně, protože přičítáním h do x se bude postupně akumulovat zaokrouhlovací chyba. Díky tomu rovnost v testu nikdy nenastane a cyklus nikdy neskončí. Pokud místo předchozí podmínky použijeme

```
if (x>b) exit
```

cyklus sice vždy skončí, ale někdy bude započítán jeden bod navíc. Řešením je podmínka

```
if (abs(x-b)<h/2) exit
```

Ještě lepší řešení je ale použití cyklu `for`

```
h = (b-a)/n
for (i=1,n) {
    x = x + i*h
    vypocet(x)
}
```

Příklad

U některých matematických úloh mohou i malé zaokrouhlovací chyby vést ke zcela chybným výsledkům. Takovýmto úlohám se říká *špatně podmíněné*. Příkladem je tzv. Wilkinsonův polynom. Tento polynom je definován takto

$$p(x) = (x+1)(x+2)\cdots(x+20) \tag{1.6}$$

Je zřejmé, že tento polynom má kořeny $x = -1, -2, \ldots, 20$. Tento polynom vyjádříme ve standardním tvaru dostaneme

$$p(x) = x^{20} + 210x^{19} + \cdots + 20! \tag{1.7}$$

Pokud v tomto tvaru pozměníme některý z koeficientů, kořeny se zcela změní. Například při změně jediného koeficientu $a_{19} = 210 - 2^{-23}$ dostaneme zcela jiné kořeny

$$x = -16.713 \pm 2.813i, -13.992 \pm 2.519i$$

1.2.2 Hornerovo schéma

I pokud odhlédneme od zaokrouhlovacích chyb nelze matematické vzorce bezmyšlenkovitě přepisovat do programů. Zápis používaný v matematice totiž často není nejrychlejší možný způsob výpočtu.
Příkladem je výpočet hodnoty polynomu

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \tag{1.8}$$

Pokud bychom postupovali přímo podle tohoto vzorce je třeba nejprve vypočítat mocniny, ty vynásobit koeficienty a sečíst. Rychlejší způsob výpočtu je tzv. *Hornerovo schéma*. Při tomto způsobu výpočtu nejprve otočíme pořadí členů polynomu

$$p(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0 \tag{1.9}$$

potom vytkneme z prvních členů proměnnou x

$$p(x) = (a_nx^{n-1} + a_{n-1}x^{n-2} + \cdots + a_2x + a_1)x + a_0 \tag{1.10}$$

Z prvních členů uvnitř závorky opět vytkneme x, a toto celkem n-krát opakujeme, nakonec dostaneme výsledné Hornerovo schéma

$$p(x) = ((\cdots((a_nx + a_{n-1})x + a_{n-2})x + \cdots + a_2)x + a_1)x + a_0 \tag{1.11}$$

Tento způsob výpočtu je rychlejší než (1.8), protože už není třeba počítat mocniny.

1.2.3 Výpočet mocniny

Jiným příkladem, kdy je vhodné matematický vzorec upravit, je výpočet mocniny. Pokud bychom počítali 11. mocninu podle matematické definice

$$x^{11} = x.x.x.x.x.x.x.x.x.x.x$$

potřebujeme 10 násobení. Při použití následujícího postupu

$$\begin{aligned} a &= x.x \\ b &= a.a \\ c &= b.b \\ x^{11} &= x.a.c \end{aligned}$$

nám stačí 5 násobení.

1.3 Cvičení

1. Napište program, který zjistí strojové epsilon.
2. Sečtěte řadu $\sum_{n=1}^{\infty} \frac{1}{n^2}$, správný součet je $\frac{\pi^2}{6}$. Jak přesný výsledek se vám podaří obdržet při jednoduché přesnosti reálných čísel? Co se stane pokud se pokusíte sečíst řadu $\sum_{n=1}^{\infty} \frac{1}{n}$?

Kapitola 2

Lineární algebra

Lineární algebra

V lineární algebře se vyskytuje řada úloh, které je třeba řešit pomocí numerické matematiky, mezi hlavní patří:

- řešení soustavy lineárních rovnic
- nalezení inverzní matice
- výpočet determinantu
- výpočet vlastních čísel a vlastních vektorů matice

V tomto kurzu se budeme zabývat pouze řešením soustav lineárních rovnic.

2.1 Řešení soustav lineárních rovnic

Úkolem této kapitoly je řešit soustavu n lineárních rovnic o n neznámých

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \tag{2.1}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \tag{2.2}$$
$$\cdots \tag{2.3}$$
$$a_{n1}x_1 + a_{n2}x_n + \cdots + a_{nn}x_n = a_{n,n+1} \tag{2.4}$$

Soustavu můžeme zapsat i ve vektorovém tvaru

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.5}$$

Při řešení této úlohy existuje celá řada přístupů

- přímé metody
- iterační metody
- speciální metody
- kombinované metody
- metody Monte Carlo

My se budeme zabývat prvními dvěma - přímými a iteračními metodami.
Přímé metody používají pro řešení soustavy algoritmus, který skončí po známém počtu kroků. Neuvažujeme-li zaokrouhlovací chybám vedou tyto metody k přesnému řešení.
Iterační metody postupně upřesňují odhady řešení. Tyto odhady konvergují k přesnému řešení. Po dosažení potřebné přesnosti iterační proces ukončíme.

2.1.1 Přímé metody

Gaussova eliminační metoda

Nejzákladnější metodou na řešení soustav lineárních rovnic je Gaussova eliminační metoda. Princip této metody spočívá v postupném vylučování proměnných z některých rovnic. Při tom využíváme toho, že můžeme k vybrané rovnici přičíst libovolnou lineární kombinaci ostatních rovnic.
Algoritmus má dvě části. Cílem první části tzv. *přímého chodu* je převést matici soustavy na horní trojúhelníkový tvar. V prvním kroku přičteme ke druhé až n-té rovnici vhodné násobky první rovnice, tak abychom z těchto rovnic vyloučili proměnnou x_1. Podobně ve druhém kroku přičteme ke třetí až n-té rovnici vhodné násobky druhé rovnice, tak abychom z těchto rovnic vyloučili proměnnou x_2. Takto postupně převedeme matici soustavy na horní trojúhelníkový tvar. Všechny operace je třeba provádět i s pravou stranou rovnice, to lze nejjednodušeji realizovat tak, že pravou stranu rovnice budeme považovat za $n+1$-ní sloupec matice $\boldsymbol{A}$. Symbolicky lze tuto část algoritmu zapsat

$$a_{ij}^{(0)} = a_{ij} \qquad i = 1, \ldots, n \qquad j = 1, \ldots, n+1 \tag{2.6}$$

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)} \tag{2.7}$$

$$k = 1, \ldots, n-1 \qquad i = k+1, \ldots, n \qquad j = k+1, \ldots, n+1 \tag{2.8}$$

Příklad

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 4 & 5 & 6 & 32 \\ 9 & 6 & 5 & 36 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 0 & -3 & -6 & -24 \\ 0 & -12 & -22 & -90 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 14 \\ 0 & -3 & -6 & -24 \\ 0 & 0 & 2 & 6 \end{array}\right] \tag{2.9}$$

Ve druhé části tzv. *zpětném chodu* postupně vypočítáváme hodnoty řešení, a to v obráceném pořadí t.j. počínaje x_n a konče x_1.

$$x_i = \frac{1}{a_{ii}^{(i-1)}} \left(a_{i,n+1}^{(i-1)} - \sum_{j=i+1}^{n} a_{kj}^{(i-1)} x_j \right) \qquad i = n, \ldots, 1 \tag{2.10}$$

V předchozím příkladě tak z nejprve z poslední rovnice dostaneme $x_3 = 3$, ze druhé $x_2 = 2$ a nakonec z první $x_1 = 1$.
Algoritmus selhává pokud na diagonále v řádku pomocí kterého chceme eliminovat proměnné je nula. V tom případě by došlo ke vzorci (2.7) k dělení nulou. Proto je potřeba provádět tzv. *pivotizaci*. Ta spočívá v tom, že zaměníme k-tý řádek za ten z řádků číslo k až n, který má v k-tém sloupci největší absolutní hodnotu. V prostřední matici (2.9) bychom například přehodili druhý a třetí řádek protože $|-12| > |-3|$. Tato metoda se nazývá částečná pivotizace. Při tzv. úplné pivotizaci přehazujeme nejen řádky, ale i sloupce, tak abychom na pozici a_{kk} dostali prvek s největší absolutní hodnotou. Ve zmiňovaném příkladu bychom přehodili nejen druhý a třetí řádek, ale i druhý a třetí sloupec, protože číslo -22 je absolutně největší z čísel (-3, -6, -12, -22). Při úplné pivotizaci je třeba dát pozor na to, že změnou pořadí sloupců dojde zároveň k změně pořadí výsledků.

Gaussova-Jordanova metoda

Při této modifikaci Gaussovy metody převádíme matici místo na trojúhelníkový na diagonální tvar. Zpětný chod je potom jednodušší. Tento přístup je však přibližně dvakrát pomalejší než původní Gaussova metoda.

Gaussova eliminace pro třídiagonální matici

Velice často se vyskytuje soustava lineárních rovnic s třídiagonální maticí (např. při interpolaci kubickými spliny, při implicitních metodách pro řešení parciálních diferenciálních rovnic). Soustava rovnic má v tomto případě tvar

$$\begin{bmatrix} a_1 & c_1 & & & & & \\ b_2 & a_2 & c_2 & & & & \\ & b_3 & a_3 & & & & \\ & & & \ddots & & & \\ & & & & a_{n-2} & c_{n-2} & \\ & & & & b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & & b_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1} \\ d_n \end{bmatrix} \tag{2.11}$$

V přímém chodu tuto soustavu převede na trojúhelníkový tvar

$$\begin{bmatrix} u_1 & c_1 & & & & & \\ 0 & u_2 & c_2 & & & & \\ & 0 & u_3 & & & & \\ & & & \ddots & & & \\ & & & & u_{n-2} & c_{n-2} & \\ & & & & 0 & u_{n-1} & c_{n-1} \\ & & & & & 0 & u_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ y_n \end{bmatrix} \tag{2.12}$$

Přímý chod provádíme pomocí vztahů

$$u_1 = a_1 \qquad y_1 = d_1 \tag{2.13}$$

$$u_i = a_i - \frac{b_i c_{i-1}}{u_{i-1}} \tag{2.14}$$

$$y_i = d_i - \frac{b_i y_{i-1}}{u_{i-1}} \qquad i = 2, \ldots, n \tag{2.15}$$

ve zpětném chodu potom vypočteme řešení

$$x_n = \frac{y_n}{u_n} \tag{2.16}$$

$$x_i = \frac{1}{u_i}(y_i - c_i x_{i+1}) \qquad i = n-1, \ldots, 1 \tag{2.17}$$

2.1.2 Iterační metody

Při iteračních metodách pro řešení $\boldsymbol{Ax} = \boldsymbol{b}$ budeme konstruovat posloupnost přibližných řešení x^i obecně nějakým vztahem

$$x^{i+1} = F_i(x^i, x^{i-1}, \ldots, x^{i-k}), \tag{2.18}$$

tak aby tato přibližná řešení konvergovala k řešení přesnému, t.j.

$$\lim_{i \to \infty} x^i = x \tag{2.19}$$

Omezíme se na tzv. jednobodové lineární stacionární metody, při kterých má rovnice (2.18) tento speciální tvar

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{B}\boldsymbol{x}^{(i)} + \boldsymbol{C}\boldsymbol{b} \tag{2.20}$$

Je třeba určit kdy iterační proces ukončíme. První možné kritérium je počkat, až se přibližná řešení příliš nemění tj. až je $||x^{i+1} - x^i||$ dostatečně malé. Druhou možností

je počkat až je řešená rovnice dostatečně přesně splněna, tj. až je $||Ax-b||$ dostatečně malé. Tyto metody lze vzájemně kombinovat.
Iterační metody bohužel nekonvergují vždy. Aby metody konvergovaly, je třeba, aby matice $\boldsymbol{A}$ splňovala určité podmínky. V metodách, které dále uvedeme stačí aby matice $\boldsymbol{A}$ byla symetrická a pozitivně definitní, nebo aby platila jedna z podmínek

$$\sum_{\substack{j=1\\ i\neq j}}^{n} |B_{ij}| < |B_i| \qquad i = 1, \ldots, n \tag{2.21}$$

$$\sum_{\substack{i=1\\ i\neq j}}^{n} |B_{ij}| < |B_j| \qquad j = 1, \ldots, n \tag{2.22}$$

(tj. součet prvků v libovolném řádku musí být menší než prvek na diagonále)

Jacobiova metoda

Základní iterační metodou je Jacobiova metoda. Matici soustavy $\boldsymbol{A}$ rozložíme na součet tří matic - diagonální matici $\boldsymbol{D}$, dolní trojúhelníkovou matici s nulami na diagonále $\boldsymbol{L}$ a horní trojúhelníkovou matici s nulami na diagonále $\boldsymbol{U}$.

$$\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U} \tag{2.23}$$

Soustavu rovnic poté upravíme.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.24}$$
$$(\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} = \boldsymbol{b} \tag{2.25}$$
$$\boldsymbol{D}\boldsymbol{x} + (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} = \boldsymbol{b} \tag{2.26}$$
$$\boldsymbol{D}\boldsymbol{x} = \boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} \tag{2.27}$$
$$\boldsymbol{x} = \boldsymbol{D}^{-1}[\boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}] \tag{2.28}$$

Poslední rovnici použijeme jako iterační proces.

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{D}^{-1}[\boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(i)}] \tag{2.29}$$

Tato rovnice ve složkovém zápisu je

$$x_j^{(i+1)} = \frac{1}{a_{jj}}\Big(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\Big) \qquad j = 1, \ldots, n \tag{2.30}$$

Gaussova-Seidelova metoda

V této variantě Gaussovy metody použijeme stejný rozklad matice $\boldsymbol{A}$, ale trochu jiné úpravy.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.31}$$
$$(\boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} = \boldsymbol{b} \tag{2.32}$$
$$(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x} + \boldsymbol{U}\boldsymbol{x} = \boldsymbol{b} \tag{2.33}$$
$$(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{U}\boldsymbol{x} \tag{2.34}$$
$$\boldsymbol{x} = (\boldsymbol{D} + \boldsymbol{L})^{-1}[\boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}] \tag{2.35}$$
$$\boldsymbol{x}^{(i+1)} = (\boldsymbol{D} + \boldsymbol{L})^{-1}[\boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}^{(i)}] \tag{2.36}$$

Ve složkovém zápisu:

$$x_j^{(i+1)} = \frac{1}{a_{jj}}\Big(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\Big) \qquad j = 1, \ldots, n \tag{2.37}$$

Příklad

Řešme soustavu

$$\begin{aligned} 6x_1+2x_2-3x_3 &= 10 \\ x_1+\ 4x_2-2x_3 &= 6 \\ 3x_1+2x_2-7x_3 &= -4 \end{aligned} \tag{2.38}$$

Jacobiho metoda v tomto případě vypadá takto

$$\begin{aligned} x_1^{(i+1)} &= (10 - 2x_2^{(i)} - 3x_3^{(i)})/6 \\ x_2^{(i+1)} &= (6 - x_1^{(i)} + 2x_3^{(i)})/4 \\ x_3^{(i+1)} &= (4 + 3x_1^{(i)} + 2x_2^{(i)})/7 \end{aligned} \tag{2.39}$$

Gaussova-Seidlova metoda vypadá takto

$$\begin{aligned} x_1^{(i+1)} &= (10 - 2x_2^{(i)} - 3x_3^{(i)})/6 \\ x_2^{(i+1)} &= (6 - x_1^{(i+1)} + 2x_3^{(i)})/4 \\ x_3^{(i+1)} &= (4 + 3x_1^{(i+1)} + 2x_2^{(i+1)})/7 \end{aligned} \tag{2.40}$$

Výsledky obou metod jsou v tabulce 2.1.2. Vidíme, že Gaussova-Seidlova metoda potřebuje 8 kroků ve srovnání s 26 kroky Jacobiho metody.

k	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$	$x_1^{(i)}$	$x_2^{(i)}$	$x_3^{(i)}$
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	1.6667	1.5000	0.5714	1.6667	1.0833	1.5952
2	1.4524	1.3690	1.7143	2.1032	1.7718	1.9790
3	2.0675	1.9940	1.5850	2.0656	1.9731	2.0204
4	1.7945	1.7757	2.0272	2.0192	2.0054	2.0098
5	2.0884	2.0650	1.8478	2.0031	2.0041	2.0025
6	1.9023	1.9018	2.0564	1.9999	2.0013	2.0003
7	2.0610	2.0527	1.9301	1.9997	2.0002	1.9999
8	1.9475	1.9498	2.0412	1.9999	2.0000	2.0000
9	2.0373	2.0337	1.9631			
10	1.9703	1.9722	2.0256			
11	2.0221	2.0202	1.9794			
	...	...	...			
25	2.0005	2.0005	1.9995			
26	1.9996	1.9996	2.0003			

Tabulka 2.1: Srovnání iteračních metod. Jacobiho metoda vlevo, Gaussova-Seidlova metoda vpravo

Superrelaxační metoda

Pro urychlení iteračních procesů se někdy používá idea tzv. superrelaxace. Mějme iterační proces (např. Jacobiovu nebo Gaussovu-Seidelovu metoda)

$$x^{i+1} = F(x^i) \tag{2.41}$$

který upravíme do tvaru předchozí iterace plus oprava

$$x^{i+1} = x^i + F(x^i) - x^i \tag{2.42}$$

$$= x^i + \Delta x_i \quad \text{kde} \quad \Delta x_i \equiv (F(x^i) - x^i) \tag{2.43}$$

Místo opravy Δx přičteme opravu větší $\omega\Delta x$, kde ω je reálný parametr $\omega > 1$.

$$x^{i+1} = x^i + \omega\Delta x_i \tag{2.44}$$

$$= (1-\omega)x^i + \omega F(x^i) \tag{2.45}$$

Při vhodné volbě parametru ω tento postup vede ke zrychlení konvergence. Někdy je vhodný i parametr $\omega < 1$, tato volba sice zpomalí konvergenci, ale může vést k větší stabilitě metody.
Pokud jako iterační proces použijeme Gaussovu-Seidelovu metodu dostaneme následující superrelaxační metodu

$$x_j^{(i+1)} = \frac{\omega}{a_{jj}}\Big(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\Big) + (1-\omega)x_j^{(i)} \qquad j = 1,\ldots,n \tag{2.46}$$

2.2 Cvičení

1. Naprogramujte Gaussovu eliminaci s částečnou pivotizací. Výsledek ověřte zkouškou.

2. Srovnejte rychlost Gaussovu metody a Gaussova-Jordanova metody.

3. Naprogramujte Jacobiho a Gaussovu-Seidelovu metodu a srovnejte jejich rychlost.

Kapitola 3

Aproximace a interpolace

Jedním ze základních úkolů numerické matematiky je nalézt k zadané funkci $f(x)$ vhodnou aproximující funkci $g(x)$. K hledání aproximační funkce může být řada důvodů, například

- Původní funkce je na výpočet příliš složitá. Tak se například nahrazují funkce $\sin x$, $\exp x$ výpočtem polynomu. (Pokud je chyba aproximace menší než přesnost reálných čísel nedopustíme se tím dokonce žádné nepřesnosti.)
- Původní funkce není známe ve všech bodech, je zadána například tabulkou. Známe tedy hodnoty funkce v bodech $x_0, x_1, \ldots, x_n$ a potřebujeme zjistit hodnotu v jiném bodě x.
- Známé hodnoty funkce $f(x)$ známe pouze přibližně nebo mohou být navíc zatíženy chybou. Tato situace nastává nejčastěji při zpracování dat z fyzikálního experimentu nebo počítačové simulace.

Při hledání aproximační funkce se omezujeme na určitou skupinu funkcí. Nejčastěji hledáme funkci $g(x)$ ve tvaru lineární kombinace předem daných funkcí tj.

$$g(x) = a_0 g_0(x) + a_1 g_1(x) + \cdots + a_n g_n(x) \tag{3.1}$$

Nejčastější volbou základních funkcí jsou mocniny. Dostaneme tak polynom

$$g(x) = a_0 + a_1 x + \cdots + a_n x^n \tag{3.2}$$

Jinou možností jsou například goniometrické funkce

$$g(x) = a_0 + a_1 \sin x + b_1 \cos x + a_2 \sin 2x + b_2 \cos 2x + \ldots \tag{3.3}$$

V některých případech je vhodnější použít racionální funkci

$$g(x) = \frac{a_0 + a_1 x + \cdots + a_n x^n}{b_0 + b_1 x + \cdots + b_m x^m} \tag{3.4}$$

Z vybrané množiny funkcí je třeba vybrat jednu, která bude naši funkci $f(x)$ nejlépe aproximovat. Abychom to mohli udělat, je třeba vybrat kritérium pomocí které rozhodneme, která z možných funkcí je lepší. Možných kriterií existuje celá řada. Předpokládejme nejprve, že máme funkci $f(x)$ zadánu tabulkou, tj. v bodech x_0, x_1, $\ldots$, x_n máme zadány hodnoty $y_0, y_1, \ldots, y_n$. Funkce $g(x)$ nemusí těmito body přímo procházet, v k-tém bodě potom bude odchylka $\epsilon_i \equiv g(x_k) - y_k$. Jednotlivá kritéria porovnávají kvalitu aproximace na základě celkové chyby aproximace E, která se získá z těchto jednotlivých odchylek. Nejpoužívanější kritéria jsou následující

- $E = \sum \epsilon_i^2$ Celková chyba je součet druhých mocnin všech odchylek. Funkce pro kterou je tato chyba nejmenší se nazývá *aproximace metodou nejmenších čtverců*.
- $E = \max |\epsilon_i|$ Pokud nepoužije součet odchylek, ale nalezneme největší odchylku dostaneme *Čebyševovu aproximaci*.
- $\epsilon_i = 0$ Požadujeme, aby hledaná funkce přesně procházela zadanými body. Všechny odchylky jsou tedy nulové. Tato metoda se nazývá *interpolace*.

