Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta – Katedra fyziky

Numerické metody ve fyzice

Diplomová práce

Vedoucí práce: doc. RNDr. Josef Blažek, CSc.

Autor: Soňa Málková

Anotace:

Práce se týká aplikace vybraných numerických metod ve fyzice. Pozornost je především věnována interpolaci funkcí, aproximaci metodou nejmenších čtverců, diferenciálním rovnicím, metodě Monte Carlo a samotnému programu MATLAB. Součástí práce jsou modelové příklady, řešené v prostředí programu MATLAB.

Abstract:

The dissertation has to do with a application of the sophisticated numerical methods in physics. The attention is devoted above all an interpolation of the functions, an approximation with the method of the smallest squares, a differential equations, a method Monte Carlo and the alone program MATLAB. A model instances solved in a medium of the program MATLAB are a constituent of the dissertation.

Prohlašuji, že svoji diplomovou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

…….. …….

Datum Podpis

Touto formou děkuji svému konzultantovi p. doc. RNDr. Josefu Blažkovi, CSc. za cenné rady a připomínky při zpracování mé diplomové práce.

Obsah

Úvod

Numerická matematika se zabývá řešením matematických problémů prostřednictvím numerických výpočtů. Je pro ní typické uplatnění výpočetní techniky.

Numerická metoda je základní pojem numerické matematiky. Jde někdy o relativně přesný postup - algoritmus, jindy o širší způsob přístupu k řešení určitého druhu numerických úloh. Základní charakteristiky každé numerické metody jsou stabilita a konvergence.

Ve skutečnosti lze jen málo problémů vzniklých matematizací reálných situací vyřešit přesně, a to i tehdy, jsou-li přesně zadána vstupní data, což také často není splněno. Pak je třeba sáhnout k numerické matematice (o to větší význam pak má přesné řešení nějakého problému v dostatečné obecnosti). K tomu ale dochází velmi zřídka, většinou je nutno řešit problém numericky. Z tohoto důvodu jsou směry výzkumu numerické matematiky určovány potřebami fyziky, chemie a ostatních exaktních vědních oborů.

Numerické metody jsou pro fyziku důležité, protože většinu fyzikálních úloh můžeme řešit pouze numericky.

Mezi základní numerické metody patří interpolace, aproximace, řešení diferenciálních rovnic nebo metoda Monte Carlo. Každá metoda je podrobně rozebrána a názorně ukázána na příkladu. Řešení některých příkladů je také znázorněno přímo v programu Matlab, který je pro řešení těchto metod ideální. Po nadefinování vstupních parametrů dané funkce nám Matlab vykreslí graf, který je možno uložit jako obrázek a dále s ním pracovat.

Matlab má uplatnění v oblastech jako je vývoj algoritmů, analýza dat a vizualizace, modelování a simulace, různé matematické výpočty nebo vědecká a inženýrská grafika.

1. Interpolace funkcí

Požadavky, podle nichž vybíráme funkci $g(x)$ jako náhradu za funkci $f(x)$, mohou být různé. Náhradu používáme za funkci, která je dána jako řešení diferenciální rovnice nebo se jedná o empirickou funkci, v jiném případě se nahradí příliš složitá funkce. Pokud chceme, aby funkce $g(x)$ měla v předepsaných bodech stejné funkční hodnoty jako má funkce $f(x)$, pak hovoříme o *interpolaci*.

Omezíme se na případ, že funkce $g(x)$ bude algebraický polynom. Existuje několik možných metod k dosažení žádané aproximace.

1.1. Přímá metoda

Nejčastěji se různé funkce nahrazují polynomy, protože polynom obsahuje tři základní operace: sčítání, odčítání a násobení. Velmi jednoduše se nalezne tzv. *interpolační polynom*. Je to algebraický polynom tvaru $P(x)=\sum_{k=0}^{n} a_k \cdot x^k$, pro který platí $P_n(x_k)=f(x_k)$, $k = 0, 1, \mathrm{K}, n$. Funkce $f(x)$ je funkce, kterou nahrazujeme interpolačním polynomem a uzly $x_0, x_1, \mathrm{K}, x_n$ jsou navzájem různé. Nalézt interpolační polynom $P_m(x)$ k funkci $f(x)$ s uzly $x_0, x_1, \mathrm{K}, x_n$ znamená určit hodnoty koeficientů a_k, $k = 0, 1, \mathrm{K}, n$. Protože některé koeficienty a_k mohou být i nulové, je interpolační polynom k funkci *f* vzhledem k uzlům nejvýše stupně *n*.

Jestliže $n=1$, tzn. máme dva uzly x_0, x_1, je interpolační polynom lineární funkce. Grafem je přímka, která prochází body $[x_0, f(x_0)], [x_1, f(x_1)]$. Hovoříme pak o lineární interpolaci. V případě $n=2$ jsou dány tři body $[x_0, f(x_0)], [x_1, f(x_1)], [x_2, f(x_2)]$, kterými bude procházet interpolační polynom $P_2(x)$. Neleží-li tyto tři body na přímce, pak grafem interpolačního polynomu $P_2(x)$ je kvadratická parabola (proto se hovoří o kvadratické interpolaci). Podobně je tomu i pro vyšší počet uzlových bodů.

1.2. Lagrangeova metoda $P(x)=\sum_{k=0}^{n} a_k \cdot x^k$

Lze dokázat, že tento interpolační polynom je právě jeden. I když postupy pro sestrojení interpolačního polynomu jsou různé, vždy se nakonec dojde k témuž polynomu. Hledání pomocí soustavy rovnic je nepohodlné.

Lepší postup navrhl Lagrange. Interpolační polynom vyjádřil ve formě

$$P_n(x)=\sum_{\substack{i=0\\ i\neq k}}^{n} f(x_k)\cdot L_k(x),$$ kde polynomy $L_k(x)$ jsou stupně n a mají tvar

$$L_k(x)=\sum_{\substack{i=0\\ i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)} = \frac{(x-x_0)...(x-x_{k-1})(x-x_{k+1})...(x-x_n)}{(x_k-x_0)...(x_k-x_{k-1})(x_k-x_{k+1})...(x_k-x_n)}$$

a nezávisí na funkci $f(x)$, ale na uzlových bodech.

$L_k(x) = 1 \quad pro \quad i = k$
$L_k(x) = 0 \quad pro \quad i \neq k, \quad kde \quad i, k = 0, 1, \mathrm{K}, n$

1.3. Newtonova metoda

Je to další často užívaná metoda. Pro její popis musíme zavést pojem *diference* $f[x_1, x_0]$.

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

pro $x_1 \neq x_0$.

Při limitním přechodu pro $x_1 \to x_0$ diference $f[x_1, x_0]$ přejde na derivaci $f'(x_0)$.

Základem této metody je vztah:

$$g(x)= f(x_0)+(x-x_0)\cdot f[x_1,x_0]+(x-x_0)\cdot(x-x_1)\cdot f[x_2,x_1,x_0]+...$$
$$...+(x-x_0)\cdot(x-x_1)...(x-x_{n-1})\cdot f[x_n,...,x_0]$$

Speciálním případem pro $n = 1$ je lineární interpolační polynom. Uzlové body mohou být v intervalu $\langle a, b\rangle$ rozděleny nerovnoměrně.

Pro pravidelné dělení se interpolační vzorce zjednoduší. Obvykle se tato metoda provádí za pomoci tabulky diferencí.

Pro výpočet počítačem můžeme použít vzorec pro diferenci řádu n:

$$f[x_n,...,x_0]=\sum_{i=0}^{n}\frac{f(x_i)}{\prod_{\substack{k=0\\k\neq i}}^{n}(x_i-x_k)}$$

Výhodou je to, že přidáním dalších vstupních dat můžeme zvyšovat přesnost interpolace. Newtonova metoda dovoluje odhadovat chybu, které se užitím vztahu daného stupně dopouštíme. Jako odhad chyby se bere nejbližší člen vyššího řádu.

1.4. Metoda interpolace kubickými splajny

Máme funkci $y = f(x)$ zadanou hodnotami v uzlových bodech, tzn. $y_k = f(x_k),\ k = 0, 1, \mathrm{K}, n$. Pro uzlové body nechť platí: $x_0 < x_1 < \mathrm{K} < x_n$.

Kubický splajn $s(x)$ je funkce, která splňuje tyto tři podmínky:

1) $s(x_k) = f(x_k),\ k = 0, 1, \mathrm{K}, n,$ graf funkce $s(x)$ prochází všemi body $[x_k, f(x_k)]$.

2) funkce je spojitá se svou 1. a 2. derivací na intervalu $\langle x_0, x_n \rangle$.

3) v každém intervalu $\langle x_{k-1}, x_k \rangle,\ k = 1, 2, \mathrm{K}, n$ tato funkce splývá s jistým polynomem 3. stupně.

Kubickému splajnu se říká též *po částech polynom 3. stupně.* To vyplývá z podmínky 3), která vyžaduje, aby mezi uzlovými body byl vždy polynom 3. stupně, ale nemusí být společný pro celý interval.