Pokud funkci $f(x)$ nemáme zadánu tabulkou, ale známe celý její průběh je třeba předchozí kritéria modifikovat. Při aproximaci se omezujeme na daný interval $\langle a, b\rangle$. Odchylku ϵ definujeme na celém intervalu $\epsilon(x) \equiv g(x) - f(x)$. Kritéria pro výběr aproximační funkce upravíme následovně

- pro aproximaci metodou nejmenších použijeme $E = \int_a^b \epsilon(x)^2 \,\mathrm{d}x$
- pro Čebyševovu aproximaci použijeme $E = \max_{x\in\langle a,b\rangle} |\epsilon(x)|$

Uvedená kritéria mají řadu dalších variant. Lze například přidávat různé váhové faktory nebo místo absolutních odchylek vzít odchylky relativní.

3.1 Interpolace

Cílem interpolace je nalezení funkce $g(x)$, která se shoduje s funkcí $f(x)$ v uzlových bodech x_i

$$g(x_i) = f(x_i) = f_i = y_i \qquad i = 0, \ldots, n \tag{3.5}$$

Tato interpolace se někdy nazývá interpolace Lagrangeova. Při tzv. Hermitově aproximaci požadujeme navíc shodu derivací v uzlových bodech tj. i $g'(x_i) = f'(x_i)$.

3.1.1 Lineární interpolace

Nejjednodušší interpolace je počástech lineární interpolace. Mezi uzlovými body aproximujeme funkci $f(x)$ úsečkami viz obr. 3.1. Pokud chceme vypočítat hodnotu interpolační funkce v bodě x, je třeba nejprve zjistit, mezi které body x_i tento bod patří, tj. potřebujeme nalézt i tak aby $x \in \langle x_i, x_{i+1}\rangle$. Pokud jsou body rozmístěny rovnoměrně ($x_{i+1} - x_i = h$) lze zjistit i jednoduchým vzorcem $\lfloor i = (x - x_0)/h \rfloor$, kde $\lfloor x \rfloor$ znamená celou část z čísla x. Pokud body nejsou rozmístěny rovnoměrně nebo dokonce nejsou seřazeny je potřeba použít jiný vyhledávací algoritmus. Když známe body x_i a x_{i+1} vypočteme aproximovanou hodnotu z následujícího vztahu

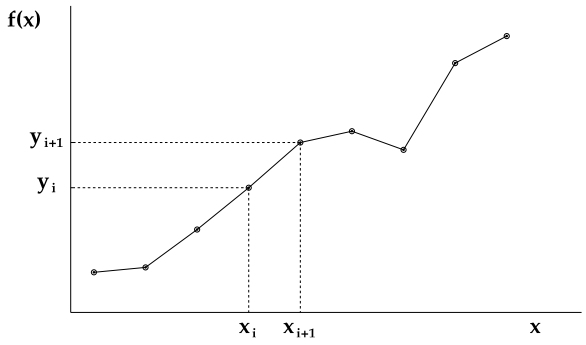
$$g(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \tag{3.6}$$

Tato metoda je velice jednoduchá. Její nevýhodou je, že aproximační funkce není hladká, tj. nemá spojitou derivaci. V řadě zejména fyzikálních aplikací však dostačuje.

3.1.2 Interpolace polynomem

Velice často používanou interpolační funkcí je polynom.

$$g(x) = p_m(x) = a_0 + a_1 x + \cdots + a_n x^n \tag{3.7}$$



Obrázek 3.1: Počástech lineární interpolace.

Požadujeme aby tento polynom procházel $n+1$ zadanými body. Máme tedy $n+1$ podmínek. Pokud použijeme polynom stupně n, máme $n+1$ neznámých koeficientů a_i. Vzniklá soustava rovnic má právě jedno řešení a polynom je tedy určen jednoznačně.
Přímým dosazením polynomu do podmínek (3.5) dostaneme soustavu lineárních rovnic pro koeficienty polynomu

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= f(x_0) \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= f(x_1) \\ \cdots & \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= f(x_n) \end{aligned} \tag{3.8}$$

Matice této soustavy se nazývá Vandermontova matice. Je tedy třeba vyřešit následující soustavu

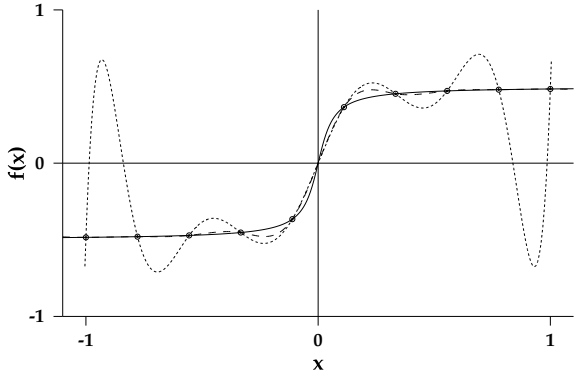
$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & & x_0^n \\ \vdots & \vdots & & \ddots & \\ 1 & x_n & x_n^2 & & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \tag{3.9}$$

Řešením získáme koeficienty a_i a z (3.7) můžeme vypočítat hodnoty aproximační funkce $g(x)$ v libovolném bodě x. Příklad polynomiální aproximace 11 body je na obrázku 3.2.
Je vidět, že interpolační polynom sice prochází požadovanými body, ale mezi nimi silně osciluje. Toto je bohužel typické chování polynomiální interpolace pro polynomy vyšších stupňů.

Příklad

Najděme interpolační polynom procházející čtyřmi body [1,-10], [2,0], [3,10] a [4,-10] viz obr. ???. Soustava lineárních rovnic (3.9) tedy je

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 0 \\ 10 \\ -10 \end{bmatrix} \tag{3.10}$$

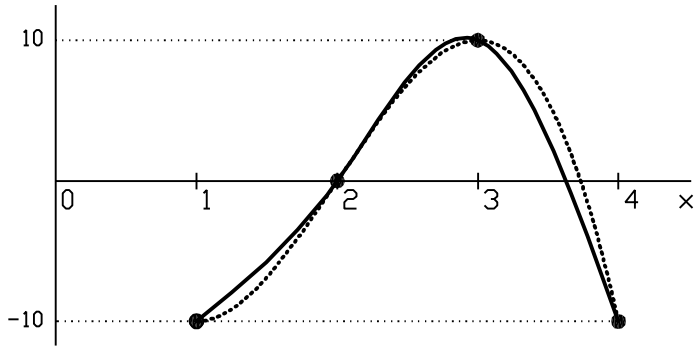


Obrázek 3.2: Interpolace polynomem (čárkovaně) a kubickým splinem (tečkovaně).

Řešením této soustavy do staneme $a_i = (10, -45, 30, -5)$, výsledný interpolační polynom tedy je

$$g(x) = 10 - 45x + 30x^2 - 5x^3 \tag{3.11}$$

Graf této interpolační funkce je na obrázku 3.3



Obrázek 3.3: Interpolace polynomem (tečkovaně) a kubickým splinem (plně).

3.1.3 Lagrangeova metoda

Výpočet polynomu pomocí jeho koeficientů může být zatížen velkými zaokrouhlovacími chybami (viz př. (xxx)). Jednou z možností jak se vyhnout výpočtu koeficientů polynomu je tzv. Lagrangeova metoda. Interpolační polynom budeme hledat ve tvaru

$$L_n(x) = \sum_{i=0}^{n} f(x_i) g_i(x) \tag{3.12}$$

kde g_i jsou polynomy splňující vztah

$$g_i(x_j) = \delta_{ij} \tag{3.13}$$

Tato podmínka zaručuje splnění podmínek (3.5). Polynomy g_i mají tvar

$$g_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \tag{3.14}$$

Pokud $j \neq i$ je čitatel v bodě x_j nulový. V bodě x_i je čitatel roven jmenovateli. Podmínka (3.13) je tedy zřejmě splněna. Výsledný interpolační polynom tedy je

$$L_n(x) = \sum_{i=0}^{n} f(x_i) \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} \tag{3.15}$$

Tento vztah lze celkem snadno naprogramovat. Oproti metodě s přímým výpočtem koeficientů nepotřebujeme řešit soustavu lineárních rovnic, výpočet jedné funkční hodnoty je ale pomalejší. Vzhledem k tomu, že interpolační polynom je určen jednoznačně měla by obě metody dávat stejné výsledky, u Lagrangeovy metody se však méně projevují zaokrouhlovací chyby.

Příklad

Stejně jako v předchozím příkladu hledáme interpolační polynom procházející čtyřmi body [1,-10], [2,0], [3,10] a [4,-10]. Lagrangeův polynom (3.15) s těmito hodnotami je

$$\begin{aligned} L_3(x) = &-10\frac{(x-2)(x-3)(x-4)}{-6} + 0\frac{(x-1)(x-3)(x-4)}{2} + \\ &+10\frac{(x-1)(x-2)(x-4)}{-2} - 10\frac{(x-1)(x-2)(x-3)}{6} \end{aligned} \tag{3.16}$$

roznásobením a sečtením tohoto vzorce bychom dostali tento polynom ve tvaru (3.11).

3.1.4 Newtonova metoda

Newtonova metoda opět vypočítává interpolační polynom. Tato metoda na rozdíl od předchozích nevytváří celý polynom najednou, ale postupně zahrnuje více bodů do interpolace a konstruuje tak polynomy vyššího stupně.
Nejprve je třeba zavést pomocný pojem *poměrné diference*. Poměrná diference prvního řádu je definována vztahem

$$f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.17}$$

Je to vlastě odhad první derivace pomocí dvou funkčních hodnot. Podobně je definována druhá poměrná diference a obecně n-tá diference.

$$f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} \tag{3.18}$$

$$f[x_n, x_{n-1}, \ldots, x_0] = \frac{f[x_n, x_{n-1}, \ldots, x_1] - f[x_{n-1}, \ldots, x_1, x_0]}{x_n - x_0} \tag{3.19}$$

Newtonova metoda začíná s konstantní interpolací jedním bodem

$$N_0(x) = f(x_0) \tag{3.20}$$

Přidáním dalšího členu získáme lineární interpolaci dvěma body

$$N_1(x) = f(x_0) + (x - x_0) f[x_1, x_0] \tag{3.21}$$

Přidáváním dalších bodů získáváme kubickou interpolaci třemi body, atd.

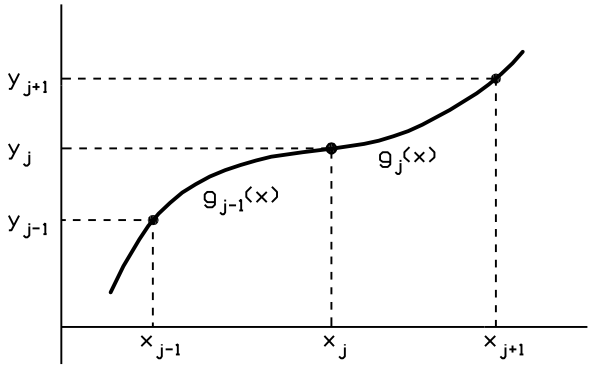
$$N_2(x) = f(x_0) + (x - x_0) f[x_1, x_0] + (x - x_0)(x - x_1) f[x_2, x_1, x_0] \tag{3.22}$$

$$\begin{aligned} N_n(x) = &f(x_0) + (x - x_0) f[x_1, x_0] + (x - x_0)(x - x_1) f[x_2, x_1, x_0] \\ &+ \cdots + (x - x_0)(x - x_1)\cdots(x - x_n) f[x_n, x_{n-1}, \ldots, x_0] \end{aligned} \tag{3.23}$$

Výhodou Newtonova interpolačního vzorce je to, že přidáváním dalších členů můžeme zvyšovat přesnost interpolace. Body x_i nemusí být ani seřazeny.

3.1.5 Kubické spliny

Hlavní nevýhodu polynomiální interpolace tj. problém zákmitů odstraňuje interpolace pomocí kubických splinů. Aproximační funkce je v tomto případě počástech kubický polynom, t.j. mezi uzlovými body použijeme polynomy třetího stupně.



Obrázek 3.4: Kubické spliny.

Mějme $n+1$ bodů, t.j. n intervalů a tudíž n kubických polynomů. Polynom na intervalu $\langle x_j, x_{j+1}\rangle$ si označíme $g_j(x)$. Tyto polynomy mají celkem $4n$ koeficientů a my tedy můžeme klást $4n$ podmínek. Polynomy musejí v uzlových bodech procházet danými hodnotami, to dává $2n$ podmínek

$$g_j(x_j) = y_j \tag{3.24}$$

$$g_j(x_{j+1}) = y_{j+1} \tag{3.25}$$

Dále budeme požadovat spojitost prvních a druhých derivací interpolační funkce, tak dostáváme dvě podmínky pro každý vnitřní uzlový bod - celkem $2n-2$ podmínky.

$$g'_{j-1}(x_j) = g'_j(x_j) \tag{3.26}$$

$$g''_{j-1}(x_j) = g''_j(x_j) \tag{3.27}$$

Zbývají dvě podmínky, většinou se tedy ještě požadují nulové druhé derivace v krajních uzlových bodech, tzv. přirozené spliny.
Předpokládejme nejprve, že známe druhé derivace y''_j v uzlových bodech. Kubický polynom na intervalu $\langle x_j, x_{j+1}\rangle$ budeme hledat ve tvaru

$$y(x) = A(x)y_j + B(x)y_{j+1} + C(x)y''_j + D(x)y''_{j+1} \tag{3.28}$$

kde A, B, C, D jsou vhodné kubické polynomy, splňující podmínky

$$A(x_j) = 1 \qquad A(x_{j+1}) = A''(x_j) = A''(x_{j+1}) = 0 \tag{3.29}$$

$$B(x_{j+1}) = 1 \qquad B(x_j) = B''(x_j) = B''(x_{j+1}) = 0 \tag{3.30}$$

$$C''(x_j) = 1 \qquad C''(x_{j+1}) = C(x_j) = C(x_{j+1}) = 0 \tag{3.31}$$

$$D''(x_{j+1}) = 1 \qquad D''(x_j) = D(x_j) = D(x_{j+1}) = 0 \tag{3.32}$$

Tyto podmínky zaručují, že polynomy procházejí zadanými body (podmínky (3.24), (3.25)) a že druhé derivace jsou spojité (podmínka (3.27)). Místo proměnné x zavedeme novou proměnou t, která na intervalu $\langle x_j, x_{j+1}\rangle$ nabývá hodnot 0 až 1.

$$t \equiv \frac{x - x_j}{x_{j+1} - x_j} \tag{3.33}$$

Polynomy A, B, C, D potom mají následující tvar

$$A \equiv \frac{x_{j+1} - x}{x_{j+1} - x_j} = 1 - t \qquad B \equiv 1 - A = \frac{x - x_j}{x_{j+1} - x_j} = t \tag{3.34}$$

$$C \equiv \frac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2 = \frac{1}{6}(-t^3 + 3t^2 - 2t)(x_{j+1} - x_j)^2 \tag{3.35}$$

$$D \equiv \frac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2 = \frac{1}{6}(t^3 - t)(x_{j+1} - x_j)^2 \tag{3.36}$$

Zbývá určit hodnoty druhých derivací. K tomu využijeme spojitost prvních derivací (podmínka (3.26)). Vyjádříme derivaci interpolační funkce (3.28) v bodě x_j pro interval $\langle x_j, x_{j+1}\rangle$ a interval $\langle x_{j-1}, x_j\rangle$

$$y'(x_j) = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} + (x_{j+1} - x_j)\Big(\frac{y''_{j+1}}{6} - \frac{y''_j}{3}\Big) \tag{3.37}$$

$$y'(x_j) = \frac{y_j - y_{j-1}}{x_j - x_{j-1}} + (x_j - x_{j-1})\Big(\frac{y''_j}{3} - \frac{y''_{j-1}}{6}\Big) \tag{3.38}$$

$$\tag{3.39}$$

Tyto derivace se musí rovnat. Po úpravě dostaneme podmínku

$$\frac{x_j - x_{j-1}}{6}y''_{j-1} + \frac{x_{j+1} - x_{j-1}}{3}y''_j + \frac{x_{j+1} - x_j}{6}y''_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}} \tag{3.40}$$

Tato podmínka musí platit pro všechna j. V krajních bodech použijeme podmínky $y''_0 = 0$ a $y''_n = 0$. Dostali jsme tedy soustavu lineárních rovnic pro neznámé druhé derivace. Matice této soustavy je třídiagonální a pro její řešení můžeme použít metody uvedené v kapitole (xxxx).

Příklad

Proložme čtyřmi body z předchozích příkladů [1,-10], [2,0], [3,10] a [4,-10] přirozený kubický spline. Nejprve je třeba vypočítat druhé derivace y''_i. Víme, že krajní hodnoty jsou nulové $y''_0 = y''_3 = 0$. Ze soustavy (3.40) je třeba vypočítat derivace v prostředních bodech y''_1 a y''_2. Po dosazení je tato soustava

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} y''_1 \\ y''_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -30 \end{bmatrix} \tag{3.41}$$

Řešením této soustavy dostaneme $y''_1 = 12$ a $y''_2 = -48$. Z (3.34)-(3.36) vypočítáme hodnoty polynomů $A(x)$, $B(x)$, $C(x)$, a $D(x)$. Například pro druhý interval $x \in \langle 2, 3\rangle$ dostaneme

$$t = x - 2 \tag{3.42}$$

$$A(x) = 1 - t \qquad = 3 - x \tag{3.43}$$

$$B(x) = t \qquad = x - 2 \tag{3.44}$$

$$C(x) = \tfrac{1}{6}(-t^3 + 3t^2 - 2t) = \tfrac{1}{6}(24 - 26x + 9x^2 - x^3) \tag{3.45}$$

$$D(x) = \tfrac{1}{6}(t^3 - t) \qquad = \tfrac{1}{6}(-6 + 11x - 6x^2 + x^3) \tag{3.46}$$

Po dosazení polynomů a derivací do (3.28), dostaneme výsledný polynom

$$g_1(x) = 76 - 130x + 66x^2 - 10x^3 \tag{3.47}$$

Podobně bychom získaly zbývající dva polynomy. Výsledný spline potom je

$$g(x) = \begin{cases} g_0(x) = -20 + 14x - 6x^2 + 2^3 & x \in \langle 1, 2\rangle \\ g_1(x) = 76 - 130x + 66x^2 - 10x^3 & x \in \langle 2, 3\rangle \\ g_2(x) = -410 + 356x - 96x^2 + 8x^3 & x \in \langle 3, 4\rangle \end{cases} \tag{3.48}$$

Graf tohoto splinu je na obrázku 3.3.

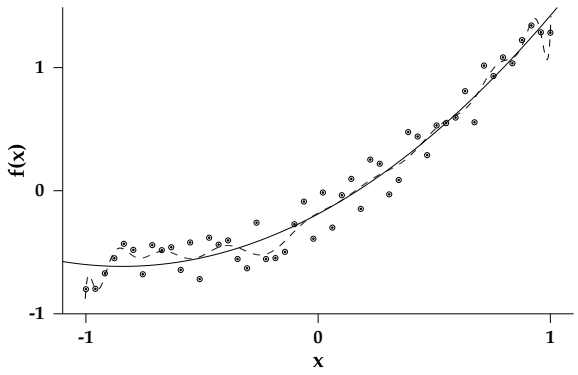
3.2 Aproximace

3.2.1 Aproximace MNČ

Předpokládejme, že máme zadánu řadu bodů $[x_i, y_i]$, $i = 0, \ldots, n$. Těmito body chceme proložit aproximační polynom ve tvaru

$$p_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m = \sum_{j=0}^{m} a_j x^j \tag{3.49}$$

Stupeň polynomu bývá většinou podstatně nižší (např m=2) než počet bodů (např. n=1000). V případě, že se počet bodů rovná stupni polynomu bude polynom body přímo procházet a bude se tedy jednat o interpolaci.



Obrázek 3.5: Aproximace MNČ polynomem stupně 2 a 19.

Budeme požadovat, aby součet kvadrátů odchylek $f(x_i)$ od aproximačního funkce $p_m(x_i)$ byl co nejmenší. Tento součet E je

$$E = \sum_{i=0}^{n} \epsilon_i^2 = \sum_{i=0}^{n} [p_m(x_i) - f(x_i)]^2 = \sum_{i=0}^{n} \sum_{j=0}^{m} [a_j x_i^j - f(x_i)]^2 \tag{3.50}$$

Tento součet je funkcí koeficientů polynomu a_k. Aby tento součet byl minimální musí být první derivace nulová.

$$\frac{\partial E}{\partial a_k} = 0 \tag{3.51}$$

Dosadíme vyjádření chyby (3.50) a upravíme

$$\frac{\partial E}{\partial a_k} = \frac{\partial}{\partial a_k}\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\left[a_j x_i^j - f(x_i)\right]^2\right) = \sum_{i=0}^{n}\sum_{j=0}^{m}\frac{\partial}{\partial a_k}\left[a_j x_i^j - f(x_i)\right]^2 = \tag{3.52}$$

$$= \sum_{i=0}^{n}\sum_{j=0}^{m} 2\left[a_j x_i^j - f(x_i)\right]\frac{\partial}{\partial a_k}\left[a_j x_i^j - f(x_i)\right] = \sum_{i=0}^{n}\sum_{j=0}^{m} 2\left[a_j x_i^j - f(x_i)\right]x_i^k \tag{3.53}$$

$$= 2\sum_{j=0}^{m}\sum_{i=0}^{n}\left[a_j x_i^{j+k} - f(x_i)x_i^k\right] = 2\Big(\sum_{j=0}^{m} a_j \sum_{i=0}^{n} x_i^{j+k} - \sum_{i=0}^{n} f(x_i)x_i^k\Big) = 0 \tag{3.54}$$

Pokud zavedeme označení

$$\begin{aligned} S_k &= \sum_{i=0}^{n} x_i^k \\ T_k &= \sum_{i=0}^{n} f(x_i)x_i^k \end{aligned} \tag{3.55}$$

dostaneme

$$\sum_{j=0}^{m} a_j S_{j+k} - T_k = 0 \tag{3.56}$$

Dostali jsme tedy soustavu lineárních rovnic

$$\begin{aligned} S_0a_0+ \quad & S_1a_1+\cdots+ \quad S_ma_m = T_0 \\ S_1a_0+ \quad & S_2a_1+\cdots+S_{m+1}a_m = T_1 \\ \vdots \quad & \\ S_ma_0+S_{m+1}a_1+\cdots+ \quad & S_{2m}a_m = T_m \end{aligned} \tag{3.57}$$

v maticovém zápisu

$$\begin{bmatrix} S_0 & S_1 & \cdots & S_m \\ S_1 & S_2 & \cdots & S_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_m & S_{m+1} & \cdots & S_{2m} \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_m \end{bmatrix} \tag{3.58}$$

Z této soustavy můžeme určit koeficienty polynomu a_k.

Příklad

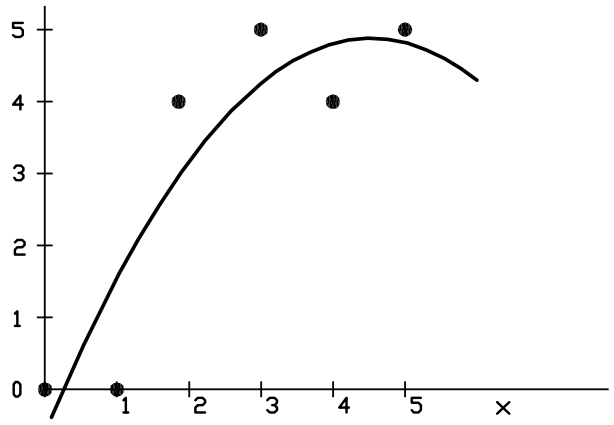
Proložme body $[1,0]$, $[2,0]$, $[3,4]$, $[4,5]$, $[5,4]$ a $[6,5]$ parabolu. Soustava (3.58) potom je

$$\begin{bmatrix} 6 & 21 & 91 \\ 21 & 91 & 441 \\ 91 & 441 & 2275 \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 18 \\ 82 \\ 396 \end{bmatrix} \tag{3.59}$$

Řešením této soustavy dostaneme $a_0 = -3.30000$, $a_1 = 2.96071$, $a_2 = -0.26785$. Výsledný polynom tedy bude

$$g(x) = -3.30000 + 2.96071x - 0.26785x^2 \tag{3.60}$$

Graf této aproximace je na obr. 3.6



Obrázek 3.6: Příklad aproximace MNČ.

Speciálním případem je lineární aproximace, kdy $m = 1$ a my tedy prokládáme přímku $g(x) = a_0 + a_1 x$. Soustavu lineárních rovnic (3.57) můžeme vyřešit Cramerovým pravidlem a dostaneme

$$a_0 = \frac{S_2 T_0 - S_1 T_1}{S_0 S_2 - S_1^2} \tag{3.61}$$

$$a_1 = \frac{S_0 T_1 - S_1 T_0}{S_0 S_2 - S_1^2} \tag{3.62}$$

Pokud dosadíme z (3.55) výrazy pro T_i a S_i, dostaneme známé vzorce pro lineární regresi

$$a_0 = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{(n+1)\sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.63}$$

$$a_1 = \frac{(n+1)\sum x_i y_i - \sum x_i \sum y_i}{(n+1)\sum x_i^2 - \left(\sum x_i\right)^2} \tag{3.64}$$

Metodu nejmenších čtverců lze zobecnit i pro aproximaci spojité funkce $f(x)$ na intervalu $\langle a, b\rangle$. Chyba je v tom případě definována vztahem

$$E(a_0, \ldots, a_m) = \int_a^b [p_m(x) - f(x)]^2 dx \tag{3.65}$$

Odvození metody je velice podobné předchozímu. Metoda opět vede na řešení soustavy (3.57), koeficienty T_i a S_i jsou ale definovány takto

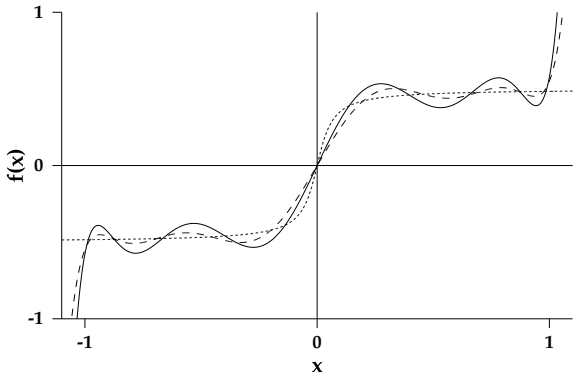
$$\begin{aligned} S_k &= \int_a^b x^k \,\mathrm{d}x = \frac{a^{k+1} - b^{k+1}}{k+1} \\ T_k &= \int_a^b x^k f(x) \,\mathrm{d}x \end{aligned} \tag{3.66}$$

3.2.2 Čebyševova aproximace

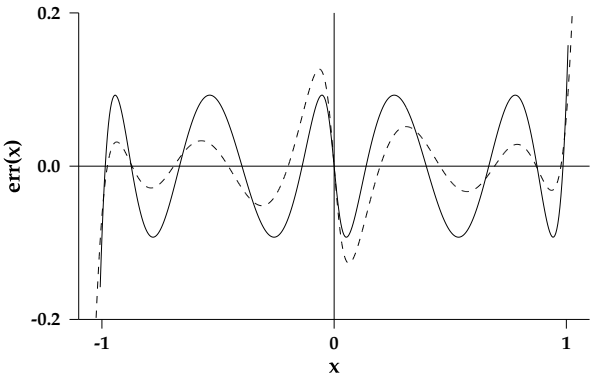
Při Čebyševově aproximaci se snažíme minimalizovat největší chybu v daných bodech nebo v daném intervalu. Chyba je tedy definována

$$E = \max_{i=0\ldots n} |\epsilon_i| = \max_{i=0\ldots n} |p_m(x_i) - f(x_i)| \tag{3.67}$$

Protože minimalizujme maximální chybu, používá se pro aproximační funkci někdy název *minimax*. Srovnání aproximace minimax a MNČ a jejich chyby je na obrázcích 3.7 a 3.8. Z těchto grafů je vidět, že chyba Čebyševovy aproximace je rovnoměrněji



Obrázek 3.7: Čebyševova aproximace (plně) a aproximace MNČ (čárkovaně).



Obrázek 3.8: Chyba Čebyševovy aproximace (plně) a aproximace MNČ (čárkovaně).

rozložena, její minima a maxima jsou stejně velká. Maximální chyba Čebyševovy aproximace je menší, střední chyba (plocha pod grafem) je naopak menší u aproximace metodou nejmenších čtverců.
Nalezení minimaxu není jednoduché, lze použít tzv. *Remezův algoritmus*. Místo Čebyševovy aproximace lze stačí někdy použít aproximaci Čebyševovými polynomy (i pro tuto aproximaci se někdy používá název Čebyševova aproximace). Tato aproximace dobře aproximuje minimax, používá se jako jeho první přiblížení. Nejprve si definujeme Čebyševovy polynomy.

Čebyševovy polynomy

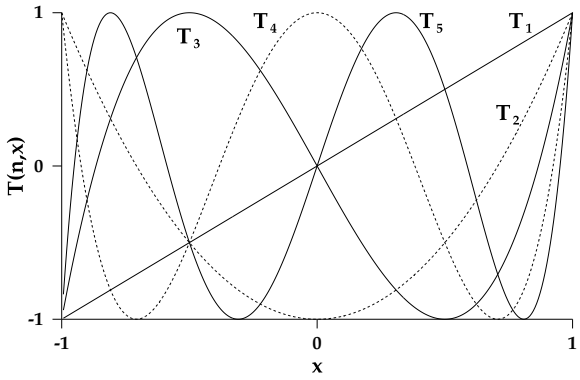
Posloupnost Čebyševových polynomů je definována vztahem

$$T_n(x) \equiv \cos(n \arccos x) \qquad n = 0, 1, 2, \ldots \tag{3.68}$$

Odtud lze odvodit explicitní tvary těchto polynomů. Tvar prvních pěti polynomů je

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \end{aligned} \tag{3.69}$$

Grafy prvních Čebyševových polynomů jsou na obrázku 3.9. Vidíme, že tyto poly-



Obrázek 3.9: Čebyševovy polynomy.

nomy oscilují mezi hodnotami -1 a 1.
Pro výpočet polynomů lze použít rekurentní vztah

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{3.70}$$

Čebyševovy polynomy jsou ortogonální v následujícím smyslu

$$\int_{-1}^{1} \frac{T_m(x)T_n(x)\,\mathrm{d}x}{\sqrt{1-x^2}} = \begin{cases} 0 & m \neq n \\ \pi/2 & m = n \neq 0 \\ \pi & m = n = 0 \end{cases} \tag{3.71}$$

Čebyševova aproximace

Ukážeme si aproximaci Čebyševovými polynomy pro spojitou funkci $f(x)$ na intervalu $\langle a, b\rangle$. Nejprve změníme proměnou x, tak abychom pracovali na intervalu $\langle -1, 1\rangle$.

$$y = \frac{2x - b - a}{b - a} \qquad x \in \langle a, b\rangle \qquad y \in \langle -1, 1\rangle \tag{3.72}$$

Čebyševovu aproximaci definujeme jako součet n Čebyševových polynomů

$$g(x) = \sum_{i=0}^{n} a_i T_i(x) \tag{3.73}$$

kde koeficienty a_i jsou definovány vztahem

$$a_j = \frac{1}{K_j} \int_{-1}^{1} \frac{f(x)T_j(x)}{\sqrt{1-x^2}}\,\mathrm{d}x \qquad K_j = \begin{cases} \pi & j = 0 \\ \pi/2 & j \neq 0 \end{cases} \tag{3.74}$$

Příklad

Najděme aproximaci funkce $\sin x$ Čebyševovými polynomy na intervalu $x \in \langle 0, \pi \rangle$. Nejprve je třeba přejít k intervalu $y \in \langle -1, 1 \rangle$.

$$\equiv \frac{2x - \pi}{\pi} \tag{3.75}$$

Budeme tedy uvažovat funkci $\sin[(y\pi + \pi)/2]$. Koeficienty a_i vypočteme z (3.74)

$$a_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{\sin[(y\pi + \pi)/2]}{\sqrt{1 - y^2}} \, \mathrm{d}y \tag{3.76}$$

$$a_1 = \frac{2}{\pi} \int_{-1}^{1} \frac{x \sin[(y\pi + \pi)/2]}{\sqrt{1 - y^2}} \, \mathrm{d}y \tag{3.77}$$

$$a_2 = \frac{2}{\pi} \int_{-1}^{1} \frac{(2y^2 - 1) \sin[(y\pi + \pi)/2]}{\sqrt{1 - y^2}} \, \mathrm{d}y \tag{3.78}$$

Numerickou integrací získáme tyto výsledky

$$a_0 = 0.4720012158 \qquad a_1 = 0 \qquad a_2 = -0.4994032583 \tag{3.79}$$

Hledaná aproximace tedy je

$$g(y) = 0.4720012158 \; T_0(y) - 0.4994032583 \; T_2(y) \tag{3.80}$$

Po dosazení Čebyševových polynomů dostaneme

$$g(y) = 0.9714044740 - 0.9988065165 \; y^2 \tag{3.81}$$

Přejdeme zpět k proměnné x a získáme konečný tvar naší aproximace funkce $\sin x$

$$g(x) = -0.0274020424 + 1.2717199543 \; x - 0.4048010339 \; x^2 \tag{3.82}$$

Maximální chyba této aproximace na intervalu $x \in \langle 0, \pi \rangle$ je 0.0286. Pokud bychom získali přesnou Čebyševovu aproximaci (minimax) např. Remezovým algoritmem byla by maximální chyba jen o málo menší a to 0.0280. Výsledná aproximace by v tom případě byla

$$g(x) = -0.0280046 + 1.2732393 \; x - 0.40528467 \; x^2 \tag{3.83}$$

Na tomto příkladu je vidět, že aproximace Čebyševovými polynomy je dobré přiblížení Čebyševovy aproximace - minimaxu.

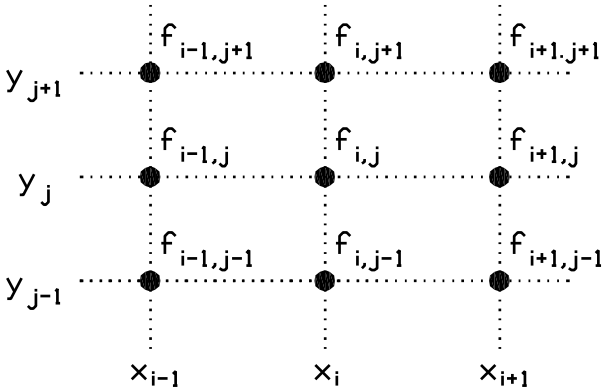
Pro stejnou úlohu bychom metodou nejmenších čtverců dostali aproximaci

$$g(x) = -0.0504654978 + 1.3122362048 \; x - 0.4176977570 \; x^2 \tag{3.84}$$

Maximální chyba této aproximace na intervalu $x \in \langle 0, \pi \rangle$ je větší než u Čebyševovy aproximace a to 0.0505.

3.3 Vícerozměrná interpolace

Aproximační úlohy, kterými jsme se doposud zabývali, pracovali s funkcemi jedné proměnné. Poměrně častou úlohou je však také dvojrozměrná interpolace. V tomto případě se snažíme najít funkci dvou proměnných $f(x, y)$, tak aby nabývala v



Obrázek 3.10: Vstupní hodnoty dvojrozměrné interpolace.

zadaných bodech daných hodnot. Hodnoty jsou nejčastěji zadány na pravoúhlé síti - máme tedy dvojrozměrou tabulku hodnot.
Předpokládejme tedy, že máme zadány hodnoty $f_{i,j}$ v bodech $[x_i, y_j]$.
V případě interpolace byla nejjednodušší možnost počástech lineární interpolace. Dvojrozměrnou analogií by tedy byla lineární interpolace na obdélnících. Obecný tvar dvojrozměrné lineární funkce je

$$g(x,y) = a + bx + cy \tag{3.85}$$

kde a, b, c jsou libovolné konstanty. Pokud se však pokusíme proložit takovouto funkci čtyřmi body v rozích obdélníka neuspějeme, protože tím máme čtyři podmínky, ale jen tři proměnné (lineární funkce reprezentuje rovinu, a ta zadána třemi nikoli čtyřmi body). Je tedy třeba použít obecnější funkci, která má čtyři nastavitelné parametry. Používá se tzv. bilineární funkce, která má obecný tvar

$$g(x,y) = a + bx + cy + dxy \tag{3.86}$$

kde a, b, c, d jsou libovolné konstanty. Tyto konstanty se většinou přímo neurčují, ale používá se následující postup výpočtu interpolace.

3.3.1 Bilineární interpolace

Provádíme tedy interpolaci na obdélníku $x \in \langle x_i, x_{i+1}\rangle$, $y \in \langle y_j, y_{j+1}\rangle$. Nejprve lineární transformací přejdeme k novým proměnným u a v, tak aby tyto na zadaném obdélníku nabývaly hodnot z intervalu $u \in \langle 0,1\rangle$ a $v \in \langle 0,1\rangle$. Tato transformace je

$$u = \frac{x - x_i}{x_{i+1} - x_i} \qquad v = \frac{y - y_j}{y_{j+1} - y_j} \tag{3.87}$$

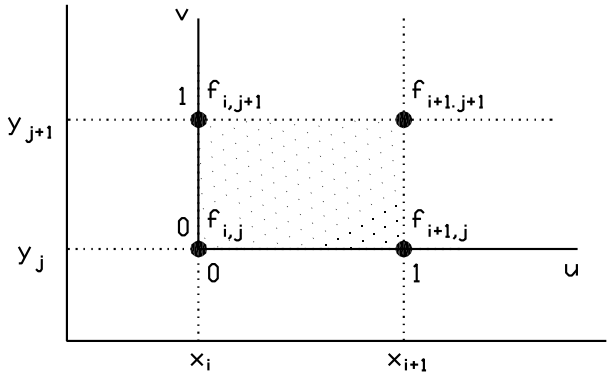
Bilineární transformaci můžeme v těchto proměnných přímo vyjádřit

$$g(u,v) = (1-u)(1-v)f_{i,j} + u(1-v)f_{i,j+1} + uv\, f_{i+1,j+1} + (1-u)v\, f_{i,j+1} \tag{3.88}$$

Vidíme, že tento vzorec má skutečně tvar (3.86). Dosazením souřadnic rohů obdélníka $[u,v] = [0,0],[0,1],[1,0],[1,1]$ se také přesvědčit, že v těchto rozích nabývá zadaných hodnot $f_{i,j}$. Vzorec (3.88) lze také zapsat ve tvaru

$$g(u,v) = (1-u)\big[(1-v)f_{i,j} + v\, f_{i,j+1}\big] + u\big[(1-v)f_{i,j+1} + v\, f_{i+1,j+1}\big] \tag{3.89}$$

Výrazy uvnitř hranatých závorek představují lineární interpolaci v proměnné y. Celkový výraz je potom interpolace v proměnné x.



Obrázek 3.11: Bilineární interpolace.

3.4 Cvičení

1. Náhodně vygenerovanými body proložte interpolační polynom. Použijte přímý výpočet koeficientů nebo Lagrangeovu metodu. Nakreslete graf.

2. Máte 5 bodů o souřadnicích $[0,0]$, $[1,0]$, $[2,1]$, $[3,0]$, $[4,0]$. Teoreticky odvoďte jak vypadá aproximace MNČ a Čebyševova aproximace polynomem nultého řádu (konstantní funkcí).

3. Naprogramujte aproximaci metodou nejmenších čtverců. Ukažte, že pokud je počet bodů roven stupni polynomu jedná se o interpolaci. Co se stane pokud je stupeň polynomu ještě větší?

4. Naprogramujte výpočet Čebyševových polynomů. Vykreslete graf prvních dvaceti.

Kapitola 4

Integrace a derivování

4.1 Kvadraturní vzorce

Velké množství integrálů nelze analyticky vypočítat (např $\int e^x \,\mathrm{d}x$). V takových případech je třeba využít metod numerické matematiky. Základní úlohou je výpočet určitého integrálu reálné funkce jedné proměnné.

$$I = \int_a^b f(x)\,\mathrm{d}x \tag{4.1}$$

Pro tuto úloha byla vyvinuta řada metod, ty nejdůležitější probereme v následujících kapitolách.
Abychom mohli jednotlivé metody srovnávat zavádí se pojem *řád metody*. Říkáme, že metoda má řád n, pokud je přesná pro všechny polynomy stupně n. Např. metoda druhého řádu integruje přesně všechny parabolické funkce.

4.1.1 Newtonovy-Cotesovy vzorce

Newtonovy-Cotesovy vzorce představují řadu základních metod pro integraci. Při těchto metodách rozdělíme interval $\langle a, b\rangle$ přes který integrujeme na n stejně velkých intervalů. Krajní body intervalů označíme x_i, délka intervalů bude h

$$x_i - x_{i-1} = h \qquad i = 1, \ldots, n \tag{4.2}$$

$$x_0 = a \qquad x_n = b \tag{4.3}$$

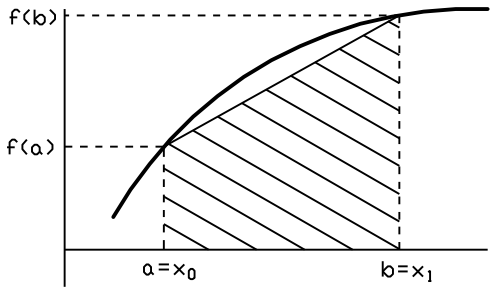
Integrál potom vyjádříme ve tvaru součtu

$$\int_a^b f(x)\,\mathrm{d}x = c_0 f(x_0) + \cdots + c_n f(x_n) + E \tag{4.4}$$

kde c_i jsou vhodně zvolené konstanty. Číslo E představuje chyba daného integračního vzorce, kterou při přibližném výpočtu integrálu zanedbáváme.
Nejjednodušší Newtonovy-Cotesovy vzorce jsou lichoběžníkové a obdélníkové pravidlo.