Pokud označíme $s''(x_k) = M_k,\ k = 0, 1, \mathrm{K}, n,$ pak 2. derivace splajnu v libovolném bodě x je

$s''(x) = M_{k-1} \cdot \frac{x_k - x}{h_k} + M_k \cdot \frac{x - x_{k-1}}{h_k}$, kde $h_k = x_k - x_{k-1}$ je vzdálenost mezi po sobě jdoucími uzly. Potom platí:

$$s'(x) = -M_{k-1} \cdot \frac{(x_k - x)^2}{2h_k} + M_k \cdot \frac{(x - x_{k-1})^2}{2h_k} + C_1$$

$$s(x) = M_{k-1} \cdot \frac{(x_k - x)^3}{6h_k} + M_k \cdot \frac{(x - x_{k-1})^3}{6h_k} + C_1 x + C_2$$

Integrační konstanty C_1 a C_2 určíme z podmínky

$$s(x_k) = M_k \cdot \frac{(x_k - x_{k-1})^3}{6h_k} + C_1 x_k + C_2 = y_k$$

$$s(x_{k-1}) = M_{k-1} \cdot \frac{(x_k - x_{k-1})^3}{6h_k} + C_1 x_{k-1} + C_2 = y_{k-1}$$

Po úpravě dostaneme soustavu lineárních rovnic:

$$C_1 x_k + C_2 = y_k - M_k \cdot \frac{h_k^2}{6}$$

$$C_1 x_{k-1} + C_2 = y_{k-1} - M_{k-1} \cdot \frac{h_k^2}{6}$$

jejíž řešení je:

$$C_1 = \frac{y_k - y_{k-1}}{h_k} - \frac{M_k - M_{k-1}}{6} \cdot h_k$$

$$C_2 = \left(y_{k-1} - \frac{M_{k-1} \cdot h_k^2}{6} \right) \cdot \frac{x_k}{h_k} - \left(y_k - \frac{M_k \cdot h_k^2}{6} \right) \cdot \frac{x_{k-1}}{h_k}$$

Po dosazení získáváme:

$$s(x) = M_{k-1} \cdot \frac{(x_k - x)^3}{6h_k} + M_k \cdot \frac{(x - x_{k-1})^3}{6h_k} + \left(y_{k-1} - \frac{M_{k-1} \cdot h_k^2}{6} \right) \cdot \frac{x_k - x}{h_k} + \left(y_k - \frac{M_k \cdot h_k^2}{6} \right) \cdot \frac{x - x_{k-1}}{h_k}$$

(1.1.)

Je zřejmé, že obě jednostranné limity zprava i zleva v uzlových bodech x_k,

$k = 1, 2, \mathrm{K}, n$, jsou stejné,

$$\lim_{x \to x_k^+} s''(x) = \lim_{x \to x_k^-} s''(x) = M_k$$

$$\lim_{x \to x_k^+} s(x) = \lim_{x \to x_k^-} s(x) = y_k$$

Jednostranné směrnice funkce $s(x)$ v bodech x_k mají velikost:

$$\lim_{x \to x_k^-} s'(x) = \frac{1}{6} \cdot h_k \cdot M_{k-1} + \frac{1}{3} \cdot h_k \cdot M_k + \frac{y_k - y_{k-1}}{h_k}$$

$$\lim_{x \to x_k^+} s'(x) = -\frac{1}{3} \cdot h_{k+1} \cdot M_k - \frac{1}{6} \cdot h_{k+1} \cdot M_{k+1} + \frac{y_{k+1} - y_k}{h_{k+1}}$$

Pro splnění podmínky spojitosti musí mít tyto limity v x_k stejnou hodnotu:

$$\frac{1}{6}\cdot h_k \cdot M_{k-1} + \frac{1}{3}\cdot\left(h_k + h_{k+1}\right)\cdot M_k + \frac{1}{6}\cdot h_{k+1} M_{k+1} = \frac{y_{k+1}-y_k}{h_{k+1}} - \frac{y_k - y_{k-1}}{h_k}$$

Pro $k = 1, 2, \mathrm{K}, n-1$ jde o soustavu $n-1$ lineárních rovnic pro $n+1$ neznámých $M_0, M_1, \mathrm{K}, M_n$. Proto musíme dodatečně zvolit dvě podmínky pro dvě neznámé. Zvolíme $M_0 = M_1 = 0$ a to odpovídá podmínkám, že v krajních bodech intervalu $\langle x_0, x_n \rangle$ jsou druhé derivace rovny nule. To znamená, že splajn přechází v přímku.

Soustava má jedno řešení a po určení hodnot neznámých získáme v každém intervalu $\langle x_{k-1}, x_k \rangle$ rovnici splajnu.

Po jednoduché úpravě lze soustavu psát v maticovém tvaru:

$$A \cdot x = b$$

$$A = \begin{pmatrix} 2\cdot(h_1+h_2) & h_2 & 0 & ..... & 0 \\ h_2 & 2\cdot(h_2+h_3) & h_3 & ..... & 0 \\ 0 & h_3 & 2\cdot(h_3+h_4) & ..... & 0 \\ ..... & ..... & ..... & ..... & ..... \\ 0 & 0 & 0 & ..... & 2\cdot(h_{n-1}+h_n) \end{pmatrix}$$

$$x = \begin{pmatrix} M_1 \\ M_2 \\ ... \\ M_{n-1} \end{pmatrix} \qquad b = \begin{pmatrix} \frac{1}{h_1}\cdot y_0 - \left(\frac{1}{h_1}+\frac{1}{h_2}\right)\cdot y_1 + \frac{1}{h_2}\cdot y_2 \\ \frac{1}{h_2}\cdot y_1 - \left(\frac{1}{h_2}+\frac{1}{h_3}\right)\cdot y_2 + \frac{1}{h_3}\cdot y_3 \\ ..... \\ \frac{1}{h_{n-1}}\cdot y_{n-2} - \left(\frac{1}{h_{n-1}}+\frac{1}{h_n}\right)\cdot y_{n-1} + \frac{1}{h_n}\cdot y_n \end{pmatrix}$$

Matice A má nenulové prvky pouze v hlavní diagonále a na dvou sousedních

diagonálách, a to nad a pod hlavní diagonálou. Takové soustavě říkáme *třídiagonální soustava.* Matice je též ostře diagonálně dominantní.

Příklad 1.:

Různými metodami nalezněte funkci $g(x)$ procházející body

x_i	1	2	3	4
$f(x_i)$	1	4	3	3

Řešení:

1) Přímá metoda

Hledáme koeficienty a_0, a_1, a_2, a_3 polynomu 3. stupně. K dispozici máme čtyři rovnice.

$$a_0 + a_1 + a_2 + a_3 = 1$$
$$a_0 + 2 \cdot a_1 + 4 \cdot a_2 + 8 \cdot a_3 = 4$$
$$a_0 + 3 \cdot a_1 + 9 \cdot a_2 + 27 \cdot a_3 = 3$$
$$a_0 + 4 \cdot a_1 + 16 \cdot a_2 + 64 \cdot a_3 = 3.$$

2) Newtonova metoda

Sestrojíme tabulku pro interpolaci

i	x_i	$f(x_i)$	$f[x_{i+1},x_i]$	$f[x_{i+2},,x_i]$	$f[x_{i+3},,,x_i]$
0	1	1			
			3		
1	2	4		-2	
			-1		0,8333
2	3	3		0,5	
			0		
3	4	3			

Tabulka diferencí pro Newtonovu interpolaci.

Aproximační polynom nalezneme tímto postupem

$$g(x) = f(x_0) + (x - x_0) \cdot f[x_1, x_0] + (x - x_0) \cdot (x - x_1) \cdot f[x_2, x_1, x_0] + (x - x_0) \cdot (x - x_1) \cdot$$
$$\cdot (x - x_2) \cdot f[x_3, x_2, x_1, x_0] = 0{,}833 \cdot x^3 - 7 \cdot x^2 + 18{,}167 \cdot x - 11.$$

3) Lagrangeova metoda

Interpolační polynom bude mít tvar

$$g(x) = 1 \cdot \frac{(x-2)\cdot(x-3)\cdot(x-4)}{(1-2)\cdot(1-3)\cdot(1-4)} + 4 \cdot \frac{(x-1)\cdot(x-3)\cdot(x-4)}{(2-1)\cdot(2-3)\cdot(2-4)} +$$
$$+ 3 \cdot \frac{(x-1)\cdot(x-2)\cdot(x-4)}{(3-1)\cdot(3-2)\cdot(3-4)} + 3 \cdot \frac{(x-1)\cdot(x-2)\cdot(x-3)}{(4-1)\cdot(4-2)\cdot(4-3)} \quad .$$

Přímá, Newtonova i Lagrangeova metoda dávají stejný výsledek.

4) Metoda kubickými splajny

Nejprve musíme spočítat druhé derivace v uzlových bodech, které se určují z podmínek spojitosti

$$g''(1) = -6{,}8$$
$$g''(2) = 3{,}2$$
$$g''(0) = g''(3) = 0 .$$

Na základě těchto hodnot nalezneme příslušné aproximace v jednotlivých dílčích intervalech:

interval $\langle 1,2 \rangle$

$$g(x) = -1{,}1333 \cdot (x-1)^3 + (2-x) + 5{,}1333 \cdot (x-1) \quad ;$$

interval $\langle 2,3 \rangle$

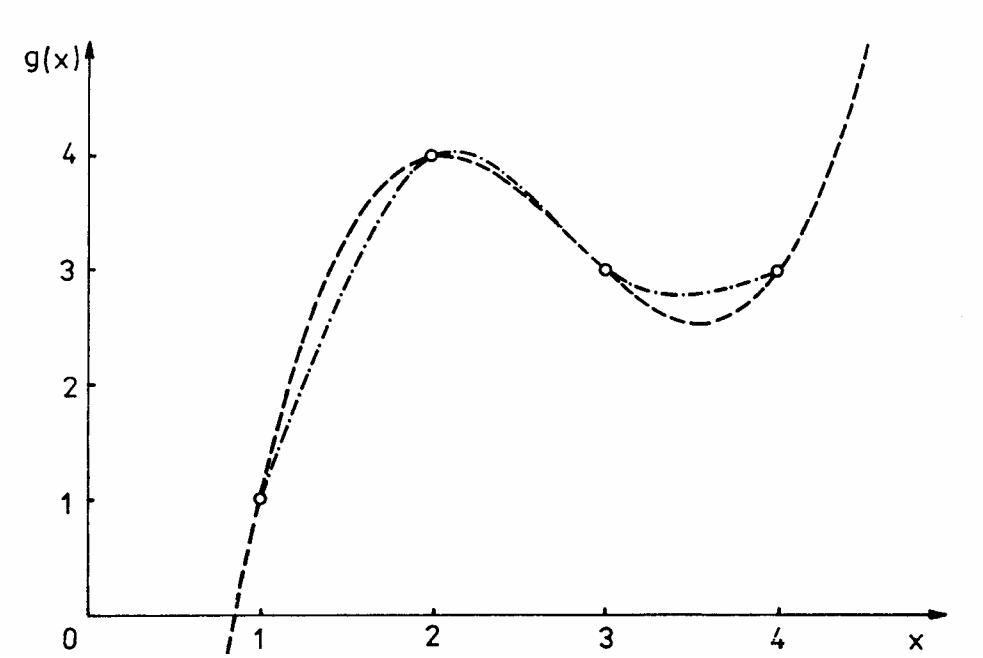
$$g(x) = -1{,}1333 \cdot (3-x)^3 + 0{,}5333 \cdot (x-2)^2 +$$
$$+ 5{,}1333 \cdot (3-x) + 2{,}4667 \cdot (x-2) \quad ;$$

interval $\langle 3,4 \rangle$

$$g(x) = 0{,}5333 \cdot (4-x)^3 + 2{,}4667 \cdot (4-x) + 3 \cdot (x-3) \quad .$$

Uvedenými postupy jsme získali dvě odlišná řešení – jednak interpolační polynom 3. stupně definovaný pro celý obor a dále kubický splajn definovaný na dílčích intervalech. Obě řešení si vyneseme do jednoho obrázku (graf 1).

Interpolační polynom je označen čárkovaně, splajn čerchovaně. Z obrázku vidíme, že interpolace splajnem má opravdu poněkud hladší průběh než obvyklý interpolační vztah.



graf 1

Příklad 2:

Funkce $f(x)$ je dána tabulkou.

k	0	1	2	3
x_k	-1,4	-0,2	1,2	2,6
$f(x_k)$	2,108	-0,484	-2,416	4,668

najděte pomocí metody přímé interpolační polynom funkce $f(x)$ s uzlovými body:

$$a) x_0, x_2$$
$$b) x_0, x_1, x_2$$
$$c) x_0, x_1, x_2, x_3$$

Řešení:

a) V tomto případě se jedná o lineární interpolaci. Rovnici přímky, procházející body [-1,4; 2,108], [1,2; -2,416] můžeme určit prostředky analytické geometrie:

$$y + 2{,}416 = \frac{-2{,}416 - 2{,}108}{1{,}2 + 1{,}4}(x - 1{,}2)$$

úpravou získáme

$$y = -1{,}74x - 0{,}328 .$$

Odtud vyplývá, že polynom pro zadanou funkci s uzly x_0, x_2 je $P_1(x) = -1{,}74x - 0{,}328$. Tento polynom se vyznačuje tím, že má s funkcí $f(x)$ společné dva body, a to $[x_0,\ f(x_0)]$, $[x_2,\ f(x_2)]$.

b) Nyní máme tři uzlové body, tudíž interpolační polynom bude nejvýše druhého stupně, tzn.

$$P_2(x) = a_0 + a_1 x + a_2 x^2 .$$

Pro $k = 0, 1, 2$ bude platit $P_2(x_k) = f(x_k)$, po dosazení do této rovnice za $x = x_k$ a $f(x_k)$ známé z tabulky dostaneme tuto soustavu rovnic se třemi neznámými a_0, a_1, a_2.

$$\begin{aligned} a_0 + a_1 \cdot (-1{,}4) + a_2 \cdot (-1{,}4)^2 &= 2{,}108 \\ a_0 + a_1 \cdot (-0{,}2) + a_2 \cdot (-0{,}2)^2 &= -0{,}484 \\ a_0 + a_1 \cdot (1{,}2) + a_2 \cdot (1{,}2)^2 &= -2{,}416 \end{aligned}$$

kterou můžeme řešit přímou metodou. Řešení této soustavy je

$$a_0 = -0{,}832, a_1 = -1{,}68, a_2 = 0{,}3$$

z toho plyne, že interpolační polynom má tvar

$$P_2(x) = 0{,}3x^2 - 1{,}68x - 0{,}832 .$$

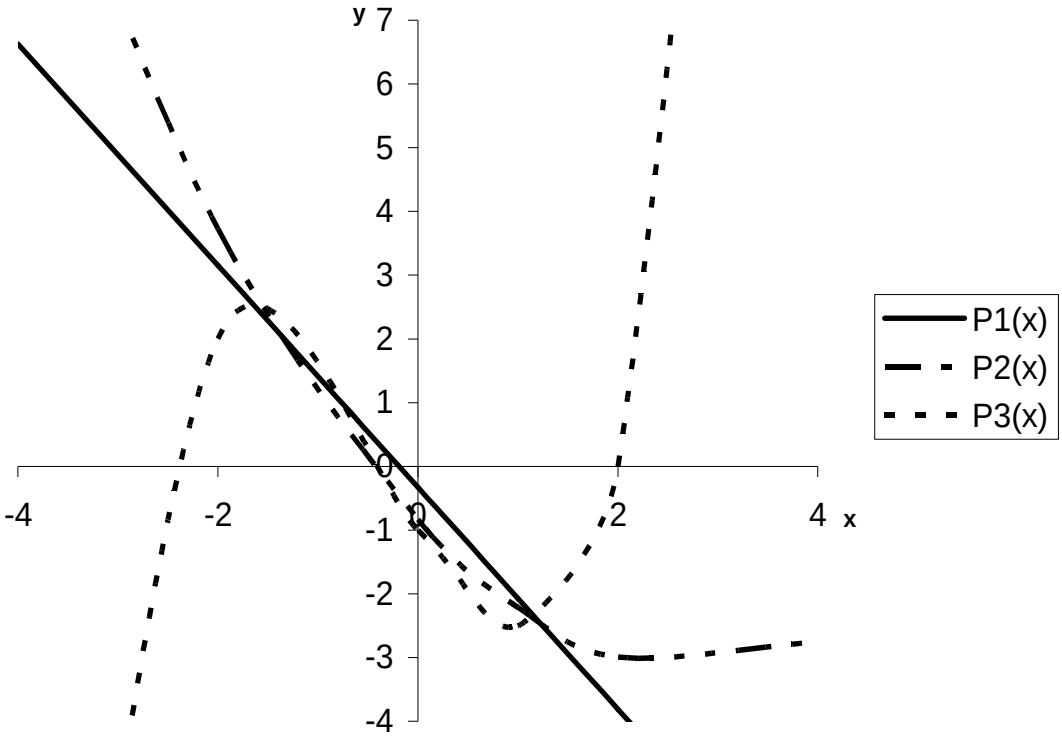
c) Uzlovými body jsou tentokrát body x_0, x_1, x_2, x_3, interpolační polynom bude tvaru

$$P_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$$

kde koeficienty a_k, $k = 0, 1, 2, 3$, splňují podmínky $P_3(x_k) = f(x_k)$ pro všechny body x_k, $k = 0, 1, 2, 3$. Když vyřešíme soustavu rovnic pro čtyři neznámé, získáme tento interpolační polynom

$$P_3(x) = 0{,}5x^3 + 0{,}5x^2 - 2{,}5x - 1{,}0\,.$$

Znázornění všech tří interpolačních polynomů si můžeme prohlédnout na grafu 2.



graf 2

Příklad 3:

Vstupní data si budeme simulovat výpočtem z funkce $f(x)$ v šesti uzlových bodech 0, 1, 2, 3, 5 a 6. Úkolem je nalézt interpolační polynom pomocí Newtonovy metody pro $x \in \langle 0, 6 \rangle$. Za zdrojovou funkci $f(x)$ si vezmeme $f(x) = 2 + 10x + 5x^2 - x^3$. Sestrojíme tabulku pro interpolaci:

i	x_i	f (x_i)	f [ ]	f [ ]	f [ ]	f [ ]
0	0	2				
			14			
1	1	16		2		
			18		-1	
2	2	34		-1		0
			16		-1	
3	3	50		-5		0
			1		-1	
4	5	52		-9		
			-26			
5	6	26				

Řešení:

Pro výpočet interpolačního výrazu sestrojíme tabulku diferencí, která bude v jednotlivých sloupcích obsahovat pořadová čísla uzlů *i*, uzlové body x_i, funkční hodnoty $f(x_i)$ a diference různých řádů.

Nejprve budeme body $f(x_i)$ interpolovat polynomem prvního stupně

$$p_1(x) = f(x_0) + (x - x_0) \cdot f[x_1, x_0] = 14x + 2$$

Chyba bude přibližně rovna

$$R_1(x) = (x - x_0) \cdot (x - x_1) \cdot f[x_2, x_1, x_0] = 2x^2 - 2x\,.$$

Zbytek je v daném intervalu příliš veliký, proto musíme pokračovat

$$p_2(x) = p_1(x) + R_1(x) = 2x^2 + 12x + 2$$
$$R_2(x) = (x - x_0) \cdot (x - x_1) \cdot (x - x_2) \cdot f[x_3, x_2, x_1, x_0] = -x^3 + 3x^2 - 2x$$

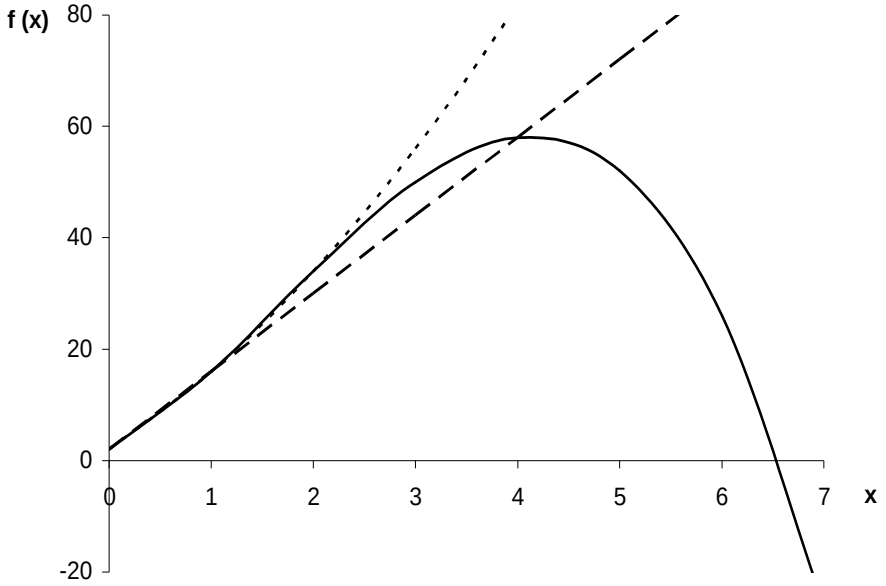
Budeme končit interpolací polynomem třetího stupně

$$p_3(x) = -x^3 + 5x^2 + 10x + 2,$$

protože zbytek $R_3(x) = 0$.