Lichoběžníkové pravidlo

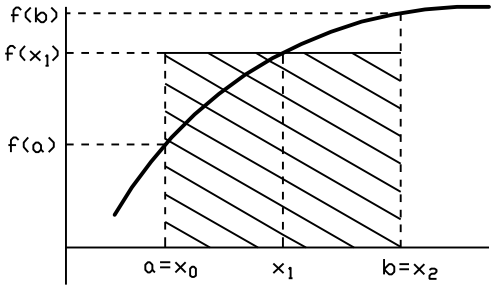
$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x = \frac{h}{2}[f(x_0) + f(x_1)] \tag{4.5}$$



Obrázek 4.1: Lichoběžníkové pravidlo.

Toto pravidlo využívá pouze jeden interval $n = 1$. Koeficienty ve vzorci (4.4) jsou $c_0 = c_1 = h/2$. Lichoběžníkové pravidlo je pravidlo prvního řádu a integruje tedy přesně všechny lineární funkce.

Obdélníkové pravidlo



Obrázek 4.2: Obdélníkové pravidlo.

$$\int_{x_0}^{x_2} f(x)\,\mathrm{d}x = 2hf(x_1) \tag{4.6}$$

Toto pravidlo využívá dva intervaly $n = 2$. Krajní body se ale ve vzorci nepoužívají, koeficienty tedy jsou $c_0 = c_2 = 0$, $c_1 = 2h$. Obdélníkové pravidlo je také pravidlo prvního řádu.

Newtonovy-Cotesovy vzorce se dělí na dvě skupiny - otevřené a uzavřené vzorce. V případě, že ve vzorci použijeme krajní body ($c_1 \neq 0$, $c_n \neq 0$), nazývá se vzorec uzavřený, v obráceném případě se nazývá vzorec otevřený. Nejjednodušším příkladem otevřeného vzorce je obdélníkové pravidlo, příkladem uzavřeného vzorce je lichoběžníkové pravidlo.
Složitější vzorce lze odvozovat dvěma způsoby. První možností je proložení interpolačního polynomu uzlovými body a integrace tohoto polynomu. Např proložit parabolu třemi body a zintegrovat ji. Druhou možností je určit přímo koeficienty vzorce (4.4) tak aby výsledný vzorec měl co nejvyšší řád. Ukážeme si tento druhý postup na příkladu Simpsonova pravidla.

Simpsonovo pravidlo

Odvodíme uzavřený Newtonův-Cotesův vzorec na dvou intervalech $n = 2$ (tedy pomocí tří bodů). Tento vzorec se nazývá *Simpsonovo pravidlo*.

Aby se výpočet zjednodušil posuneme interval integrace, tak aby jeho střed ležel v nule. Tento posun výsledek nijak neovlivní. Bude tedy hledat koeficienty a, b, c aby platilo

$$\int_{-h}^{h} f(x)\,\mathrm{d}x = af(-h) + bf(0) + cf(h) \tag{4.7}$$

Chceme aby tento vztah platil pro polynomy co nejvyššího stupně. Dosadíme tedy za $f(x)$ postupně vybrané polynomy $f(x) = 1$, $f(x) = x$, $f(x) = x^2$, ... Dostaneme tak tyto podmínky

$$\begin{aligned} a + b + c &= 2h \\ a(-h) + ch &= 0 \\ ah^2 + ch^2 &= \tfrac{2}{3}h^3 \end{aligned} \tag{4.8}$$

Získali jsme soustavu tří rovnic pro tři neznámé. Přidáním další podmínky ($f(x) = x^3$) bychom již měli víc rovnic než proměnných. Řešením této soustavy dostaneme

$$a = c = \frac{h}{3} \qquad b = \frac{4h}{3} \tag{4.9}$$

Pravidlo tedy je

$$\int_{-h}^{h} f(x)\,\mathrm{d}x = \frac{h}{3}\Big(f(-h) + 4f(0) + f(h)\Big) \tag{4.10}$$

Po posunutí zpět na interval $\langle a, b\rangle$ dostane výsledné Simpsonovo pravidlo

$$\int_{a}^{b} f(x)\,\mathrm{d}x = \frac{h}{3}\Big(f(a) + 4f(\tfrac{a+b}{2}) + f(b)\Big) \tag{4.11}$$

Chyby

Předchozí vzorce obecně neplatí přesně. Pokud budeme integrovat polynom vyššího řádu nebo funkci které není polynomem dopustíme se chyby. Např. Simpsonovo pravidlo by správě mělo být

$$\int_{a}^{b} f(x)\,\mathrm{d}x = \frac{h}{3}\Big(f(a) + 4f(\tfrac{a+b}{2}) + f(b)\Big) + E \tag{4.12}$$

kde E je chyba. Je třeba umět tuto chybu odhadnout. Odhad chyby se provádí tak, že do integračního vzorce dosadíme část Taylorovy řady integrované funkce. Ukážeme si to právě na příkladu Simpsonova vzorce.
Vyjdeme z posunutého tvaru (4.11). Funkci $f(x)$ rozvineme do Taylorovy řady v bodě $x = 0$.

$$f(x) = f(0) + xf'(0) + \tfrac{1}{2}x^2 f''(0) + \tfrac{1}{6}x^3 f'''(\theta) \tag{4.13}$$

Tento rozvoj dosadíme do levé i pravé strany integračního vzorce.

$$\int_{-h}^{h} f(x)\,\mathrm{d}x = 2hf(0) + \frac{h^3}{3}f''(0) + \frac{h^5}{60}f^{(4)}(\theta) \tag{4.14}$$

$$f(-h) = f(0) - hf'(0) + \tfrac{1}{2}h^2 f''(0) - \tfrac{1}{6}h^3 f'''(0) + \tfrac{1}{24}h^4 f^{(4)}(\theta) \tag{4.15}$$

$$f(h) = f(0) + hf'(0) + \tfrac{1}{2}h^2 f''(0) + \tfrac{1}{6}h^3 f'''(0) + \tfrac{1}{24}h^4 f^{(4)}(\theta) \tag{4.16}$$

$$\frac{h}{3}(f(-h) + 4f(0) + f(h)) = 2hf(0) + \frac{h^3}{3}f''(0) + \frac{h^5}{36}f^{(4)}(\theta) \tag{4.17}$$

Odečtením dostaneme výraz pro chybu

$$E = -\frac{h^5}{90} f^{(4)}(\theta) \tag{4.18}$$

Tento výraz je úměrný h^5, pokud se bude zmenšovat krok h bude se zmenšovat i chyba. Chyba je dále úměrná čtvrté derivaci integrované funkce. Pokud budeme tedy integrovat polynom třetího stupně bude chyba nulová - Simpsonovo pravidlo je tedy třetího řádu.

Uzavřené Newtonovy-Cotesovy vzorce

Podobně jako jsme odvodili Simpsonovo pravidlo je možné odvozovat složitější vzorce. Další v řadě je pravidlo na třech intervalech tzv. *tříosminové pravidlo*

$$\int_{x_0}^{x_3} f(x)\,\mathrm{d}x = \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] \tag{4.19}$$

Toto pravidlo je také třetího řádu.
Obecně platí, že pro lichý počet intervalů n je řád vzorce také n, pro sudý počet intervalů n je řád vzorce o jedna větší $n+1$. Pro vyšší hodnoty n ($n = 8$ a $n \geq 10$) jsou některé koeficienty c_i záporné, v důsledku toho dochází při použití těchto vzorců k velkým zaokrouhlovacím chybám.
Přehled koeficientů uzavřených vzorců je v tabulce 4.1.1.

n	A	c_0	c_1	c_2	c_3	c_4	c_5	c_6
1	$\frac{1}{2}$	1	1					
2	$\frac{1}{3}$	1	4	1				
3	$\frac{3}{8}$	1	3	3	1			
4	$\frac{2}{45}$	7	32	12	32	7		
5	$\frac{5}{288}$	19	75	50	50	75	19	
6	$\frac{1}{140}$	41	216	27	272	27	216	41
7	$\frac{7}{1728}$	751	3577	1323	2989	2989	1323	3577
8	$\frac{4}{14175}$	989	5888	-928	10496	-4540	10496	-928

Tabulka 4.1: Koeficienty uzavřených Newton-Cotesových vzorců

Otevřené Newtonovy-Cotesovy vzorce

Stejně jako uzavřené vzorce lze odvodit i další vzorce otevřené, např.

$$\int_{x_0}^{x_3} f(x)\,\mathrm{d}x = \frac{3h}{2}[f(x_1) + f(x_2)] \tag{4.20}$$

Podobně jako u uzavřených jsou pro vyšší n ($n = 4$ a $n \geq 6$) některé koeficienty záporné.

Složené vzorce

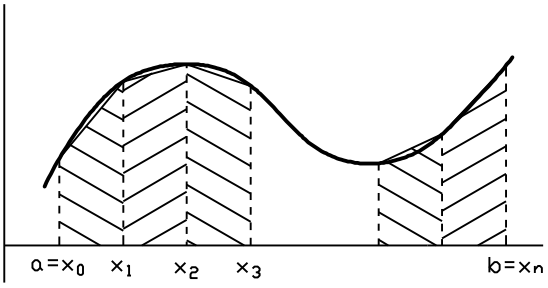
Při výpočtu integrálu chceme pochopitelně dosáhnout co nejvyšší přesnosti. První možnost pro zvýšení přesnosti, která se nabízí je použít metodu co nejvyššího řádu. Tento postup bohužel nefunguje. Ve vzorcích vysokých řádů se příliš projevují zaokrouhlovací chyby, ve vzorci pro chybu je také derivace příliš velikého stupně.

V tabulce 4.1.1 je ukázka řešení integrálu $\int_0^\pi \sin x \, \mathrm{d}x = 2$ Newton-Cotesovými vzorci různých řádů. Vidíme, že chyba s rostoucím počtem bodů klesá. Nejmenší je pro 12-ti bodové pravidlo. Pokud pokud ale použijeme větší počet bodů, chyba se začne opět zvětšovat.

n	I_n	n	I_n
2	0.0000000000000001	12	2.0000000009795653
3	2.0943951023931957	13	1.9999999418208351
4	2.0405242847634958	14	1.9999998084055956
5	1.9985707318238394	15	1.9999962440169665
6	1.9992030939158285	16	2.0000184626376311
7	2.0000178136377146	17	2.0000281591646893
8	2.0000108655419333	18	2.0007391474080340
9	1.9999998352723631	19	1.9554170445815453
10	1.9999998947797497	20	1.7668132049017497
11	2.0000000018133441	21	0.8017914165690169

Tabulka 4.2: Integrace n-bodovým Newton-Cotesovým vzorcem

Pokud tedy chceme dosáhnout větší přesnosti, je třeba rozdělit celkový interval $\langle a, b\rangle$ na několik intervalů a integrál spočítat jako součet dílčích integrálů. Vzorce, které tímto postupem dostaneme, se nazývají složené vzorce.



Obrázek 4.3: Složené lichoběžníkové pravidlo.

Příklad - složené Simpsonovo pravidlo:
Interval $\langle a, b\rangle$ rozdělíme na n dílčích intervalů. Na každé dvojici intervalů potom použijeme Simpsonovo pravidlo (počet intervalů musí tedy být sudý), dostaneme

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] + \frac{h}{3}[f(x_2) + 4f(x_3) + f(x_4)] + \dots \tag{4.21}$$

$$\int_a^b f(x)\,\mathrm{d}x = \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \dots + f(x_n)] \tag{4.22}$$

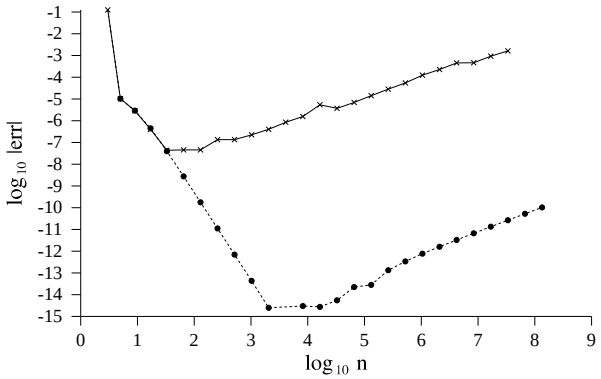
Chybu složeného pravidla je součtem chyb jednotlivých vzorců. Celkem tedy je

$$E = -\frac{n}{2}\frac{h^5}{90}f^{(4)}(\theta) = -\frac{(b-a)}{180}h^4 f^{(4)}(\theta) \tag{4.23}$$

Podobně získáme složené lichoběžníkové pravidlo

$$\int_a^b f(x)\,\mathrm{d}x = h[\tfrac{1}{2}f(x_0) + f(x_1) + \dots + f(x_{n-1}) + \tfrac{1}{2}f(x_n)] - \tag{4.24}$$

$$-\frac{(b-a)}{12}h^2 f''(\xi) \tag{4.25}$$



Obrázek 4.4: Závislost chyby složeného Simpsonova vzorce na počtu kroků (jednoduchá a dvojitá přesnost).

Z výrazu pro chybu (např. 4.23) vidíme, že s klesajícím krokem h se chyba zmenšuje. V celkové chybě se ale projeví i chyby zaokrouhlovací, tyto chyby s klesajícím krokem rostou a pro malé kroky převáží. Příklad závislosti celkové chyby na počtu kroků je na obrázku 3.1.

4.1.2 Odhad chyby

Chybu vypočítaného integrálu lze teoreticky odhadnout chybovými členy uvedenými u předchozích vzorců (např. 4.23). V těchto vzorcích se vyskytuje maximální hodnota derivace, která se zjišťuje obtížně. I když se podaří odhad derivace získat jsou výsledné odhady chyby zbytečně pesimistické. Proto se vzorce tohoto typu přímo k odhadu chyby nepoužívají.
Pro odhad chyby se používá nejčastěji *metoda polovičního kroku*. Předpokladem této metody je, že závislost chyby na délce kroku lze vyjádřit ve tvaru řady začínající k-tou mocninou, tj.

$$E(h) = a_k h^k + a_{k+1} h^{k+1} + \cdots \tag{4.26}$$

Všechny vzorce se kterými jsme se v této kapitole setkali tomuto předpokladu vyhovují, např. složené Simpsonovo pravidlo $k = 4$, $a_k = -\frac{(b-a)}{180} f^{(4)}$.
Spočteme odhad $I(h)$ našeho integrálu, pro dva různé kroky h_1 a h_2. Pro přesnou hodnotu integrálu I bude tedy platit

$$I = I(h_1) + a_k h_1^k + a_{k+1} h_1^{k+1} + \cdots \tag{4.27}$$

$$I = I(h_2) + a_k h_2^k + a_{k+1} h_2^{k+1} + \cdots \tag{4.28}$$

Tj. "přesný výsledek je přibližný výsledek plus chyba". Zanedbáme-li vyšší členy v rozvoji chyby, lze "přesnou" hodnotu integrálu I_e vyjádřit ve tvaru

$$I_e = I(h_1) + a_k h_1^k \tag{4.29}$$

$$I_e = I(h_2) + a_k h_2^k \tag{4.30}$$

V těchto vztazích neznáme pouze koeficient a_k a hodnotu I_e. Máme ale dvě rovnice pro dvě neznámé a ty můžeme vyřešit. Výsledkem je odhad chyby E

$$E(h_1) \equiv a_k h_1^k = \frac{h_1^k}{h_1^k - h_2^k} [I(h_2) - I(h_1)] \tag{4.31}$$

Druhý krok většinou volíme jako polovinu kroku prvního, odtud název metoda poloviční kroku, tj. $h_1 = 2h_2 = 2h$. Vzorec pro výpočet chyby potom je

$$E(2h) = \frac{2^k}{2^k - 1}[I(h) - I(2h)] \tag{4.32}$$

Tento vzorec je základem metody poloviční kroku. Ze dvou odhadů integrálu ($I(2h)$,$I(h)$) nám umožňuje odhadnou velikost chyby.

V rozvoji chyby se často vyskytují pouze sudé či pouze liché mocniny délky kroku (např. u Newtonových-Cotesových vzorců).

$$E(h) = \sum_{j=k}^{\infty} a_j h^{2j} \tag{4.33}$$

Výsledný vzorec v tom případě je

$$E(2h) = \frac{4^k}{4^k - 1}[I(h) - I(2h)] \tag{4.34}$$

Při správné implementaci funkční hodnoty pro jemný krok nevypočítáváme všechny znovu, ale použijeme hodnoty z integrace pro hrubý krok.

4.1.3 Gaussovy vzorce

Gaussovy vzorce jsou podobné Newton-Cotesovým vzorcům. Rozdíl spočívá v tom, že body v kterých počítáme funkční hodnoty, nejsou rozděleny ekvidistantně.

$$\int_a^b w(x)f(x)\,\mathrm{d}x = c_1 f(x_1) + \cdots + c_n f(x_n) \tag{4.35}$$

Polohy uzlových bodů jsou voleny optimálně tak, abychom dosáhli co nejvyššího řádu metody. Hodnoty konstant c_i a x_i můžeme vypočítat postupem podobným jako u Newtonových-Cotesových vzorců. Výsledná soustava rovnic v tomto případě však není lineární.
Polohy uzlových bodů závisí na integračních mezích, proto se tyto polohy uvádějí pro integraci na intervalu $\langle -1, 1\rangle$. Jiné integrály se převedou na tento případ substitucí

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(t\frac{b-a}{2} + \frac{b+a}{2}\right)\,\mathrm{d}t \tag{4.36}$$

Koeficienty nejjednodušších Gaussových vzorců jsou uvedeny v tabulce (4.3).
Například 3-bodový Gaussův vzorec je

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right) + \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right) \tag{4.37}$$

a pro obecný interval odtud dostaneme ($h \equiv (b-a)/2$)

$$\int_a^b f(x)\,\mathrm{d}x = \frac{8h}{9}f(\frac{a+b}{2}) + \frac{5h}{9}f\left(\frac{a+b}{2} + h\sqrt{\frac{3}{5}}\right) + \frac{5h}{9}f\left(\frac{a+b}{2} - h\sqrt{\frac{3}{5}}\right) \tag{4.38}$$

Gaussův vzorec který využívá n bodů má řád $2n-1$. Předchozí tříbodový vzorec má tedy řád 5. Gaussovy vzorce mají tedy přibližně dvojnásobný řád oproti Newtonovým-Cotesovým vzorcům.
Gaussovy vzorce se v praxi používají opět ve formě složených vzorců, ze stejných důvodů jako Newtonovy-Cotesovy vzorce.

n	x_i		c_i	
2	$\pm\sqrt{3}/3$	0.577350	1	1.000000
3	0	0.000000	8/9	0.888889
	$\pm\sqrt{3/5}$	0.774597	5/9	0.555556
4	$\pm[(15-2\sqrt{30})/35]^{1/2}$	0.339981	$(18+\sqrt{30})/36$	0.652145
	$\pm[(15+2\sqrt{30})/35]^{1/2}$	0.861136	$(18-\sqrt{30})/36$	0.347855
5	0	0.000000	128/225	0.568889
	$\pm[(35-2\sqrt{70})/63]^{1/2}$	0.538469	$(322+13\sqrt{70})/900$	0.478629
	$\pm[(35+2\sqrt{70})/63]^{1/2}$	0.906180	$(322-13\sqrt{70})/900$	0.236927

Tabulka 4.3: Koeficienty Gaussových vzorců

4.2 Rombergova kvadratura

4.2.1 Rombergův kvadraturní vzorec

Další zdokonalení integrace přináší tzv. Rombergova metoda. Tato metoda kombinuje některý z předchozích složených vzorců s tzv. Richardsonovou extrapolací.
Předpoklady této metody jsou stejné jako u metody poloviční kroku, tj. předpokládáme, že chybu lze vyjádřit ve tvaru řady (4.26). Stejně jako u metody poloviční kroku vypočítáme složeným pravidlem dva odhady integrálu. Ze vztahů (4.29) potom můžeme vyjádřit nejen chybu, ale i přesnější hodnotu integrálu I_e. Dostaneme

$$I_e = \frac{1}{h_1^k - h_2^k}[h_1^k I(h_2) - h_2^k I(h_1)] \tag{4.39}$$

Pokud opět použijeme poloviční krok dostaneme

$$I_e = \frac{1}{2^k - 1}[2^k I(h) - I(2h)] \tag{4.40}$$

Pokud by rozvoj chyby obsahoval pouze sudé mocniny dostali bychom místo (4.40)

$$I_e = \frac{1}{4^k - 1}[4^k I(h) - I(2h)] \tag{4.41}$$

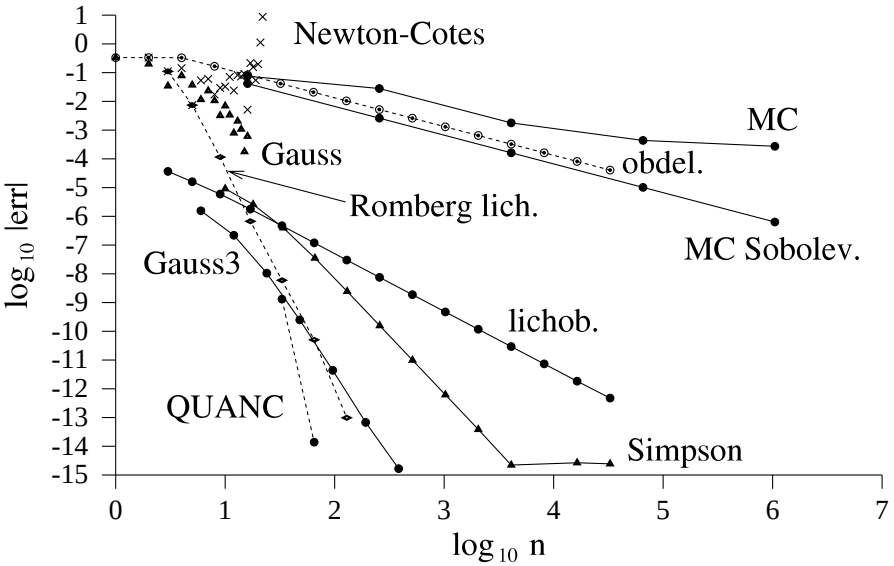
Tyto vzorce nám umožňují ze dvou odhadů integrálu $I(h)$, $I(2h)$ získat odhad přesnější I_e. Pokud původní vzorec měl řád k nový vzorec má řád $k+1$.
Celý tento postup můžeme opakovat. Spočítáme odhad integrálů T_{0i} pro několik postupně se zmenšujících hodnot kroku. Z (4.40) vypočteme odhady T_{1i} o řád přesnější. Z těchto opět odhady o řád přesnější, atd.. Celý proces můžeme znázornit následujícím schématem.

$$\begin{array}{lccccc} h = d & T_{00} & & & & \\ h = d/2 & T_{01} & T_{10} & & & \\ h = d/4 & T_{02} & T_{11} & T_{20} & & \\ & \vdots & \vdots & \vdots & & \\ h = d/2^m & T_{0m} & T_{1,m-1} & T_{2,m-2} & \cdots & T_{m0} \end{array} \tag{4.42}$$

První sloupec je vypočítán přímo integračním pravidlem a další sloupce jsou vypočítány rekurzivně

$$T_{mk} = \frac{1}{4^m - 1}(4^m T_{m-1,k+1} - T_{m-1,k}), \qquad k = 0, 1, \ldots \tag{4.43}$$

Hodnoty na diagonále tohoto schématu konvergují rychle k přesné hodnotě integrálu.



Obrázek 4.5: Srovnání relativní chyby integračních metod v závislosti na počtu použitých bodů.

4.3 Adaptivní metody

V dosud uvedených metodách jsme předpokládali, že velikost integračního kroku je konstantní na celém intervalu $\langle a, b\rangle$. Průběh funkce se ale může na tomto intervalu značně měnit. Velikost chyby však závisí na integrované funkci, např. chyba Simpsonova metody je úměrná čtvrté derivaci. Je tedy vhodné velikost integračního kroku h přizpůsobovat průběhu funkce. Tam kde se funkce prudce mění (tj. např. u Simpsonova pravidla je velká čtvrtá derivace) použijeme malý krok, tam kde se funkce moc nemění použijeme malý krok. Metodám, které automaticky používají tuto strategii se říká *adaptivní metody.*

Vstupem pro adaptivní metody není velikost integračního kroku h, ale např. požadovaná velikost relativní chyby ε. Vhodným integračním pravidlem a metodou poloviního kroku spočítáme hodnotu integrálu I a odhad jeho chyby ΔI. Pokud je získaná relativní chyba $\Delta I/I$ příliš velká, rozdělíme integrovaný interval na poloviny, které integrujeme zvlášť stejným postupem. Tímto způsobem se budou intervaly, kde je chyba velká, dělit na kratší, zatímco intervaly s malou chybou zůstanou neděleny.

4.4 Derivování

V této části se budeme zabývat opačnou operací než doposud a to derivováním. Metody pro numerické derivování vycházejí z definice derivace, jako například následující přibližný vzorec.

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{4.44}$$

V tomto vzorci nahrazujeme derivaci jejím přibližným vyjádřením - diferencí. Čím bude krok h menší tím přesnější odhad derivace dostaneme. Při velmi malém kroku se ale začnou hodnoty $f(x+h)$, $f(x)$ přibližovat a v předchozím vzorci dojde k odčítání s přibližně stejnou hodnotou a tím k nárůstu zaokrouhlovací chyby. V

důsledku toho bude výpočet derivace nepřesný. Tento nedostatek se projevuje u všech numerických metod pro výpočet derivace. Numerickému výpočtu derivace bychom se proto měli pokud je to možné vyhýbat.

Příklad

Vypočteme pomocí vzorce (4.44) derivace funkce $f(x) = e^x$ v bodě nula pro různě velké kroky. Výsledky jsou v tabulce 4.4. Je vidět, že v tomto případě je op-

h	$f'(0)$
10^0	1.7182817
10^{-1}	1.0517097
10^{-2}	1.0050178
10^{-3}	1.0005237
10^{-4}	1.0001661
10^{-5}	1.0013582
10^{-6}	0.9536744
10^{-7}	1.1920930

Tabulka 4.4: Odhady derivace funkce e^x

timální délka kroku h přibližně 10^{-4}. Pokud je krok ještě menší chyba narůstá díky zaokrouhlovacím chybám.

Vzorce pro výpočet n-té derivace mají v obecném případě tvar

$$f^{(n)}(x) = \sum_{i=s}^{r} c_i f(x_i) \tag{4.45}$$

kde c_i jsou vhodně zvolené konstanty a x_i jsou body v okolí bodu x ve kterém chceme vypočítat derivaci. Nejčastěji se body volí rovnoměrně rozmístěné okolo bodu x s krokem h tj.

$$x_i = x + ih \tag{4.46}$$

Koeficienty c_i volíme tak, aby vzorec derivoval přesně polynomy co nejvyššího řádu, podobně jako jsme odvozovali Newtonovy-Cotesovy vzorce.

Příklad

Odvoďme vzorec pro výpočet druhé derivace pomocí tří bodů, derivaci chceme určit v prostředním z nich. Tento vzorec má tedy obecně tvar

$$f''(x) = c_{-1} f(x_{-1}) + c_0 f(x_0) + c_1 f(x_1) \tag{4.47}$$

jednodušeji zapsáno

$$f''(x) = af(x-h) + bf(x) + cf(x+h) \tag{4.48}$$

Aby byl výpočet jednodušší posuneme celý vzorec, tak aby prostřední bod ležel v nule.

$$f''(0) = af(-h) + bf(0) + cf(h) \tag{4.49}$$

Nyní dosadíme za $f(x)$ postupně vybrané polynomy $f(x) = 1$, $f(x) = x$, $f(x) = x^2$. Dostaneme tak tyto podmínky

$$\begin{aligned} 0 &= a + b + c \\ 0 &= -ha + hc \\ 2 &= h^2 a + h^2 c \end{aligned} \tag{4.50}$$

Řešení této soustavy je $a = c = \frac{1}{h^2}$, $b = -\frac{2}{h^2}$. Výsledný vzorec pro derivování tedy je

$$f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} \tag{4.51}$$

Vzorce jako (4.44) či (4.51) samozřejmě neplatí pro obecné funkce přesně, ale obsahují chybu. Odhad této chyby lze provést stejně jako odhad chyby integračních vzorců, tj. dosazením Taylorovy řady.

Příklad

Odvodíme chybu vzorce pro výpočet druhé derivace z předchozího příkladu. Funkci f rozvineme do řady v okolí bodu 0

$$f(h) = f(0) + hf'(0) + \frac{h^2}{2}f''(0) + \frac{h^3}{6}f'''(0) + \frac{h^4}{24}f^{(4)}(\theta) \tag{4.52}$$

$$f(-h) = f(0) - hf'(0) + \frac{h^2}{2}f''(0) - \frac{h^3}{6}f'''(0) + \frac{h^4}{24}f^{(4)}(\theta) \tag{4.53}$$

Tento rozvoj dosadíme do vzorce (4.51) do kterého ještě přidáme chybu E a vzorec opět pro snadnější výpočet posuneme do nuly

$$f''(0) = \frac{f(-h) - 2f(0) + f(h)}{h^2} + E \tag{4.54}$$

Po dosazení a úpravě dostaneme

$$E = -\frac{h^2}{12}f^{(4)}(\theta) \tag{4.55}$$

U vzorců na derivování se většinou nemluví řádu, ale uvádí se jaká mocnina kroku h je v odhadu chyby. O předchozím vzorci bychom tedy řekli, že je $O(h^2)$.
Vzorec (4.44) pro výpočet první derivace má chybu $E = -h/2f''$ je to tedy vzorec pouze O(h). První derivaci lze ale vypočítat pomocí dvou bodů i přesněji a to vzorcem

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{4.56}$$

Chyba toho vzorce je $E = -h^2/6f'''(x)$ a je tedy přesnější než (4.44). Obecně platí, že symetrické vzorce bývají přesnější než vzorce nesymetrické a proto je dobré jim dávat přednost.
V tabulce 4.4 jsou uvedeny koeficienty a chyby dalších vzorců pro derivování

4.5 Cvičení

1. Odvoďte otevřený Newtonův-Cotesův vzorec na třech intervalech.

2. Odvoďte chybu obdélníkového a lichoběžníkového pravidla.

	c_{-2}	c_{-1}	c_0	c_1	c_2		
f'_0			-1	1		h	$-h/2f''$
f'_0		-1		1		$2h$	$-h^2/6f^{(3)}$
f'_0			-3	4	-1	$2h$	$h^2/3f^{(3)}$
f'_0	1	-8	0	8	-1	$12h$	$h^4/30f^{(5)}$
$f'_{1/2}$		1	-27	27	-1	$24h$	? $h^4f^{(5)}$
f''_0		1	-2	1		h^2	$-h^2/12f^{(4)}$
f''_0			1	-2	1	h^2	$-hf^{(3)}$
f''_0	-1	16	-30	16	-1	$12h^2$	$h^4/90f^{(6)}$
$f^{(3)}_0$		-1	3	-3	1	h^3	$-h/2f^{(4)}$
$f^{(3)}_0$	-1	2		-2	1	$2h^3$	$-h^2/4f^{(5)}$
$f^{(4)}_0$	1	-4	6	-4	1	h^4	$-h^2/6f^{(6)}$

Tabulka 4.5: Koeficienty a chyby diferenčních vzorců

3. Odvoďte dvojbodový Gaussův vzorec. Jakého je řádu?

4. Naprogramujte složené lichoběžníkové a Simpsonovo pravidlo. Pro jaký krok dostaneme nejpřesnější výsledek? (zkuste např. $\int_0^\pi \sin x \,\mathrm{d}x = 2$).

5. Naprogramujte složené Simpsonovo pravidlo i s odhadem jeho chyby metodou poloviční kroku.

6. Odvoďte vzorec pro výpočet první derivace $f'(x)$ pomocí hodnot $f(x)$, $f(x+h)$, $f(x+2h)$.

7. Odvoďte chyby vzorců (4.44) a (4.51).

Kapitola 5

Řešení nelineárních rovnic

5.1 Řešení nelineárních rovnic

Celá řada rovnic, které vyskytují v matematice a fyzice je analyticky neřešitelná. Polynomiální rovnice jsou například řešitelné jen do čtvrtého stupně. V této kapitole si ukážeme metody jak takovéto rovnice řešit alespoň přibližně.
Budeme hledat kořeny algebraické rovnice

$$f(x) = 0 \tag{5.1}$$

Kořen budeme hledat iteračními metodami. Začne s přibližným odhadem kořene x_0 a tento odhad budeme postupně upřesňovat x_1,x_2,.... Nový odhad vypočítáme z několika odhadů předchozích. Tyto iteračních metody mají tedy tvar

$$x_{i+1} = F_i(x_i, x_{i-1}, \ldots, x_{i-n+1}) \tag{5.2}$$

Chybu i-té aproximace ε_i je rozdíl našeho odhadu x_i a přesného řešení x

$$\varepsilon_i \equiv x_i - x \tag{5.3}$$

Podobně jako u jiných úloh numerické matematiky i u metod na hledání kořene zavádíme řád metody. Řekneme, že metoda je řádu p, pokud platí

$$\lim_{i\to\infty} \frac{|\varepsilon_{i+1}|}{|\varepsilon_i|^p} = C \qquad \text{kde} \qquad 0 < C < \infty \tag{5.4}$$

Pro pochopení metod jsou nejdůležitější dva případy - řád 1 a 2.
Pro metody prvního řádu v limitě platí, že

$$|\varepsilon_{i+1}| = C|\varepsilon_i| \tag{5.5}$$

Pokud by bylo např $C = \frac{1}{10}$ klesla by chyba v každém kroku 10-krát. V každém kroku bychom tedy získali jednu platnou číslici výsledku.
Pro metody druhého řádu v limitě platí, že

$$|\varepsilon_{i+1}| \approx |\varepsilon_i|^2 \tag{5.6}$$

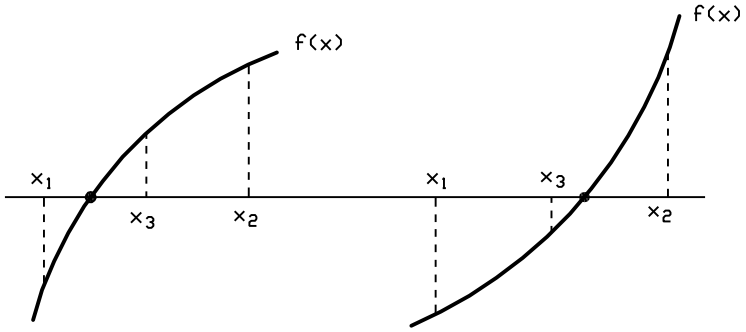
Nová chyba je tedy mocninou předchozí chyby, např. $\varepsilon_i = 0.1, 0.01, 0.0001, 0.000001$.
Počet platných číslic výsledku se tedy v každém kroku zdvojnásobuje.

5.1.1 Metoda půlení intervalu

Nejzákladnější metodou je metoda půlení intervalu. Do celého algoritmu a poté i do každé iterace vstupují dva body x_1 a x_2 ve kterých funkce nabývá opačných znamének, t.j. musí platit

$$f(x_1)f(x_2) < 0 \tag{5.7}$$

V každé iteraci sestrojíme prostřední bod x_3. Tu z dvojic x_1,x_3 a x_3,x_2, která splňuje podmínku (5.7) použijeme do další iterace.



Obrázek 5.1: Metoda půlení intervalu.

$$x_3 = (x_1 + x_2)/2 \tag{5.8}$$

$$f(x_1)f(x_3) < 0 \qquad x_3 \to x_2 \tag{5.9}$$

$$f(x_1)f(x_3) > 0 \qquad x_3 \to x_1 \tag{5.10}$$

Tato metoda je vždy kovergentní, je ale pouze 1. řádu. Konstanta C ve vztahu (5.4) je rovna $\frac{1}{2}$. Na každou iteraci se tedy odhad kořene zpřesní o jednu binární číslici.

5.1.2 Metoda jednoduché iterace

Řešenou rovnici $f(x) = 0$ vhodným způsobem upravíme do tvaru

$$g(x) = x \tag{5.11}$$

Potom provádíme iterace

$$x_{i+1} = g(x_i) \tag{5.12}$$

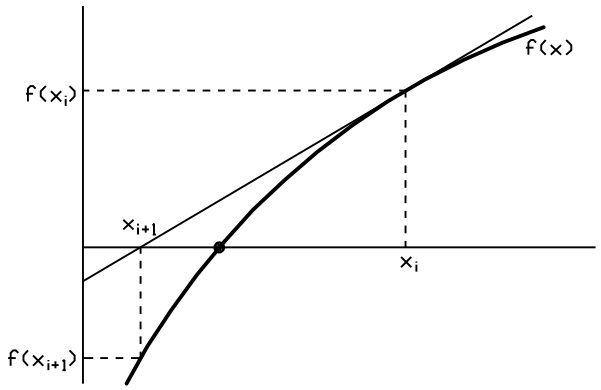
Tento proces je kovergentní pouze pokud platí

$$|g'(x)| < 1 \tag{5.13}$$

5.1.3 Newtonova metoda

Newtonova metoda se též nazývá metoda tečen. Začneme s odhadem kořene x_0. V tomto bodě sestrojíme tečnu ke grafu funkce f. Průsečík této tečny s x-ovou osou použijeme jako další odhad kořene.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{5.14}$$



Obrázek 5.2: Newtonova metoda.

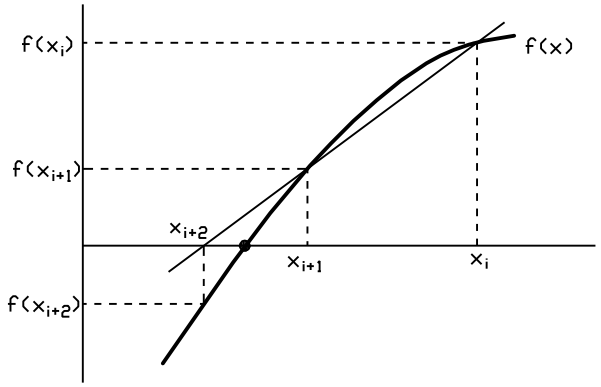
Pokud tato metoda koverguje, konverguje velice rychle. Je to metoda 2. řádu, počet platných desetinných míst výsledky se v každé iteraci zdvojnásobuje. Bohužel tato metoda není globálně konvergentní. Pro hladkou funkci však vždy existuje okolí kořene ve kterém metoda konvergentní je. Další nevýhodou je potřeba výpočtu derivace funkce. Tato metoda se většinou používá v kombinaci s jinou metodou zaručující konvergenci.

5.1.4 Metoda sečen

Metoda sečen vychází z Newtonovy metody. Používáme však dva odhady kořene a na místo tečny používáme sečnu procházející těmito body. Starší odhad kořene nahradíme průsečíkem sečny a osy x.

$$x_{i+2} = x_i - \frac{x_{i+1} - x_i}{f(x_{i+1}) - f(x_i)} f(x_i) \tag{5.15}$$

Tato metoda není globálně konvergentní. Řád této metody je $(1 + \sqrt{5})/2 \approx 1.618$.



Obrázek 5.3: Metoda sečen.

5.1.5 Metoda regula fasci

Jinou variantou je metoda regula fasci. Je téměř stejná jako metoda půlení intervalu. Rozdíl spočívá v tom, že nový bod není uprostřed předchozích odhadů, ale je to

průsečík sečny procházející těmito body.

$$f(x_0)f(x_1) < 0 \tag{5.16}$$

$$x_2 = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0) \tag{5.17}$$

Tato metoda je vždy konvergentní, ke kořeni však konverguje pouze jeden odhad, druhý zůstává konstantní. Řád této metodu je 1.

Příklad

Řešme rovnici

$$x + \sin x - \tfrac{1}{2} = 0 \tag{5.18}$$

tato rovnice má kořen $x = 0.251318624524097$. V tabulce 5.1.5 je srovnání metody půlení intervalu a Newtonovy metody.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x - x_3^{(k)}$	$x^{(k)}$	$x - x^{(k)}$
0	0.000000	0.500000	0.001319	1.5000000	-1.2486813
1	0.250000	0.500000	-0.123681	-0.3655323	0.6168510
2	0.250000	0.375000	-0.061181	0.2668466	-0.0155280
3	0.250000	0.312500	-0.029931	0.2513027	0.0000159
4	0.250000	0.281250	-0.014306	0.2513186	0.0000000
5	0.250000	0.265625	-0.006494		
6	0.250000	0.257812	-0.002588		
7	0.250000	0.253906	-0.000634		
8	0.250000	0.251953	0.000342		
9	0.250977	0.251953	-0.000146		
10	0.250977	0.251465	0.000098		
11	0.251221	0.251465	-0.000024		
12	0.251221	0.251343	0.000037		
13	0.251282	0.251343	0.000006		

Tabulka 5.1: Srovnání půlení intervalu (vlevo) a Newtonovy metody (vpravo).

5.1.6 Násobné kořeny

Připomeňme definici n-násobného kořene rovnice $f(x) = 0$

$$f^{(i)} = 0 \qquad i = 0, 1, \ldots, n-1 \tag{5.19}$$

$$f^{(n)} \neq 0 \tag{5.20}$$

V případě, že hledané kořeny jsou vícenásobné, většina metod selhává. Jednou možností je využití toho, že jestliže $f(x) = 0$ má vícenásobný kořen potom rovnice $u(x) = 0$, kde $u(x) = \frac{f(x)}{f'(x)}$, má jednonásobný kořen. Jinou možností je následující modifikace Newtonovy metody

$$x_{i+1} = x_i - n\frac{f(x_i)}{f'(x_i)} \tag{5.21}$$

5.1.7 Aitkenův δ^2-proces

Pro metody prvního řádu existuje možnost urychlení, zvaná Aitkenův δ^2-proces. Označme α řešení naší rovnice. Pro metodu prvního řádu platí

$$\alpha - x_{i+1} = C_i(\alpha - x_i) \tag{5.22}$$

Předpokládejme, že konstanty C_i se již příliš nemění $C_i \approx C$, potom

$$\frac{\alpha - x_{i+2}}{\alpha - x_{i+1}} \approx \frac{\alpha - x_{i+1}}{\alpha - x_i} \tag{5.23}$$

Z této rovnice můžeme určit nový odhad kořene

$$\alpha \approx \frac{x_i x_{i+2} - x_{i+1}^2}{x_{i+2} - 2x_{i+1} + x_i} = x_{i+2} - \frac{(\Delta x_{i+1})^2}{\Delta^2 x_i} \tag{5.24}$$

Odhady vypočítané z toho vztahu konvergují rychleji než původní iterace x_i.