Při tomto výpočtu jsme v tabulce diferencí sledovali vytaženou čáru mezi čísly. Tabulkou je však možno procházet libovolnou cestou při zachování směru k vyšším diferencím a vždy dostaneme stejný výsledek.

Průběh nalezených interpolačních polynomů různých stupňů je znázorněn na grafu 3.



graf 3

Interpolační polynomy $p_k(x)$ postupně zahrnují další body a interpolace se zpřesňuje. Tabulka diferencí nám ukáže, jaký stupeň polynomu máme pro interpolaci skutečně použít. Může se stát, že všechny diference od určitého stupně počínaje budou nulové a my pak můžeme zadané body aproximovat zcela přesně již interpolačním polynomem tohoto stupně.

Pokud zjistíme, že diference v určitém sloupci tabulky jsou již tak malé, že zbytek *R* bude pro každé *x* z výpočetního intervalu zanedbatelný, můžeme interpolaci na tomto stupni ukončit a získaný polynom použít jako aproximační funkci.

2. Aproximace metodou nejmenších čtverců

Při interpolaci byla nahrazovaná funkce dána tabulkou se zcela přesnými funkčními hodnotami. Nejsou-li hodnoty funkce $y_k = f(x_k),\ k = 0, 1, \mathrm{K}, n$ v uzlových bodech $x_0, x_1, \mathrm{K}, x_n$ dány přesně, není vhodné volit za aproximační funkci interpolační polynom či splajn (reprodukují chyby měření). Potřebujeme chyby „vyhladit“.

Máme uzlové body $x_0, x_1, \mathrm{K}, x_n$ a příslušné funkční hodnoty $y_0, y_1, \mathrm{K}, y_n$ funkce $f(x)$, kterou neznáme, ale pro kterou platí $y_k = f(x_k),\ k = 0, 1, \mathrm{K}, n$.

Polynom $P_m(x)$ stupně $m \le n$ určíme metodou nejmenších čtverců tak, že ze všech polynomů stupně *m* vybereme ten, pro který je hodnota následujícího výrazu nejmenší.

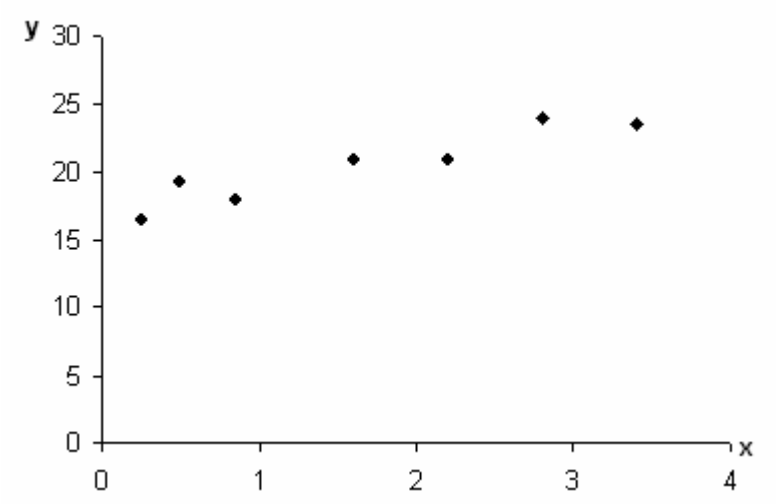
$$Q = \sum_{k=0}^{n} (y_k - P_m(x_k))^2 \qquad (1.2.)$$

Lze aproximovat i jinými funkcemi než jen polynomy, musí ale rovněž obsahovat volné parametry, které stanoví metoda nejmenších čtverců.

Při aproximaci funkce $f(x)$ polynomem $P(x)$ musíme volit kompromis mezi dvěma protichůdnými skutečnostmi:

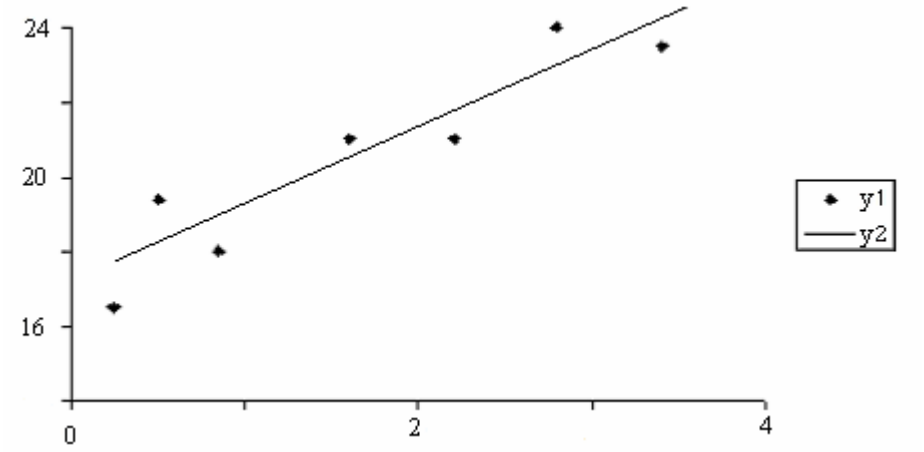
1) Polynom $P(x)$ by měl být dostatečně vysokého stupně, aby byl dobrou aproximací funkce $f(x)$.
2) Polynom $P(x)$ by neměl být příliš vysokého stupně, protože by vystihoval měřené hodnoty příliš věrně, čímž by se nepřesnosti měření zachovaly.
3) Polynomy vysokých stupňů vykazují oscilace, které aproximované funkce nemají.

Tento rozpor se řeší často statistickými metodami.



graf 4

Situaci máme znázorněnou na grafu 4, kde 7 bodů $[x_k, y_k]$ jsme dostali jako výsledek měření. Víme, že s rostoucí veličinou *x* hodnoty veličiny *y* také porostou. (Hledaná funkce je ryze monotónní.) Musíme najít předpis pro vyjádření závislosti *y* na *x*. Interpolační polynom ani splajn není vhodný. Vhodnou aproximací bude polynom prvního stupně (přímka), $m = 1$ (graf 5).



graf 5

Víme, že hledání minima funkce *Q* se převádí na řešení soustavy rovnic. Každá z těchto rovnic vypadá takto: $\frac{\partial Q}{\partial a_i} = 0$, $i = 0, 1, \ldots, m$ (1.3.)

kde a_i jsou koeficienty polynomu $P_m(x) = \sum_{i=0}^{m} a_i \cdot x^i$. Rovnicím se říká normální rovnice.

Příklad 1:

Pomocí způsobu interpolace přímky procházející počátkem zpracujte měření tuhosti pružiny. Při určování tuhosti pružiny *k* statickou metodou vycházíme z Hookova zákona pro pružinu

$$|F| = k \cdot |y| \quad ,$$

kde *F* je působící síla realizovaná závažím o hmotnosti *m* a *y* je prodloužení pružiny. Měříme ovšem statické prodloužení pružiny v závislosti na zatížení

$$y = u \cdot m \quad .$$

Dále víme, že konstanta úměrnosti *u* souvisí s hledanou tuhostí pružiny vztahem

$$k = \frac{g}{u} \quad .$$

Je tedy zřejmé, že jde o interpolaci lineární závislosti přímkou procházející počátkem. Hmotnost *m* je realizována závažími, jejichž přesnost je výrazně vyšší, než je přesnost odečítání prodloužení pružiny *y*.

Měřením prodloužením pružiny v závislosti na působící síle (tíze závaží) byly získány hodnoty uvedené v tabulce:

m [g]	0	5	10	15	20	25	30	35	40	45
y [cm]	0	2,5	4,7	7,1	9,4	11,7	13,7	17	18,4	19,7

m [g]	50	55	60	70	80	90	100	110	120	130
y [cm]	22,8	25	27,1	31,3	34,8	39,9	44,5	48,5	52,9	57,3

Pro zpracování závislosti $y = u \cdot m$ uspořádáme měřené hodnoty do tabulky:

Č. měř.	m (x_i) (g)	y_i (cm)	$x_i y_i$ (g cm)	$x_i x_i$ (g^2)	$(\tilde{u}\, x_i - y_i)^2$ $(cm)^2$
1	0,00	0,00	0,00	0,0	0
2	5,00	2,50	12,50	25,0	0,07756225
3	10,00	4,70	47,00	100,0	0,066049
4	15,00	7,10	106,50	225,0	0,18966025
5	20,00	9,40	188,00	400,0	0,264196
6	25,00	11,70	292,50	625,0	0,35105625
7	30,00	13,70	411,00	900,0	0,137641
8	35,00	17,00	595,00	1225,0	2,10105025
9	40,00	18,40	736,00	1600,0	0,394384
10	45,00	19,70	886,50	2025,0	0,08614225
11	50,00	22,80	1140,00	2500,0	0,342225
12	55,00	25,00	1375,00	3025,0	0,31753225
13	60,00	27,10	1626,00	3600,0	0,195364
14	70,00	31,30	2191,00	4900,0	0,039601
15	80,00	34,80	2784,00	6400,0	0,553536
16	90,00	39,90	3591,00	8100,0	0,007569
17	100,00	44,50	4450,00	10000,0	0,0049
18	110,00	48,50	5335,00	12100,0	0,139129
19	120,00	52,90	6348,00	14400,0	0,173056
20	130,00	57,30	7449,00	16900,0	0,210681
n	*X*	*Y*	*XY*	*XX*	R_1^2
20	1090,00	488,30	39564,00	89050,0	5,6513345

Hodnoty *XY* a *XX* jsou vypočteny v posledním řádku tabulky. Vypočítáme tedy:

$$\tilde{u} = \frac{XY}{XX} = \frac{39564{,}00\ g \cdot cm}{89050{,}00\ g^2} = 0{,}4443\ cm \cdot g^{-1}.$$

Využitím vypočtené hodnoty $\tilde{u}$ můžeme vytvořit poslední sloupec tabulky a sečtením řádků získáme minimální sumu čtverců odchylek R_1^2.