5.1.8 Soustavy rovnic

Budeme nyní řešit soustavu n rovnic o n neznámých

$$F_i(x_1, x_2, \ldots, x_n) = 0 \qquad i = 1, 2, \ldots, n \tag{5.25}$$

Newtonova metoda

Základní metodou je zobecnění Newtonovy metody. Nejprve rozvineme funkce levých stran do řady

$$F_i(\boldsymbol{x} + \delta\boldsymbol{x}) = F_i(\boldsymbol{x}) + \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j} + O(\delta\boldsymbol{x}^2) \tag{5.26}$$

Zavedeme matici derivací - *Jacobián* J_{ij}

$$J_{ij} \equiv \frac{\partial F_i}{\partial x_j} \tag{5.27}$$

$$\boldsymbol{F}(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{J} \cdot \delta\boldsymbol{x} + O(\delta\boldsymbol{x}^2) \tag{5.28}$$

Budeme uvažovat iterační proces $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \delta\boldsymbol{x}$. Naším cílem je zvolit $\delta\boldsymbol{x}$ tak, aby $\boldsymbol{F}(\boldsymbol{x} + \delta\boldsymbol{x}) = 0$. Z rovnice (5.28) po zanedbání posledního členu dostáváme podmínku $\boldsymbol{J} \cdot \delta\boldsymbol{x} = -\boldsymbol{F}$. Celý iterační proces tedy je

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \boldsymbol{J}^{-1}(\boldsymbol{x}_i) \cdot \boldsymbol{F}(\boldsymbol{x}_i) \tag{5.29}$$

Dostali jsme tedy vztah podobný původní Newtonově metodě. Namísto dělení derivací nyní však musíme násobit inverzním Jacobiánem.
Metoda není globálně konvergentní, pokud však máme dostatečně dobrý odhad kořene, metoda konverguje rychle. V každém kroku je třeba vypočítat Jacobián a řešit soustavu lineárních rovnic.

Příklad

Řešíme soustavu rovnic

$$2x^3 - y^2 - 1 = 0 \tag{5.30}$$

$$xy^3 - y - 4 = 0 \tag{5.31}$$

k	$x^{(k)}$	$y^{(k)}$
0	1.00000000000000	1.00000000000000
1	1.57142857142857	2.71428571428571
2	1.38455347014689	2.09253530359103
3	1.27276925577104	1.76850431820878
4	1.23740148731123	1.67018786017900
5	1.23429695463203	1.66158901065441
6	1.23427448529044	1.66152647008248
7	1.23427448411448	1.66152646679593

Tabulka 5.2: Řešení soustavy rovnic Newtonovou metodou.

Vypočítáme parciální derivace a sestavíme Jakobián této soustavy

$$J = \begin{bmatrix} 6x^2 & -2y \\ y^3 & 3xy^2 - 1 \end{bmatrix} \tag{5.32}$$

V každém kroku metody je třeba nejprve vyřešit soustavu lineárních rovnic

$$\begin{bmatrix} 6x^2 & -2y \\ y^3 & 3xy^2 - 1 \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} 2x^3 - y^2 - 1 \\ xy^3 - y - 4 \end{bmatrix} \tag{5.33}$$

a poté vypočítat novou aproximaci řešení

$$x \leftarrow x - dx \qquad y \leftarrow y - dy \tag{5.34}$$

Výsledky tohoto postupu jsou v tabulce 5.1.8. Výsledky v posledním řádku jsou platné na plný počet číslic. Vidíme, že metoda konverguje velice rychle.

5.2 Cvičení

1. Naprogramujte metodu půlení intervalu, metodu sečen a Newtonovu metodu, srovnejte jejich rychlost a přesnost. (řešte např. $\arctan 10x - 1.2 = 0$)
2. Najděte kořeny rovnice $x^4 - 3x^3 - 2x^2 + 2x + 1 = 0$. (řešení je x=-0.77222289576562, -0.45588678010287, 0.83785279135297, 3.39025688451552)

5.3 Hledání minima a maxima

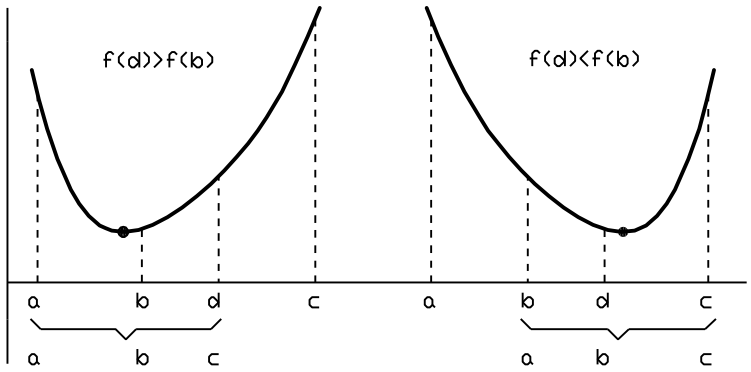
5.3.1 Metoda zlatého řezu

Metoda zlatého řezu je iterační metoda pro nalezení minima funkce jedné proměnné. Vstupem do algoritmu a poté do každé iterace jsou tři body $x = a, b, c$. Funkční hodnota v prostředním bodě musí být menší než funkční hodnoty v krajních bodech.

$$a < b < c \tag{5.35}$$

$$f(b) < f(a) \qquad f(b) < f(c) \tag{5.36}$$

K těmto třem bodům vybereme vhodným způsobem čtvrtý bod d. Z těchto čtyř bodů použijeme do další iterace levé nebo pravé tři body tak, aby podmínka (5.36) byla opět splněna.



Obrázek 5.4: Dvě situace metody zlatého řezu.

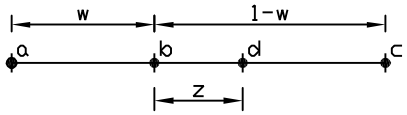
V každé iteraci mohou nastat dva případy (levé nebo pravé body), čtvrtý bod zvolíme tak, aby výsledný interval byl v obou případech stejně dlouhý. Předpokládejme, že interval ab je kratší než interval bc. Poměrnou délku intervalu ab označíme w.

$$\frac{b-a}{c-a} = w \qquad \frac{c-b}{c-a} = 1 - w \tag{5.37}$$

Čtvrtý bod d umístíme do delšího intervalu (t.j. bc). Relativní vzdálenost od bodu b označíme z.

$$\frac{d-b}{c-a} = z \tag{5.38}$$

Označení relativních délek je tedy následující



Aby délky intervalů ad a bc byly stejné musí být

$$w + z = 1 - w \tag{5.39}$$

$$z = 1 - 2w \tag{5.40}$$

I když začneme s libovolným poměrem délek intervalů, poměr w konverguje ke konstantní hodnotě. Pokud je již poměr délek na začátku a po provedení iterace stejný, platí

$$\frac{z}{w+z} = \frac{z}{1-w} = w \tag{5.41}$$

Řešením dostaneme

$$w^2 - 3w + 1 = 0 \tag{5.42}$$

$$w = \frac{3-\sqrt{5}}{2} \approx 0.38197 \tag{5.43}$$

Tento poměr je znám jako poměr zlatého řezu, odtud název metody. Shrňme si celý postup:

- vstup body $a < b < c$
- opakovat následující
  - pokud $b - a > c - b$ (tj. interval ab je větší než bc)
    - nový bod $d = b - w(b - a)$ (tj. body jsou v pořadí $adbc$)
    - pokud $f(d) < f(b)$ (použijeme levé tři body adb)
      - $c \leftarrow b$, $b \leftarrow d$
    - jinak (tj. $f(d) > f(b)$; použijeme pravé tři body dbc)
      - $a \leftarrow d$
  - jinak (tj. interval ab je menší než bc)
    - nový bod $d = b + w(c - b)$ (tj. body jsou v pořadí $abdc$)
    - pokud $f(b) < f(d)$ (použijeme levé tři body abd)
      - $c \leftarrow d$
    - jinak (tj. $f(b) > f(d)$; použijeme pravé tři body bdc)
      - $a \leftarrow b$, $b \leftarrow d$

Při této přímočaré implementaci algoritmu nevyužíváme informaci o délce intervalů z předchozí iterace, a proto je třeba vždy testovat, který ze dvou intervalů je delší. Tento nedostatek odstraňuje následující implementace metody zlatého řezu

- vstup body $a < b < c$
- nový bod do většího segmentu $d = b + w(c - b)$ nebo $d = b - w(b - a)$
- seřadit $a < b < d < c$
- opakovat následující
  1. pokud $f(d) < f(b)$ – beru pravé tři body; nový mezi d a c, blíž k d
     $a \leftarrow b$, $b \leftarrow d$, $d \leftarrow (1 - w)b + wc$
  2. pokud $f(d) \geq f(b)$ – beru levé tři body; nový mezi a a b, blíž k b
     $c \leftarrow d$, $d \leftarrow b$, $b \leftarrow (1 - w)d + wa$

Příklad

Hledáme minimum funkce

$$f(x) = x^2 - x - 2\cos x + 3 \tag{5.44}$$

minimum se nachází v bodě $x = 0.251318624524097$ a má hodnotu
$f(x) = 0.874671733470503$.
V tabulce 5.3.1 jsou postupné iterace metody zlatého řezu při počátečním odhadu $a = 0$, $b = \frac{1}{4}$, $c = \frac{1}{2}$. Vidíme, že hodnota ke které algoritmus konverguje se shoduje s přesným výsledkem pouze na tři desetinná místa. Přesnějšího výsledku nelze při dané přesnosti reálných čísel dosáhnout, protože funkční hodnoty jsou ve všech bodech v okolí minima stejné.

k	$a^{(k)}$	$c^{(k)}$	$f(a^{(k)})$	$f(c^{(k)})$
1	0.0000000	0.5000000	1.0000000	0.9948349
2	0.1545085	0.5000000	0.8931898	0.9948349
3	0.1545085	0.3454915	0.8931898	0.8920546
4	0.2135255	0.3454915	0.8774878	0.8920546
5	0.2135255	0.2864745	0.8774878	0.8771011
6	0.2360680	0.2864745	0.8751299	0.8771011
7	0.2360680	0.2639320	0.8751299	0.8749847
8	0.2446784	0.2639320	0.8747585	0.8749847
9	0.2446784	0.2553216	0.8747585	0.8747033
10	0.2479673	0.2553216	0.8746939	0.8747033
11	0.2500000	0.2553216	0.8746752	0.8747033
12	0.2500000	0.2532889	0.8746752	0.8746794
13	0.2500000	0.2520327	0.8746752	0.8746727
14	0.2507764	0.2520327	0.8746723	0.8746727
15	0.2507764	0.2515528	0.8746723	0.8746718
16	0.2510730	0.2515528	0.8746719	0.8746718
17	0.2510730	0.2513695	0.8746719	0.8746718
18	0.2510730	0.2512563	0.8746719	0.8746718
19	0.2511430	0.2512563	0.8746718	0.8746718

Tabulka 5.3: Metoda zlatého řezu.

Kapitola 6

Řešení obyčejných diferenciálních rovnic

Většina fyzikálních zákonů je formulována ve formě diferenciálních rovnic. Jsou to buď parciální nebo obyčejné diferenciální rovnice. V této kapitole se budeme zabývat obyčejnými diferenciálními rovnicemi. Tyto rovnice nejčastěji popisují závislost fyzikálních veličin na čase. Hlavním představitelem těchto rovnic ve fyzice jsou rovnice pohybové, které popisují pohyb těles pod vlivem vzájemných a vnějších sil. Tyto rovnice není často možné analyticky řešit a tedy třeba použít metod numerické matematiky. Jako příklad lze uvést problém n těles, která na sebe gravitačně působí. Tento problém není pro více než dvě tělesa analyticky řešitelný.

V těchto skriptech se omezíme se na diferenciální rovnice prvního řádu, ve kterých je derivace přímo vyjádřena. Hledáme tedy reálnou funkci $y(x)$, která splňuje následující rovnici

$$\frac{\mathrm{d}y(x)}{\mathrm{d}x} = f(x, y) \tag{6.1}$$

Funkcí které splňují tuto rovnici existuje většinou nekonečně mnoho. Jedno konkrétní řešení určíme tzv. *počáteční podmínkou*, které je nezbytnou součástí zadání úlohy. Počáteční podmínka má následující tvar

$$y(x_0) = y_0 \tag{6.2}$$

kde x_0 a y_0 jsou zadané hodnoty.
Příkladem diferenciální rovnice s počáteční podmínkou je např.

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -2(y - x) + 1 \qquad y(0) = 1 \tag{6.3}$$

Řešením této rovnice je $y(x) = e^{-2x} + x$
Všechny metody které si ukážeme, lze jednoduše zobecnit i na soustavy diferenciálních rovnic. Soustava dvou rovnic pro dvě neznámé funkce $y_1(x)$, $y_2(x)$ má tvar

$$\frac{\mathrm{d}y_1}{\mathrm{d}x} = f_1(x, y_1, y_2) \qquad y_1(x_0) = y_{10} \tag{6.4}$$

$$\frac{\mathrm{d}y_2}{\mathrm{d}x} = f_2(x, y_1, y_2) \qquad y_2(x_0) = y_{20} \tag{6.5}$$

Podobně definujeme soustavu více rovnic.

Pokud je třeba řešit diferenciální rovnici, ve které vystupuje vyšší derivace, lze zavedením další proměnné převézt tuto rovnici na soustavu rovnic prvního řádu. Například pohybovou rovnici

$$\frac{\mathrm{d}^2 r}{\mathrm{d}t^2} = F(t, r, r') \tag{6.6}$$

lze zavedením rychlosti převézt na soustavu

$$\frac{\mathrm{d}r}{\mathrm{d}t} = v \tag{6.7}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = F(t, r, v) \tag{6.8}$$

a tuto soustavu již můžeme řešit některou z dále uvedených metod.
Při numerickém řešení diferenciální rovnice vyjdeme z bodu x_0 ve kterém máme zadánu počáteční podmínku. Poté se budeme snažit najít řešení v dalších bodech x_i. V těchto bodech získáme odhady přesného řešení y_i. Pokud k výpočtu nové hodnoty použijeme pouze jednu předchozí hodnotu, tj.

$$y_{i+1} = F(x_i, y_i, x_{i+1}) \tag{6.9}$$

nazývá se tato metoda *jednokroková*. Pokud použijeme n předchozích bodů

$$y_{i+1} = F(x_{i+1}, x_i, y_i, x_{i-1}, y_{i-1}, \ldots, x_{i-n+1}, y_{i-n+1}) \tag{6.10}$$

nazývá se tato metoda *mnohokroková*, přesněji n-kroková.
Vzdálenost dvou sousedních bodů budeme označovat $h \equiv x_{i+1} - x_i$. Pro zjednodušení zavedeme také značení $f_i \equiv f(x_i, y_i)$.

6.1 Chyby

Námi vypočítané hodnoty řešení y_i nejsou samozřejmě zcela přesné, ale jsou zatíženy chybou. U diferenciálních rovnic mluvíme o třech druzích chyb. Přesnost výsledku vyjadřuje tzv. *akumulovaná diskretizační chyba*. Tato chyba udává konečný rozdíl numerického řešení y_i a přesného řešení $y(x_i)$ tj.

$$e_n = y_n - y(x_n) \tag{6.11}$$

Akumulovaná chyba vzniká jakou součet dílčích chyb v jednotlivých krocích řešení. Chybě, která vznikne v jednom kroku se říká *lokální diskretizační chyba*. Tyto chybu definujeme jako chybu našeho přibližného vzorce po dosazení přesného řešení. Např. chyba vzorce (6.9) je

$$L(y(x), h) \equiv y(x + h) - F(x, y(x), x + h) \tag{6.12}$$

Tuto chybu většinou upravujeme rozvojem do Taylorovy řady. Pokud rozvoj chyby začíná mocninou h^{p+1}

$$L(y(x), h) = O(h^{p+1}) \tag{6.13}$$

říkáme, že metoda má řád p.

6.2 Rungovy-Kuttovy metody

6.2.1 Eulerova metoda

Nejjednodušší metodou na řešení diferenciálních rovnic je Eulerova metoda. Pokud v naší diferenciální rovnici (6.1) nahradíme derivaci přibližným vzorcem dostaneme

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) \tag{6.14}$$

Když vyjádříme y_{i+1} dostaneme Eulerovu metodu

$$y_{i+1} = y_i + hf(x_i, y_i) \tag{6.15}$$

Teto vzorec nám umožňuje při znalosti řešení v x_i vypočítat řešení v x_{i+1}, jedná se tedy o jednokrokovou metodu.
Pokusme se odhadnou lokální diskretizační chybu. Podle definice chyby je

$$L = y(x + h) - y(h) - hf(x, y(x)) \tag{6.16}$$

Dosadíme rozvoj funkce $y(x)$

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \dots \tag{6.17}$$

využijeme toho, že $y'(x) = f(x, y(x))$ a dostaneme po úpravě následující vyjádření lokální chyby

$$L = \frac{h^2}{2}y''(x) + \dots \tag{6.18}$$

V tomto výrazu je druhá mocnina h a Eulerova metoda je tedy metoda prvního řádu.

6.2.2 Modifikace Eulerovy metody

Eulerova metoda je velice jednoduchá, ale bohužel velice nepřesná. Hlavní nepřesnost je způsobena tím, že při kroku z x_i do x_{i+1} používáme na celém intervalu konstantní hodnotu derivace $f(x_i, y_i)$. Tato hodnota by se však správně v průběhu kroku mění. Tuto chybu se snaží odstranit modifikace Eulerovy metody.
První modifikace se snaží vystihnou změnu derivace tím, že místo derivace na začátku intervalu použije derivaci uprostřed intervalu. Doprostřed intervalu se dostaneme obyčejnou Eulerovou metodou s poloviční délkou kroku. Výsledný vzorec je

$$y_{n+1} = y_n + hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hf(x_n, y_n)) \tag{6.19}$$

Druhá modifikace používá průměr z derivace na začátku a na konci intervalu

$$y_{n+1} = y_n + \tfrac{1}{2}h[f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))] \tag{6.20}$$

Lze ukázat, že obě tyto modifikace jsou metody druhého řádu. Jsou to opět jednokrokové metody.

6.2.3 Rungovy-Kuttovy metody

Dalšími modifikace lze Eulerovu metodu dále vylepšovat. Získáme tak tzv. *Rungovy-Kuttovy metody*. Při těchto metodách získáme několik odhadů derivace k_i v s různých bodech. Pro celý krok potom použijeme vážený průměr těchto derivací s váhami w_i tj.

$$y_{n+1} = y_n + h(w_1 k_1 + \ldots + w_s k_s) \tag{6.21}$$

Odhady derivací se vypočítají náledujícími vztahy

$$\begin{aligned} k_1 &= f(x, y) \\ k_2 &= f(x + \alpha_2 h, y + h\beta_{21} k_1) \\ k_3 &= f(x + \alpha_3 h, y + h\beta_{31} k_1 + h\beta_{32} k_2) \\ k_i &= f(x + \alpha_i h, y + h \sum_{j=1}^{i-1} \beta_{ij} k_j) \end{aligned} \tag{6.22}$$

Pokud použijeme jen jeden bod (tj. $s = 1$) dostaneme metodu prvního řádu - Eulerovu metodu. Pokud použijeme dva body můžeme dostat modifikované Eulerovy metody. První modifikaci odpovídá volba

$$w_1 = 0 \qquad w_2 = 1 \qquad \alpha_2 = \beta_{12} = \frac{1}{2} \tag{6.23}$$

druhé modifikaci odpovídá

$$w_1 = w_2 = \frac{1}{2} \qquad \alpha_2 = \beta_{12} = 1 \tag{6.24}$$

S rostoucím počtem bodů dostáváme metody vyšších řádů. Například při čtyřech odhadech derivace můžeme dostat následující metodu čtvrtého řádu

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{h}{2}k_1) \\ k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{h}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + h\Big(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\Big) \end{aligned} \tag{6.25}$$

U dosud uvedených Rungových-Kuttových metod se řád metody vždy rovnal použitému počtu odhadů derivace. Tak to ale vždy není, například při pěti odhadech derivace se dá sestrojit pouze metoda čtvrtého řádu. I z toho důvodu patří uvedená metoda čtvrtého řádu k velice oblíbeným.

6.3 Mnohokrokové metody

6.3.1 Metoda středního bodu

Metoda středního bodu je poměrně jednoduchá metoda druhého řádu. První krok v této metody se provede jednoduchou Eulerovou metodou.

$$y_1 = y_0 + hf(x_0, y_0) \tag{6.26}$$

Další kroky vycházejí z předposledního bodu x_{i-1} za použití derivace z posledního bodu x_i.

$$y_{i+1} = y_{i-1} + hf(x_i, y_i) \tag{6.27}$$

Během jednoho kroku tedy používáme derivaci uprostřed intervalu, odtud název metody.
K výpočtu nové hodnoty y_{i+1} potřebujeme znát dvě předchozí hodnoty y_i a y_{i-1}. Tato metoda již tedy není jednokroková ale je dvoukroková.