Dále můžeme stanovit disperzi hodnoty $\tilde{u}$:

$$D_{\tilde{u}}^{*} \equiv \left(\tilde{\sigma}_{\tilde{u}}^{2}\right)^{*} = \frac{1}{XX} \cdot \frac{R_1^2}{n-1} = \frac{1}{89050{,}0} \cdot \frac{5{,}65133}{19}\ cm^2 \cdot g^{-2} = 3{,}34011 \cdot 10^{-6}\ cm^2 \cdot g^{-2}.$$

Po zaokrouhlení, odmocnění disperze a úpravě na jednotky systému SI dostaneme

$$\tilde{u} = (4{,}443 \pm 0{,}018)\ m \cdot kg^{-1}.$$

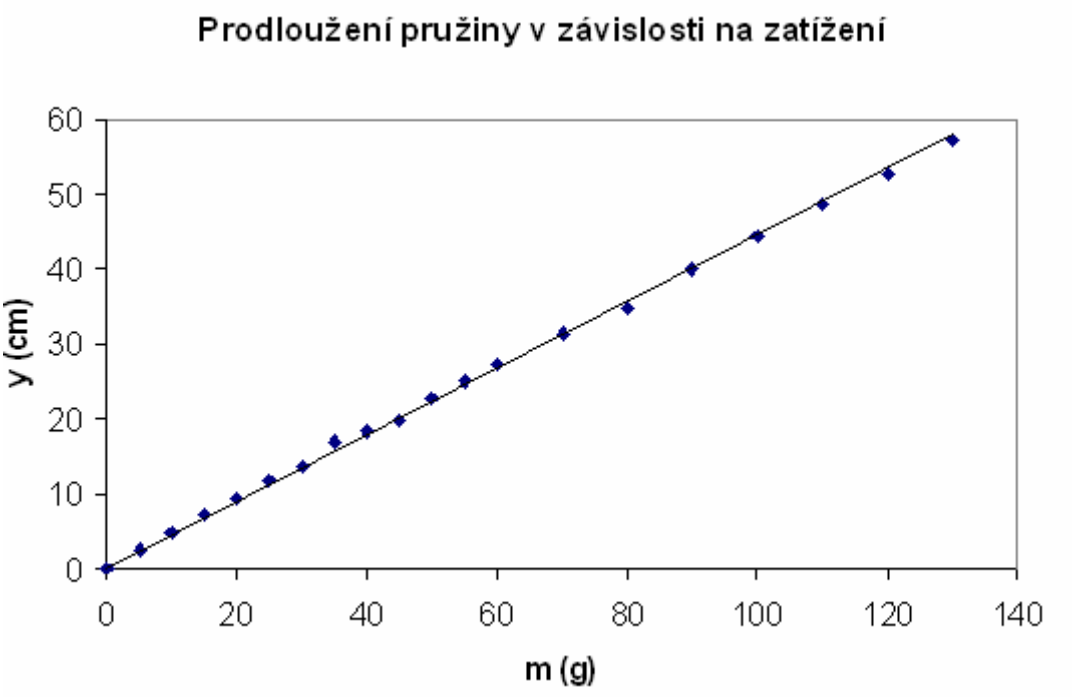
Pro výpočet hledané tuhosti využijeme hodnotu gravitačního zrychlení zaokrouhlenou tak, aby zaokrouhlovací chyba byla nejméně o řád menší, než chyba hodnoty $\tilde{u}$.

Při zaokrouhlení gravitačního zrychlení na hodnotu $g = 9{,}81$ je relativní zaokrouhlovací chyba $n_g = 5 \cdot 10^{-5}$.

Protože relativní chyba hodnoty $\tilde{u}$ je $n_g \cong 4 \cdot 10^{-3}$, je možno zaokrouhlovací chybu gravitačního zrychlení zanedbat. Potom

$$\tilde{k} = (2{,}208 \pm 0{,}009)\ N \cdot m^{-1}\ .$$

Výsledky měření prodloužení pružiny v závislosti na zatížení znázorníme graficky.



Příklad 2:

Napište normální rovnice:

a) pro polynom 1. stupně

b) pro polynom 2. stupně

Řešení:

a) Hledáme lineární polynom $P_1(x) = a_0 + a_1 x$. Funkce Q má tedy tvar:

$$Q = \sum_{k=0}^{n} \left(y_k - a_0 - a_1 x_k\right)^2$$

Podle vztahu 1.3. dostaneme soustavu:

$$\frac{\partial Q}{\partial a_0} = 2\sum_{k=0}^{n} \left(y_k - a_0 - a_1 x_k\right) \cdot (-1) = 0$$
$$\frac{\partial Q}{\partial a_1} = 2\sum_{k=0}^{n} \left(y_k - a_0 - a_1 x_k\right) \cdot (-x_k) = 0$$

Soustavu převedeme pomocí jednoduché úpravy na tento tvar:

$$(n+1) \cdot a_0 + \sum_{k=0}^{n} x_k \cdot a_1 = \sum_{k=0}^{n} y_k$$
$$\sum_{k=0}^{n} x_k \cdot a_0 + \sum_{k=0}^{n} x_k^2 \cdot a_1 = \sum_{k=0}^{n} x_k \cdot y_k$$

protože $\sum_{k=0}^{n} a_0 = a_0 \cdot \sum_{k=0}^{n} 1 = a_0 \cdot (1+1+\ldots+1) = (n+1) \cdot a_0$.

Tuto soustavu můžeme zapsat v maticovém tvaru:

$$\begin{pmatrix} \sum_{k=0}^{n} x_k^0 & \sum_{k=0}^{n} x_k^1 \\ \sum_{k=0}^{n} x_k^1 & \sum_{k=0}^{n} x_k^2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{n} x_k^0 \cdot y_k \\ \sum_{k=0}^{n} x_k^1 \cdot y_k \end{pmatrix}$$

kde platí, že $x_k^{\,0} = 1$ a $x_k^{\,1} = x_k$. Ponecháváme tyto výrazy v naznačených mocninách, aby vynikla zákonitost prvků matic.

b) Hledáme kvadratický polynom $P_2(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2$.

Výraz 1.2. bude vypadat takto:

$$Q = \sum_{k=0}^{n} \left(y_k - a_0 - a_1 \cdot x_k - a_2 \cdot x_k^2\right)^2$$

Využitím parciálních derivací dostaneme tento vztah:

$$\sum_{k=0}^{n}\left(y_k - a_0 - a_1 \cdot x_k - a_2 \cdot x_k^2\right) = 0$$
$$\sum_{k=0}^{n}\left(x_k \cdot y_k - a_0 \cdot x_k - a_1 \cdot x_k^2 - a_2 \cdot x_k^3\right) = 0$$
$$\sum_{k=0}^{n}\left(x_k^2 \cdot y_k - a_0 \cdot x_k^2 - a_1 \cdot x_k^3 - a_2 \cdot x_k^4\right) = 0$$

A odtud už získáme normální soustavu pro tři neznámé a_0, a_1, a_2:

$$(n+1)\cdot a_0 + \sum_{k=0}^{n} x_k \cdot a_1 + \sum_{k=0}^{n} x_k^2 \cdot a_2 = \sum_{k=0}^{n} y_k$$
$$\sum_{k=0}^{n} x_k \cdot a_0 + \sum_{k=0}^{n} x_k^2 \cdot a_1 + \sum_{k=0}^{n} x_k^3 \cdot a_2 = \sum_{k=0}^{n} x_k \cdot y_k$$
$$\sum_{k=0}^{n} x_k^2 \cdot a_0 + \sum_{k=0}^{n} x_k^3 \cdot a_1 + \sum_{k=0}^{n} x_k^4 \cdot a_2 = \sum_{k=0}^{n} x_k^2 \cdot y_k$$

Většina fyzikálních závislostí je ve formě tabulky, a proto je interpolace a aproximace empirických vztahů pro fyziku tak důležitá.

3. Richardsonova extrapolace (obecná idea)

Při numerické derivaci (ale i integraci – obecně pro jakoukoli operaci, která je diskretizována krokem $h \to 0$) jsou dávány rozporné požadavky na volbu kroku. Ten má na jedné straně být co nejmenší, abychom snížili chybu metody, na druhé straně nesmí být příliš malý, abychom neznehodnotili výsledek zaokrouhlovacími chybami.
Jedním z obecných postupů, které lze v takových případech uplatnit, je *Richardsonova extrapolace*. Jejím cílem je zpřesnit odhad výsledku pro krok jdoucí k nule, aniž bychom ve výpočtu přímo použili malé hodnoty kroku.

Úloha:
Nechť správný výsledek nějakého výpočtu je funkce *g* kroku *h*. Předpokládáme, že *g* má v okolí bodu *o* Taylorův rozvoj tvaru

$$g(h) = g(o) + \frac{h^p}{p!} \cdot g^{(p)}(o) + \frac{h^r}{r!} \cdot g^{(r)}(o) + \dots ,$$

kde *p* (řád metody) známe a $r > p$. Z hodnot funkce *g* v konečně mnoha nenulových bodech máme odhadnout hodnotu $g(0)$.

Nejjednodušší řešení dostaneme, pokud zanedbáme členy řádů vyšších než *p* a aproximujeme *g* polynomem

$$\varphi(h) = s + c \cdot h^p , \qquad s, c \in R.$$

Ke stanovení dvou neznámých parametrů s, c zvolíme dva uzlové body $h, \frac{h}{q}$, *kde* $q \neq 1$:

$$\varphi(h) = s + c \cdot h^p = g(h),$$
$$\varphi\left(\frac{h}{q}\right) = s + c \cdot \frac{h^p}{q^p} = g\left(\frac{h}{q}\right),$$

což je regulární soustava dvou lineárních rovnic pro dvě neznámé s, c, z nichž nás zajímá pouze $s = \varphi(0)$; stačí vynásobit druhou rovnici q^p a odečíst od ní první rovnici:

$$\left(q^p - 1\right) \cdot s = q^p \cdot g \cdot \left(\frac{h}{q}\right) - g(h),$$

$$s = \frac{q^p \cdot g \cdot \left(\frac{h}{q}\right) - g(h)}{q^p - 1}.$$

(1.4.)

Dostali jsme odhad *s* hodnoty $g(0)$, který je zatížen pouze chybami vyšších řádů než *p*. Richardsonova extrapolace umožnila zvýšit řád metody.

Jedná se většinou o extrapolaci, neboť obvykle nemůžeme měnit znaménko veličiny *h* (kroku). Extrapolaci bychom se měli vyhýbat, ale zde činíme výjimku díky následujícím argumentům:

1) extrapolujeme v blízkosti uzlových bodů,
2) předpokládáme, že známe teoretickou závislost; navíc v aplikacích tato závislost platí tím přesněji, čím blíže jsme k nule.

Nepostradatelný je předpoklad, že známe řád metody.

Dále je nutno se zabývat volbou poměru kroků *q*. Bude-li tento poměr velký (např. 10), pak používáme jednu hodnotu, která je blízko nuly $\left(\frac{h}{q}\right)$, druhou (h), která je natolik vzdálená, že by aproximace prvními členy Taylorova rozvoje mohla být velmi nepřesná. To nelze doporučit. Totéž lze říci o velmi malých hodnotách *q*.

Bude-li poměr *q* blízký jedné, pak extrapolujeme z krátkého intervalu do poměrně vzdáleného bodu, což opět nelze doporučit. Často volíme $q = 2 \left(resp.\ q = \frac{1}{2}\right)$, pak rovnice (1.4.) nabývá tvaru

$$s = \frac{2^p \cdot g \cdot \left(\frac{h}{2}\right) - g(h)}{2^p - 1}.$$

Pokud nám dosažený řád metody nestačí, můžeme Richardsonovu extrapolaci opakovat (pokud známe řád chyby). K tomu potřebujeme vypočítat hodnotu funkce *g* v alespoň jednom dalším bodě, obvykle volíme bod $\frac{h}{q^2}$. Výše uvedený postup nám poskytne 2 odhady řádu *r*, jeden z hodnot $g(h), g\left(\frac{h}{q}\right)$, druhý z hodnot $\frac{g(h)}{q}, g\left(\frac{h}{q^2}\right)$, tj. pro krok

q-krát menší. Z nich pak můžeme další Richardsonovou extrapolací vypočítat odhad, v němž bude kompenzována i chyba *r*-tého řádu, zbudou chyby vyšších řádů.

Nyní bude *p* ve vzorci nahrazeno *r*, což je nový řád chyby. Takto můžeme teoreticky libovolně zvyšovat řád metody, pokud víme, které členy Taylorova rozvoje chyby metody mohou být nenulové. Tento postup se někdy využívá, na druhé straně lze namítnout, že vyžadujeme spojitost stále vyšších derivací a výsledek odpovídá extrapolaci při aproximaci polynomem stále vyššího stupně, což může být problematické.

4. Diferenciální rovnice

Omezíme se na obyčejné diferenciální rovnice, navíc na diferenciální rovnici 1. řádu. Pro ni budeme řešit nejjednodušší možnou úlohu, Cauchyho počáteční úlohu. Na daném reálném intervalu $\langle x_0, x_n \rangle$ máme řešit diferenciální rovnici

$$y'(x) = f(x, y(x)) \qquad (1.5.)$$

s počáteční podmínkou

$$y(x_0) = y_0 , \qquad (1.6.)$$

kde f je funkce dvou reálných proměnných (pravá strana diferenciální rovnice) a $y_0 \in R$. Pokud f nezávisí na y, tj. $f(x, y) = g(x)$ pro nějakou funkci g jedné reálné proměnné, pak dostáváme výpočet určitého integrálu jako speciální případ řešení diferenciální rovnice

$$y'(x) = g(x)$$

s počáteční podmínkou (1.6.); a ta má řešení

$$y(x) = y_0 + \int_{x_0}^{x} g(t) \cdot dt , \qquad x \in \langle x_0, x_n \rangle .$$

Obyčejné diferenciální rovnice vyšších řádů lze vhodnou volbou pomocných proměnných převést na soustavy diferenciálních rovnic prvního řádu. Ty lze řešit analogicky jako pro jednu proměnnou s příslušně upravenými typy proměnných,
tj. y je vektorová funkce reálného argumentu.
Nejdříve bychom se měli ujistit, že řešení Cauchyho počáteční úlohy existuje a že je jediné. Obecně tomu tak být nemusí.
Příkladem je diferenciální rovnice s počáteční podmínkou:

$$y'(x) = \sqrt{y(x)}, \qquad y(o) = 0 .$$

Pro $x \geq 0$ má triviální řešení $y(x) = 0$, ale řeší ji i funkce $y(x) = \frac{x^2}{4}$.

Nedostatkem předchozího případu je, že pravá strana není definována pro $y(x)<0$, lze však najít podobné příklady, které jsou definovány všude. Např. diferenciální rovnice s počáteční podmínkou:

$$y'(x)=\sqrt[3]{y(x)}, \qquad y(o)=0,$$

kde třetí odmocninu považujeme za reálnou funkci definovanou i pro záporný argument. Má řešení např. $y(x)=0$ *a* $y(x)=\pm\left(\frac{2}{3}\cdot x\right)^{\frac{3}{2}}$.

Z fyziky jsme zvyklí, že diferenciálními rovnicemi popisujeme systémy, jejichž chování se zdá být plně určeno počátečními podmínkami. Z předchozích příkladů je vidět, že po matematické stránce by počáteční podmínky nemusely stačit k jednoznačnému určení řešení. Následující podmínka vyloučí nejednoznačnost řešení v mnoha důležitých případech.

Lipschitzova podmínka:

Nechť funkce *f* je definovaná a spojitá na $\langle x_0, x_n\rangle \times R$ (tj. $f(x,y)$ je určeno pro všechna $x\in\langle x_0, x_n\rangle$ a pro všechna $y\in R$). Nechť

$$\exists L\in R\ \forall x\in\langle x_0, x_n\rangle\ \forall y_1, y_2\in R: |f(x,y_1)-f(x,y_2)|\le L|y_1-y_2|.$$

Pak existuje právě jedna reálná funkce *y* na intervalu $\langle x_0, x_n\rangle$, která je řešením rovnice (1.5.) s počáteční podmínkou (1.6.).

Cauchyho počáteční úlohu lze přepsat na ekvivalentní integrální rovnici

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))\cdot dt.$$

Integrál na pravé straně lze chápat jako integrál funkce *g* jedné proměnné, $g(t)=f(t,y(t))$; tuto funkci však obecně neznáme, neboť obsahuje hledané řešení *y*. Můžeme jej také chápat jako křivkový integrál přes křivku s parametrizací $(t, y(t))$, $t\in\langle x_0, x_n\rangle$; tuto křivku však neznáme.

Protože můžeme pracovat jen s konečně mnoha body, rozdělíme interval $\langle x_0, x_n \rangle$ na n dílčích intervalů, pro jednoduchost stejné délky $h = \frac{(x_n - x_0)}{n}$. Získáme uzlové body $x_i = x_0 + ih, \; kde \; i = 0, \mathrm{K}, n$. Správné hodnoty řešení v uzlových bodech, $y(x_i)$, nahradíme jejich odhady, které budeme značit y_i. Odpovídající hodnoty pravé strany diferenciální rovnice budeme pro zjednodušení značit

$$f_i = f(x_i, y_i).$$

Postupně budeme generovat posloupnost $y_i, i = 0, \mathrm{K}, n$. V kroku $i+1$ známe odhady y_0, K, y_i, s jejichž pomocí počítáme odhad $y_i + 1$. Podle analogie se vztahem pro přesné řešení

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(t, y(t)) \cdot dt$$

budeme hledat složky řešení takové, že

$$y_{i+1} - y_i = \Delta y_i \; ,$$

kde Δy_i bude odhad integrálu

$$\int_{x_i}^{x_{i+1}} f(t, y(t)) \cdot dt \; . \qquad (1.7.)$$

Pomocí tohoto odhadu počítáme další hodnotu

$$y_{i+1} = y_i + \Delta y_i \; .$$

Jednotlivé metody se liší pouze způsobem odhadu Δy_i integrálu (1.7.).

4.1. Jednokrokové metody

Pomocí těchto metod počítáme nové hodnoty pouze na základě výsledku předchozího kroku řešení. Typickými a nejčastěji používanými metodami z této skupiny jsou *Rungovy-Kuttovy metody.*

Rungovy-Kuttovy metody

1) Eulerova metoda

Je to nejjednodušší metoda řešení diferenciálních rovnic. Jedná se o situaci, kdy integrál (1.7.) počítáme metodou levého odhadu (s jedním intervalem), tj. funkci $f(t, y(t))$ nahrazujeme konstantou rovnou její hodnotě v levém krajním bodě uvažovaného intervalu x_i. Druhý argument y_i máme již odhadnutý z předchozího kroku (nebo pro $i = 0$ vyplývá z počáteční podmínky).

$$\Delta y_i = \int_{x_i}^{x_{i+1}} f(x_i, y_i) \cdot dt = h \cdot f(x_i, y_i) ,$$
$$y_{i+1} = y_i + h \cdot f(x_i, y_i) = y_i + h \cdot f_i \ .$$

Uvedený vzorec má názorný geometrický význam: hodnota $f_i = f(x_i, y_i)$ je směrnice úsečky vedené z bodu (x_i, y_i) do bodu (x_{i+1}, y_{i+1}).