6.3.2 Mnohokrokové metody

Mnohokrokové metody využívají k výpočtu nové hodnoty řešení znalost řešení ve více předchozích bodech. Obecný tvar k-krokové metody je

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i} \qquad n = 0, 1, \ldots \tag{6.28}$$

kde α_i a β_i jsou vhodně zvolené koeficienty. Například

$$y_{n+1} = y_n + \tfrac{1}{2}h(3f_n - f_{n-1}) \tag{6.29}$$
$$y_{n+1} = y_n + \tfrac{1}{12}h(23f_n - 16f_{n-1} + 5f_{n-2}) \tag{6.30}$$
$$y_{n+1} = y_n + \tfrac{1}{2}h(f_{n+1} + f_n) \tag{6.31}$$
$$y_{n+1} = y_n + \tfrac{1}{12}h(5f_{n+1} + 8f_n - f_{n-1}) \tag{6.32}$$
$$y_{n+2} = y_n + 2hf_{n+1} \tag{6.33}$$
$$y_{n+2} = y_n + \tfrac{1}{3}h(f_{n+2} + 4f_{n+1} + f_n) \tag{6.34}$$
$$y_{n+1} = \tfrac{4}{3}y_n - \tfrac{1}{3}y_{n-1} + \tfrac{2}{3}hf_{n+1} \tag{6.35}$$
$$y_{n+1} = \tfrac{18}{11}y_n - \tfrac{9}{11}y_{n-1} + \tfrac{2}{11}y_{n-2} + \tfrac{6}{11}hf_{n+1} \tag{6.36}$$

Metody (6.30) a (6.36) jsou tříkrokové, ostatní metody jsou dvoukrokové.
Pokud k výpočtu řešení v novém bodě není třeba znát v tomto bodě derivaci nazýváme metodu *explicitní*. V opačném případě se jedná o metodu implicitní. Například metoda (6.31) je metoda *implicitní*, protože k výpočtu y_{n+1} je třeba znát $f_{n+1} = f(x_{n+1}, y_{n+1})$. Z předchozích příkladů jsou implicitní metody (6.31), (6.32), (6.34), (6.35) a (6.36).
Explicitní metody lze přímo použít k výpočtu řešení y_{n+1} v dalším bodě. U implicitních metod je třeba pro výpočet y_{n+1} využít některou z metod na hledání kořene. Většinou stačí metoda prostých iterací, někdy se používá Newtonova metoda.

6.3.3 Metody prediktor-korektor

Mnohokrokové metody se většinou používají způsobem, který se nazývá prediktor-korektor. Pracuje se vždy s dvojicí metod jedna je explicitní, druhá je implicitní. Nejprve se pomocí explicitní metody (tzv. prediktoru) získá první odhad nového řešení. Tento odhad se použije jako start iteračního procesu kterým se řeší implicitní metoda (tzv. korektor). Tímto iteračním procesem se první odhad z prediktoru zpřesní. Například

$$\text{Prediktor :} \qquad y^P_{n+1} = y_{n-2} + \tfrac{3}{2}h(f_n + f_{n-1}) \tag{6.37}$$
$$f^P_{n+1} = f(x_{n+1}, y^P_{n+1}) \tag{6.38}$$
$$\text{Korektor :} \qquad y^C_{n+1} = y_n + \tfrac{1}{2}h(f^P_{n+1} + f_n) \tag{6.39}$$

První odhad řešení y^P_{n+1} se vypočítá prediktorem (6.37). V tomto bodě se vypočítá hodnota derivace (6.38). Odhad řešení se zpřesní korektorem (6.39). Výpočty (6.38) a (6.39) se opakují, dokud nedosáhneme požadované přesnosti řešení korektoru.

Dvojice prediktor a korektor se volí tak, aby řád prediktoru byl o jedna menší než řád korektoru. Metody prediktor-korektor se ještě vylepšují přidáním takzvaného modifikátoru, např. Hammingova metoda

$$\text{Pred}: \quad y_{n+1}^P = y_{n-3} + \tfrac{4}{3}h(2f_n - f_{n-1} + 2f_{n-2}) \tag{6.40}$$

$$\text{Mod}: \quad y_{n+1}^M = y_{n+1}^P + \tfrac{112}{121}(y_n^C - y_n^P) \tag{6.41}$$

$$f_{n+1}^M = f(x_{n+1}, y_{n+1}^M) \tag{6.42}$$

$$\text{Kor}: \quad y_{n+1}^C = \tfrac{1}{8}(9y_n - y_{n-2}) + \tfrac{3}{8}h(f_{n+1}^M + 2f_n - f_{n-1}) \tag{6.43}$$

$$\text{Mod}: \quad y_{n+1} = y_{n+1}^C + \tfrac{9}{121}(y_{n+1}^P - y_{n+1}^C) \tag{6.44}$$

$$f^{n+1} = f(x_{n+1}, y_{n+1}) \tag{6.45}$$

Vícekrokové metody jsou většinou rychlejší než metody jednokrokové. Mají však také několik nevýhod. První nevýhodou je jejich obtížné startování. Než začneme vícekrokovou metodu používat je třeba znát řešení v několika bodech. Počáteční podmínka nám však udává řešení pouze v jednom bodě. Potřebná řešení v dalších bodech je tedy třeba získat vhodnou jednokrokovou metodou. Další nevýhodou je obtížná změna velikosti kroku. Při ní je totiž třeba zpětně dopočítat hodnoty řešení v chybějících bodech, a to buď interpolací nebo opět jednokrokovou metodou.

6.4 Příklad - pohyb planety

Řešme klasický problém pohybu tělesa v gravitačním poli nehybného centrálního tělesa, například pohyb planety okolo slunce. Tato úloha je téměř řešitelná analyticky (viz Keplerovy zákony), ale její zobecnění na více těles srovnatelné váhy již není.
Celou úlohu omezíme na pohyb v rovině. Mějme nehybné těleso o hmotnosti M v počátku souřadnic. Okolo tohoto tělesa se pohybuje druhé těleso o hmotnosti m, které se v daný okamžik nachází v bodě o souřadnicích $\boldsymbol{r} = [r_x, r_y]$ a má rychlost $\boldsymbol{v} = [v_x, v_y]$. Na toto tělesa působí gravitační síla

$$\boldsymbol{F} = \kappa \frac{mM}{r^3}\boldsymbol{r} \tag{6.46}$$

Podle pohybového zákona je zrychlení úměrné působící síle

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \frac{\boldsymbol{F}(\boldsymbol{r})}{m} \tag{6.47}$$

Tato pohybová rovnice je rovnice druhého řádu, zavedením rovnice pro rychlost převedeme tuto rovnici na soustavu rovnic prvního řádu. Získáme soustavu

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{v} \tag{6.48}$$

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \frac{\boldsymbol{F}(\boldsymbol{r})}{m} \tag{6.49}$$

Po rozepsání do složek a dosazení výrazu pro sílu dostaneme soustavu čtyř diferenciálních rovnic pro čtyři neznámé funkce $r_x(t), r_y(t), v_x(t), v_y(t)$

$$r_x' = v_x \tag{6.50}$$

$$r_y' = v_y \tag{6.51}$$

$$v_x' = F_x/m = \kappa M(r_x^2 + r_y^2)^{-3/2} r_x \tag{6.52}$$

$$v_y' = F_y/m = \kappa M(r_x^2 + r_y^2)^{-3/2} r_y \tag{6.53}$$

Apostrofem značíme derivace podle času t. Přejdeme nyní ke značení, které jsme používali v předchozích kapitolách. Hledané funkce označíme jako vektor $\boldsymbol{y}(t)$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \equiv \begin{bmatrix} r_x \\ r_y \\ v_x \\ v_y \end{bmatrix} \tag{6.54}$$

jeho derivace potom je

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}x} = \begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \end{bmatrix} = \begin{bmatrix} y_3 \\ y_4 \\ \kappa M(y_1^2+y_2^2)^{-3/2} y_1 \\ \kappa M(y_1^2+y_2^2)^{-3/2} y_2 \end{bmatrix} = \boldsymbol{f}(x,\boldsymbol{y}) \tag{6.55}$$

K této soustavě rovnic je třeba ještě zadat počáteční podmínky - tj. počáteční polohu a rychlost tělesa. Soustavu můžeme řešit libovolnou z dříve probraných metod, viz program `planeta.f90`.

6.5 Cvičení

1. Naprogramujte Eulerovu metodu a její modifikaci. Řešte rovnici (6.3) a srovnejte přesnost řešení v $x = 1$.

2. Naprogramujte Rungovu-Kuttovu metodu 4. řádu pro soustavu diferenciálních rovnic. Metodu použijte na řešení soustavy rovnic (tzv. Lorentzův atraktor)

$$\begin{aligned} x' &= -10x + 10y \\ y' &= -xz + 40x - y \\ z' &= xy - \tfrac{8}{3}z \end{aligned}$$

   nakreslete graf řešení.

3. Předchozí program použijte na simulaci pohybu planet.

4. Nakreslete řešení Van der Polovy rovnice $y'' - \mu(1-y^2)y' + y = 0$, počáteční podmínka je $y(0) = 2$, $y'(0) = 0$. Parametr μ je libovolné reálné číslo (např. $\mu = 1$).

Kapitola 7

Parciální diferenciální rovnice

Ve fyzice parciální diferenciální rovnice (PDR) popisují chování veličiny, která kromě času závisí také na prostorových proměnných. Obecná parciální diferenciální rovnice r-tého řádu udává vztah mezi parciálními derivacemi hledané funkce

$$F(x_1, \ldots, x_n, u, \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, \ldots, \frac{\partial^r u}{\partial x_n^r}) = 0 \tag{7.1}$$

My se omezíme na lineární PDR druhého řádu. Pro dvě proměnné mají tyto rovnice obecně tvar

$$a_1 \frac{\partial^2 u}{\partial x_1^2} + a_2 \frac{\partial^2 u}{\partial x_2^2} + a_{12} \frac{\partial^2 u}{\partial x_1 \partial x_2} + b_1 \frac{\partial u}{\partial x_1} + b_2 \frac{\partial u}{\partial x_1} + c = 0 \tag{7.2}$$

kde a_i, b_i a c jsou zadané funkce x_1 a x_2.
Lineární PDR druhého řádu lze rozdělit do tří skupin - na rovnice parabolické, hyperbolické a eliptické. Detaily tohoto dělení se zabývat nebudeme. Ukážeme si jak numericky řešit vybrané zástupce těchto skupin.

7.1 Parabolické rovnice

Budeme se zabývat rovnicí

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} \tag{7.3}$$

tato rovnice určuje chování funkce $u(t, x)$, která závisí na dvou proměnných. První proměnná t mívá význam času, druhá x bývá prostorová souřadnice. Předchozí rovnice lze snadno zobecnit pro více prostorových proměnných např.

$$\frac{\partial u}{\partial t} = D\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) \tag{7.4}$$

Příkladem parabolické rovnice je rovnice vedení tepla

$$\frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) = \rho c \frac{\partial T}{\partial t} \tag{7.5}$$

tato rovnice popisuje časový vývoj teploty $T(t, x)$ uvnitř nekonečné stěny, k je tepelná vodivost materiálu, c je tepelná vodivost a ρ je hustota.

Jiným příkladem je rovnice difuze pro hustotu částic $n(t,x)$

$$\frac{\partial n}{\partial x} - \frac{1}{k^2}\frac{\partial n}{\partial t} = 0 \tag{7.6}$$

kde k je ???.
Parabolická rovnice nám vlastně říká, že tam kde je druhá derivace záporná, hledaná funkce v čase klesá a naopak. Dochází tak k vyhlazování průběhu funkce - teploty se vyrovnávají, hustoty částic v různých bodech se srovnávají.
Parabolické rovnice většinou řešíme na omezeném intervalu. Budeme předpokládat, že se jedná o interval $x \in \langle 0, L\rangle$. Např. při řešení rovnice vedení tepla, bude $x = 0$ reprezentovat vnitřní povrch stěny a $x = L$ vnější povrch stěny.
Abychom mohli rovnici řešit je třeba znát hodnoty hledané funkce v počátečním čase $t = 0$ tzv. *počáteční podmínku*. Například je třeba zadat počáteční hodnoty ve stěně. Počáteční podmínka má obecně tvar

$$u(0,x) = p(x) \tag{7.7}$$

kde $p(x)$ je známá funkce.
Zadání počáteční podmínky však pro výpočet nedostačuje. Je třeba také vědět co se odehrává na okrajích studované oblasti. Je tedy třeba zadat chování funkce pro $x = 0$ a $x = L$, tomuto říkáme *okrajové podmínky*. Např. je třeba zadat teplotu na vnitřním a vnějším povrchu stěny.
Okrajová podmínka může mít dva základní tvary. První možností je přímo zadání hodnot na hranici oblasti tj.

$$u(t,0) = g_0(t) \qquad \text{nebo} \qquad u(t,L) = g_1(t) \tag{7.8}$$

takovéto podmínce se říká *Dirichletova podmínka*. Druhou možností je zadání prostorové derivace funkce tj.

$$\frac{\partial u}{\partial x}(t,0) = g_2(t) \qquad \text{nebo} \qquad \frac{\partial u}{\partial x}(t,L) = g_3(t) \tag{7.9}$$

takovéto podmínce se říká *Neumannova podmínka*. Například u rovnice vedení tepla této podmínce odpovídá zadání tepelného toku, např. nulová derivace odpovídá dokonale izolovanému povrchu stěny.
Naším cílem tedy bude nalezení funkce $u(t,x)$ pro $x \in \langle 0, L\rangle$ a $t \in \langle 0, \infty)$, která splňuje rovnici (7.3), počáteční podmínku (7.7) a okrajové podmínky (7.8) či (7.9).

Příklad

Řešme diferenciální rovnici

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{7.10}$$

na intervalu $\langle 0, L\rangle$ s okrajovými podmínkami

$$u(t,0) = 0 \qquad u(t,L) = 1 \tag{7.11}$$

a s počáteční podmínkou

$$u(0,x) = x + \sin\left(\frac{\pi x}{L}\right) \tag{7.12}$$

Snadno se přesvědčíme, že počáteční a okrajové podmínky jsou v souladu tj. podle obojího je $u(0,0) = 0$ a $u(0,L) = 1$. Dosazením se lze přesvědčit, že řešením dané rovnice s těmito podmínkami je

$$u(t,x) = x + \sin\left(\frac{\pi x}{L}\right) e^{-\frac{D\pi^2}{L^2}t} \tag{7.13}$$

Pro $T \to \infty$ se toto řešení plíží k ustálenému stavu

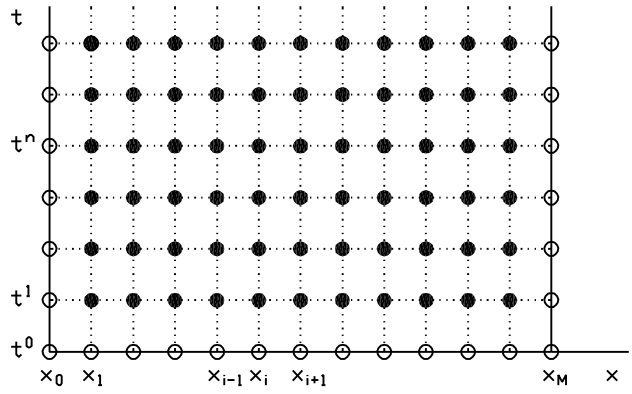
$$u(\infty, x) = x \tag{7.14}$$

7.2 Metoda sítí pro parabolické rovnice

Ukážeme si základní metodu na numerické řešení PDR, tzv. *metodu sítí* přesněji zvanou metodu konečných diferencí. V této metodě se místo spojité funkce $u(t,x)$ hledají pouze odhady řešení v konečném počtu bodů. Tyto body tvoří v oblasti řešení síť, odtud je název metody. Existuje celá řada druhů sítě, my se omezíme na nejjednodušší případ pravoúhlé rovnoměrné sítě. Body této sítě mají souřadnice $[t^n, x_i]$. Časové okamžiky t^n jsou rovnoměrně rozmístěny s časovým krokem τ a prostorové body jsou rovnoměrně rozmístěny s krokem h tj.

$$x_j = jh \qquad j = 0, 1, \ldots, M \qquad h = L/M \tag{7.15}$$

$$t^n = n\tau \qquad n = 0, 1, \ldots, N \tag{7.16}$$



Obrázek 7.1: Konstrukce rovnoměrné ortogonální sítě.

Místo spojité funkce $u(t,x)$ tedy budeme hledat odhady tohoto řešení u_i^n, tak aby v ideálním případě platilo $u_i^n = u(t^n, x_i)$. Tento vztah nebude platit přesně, protože numerická metoda bude opět zatížena chybou. Místo diferenciálních rovnic pro $u(t,x)$ je třeba získat tzv. *diferenční rovnici* pro u_i^n. Tuto diferenční rovnici získáme tak, že parciální derivace v diferenciální rovnici nahradíme některým přibližným vzorcem z kapitoly ???.

7.2.1 Explicitní metoda

V nejjednodušší tzv. explicitní metodě použijeme následující přibližné vzorce

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\tau} \tag{7.17}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} \tag{7.18}$$

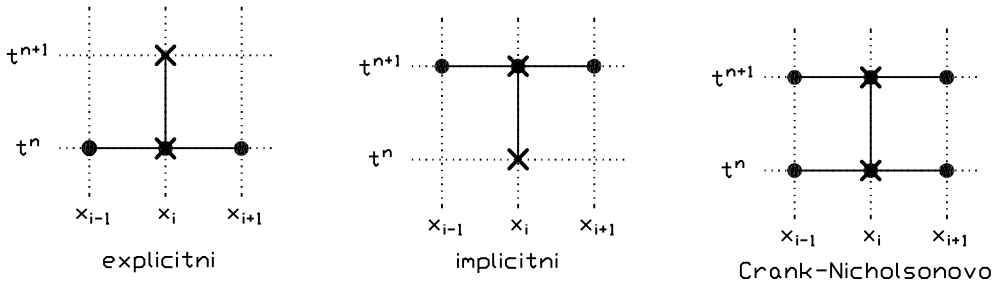
Dosazením do (7.3) dostaneme diferenční rovnici pro u_i^n

$$\frac{u_i^{n+1} - u_i^n}{\tau} = D\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} \tag{7.19}$$

Z této rovnice můžeme vyjádřit u_i^{n+1}

$$u_j^{n+1} = u_j^n + \frac{D\tau}{h^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) \tag{7.20}$$

Pokud známe řešení u_j^n v časový okamžik t^n, můžeme z této rovnice vypočítat řešení u_j^{n+1} v následující časový okamžik t^{n+1}. Výpočet začneme v čase $t = t_0 = 0$, kde máme zadáno řešení počáteční podmínkou. Poté postupujeme časem a ze vztahu (7.20) vypočítáme řešení v čase $t^1, t^2, \ldots$. Vztah (7.20) však nelze použít v krajních hodnotách, protože například k výpočtu u_0^{n+1} potřebujeme znát hodnotu u_{-1}^n, která ale leží mimo studovanou oblast. Pro výpočet krajních hodnot u_0^{n+1} a u_J^{n+1} je tedy třeba použít okrajových podmínek.



Obrázek 7.2: Schemata pro řešení parabolické rovnice.

Explicitní metoda je celkem jednoduchá, má ale jedno podstatné omezení. Pokud je časový krok příliš velký metoda není tzv. *stabilní*. To se projeví po několika krocích metody, kdy se objeví na průběhu funkce zákmity, které se postupně zesilují, až řešení diverguje. Takové chování je samozřejmě nežádoucí. Aby toto chování nenastalo tj. aby metoda byla stabilní, je třeba, aby byla splněna následující podmínka

$$\frac{\tau D}{h^2} \leq \frac{1}{2} \tag{7.21}$$

Takováto podmínka, která zajišťuje stabilitu metody, se nazývá podmínka stabilita. Metody se z tohoto hlediska dělí na tři druhy - metody stabilní, nestabilní a podmíněně stabilní. Metoda se nazývá podmíněně stabilní pokud potřebuje k zajištění stability nějakou podmínku, jako výše uvedená metoda.
Podmínka (7.21) omezuje velikost časového kroku. Tato podmínka je bohužel dost přísná. Pokud je například tato podmínka přesně splněna a my chceme zmenšit prostorový krok na polovinu je třeba zmenšit časový krok čtyřikrát, v důsledku toho vzroste časová náročnost osmkrát.

7.2.2 Implicitní metoda

Implicitní metoda vylepšuje explicitní metodu tím, že odstraňuje podmínku stability. Je tedy stabilní, nikoli jen podmíněně stabilní. Tato metoda používá jinou aproximaci druhé prostorové derivace než je (7.18) a to

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{h^2} \tag{7.22}$$

Použijeme tedy stejný derivační vzorec, ale vztažený k jinému časovému okamžiku. Dostaneme tak

$$\frac{u_i^{n+1} - u_i^n}{\tau} = D\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{h^2} \tag{7.23}$$

Z této rovnice nemůžeme přímo vypočítat řešení v novém čase t^{n+1} protože k výpočtu u_i^{n+1} je třeba již znát řešení v sousedních bodech u_{i+1}^{n+1} a u_{i-1}^{n+1}. Rovnice

(7.23) představuje tedy vztah mezi třemi hledanými hodnotami v novém čase t^{n+1} ve třech sousedních bodech.
Označme si $\alpha \equiv \frac{\tau D}{h^2}$, z předchozího vztahu dostaneme

$$u_j^{n+1} - u_j^n = \alpha(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) \tag{7.24}$$

$$-\alpha u_{j-1}^{n+1} + (1+2\alpha)u_j^{n+1} - \alpha u_{j+1}^{n+1} = u_j^n \tag{7.25}$$

Pro různá i máme různé rovnice, dohromady tvoří tyto rovnice soustavu lineárních rovnic pro neznáme u_i^{n+1}. V maticovém zápisu

$$\begin{bmatrix} 1+2\alpha & -\alpha & & & & & \\ -\alpha & 1+2\alpha & -\alpha & & & & \\ & -\alpha & 1+2\alpha & & & & \\ & & & \ddots & & & \\ & & & & 1+2\alpha & -\alpha & \\ & & & & -\alpha & 1+2\alpha & \end{bmatrix} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{J-1}^{n+1} \\ u_J^{n+1} \end{bmatrix} = \begin{bmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ u_{J-1}^n \\ u_J^{nq} \end{bmatrix} \tag{7.26}$$

Řešením této soustavy dostaneme řešení v novém časovém okamžiku t^{n+1}.