2) První modifikace Eulerovy metody

Tak jako Eulerova metoda je zobecněním integrace metodou levého odhadu, následující metoda je zobecněním obdélníkové metody. Funkci $f(t, y(t))$ v ní nahradíme opět konstantou, kterou však bude hodnota uprostřed intervalu integrace, tedy v bodě

$$\frac{x_i + x_{i+1}}{2} = x_i + \frac{h}{2} .$$

Problémem však zůstává, co dosadit za druhý argument funkce f; měla by to být hodnota hledaného řešení. Tento nedostatek se vyřeší tím, že provedeme pomocný krok poloviční délky (Eulerovou metodou), čímž dostaneme odhad řešení v bodě $x_i + \frac{h}{2}$:

$$\eta_i = y_i + \frac{h}{2} \cdot f_i \ .$$

Funkci $f(t, y(t))$ nahradíme konstantou $f\left(x_i + \frac{h}{2}, \eta_i\right)$ a dostaneme

$$\Delta y_i = \int_{x_i}^{x_{i+1}} f\left(x_i + \frac{h}{2}, \eta_i\right) \cdot dt = h \cdot f\left(x_i + \frac{h}{2}, \eta_i\right).$$

Tato metoda je 2. řádu.

3) Druhá modifikace Eulerovy metody (Heunova metoda)

Metoda je zobecněním lichoběžníkové metody integrace. Funkci $f(t, y(t))$ v ní nahradíme lineární funkcí, proloženou hodnotami v krajních bodech intervalu. Hodnota v levém krajním bodě je $f_i = f(x_i, y_i)$. V pravém krajním bodě však narážíme na neznalost *y*-ové souřadnice, což opět řešíme pomocným krokem (tentokrát délky *h*) Eulerovou metodou, kterým dostaneme odhad řešení v bodě x_{i+1}:

$$\theta_i = y_i + h \cdot f_i .$$

Funkci $f(t, y(t))$ nahradíme lineární funkcí, jejíž graf prochází body $(x_i, f(x_i, y_i)), (x_{i+1}, f(x_{i+1}, \theta_i))$. Dostaneme

$$\Delta y_i = \frac{h}{2} \cdot \left(f(x_i, y_i) + f(x_{i+1}, \theta_i)\right).$$

Tato metoda je 2. řádu.

4) Rungova-Kuttova metoda 4. řádu

Podstatou uvedených modifikací Eulerovy metody bylo, že jsme v pomocných krocích stanovili hodnoty, které umožnily kompenzovat nejnižší členy Taylorova rozvoje chyby. Takto lze pokračovat dále a s více pomocnými kroky získat metody vyšších řádů. To je princip obecných Rungových-Kuttových metod.

Z nich nejrozšířenější je Rungova-Kuttova metoda 4. řádu, která používá hodnoty pravé strany ve 4 bodech.

Postup je následující: nejprve vypočteme pomocné body a hodnoty pravé strany v nich,

$$k_{i,1} = f(x_i, y_i),$$
$$k_{i,2} = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k_{i,1}\right),$$
$$k_{i,3} = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot k_{i,2}\right),$$
$$k_{i,4} = f(x_i + h, y_i + h \cdot k_{i,3}).$$

Integrál (1.7.) nahradíme lineární kombinací těchto hodnot:

$$\Delta y_i = \frac{h}{6} \cdot (k_{i,1} + 2 \cdot k_{i,2} + 2 \cdot k_{i,3} + k_{i,4}).$$

Začátek výpočtu připomíná první modifikaci Eulerovy metody. Hodnotu $k_{i,2}$ počítáme v bodě, který jsme dostali Eulerovou metodou s poloviční krokem, tu pak použijeme při výpočtu $k_{i,3}$ podobně jako v první modifikaci Eulerovy metody,

s tím rozdílem, že uděláme pouze krok poloviční délky. S hodnotou $k_{i,3}$ pak naložíme obdobně jako v první modifikace Eulerovy metody.

Všechny dosud uvedené metody patří do rozsáhlejší třídy Rungových-Kuttových metod. Jejich společným rysem je, že odhadují integrál $\int_{x_i}^{x_{i+1}} f(t, y(t)) \cdot dt$ z několika hodnot funkce f v bodech, získaných z výchozích hodnot x_i, y_i a pomocných kroků. Tyto hodnoty jsou zkombinovány tak, aby se vykompenzovaly chyby nejnižších řádů. Takto lze za cenu větší složitosti obdržet metody libovolně vysokého řádu.

Rungova-Kuttova metoda 4. řádu se obvykle jeví jako optimální kompromis, když při výpočtu používá hodnota pravé strany ve čtyřech bodech; je dokázáno, že pro řád $p > 4$ neexistuje obdobný postup, který by vystačil pouze s hodnotami v p bodech.

Rungovy-Kuttovy metody řadíme mezi metody jednokrokové, neboť při výpočtu hodnota y_{i+1} používáme z dříve vypočtených hodnot pouze jedinou, y_i .

Tabulka řádů probíraných metod

metoda	řád
Eulerova	1
první modifikace Eulerovy	2
druhá modifikace Eulerovy	2
Rungova-Kuttova 4. řádu	4

4.2. Vícekrokové metody

Tyto metody využívají obecně více než jednu posledně vypočtenou hodnotu. Myšlenka je založena na tom, že z posloupnosti dříve vypočtených hodnot dostáváme informaci o průběhu řešení, která nám umožňuje lépe odhadnout integrál $\int_{x_i}^{x_{i+1}} f(t, y(t)) \cdot dt$. To dovoluje například zvýšit řád metody, aniž bychom potřebovali další pomocné kroky, jaké jsme používali v Rungových-Kuttových metodách.

Jednoduchost některých vícekrokových metod je lákavá. Bohužel se ztrácí, neboť k nastartování *s*-krokové metody potřebujeme *s* hodnot $y_0, y_1, \mathrm{K}, y_{i-1}$. Ty získáváme zvolenou *startovací metodou*, obvykle některou z jednokrokových metod, např. Rungovou-Kuttovou, jejímuž naprogramování se tedy nevyhneme. Na druhé straně nasazení vícekrokové metody může snížit výpočetní složitost. Hlavní výhodou je možnost zpřesnění řešení v metodách *prediktor-korektor*.

Z hlediska závislosti chyb na kroku by bylo žádoucí, aby startovací metoda byla zhruba stejného řádu jako vícekroková metoda, kterou se bude pokračovat. Zejména by bylo nevhodné zvolit startovací metodu podstatně nižšího řádu, neboť pak hrozí, že hned v prvních krocích připustíme velkou chybu, která znehodnotí následný výpočet, provedený s vyšší přesností. Přesnější analýza chyb vede k doporučení, aby řád startovací metody byl stejný nebo o jednu nižší než řád následné vícekrokové metody. O jednu nižší proto, že startovací metodu použijeme v konstantním počtu kroků (minimálním pro nastartovaní vícekrokové metody), zatímco počet kroků vícekrokové metody je nepřímo úměrný délce kroku.

1) Adamsovy-Bashforthovy metody (explicitní)

Tyto metody vychází z toho, že s hodnotami pravé strany $f_i, f_{i-1}, \mathrm{K}, f_{i-s+1}$ (odpovídajícími uzlovým bodům $x_i, x_{i-1}, \mathrm{K}, x_{i-s+1}$) proložíme interpolační polynom φ_i a ten integrujeme místo neznámé funkce $f(t, y(t))$.

Tedy

$$\Delta y_i = \int_{x_i}^{x_{i+1}} \varphi_i(t) \cdot dt \,.$$

Ve skutečnosti není potřeba přímo počítat interpolační polynom φ_i. Podle úvahy o linearitě integrálu závisí výsledek lineárně na hodnotách f_i, f_{i-1}, K , f_{i-s+1}, je tedy tvaru

$$\Delta y_i = h \cdot \sum_{j=0}^{s-1} \varpi_j \cdot f_{i-j} \,,$$

kde ϖ_j jsou předem vypočítané konstanty pro danou metodu. (Lze ji získat integrací polynomů z Lagrangeova tvaru interpolačního polynomu nebo metodou neurčitých koeficientů.) Jeden krok metody tedy představuje pouze jeden výpočet pravé strany a jednu lineární kombinaci *s* hodnot. Řád metody je *s*, tedy počet použitých bodů.
Je nutné ještě zdůraznit, že náhrada polynomem se zde používá pro pravou stranu $f(t, y(t))$, nikoli pro řešení $y(t)$.

2) Adamsovy-Moultonovy metody (implicitní)

Tyto metody používají obdobnou myšlenku jako metody Adamsovy-Bashforthovy, pravou stranu $f(t, y(t))$ aproximují interpolačním polynomem. Abychom se vyhnuli velké chybě způsobené extrapolací, interpolujeme (v užším smyslu) díky tomu, že interpolační polynom φ_i proložíme nejen hodnotami f_i, f_{i-1}, K , f_{i-s+1}, ale i hodnotou pravé strany v bodě x_{i+1}, tj.

$$f_{i+1} = f(x_{i+1}, y_{i+1}) \,.$$

Ta ovšem závisí i na výsledné hodnotě y_{i+1}. Stejně jako u Adamsovy-Bashforthovy metody se postup redukuje na výpočet lineární kombinace hodnot pravé strany

$$y_{i+1} - y_i = \Delta y_i = h \cdot \sum_{j=-1}^{s-1} \varpi_j \cdot f_{i-j} \,,$$

kde ϖ_j jsou předem vypočtené koeficienty pro daný řád metody. Po dosazení za f_{i+1} dostáváme rovnici

$$y_{i+1} = y_i + h \cdot \varpi_{-1} \cdot f\left(x_{i+1}, y_{i+1}\right) + h \cdot \sum_{j=0}^{s-1} \varpi_j \cdot f_{i-j} \tag{1.8.}$$

pro neznámou hodnotu y_{i+1}, která je tímto určena implicitně; proto se takovým metodám říká implicitní. Nevýhodou je, že jen zřídka se rovnici (1.8.) daří řešit analyticky a tím realizovat původní záměr. Případné numerické řešení této rovnice zvyšuje výpočetní náročnost. Používá se ve zjednodušené podobě jako součást metod *prediktor-korektor*; ty jsou hlavním místem uplatnění implicitních metod.