7.2.3 Crankovo-Nicholsonovo schéma

Předchozí implicitní metoda je vždy stabilní. Z hlediska stability lze v této metodě volit libovolně velký krok. Pokud však zvolíme větší krok diskretizační chyba se zvětší. Je tedy vhodné použít metodu u které poroste chyba co nejpomaleji. Pokud se podíváme do tabulky 4.4, zjistíme, že chyba implicitní i explicitní metody je $O(\tau, h^2)$. Chyba je tedy úměrná pouze první mocnině časového kroku, to je způsobeno tím, že obě metody jsou nesymetrické v čase. Tuto asymetrii odstraňuje Crankova-Nicholsonova metoda. Tato metoda místo prostorové derivace v čase t^n (jako explicitní metoda) nebo v čase t^{n+1} (jako implicitní metoda) používá průměr z obou možností tj.

$$\frac{u_i^{n+1} - u_i^n}{\tau} = D\frac{1}{2}\left[\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{h^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}\right] \tag{7.27}$$

neboli

$$u_j^{n+1} - u_j^n = \frac{\alpha}{2}\left[(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}) + (u_{j+1}^n - 2u_j^n + u_{j-1}^n)\right] \tag{7.28}$$

Z této rovnice opět nelze přímo vypočítat řešení v novém čase, rovnici tedy nejprve upravíme

$$-\frac{\alpha}{2}u_{j-1}^{n+1} + (1+\alpha)u_j^{n+1} - \frac{\alpha}{2}u_{j+1}^{n+1} = u_j^n + \frac{\alpha}{2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) \tag{7.29}$$

Máme tedy opět soustavu lineárních rovnic pro řešení v čase t^{n+1}. V maticovém zápisu je tato soustava

$$\begin{bmatrix} 1+\alpha & -\frac{\alpha}{2} & & \\ -\frac{\alpha}{2} & 1+\alpha & -\frac{\alpha}{2} & \\ & -\frac{\alpha}{2} & 1+\alpha & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} u_0^n + \frac{\alpha}{2}(u_1^n - 2u_0^n) \\ u_1^n + \frac{\alpha}{2}(u_2^n - 2u_1^n + u_0^n) \\ u_2^n + \frac{\alpha}{2}(u_3^n - 2u_2^n + u_1^n) \\ \vdots \end{bmatrix} \tag{7.30}$$

Crankova-Nicholsonova metoda je nepodmíněně stabilní stejně jako implicitní metoda. Její chyba je ale menší je řádu $O(\tau^2, h^2)$. Tato metoda je téměř stejně časově náročná jako implicitní metoda, je ale přesnější, a proto bychom jí měli dávat přednost.

7.3 Hyperbolické rovnice

Hyperbolická rovnice, kterou se nyní budeme zabývat má tvar

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{7.31}$$

nebo ve dvou prostorových proměnných

$$\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \tag{7.32}$$

Opět hledáme funkci $u(t, x)$, která je funkcí času t a prostorové souřadnice x. Rovnice (7.31) se často nazývá vlnová rovnice. Ve fyzice se totiž touto rovnicí popisuje šíření vlnění, například na napnuté struně. Funkce $u(t, x)$ potom reprezentuje výchylku struny z rovnovážné polohy v čase t a v místě x. Konstanta a představuje rychlost šíření vlny po struně.
Vlnová rovnice pro strunu je pohybová rovnice - říká nám, že zrychlení úseku struny ($\frac{\partial^2 u}{\partial t^2}$) je úměrné síle, která na něj působí. Tam, kde má výchylka struny zápornou druhou derivaci, je struna urychlována směrem dolů a naopak.
Pro řešení rovnice je třeba zadat oblast řešení, okrajové a počáteční podmínky stejně jako u parabolické rovnice. Zadání počátečních podmínek se ale liší od parabolických rovnic. Na počátku nestačí zadat hodnoty funkce $u(t, x)$, např. výchylku struny. Je třeba též zadat hodnotu derivace $\frac{\partial u}{\partial t}$, tj. např. rychlost pohybu struny na počátku. Počáteční podmínky mají tedy tvar

$$u(0, x) = p(x) \tag{7.33}$$

$$\frac{\partial u}{\partial t}(0, x) = q(x) \tag{7.34}$$

7.4 Metoda sítí pro hyperbolické rovnice

Konstrukce sítě je stejná jako v případě parabolické rovnice.

7.4.1 Explicitní metoda

Nejjednodušší je opět explicitní metoda. Při této metodě použijeme následující aproximace derivací

$$\frac{\partial^2 u}{\partial t^2} = \frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\tau^2} \tag{7.35}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2} \tag{7.36}$$

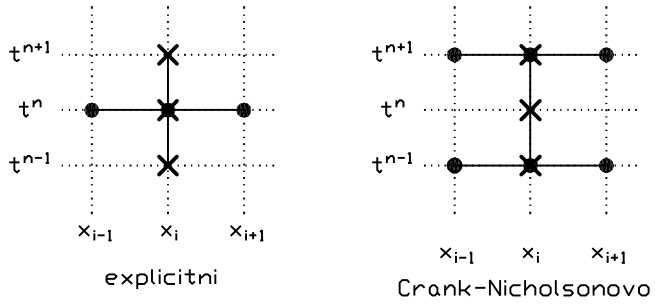
Dosazením do (7.31) dostaneme diferenční rovnici pro u_i^n

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\tau^2} = a^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} \tag{7.37}$$

Z této rovnice můžeme vyjádřit u_i^{n+1}

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + \left(\frac{a\tau}{h} \right)^2 (u_{j+1}^n - 2u_j^n + u_{j-1}^n) \tag{7.38}$$

Tato rovnice nám umožňuje vypočítat řešení v čase t^{n+1} z řešení v čase t^n a t^{n-1}. Oproti parabolickým rovnicím nestačí tedy znát jeden předchozí krok, ale je třeba



Obrázek 7.3: Schemata pro řešení hyperbolické rovnice.

znát předchozí kroky dva. To je problém na začátku, kdy potřebujeme inicializovat první dvě řešení pomocí počátečních podmínek. První časovou vrstvu získáme přímo z počáteční podmínky

$$u_i^0 = p(x_i) \tag{7.39}$$

Řešení v druhém čase získáme pomocí počáteční hodnoty a derivace za předpokladu, že derivace (rychlost struny) je konstantní

$$u_i^1 = p(x_i) + \tau q(x_i) \tag{7.40}$$

Tato explicitní metoda pro hyperbolické rovnice je pouze podmíněně stabilní. K zajištění stability je třeba aby platilo tzv. *Courantovo-Friedrichsovo-Lewyho kritérium*

$$\tau \leq \frac{h}{|a|} \tag{7.41}$$

Díky tomu, že v této podmínce je pouze první mocnina h, není tato podmínka tak omezující jako podmínka pro explicitní metodu u parabolických rovnic.

7.4.2 Crankovo-Nicholsonovo schéma

Dokonalejší než explicitní metoda jsou opět metody implicitní, protože odstraňují nutnost podmínky stability. Základní implicitní metoda, která je symetrická v čase je Crankova-Nicholsonova metoda. Tato místo aproximace prostorové derivace v čase t^n používá průměr z derivací v časech t^{n+1} a t^{n-11}. Místo (7.36) tedy použijeme

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{2}\left(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} + \frac{u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}}{h^2}\right) \tag{7.42}$$

Výsledné schéma tedy je

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\tau^2} = a^2\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{2h^2} + a^2\frac{u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}}{2h^2} \tag{7.43}$$

Opět jsme získali soustavu lineárních rovnic. Zavedeme značení

$$\sigma \equiv \frac{a\tau}{h} \tag{7.44}$$

a soustavy upravíme

$$-\frac{\sigma^2}{2}u_{j-1}^{n+1} + (1+\sigma^2)u_j^{n+1} - \frac{\sigma^2}{2}u_{j+1}^{n+1} = 2u_j^n - u_j^{n-1} + \frac{\sigma^2}{2}(u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}) \tag{7.45}$$

Matice této soustavy je velice podobná matici soustavy pro Crankovo-Nicholsonovo schéma u parabolických rovnic.

7.5 Eliptické rovnice

Budeme se zabývat eliptickými rovnicemi ve tvaru

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{7.46}$$

V případě eliptických rovnic hledáme funkci $u(x, y)$, která je funkcí dvou prostorových proměnných x a y. Nemáme zde žádnou proměnnou s významem času. Funkce $f(x, y)$ je známá zadaná funkce. Takováto eliptická rovnice se také nazývá Poissonova rovnice. V případě, že pravá strana rovnice je nulová mluvíme o Laplaceově rovnici

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{7.47}$$

Jako fyzikální příklad lze uvést rovnici pro elektrostatický potrnciál φ

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = -\frac{\varrho}{\varepsilon_0} \tag{7.48}$$

kde $\varrho(x, y)$ je hustota náboje. Eliptické rovnice také vznikají jako stacionární řešení parabolické rovnice. Pokud se totiž řešení (7.4) již nemění, tj. časová derivace je nulová, dostaneme rovnici (7.47). Tak dostaneme např. rovnici pro stacionární rozložení teploty $T(x, y)$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = q(x, y) \tag{7.49}$$

kde $q(x, y)$ je dané rozložení tepelných zdrojů.
Díky tomu, že u eliptických rovnic není proměnná s významem času, nejsou zde žádné počáteční podmínky. Je třeba zadat oblast na které budeme hledat řešení. V našem případě dvou prostorových proměnných to bude část roviny. My se omezíme na nejjednodušší případ, kdy oblastí řešení je obdélník. Na hranicích oblasti řešení je třeba zadat okrajové podmínky. Okrajové podmínky mohou být několika druhů stejně jako u parabolických a hyperbolických rovnic.

7.6 Metoda sítí pro eliptické rovnice

V oblasti řešení sestrojíme nejprve síť. Naše síť bude opět pravoúhlá a rovnoměrná. Pro jednoduchost budeme předpokládat, že krok sítě je v obou směrech stejný a má velikost h

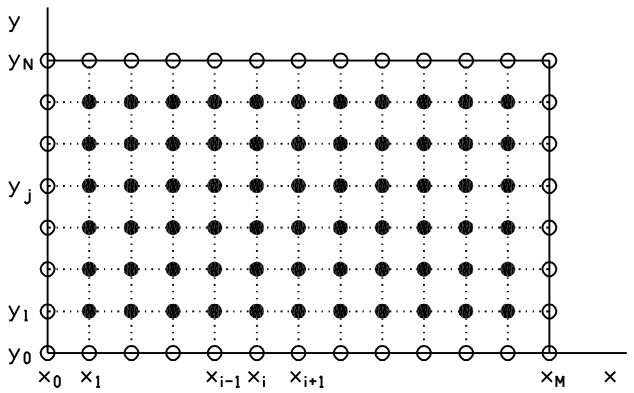
$$x_i = ih \qquad i = 0, 1, \ldots, M \tag{7.50}$$

$$y_j = jh \qquad j = 0, 1, \ldots, N \tag{7.51}$$

Místo spojitého řešení $u(x, y)$ budeme opět hledat jeho odhady $u_{i,j}$ tak aby přibližně bylo $u(x_i, y_j) = u_{i,j}$.
Derivace v rovnici (7.46) nahradíme přibližnými vzorci

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \tag{7.52}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \tag{7.53}$$



Obrázek 7.4: Konstrukce rovnoměrné ortogonální sítě.

a dostaneme

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = f_{ij} \tag{7.54}$$

po malé úpravě máme

$$u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} - 4u_{i,j} = h^2 f_{ij} \tag{7.55}$$

Získali jsme soustavu lineárních rovnic pro neznámé hodnoty u_{ij}. Hledané hodnoty u_{ij} tvoří matici. Abychom mohli vyjádřit soustavu (7.55) v maticovém zápisu je třeba hodnoty u_{ij} seřadit do vektoru. Lze postupovat po řádcích, tak dostaneme vektor

$$\boldsymbol{u} = (u_{01}, u_{02}, \ldots, u_{0N}, u_{11}, u_{12}, \ldots, u_{1N}, u_{20}, \ldots, u_{MN}) \tag{7.56}$$

Pokud máme hledané proměnné takto seřazeny je matice soustavy tzv. blokově třídiagonální matice. Příklad takové matice je na obr. 7.5 (kvůli přehlednosti tečky značí nulu). V každém řádku této matice je nejvýše 5 nenulových prvků, které odpovídají pěti členům rovnice (7.55). Na diagonále je vždy 4 odpovídající prostřednímu prvku u_{ij}. Hodnoty -1 odpovídají sousedním prvků. U rovnic, které odpovídají vnitřním uzlům sítě a které tedy mají čtyři sousedy, jsou v řádku čtyři -1. U rovnic, které odpovídají hraničním resp. rohovým bodům jsou -1 jen tři resp. dvě.
Blokově třídiagonální matice je příkladem tzv. řídké matice. Řídké matice jsou matice, jejichž většina prvků je nulová. Řešení soustavy s takovouto maticí pomocí přímých metod je zbytečně obtížné. Během řešení se totiž nulové prvky nahradí nenulovými a řídkost matice se nijak nevyužije. Pro řešení se využívají iterační metody. Jacobiova metoda pro soustavu (7.55) je velice jednoduchá

$$u_{i,j}^{(n+1)} = -\frac{h^2}{4} f_{ij} + \frac{1}{4}(u_{i,j+1}^{(n)} + u_{i,j-1}^{(n)} + u_{i+1,j}^{(n)} + u_{i-1,j}^{(n)}) \tag{7.57}$$

Gaussova-Seidelova metoda pro stejnou soustavu je

$$u_{i,j}^{(n+1)} = -\frac{h^2}{4} f_{ij} + \frac{1}{4}(u_{i,j+1}^{(n)} + u_{i,j-1}^{(n+1)} + u_{i+1,j}^{(n)} + u_{i-1,j}^{(n+1)}) \tag{7.58}$$

$$\left[\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & . & . & -1 & . & . & . & . & . & . & . & . & . & . & . \\
-1 & 4 & -1 & . & . & -1 & . & . & . & . & . & . & . & . & . & . \\
. & -1 & 4 & -1 & . & . & -1 & . & . & . & . & . & . & . & . & . \\
. & . & -1 & 4 & . & . & . & -1 & . & . & . & . & . & . & . & . \\
\hline
-1 & . & . & . & 4 & -1 & . & . & -1 & . & . & . & . & . & . & . \\
. & -1 & . & . & -1 & 4 & -1 & . & . & -1 & . & . & . & . & . & . \\
. & . & -1 & . & . & -1 & 4 & -1 & . & . & -1 & . & . & . & . & . \\
. & . & . & -1 & . & . & -1 & 4 & . & . & . & -1 & . & . & . & . \\
\hline
. & . & . & . & -1 & . & . & . & 4 & -1 & . & . & -1 & . & . & . \\
. & . & . & . & . & -1 & . & . & -1 & 4 & -1 & . & . & -1 & . & . \\
. & . & . & . & . & . & -1 & . & . & -1 & 4 & -1 & . & . & -1 & . \\
. & . & . & . & . & . & . & -1 & . & . & -1 & 4 & . & . & . & -1 \\
\hline
. & . & . & . & . & . & . & . & -1 & . & . & . & 4 & -1 & . & . \\
. & . & . & . & . & . & . & . & . & -1 & . & . & -1 & 4 & -1 & . \\
. & . & . & . & . & . & . & . & . & . & -1 & . & . & -1 & 4 & -1 \\
. & . & . & . & . & . & . & . & . & . & . & -1 & . & . & -1 & 4
\end{array}\right]$$

Obrázek 7.5: Blokově třídiagonální matice.

Kapitola 8

Závěr

8.1 Numerický software

V těchto skriptech jsme se zabývali různými algoritmy numerické matematiky. V počátcích tyto algoritmy sloužily přímo k ručnímu počítání. V současné době se tyto algoritmy realizují na počítači, vniká tak numerický software. Tvorba numerického softwaru není pouhé přepsání vzorců numerické matematiky do počítače. Kvalitní numerický software musí totiž splňovat celou řadu požadavků:

- spolehlivost – přesnost, efektivita
  Software musí pro všechny vstupní hodnoty poskytovat správné výsledky, a to v co nejkratším čase.
- robusnost – blbuvzdornost
  Software musí být odolný proti zadávání špatných hodnot. Při zadání hodnot, které není schopen zpracovat musí zahlásit chybu a ne se zhroutit.
- kvalita dokumentace
  Význam a formát vstupních hodnot, které program požaduje musí být jasně popsán, stejně tak výstupní hodnoty z programu.
- univerzálnost
  Software by měl řešit pokud možno co nejširší třídu problémů.
- strojová nezávislost – přenositelnost
  Software by neměl záviset na operačním systému, případně na dalších podpůrných programech. Měl by být k dispozici ve zdrojovém tvaru a využívat pouze standardní rysy jazyka.
- programovací styl
  Zdrojový kód programu by se měl řídit zásadami správného programování. Minimálně by měl být dobře čitelný (pochopitelný), strukturovaný, formátovaný a komentovaný.

8.1.1 Numerické knihovny

Numerické metody zmiňované v těchto skriptech z převážné části patří ke standardním metodám. Odpovídající programy lze nalézt v řadě numerických knihoven. Za nejlepším zdroj numerického softwaru považujeme GAMS - Guide to Available Mathematical Software, který lze nalézt na adrese `http://gams.nist.gov`. Dva nejznámější komerční numerické balíky jsou:

`NAG` The Numerical Algorithms Group

`IMSL` Numerical and Statistics Libraries; fa Visual Numerics

Mezi komerční produkty také patří kniha a sada programů "Numerical Recepies" [xxx]. Tento software je dostupný v jazyce C, FORTRAN 77 i 90 a Pascal. Tato kniha získala ve fyzikální veřejnosti oblibu díky svému intuitivnímu podání, ale o kvalitě programů lze někdy pochybovat.
Mimo komerční software existuje celá řada velmi kvalitních numerických knihoven. Většina programů je napsána v jazyce FORTRAN 77, ale většinu metod lze nalézt i v jazyce C. Mezi nejznámější specializované knihovny patří:

`BLAS`	Základní operace s maticemi a vektory
`EISPACK`	Vlastní čísla a vektory matic
`FFTPACK`	Rychlá Fourierova transformace
`FISHPACK`	Eliptické parciální diferenciální rovnice
`LAPACK`	Lineární algebra (dříve `LINPACK` a `EISPACK`)
`MINPACK`	Řešení nelineárních rovnic
`ODEPACK`	Počáteční úlohy obyčejných diferenciálních rovnic
`QUADPACK`	Integrace (adaptivní i singularity)
`FN`	Speciální funkce
`RANDOM`	Generátory náhodných čísel a jejich testování

Mimo těchto specializovaných knihoven existují i soubory takovýchto knihoven:

`CMLIB`	NIST Core Math Library
`SLATEC`	Common Mathematical Library Committee

8.2 Seznam programů

Hlavní probrané numerické algorimy jsou demonstrovány přiloženými vzorovými programy v jazyce Fortran 90

- `gauss.f90` Gaussova eliminace
- `jacobi.f90` Iterační metody pro soustavu lineárních rovnic
- `interp.f90` Polynomiální interpolace
- `spline.f90` Výpočet kubického splinu
- `integ.f90` Integrace složenými Newton-Cotesovými vzorci
- `root.f90` Hledání kořene Newtonovou metodou a půlením intervalu
- `newton.f90` Newtonova metoda pro soustavu nelineárních rovnic
- `gold.f90` Metoda zlatého řezu
- `odr.f90` Srovnání Runge-Kutttových metod různého řádu
- `planeta.f90` Simulace pohybu planety
- `telesa.f90` Simulace problému n-těles
- `vlny.f90` Řešení vlnové rovnice explicitní metodou
- `vlny_impl.f90` Řešení vlnové rovnice Crank-Nicholsonovou metodou

Vzorové programy byly napsány tak, aby umožňovaly co nejsnadnější pochopení algoritmu. Nesplňují výše uvedené požadavky pro kvalitní numerický software - nejsou robusní, rychlé a dokumentované, jsou určeny pouze pro demonstraci probraných algoritmů.

8.3 Literatura

Hrach R.: Numerické metody ve fyzikální elektronice I., Skripta MFF UK, Praha 1981
Ralston A.: Základy numerické matematiky, Academia, Praha 1978
Vitásek E.: Numerické metody, Nakl. techn. lit., Praha l987
Marčuk G.I.: Metody numerické matematiky, Academia, Praha l987
Rektorys K.: Přehled užité matematiky, SNTL, Praha 1981

[**?**]