3) Metody prediktor-korektor

Jedná se o široké spektrum metod, které se hojně používají, zejména v úlohách, kde je dosažení požadované přesnosti obtížné. Základem je korektor, což je některá z implicitních metod, modifikovaná tak, že se rovnice (1.8.) řeší numericky. V *j*-té iteraci z ní vypočítáme odhad y_{i+1}, hodnoty y_{i+1}, přičemž na pravé straně použijeme odhad $y_{i+1,\,j-1}$ získaný v předchozí iteraci. Je-li korektorem Adamsova-Moultonova metoda, dostaneme předpis

$$y_{i+1,\,j} = y_i + h \cdot \sum_{j=0}^{s-1} \varpi_j \cdot f_{i-j} + h \cdot \varpi_{-1} \cdot f\left(x_{i+1}, y_{i+1,\,j-1}\right).$$

Jedná se vlastně o řešení rovnice (1.8.) metodou prosté iterace. Pro popis algoritmu zbývá stanovit řídící mechanismus iteračního použití korektoru. Jednotlivé kroky výpočtu je zvykem označovat zkratkami jejich anglických názvů: P pro prediktor (predictor), C pro korektor (corrector). Kromě nich může být (z hlediska složitosti) nezanedbatelnou částí výpočtu i vyhodnocení pravé strany (evaluation), tj. hodnot

$$f_{i+1,\,j} = f\left(x_{i+1}, y_{i+1,\,j}\right), \qquad j = 0, 1, ...,$$

označované písmenem E.

Jak je zvykem u metody prosté iterace, měl by se cyklus korektoru provádět tak dlouho, dokud nedosáhneme dostatečně malého rozdílu po sobě následujících výsledků $y_{i+1,j} - y_{i+1,j-1}$. Takový postup má však problém v tom, že předem nevíme, kolik iterací budeme potřebovat. V praxi se to může projevit radikálním zpomalením metody v průběhu řešení. Přitom nemusí mít smysl usilovat o velmi přesné řešení, neboť samotná rovnice (1.8.) je již jen nepřesnou aproximací správného řešení původní diferenciální rovnice. Z těchto důvodů se obvykle volí konstantní počet *k* opakování korektoru a odpovídající řídící mechanismus se formálně zapisuje $P(EC)^k E$. Tento postup odpovídá jednomu kroku numerického řešení diferenciální rovnice. Často se volí $k = 1$, pak se korektor – bez ohledu na obdržené výsledky – použije jen jednou a řídící mechanismus lze zapsat jako PECE.

Příkladem metod prediktor-korektor jsou Adamsovy metody. V nich je predikátorem Adamsova-Bashforthova metoda, korektorem Adamsova-Moultonava metoda. Kombinací různých prediktorů a korektorů dostáváme velmi širokou škálu metod.

Metody prediktor-korektor se v posledních letech podílely na řadě nových úspěšných aplikací tam, kde dřívější postupy selhávaly, jako např. při předpovědi vývoje sluneční soustavy na desítky miliard let.

Příklad:

Na těleso působí tíha $m \cdot g$ směrem dolů a odporová síla prostředí $\lambda \cdot v$ směrem vzhůru. Počáteční rychlost tělesa je nulová. Pak platí

$$m \cdot \frac{dv}{dt} = m \cdot g - \lambda \cdot v \quad , v(0) = 0$$

Pokud

$$y \equiv v$$

můžeme napsat řešení

$$f(t,v) = \frac{m \cdot g - \lambda \cdot v}{m} \quad , y_0 \equiv 0 .$$

Veličiny y, y_0 a funkce *f* mohou být také vektory. Zapisujeme je do sloupců.

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \mathrm{M} \\ y_n(t) \end{bmatrix}, \quad y(0) = \begin{bmatrix} y_{01} \\ y_{02} \\ \mathrm{M} \\ y_{0n} \end{bmatrix}, \quad f(t,y) = \begin{bmatrix} f_1(t,y) \\ f_2(t,y) \\ \mathrm{M} \\ f_n(t,y) \end{bmatrix}$$

Derivaci značíme takto

$$\frac{dy}{dt} = y'$$

Ve fyzice se značí derivace podle času takto

$$\frac{dy}{dt} = \dot{y}$$

Nyní porovnáme vektorový zápis

$$y' = f(t,y)$$
$$y(t_0) = y_0$$

a složkový zápis

$$y_1' = f_1(t, y_1, y_2, \mathrm{K}\ , y_n)$$
$$y_2' = f_2(t, y_1, y_2, \mathrm{K}\ , y_n)$$
$$\mathrm{M}$$
$$y_n' = f_n(t, y_1, y_2, \mathrm{K}\ , y_n)$$

$$y_1(t_0) = y_{01}$$
$$y_2(t_0) = y_{02}$$
$$\mathrm{M}$$
$$y_n(t_0) = y_{0n}$$

Rovnice vyšších řádů než prvního lze vhodnými substitucemi převést na soustavu rovnic prvního řádu. Rovnice n-tého řádu $y^{(k)}$…k-tá derivace $y^{(n)} = g\left(t, y, y', \mathrm{K}\ , y^{(n-1)}\right)$ s počátečními, resp. okrajovými podmínkami

$$
\begin{aligned}
&y(t_0) = y_{01}\\
&y'(t_0) = y_{02}\\
&\mathrm{M}\\
&y^{(n-1)}(t_0) = y_{0n}
\end{aligned}
$$

Po přepsání do systému *n* rovnic 1. řádu substitucemi:

$$y_1 = y,\ y_2 = y',\ \mathrm{K}\ ,\ y_n = y^{(n-1)}$$

$$
\left.\begin{aligned}
y_1' &= y_2\\
y_2' &= y_3\\
&\mathrm{M}\\
y_n' &= g(t, y_1, y_2, \mathrm{K}\ , y_{n-1})
\end{aligned}\right\}\quad f(t, y) = \begin{bmatrix} y_2\\ y_3\\ \mathrm{M}\\ g(t, y)\end{bmatrix}
$$

$$
\left.\begin{aligned}
y_1(t_0) &= y_{01}\\
y_2(t_0) &= y_{02}\\
&\mathrm{M}\\
y_n(t_0) &= y_{0n}
\end{aligned}\right\}\quad y_0 = \begin{bmatrix} y_2\\ y_3\\ \mathrm{M}\\ y_{0n}\end{bmatrix}\quad .
$$

5. Metoda Monte Carlo

Metodou Monte Carlo se nazývá souhrn postupů dovolujících pomocí mnohonásobných náhodných pokusů získat řešení úloh z nejrůznějších oblastí vědy a techniky (matematiky, fyziky, chemie). Základem metody je experimentální simulace nějaké události.

Tato metoda patří mezi metody numerické matematiky pouze částečně. Při použití jiných numerických metod musíme znát algoritmus řešení úlohy a s jeho pomocí nalezneme hledanou veličinu s danou přesností. Celý postup výpočtu je přísně determinován. Metodou Monte Carlo řešíme úlohy tak, že pro každý problém sestrojíme náhodný proces se statistickými parametry (charakteristikami), které se rovnají hledaným veličinám úlohy. Provedeme řadu náhodných experimentů a výsledky najdeme statistickým vyhodnocením získaného materiálu. Při vyhodnocování se využívají zákony teorie pravděpodobnosti. Výpočtový proces je nedeterminovaný – řešení většinou nemůžeme přesně reprodukovat. Při opakování výpočtu postupně dochází k vyhlazování chyb.

Metoda se používá v případech, kdy přesný algoritmus řešení úlohy neznáme nebo je příliš složitý. K provedení výpočtu stačí znát celý komplex podmínek, za kterých k danému jevu dojde, a tyto podmínky napodobit. Náhodný proces nebývá realizován skutečným experimentem, ale modelováním v samočinném počítači, tj. prováděním operací s náhodnými čísly. Na základě analýzy skutečného procesu vybudujeme dostatečně jednoduchý model, který již našimi prostředky dokážeme vyřešit.

Základním a nejběžnějším obratem metody je modelování takové náhodné veličiny ξ, jejíž střední hodnota $E\xi$ je rovna hledané veličině.

Budeme hledat neznámou hodnotu *a*. Podle schématu metody Monte Carlo vytvoříme náhodnou veličinu ξ se střední hodnotou $E\xi = a$. Očekávanou hodnotu $E\xi$ (a tím i hledané *a*) odhadneme tak, že provedeme *n* nezávislých realizací náhodné veličiny $\xi : \xi_1, \mathrm{K}\ , \xi_n$. Podle zákona velkých čísel platí, že jejich průměr

$$\overline{\xi}_n = \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i$$

pro dosti velká *n* se blíží k *a*. Dále budeme předpokládat, že veličina ξ má konečný rozptyl $D\xi = E\left(\xi^2\right) - \left(E\xi\right)^2$. Za pomoci centrální limitní věty teorie pravděpodobnosti dostaneme

$$\lim_{n\to\infty} P\left\{ -x < \frac{1}{\sqrt{n \cdot D\xi}} \cdot \sum_{i=1}^{n} \left(\xi_i - a\right) < x \right\} = 2 \cdot \Phi_0\left(x\right) .$$

V tomto vztahu je integrál pravděpodobnosti

$$\Phi_0\left(x\right) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot \int_0^x \exp\left(-\frac{t^2}{2} \right) \cdot dt \ .$$

Jeho hodnoty též můžeme určit pomocí funkce $\Phi\left(x\right)$. V tabulce je uvedeno několik hodnot.

Hodnoty integrálu pravděpodobnosti

x	0,675	1,212	1,645	1,96	2,58	3,0	3,29
$2 \cdot \Phi_0(x)$	0,50	0,80	0,90	0,95	0,99	0,997	0,999

Pro dosti velká *n* tedy platí

$$P\left\{ \left| \overline{\xi}_n - a \right| < x \cdot \sqrt{\frac{D\xi}{n}} \right\} = 2 \cdot \Phi_0\left(x\right) .$$

Při použití tohoto vzorce pro odhad přesnosti výpočtu si nejprve stanovíme dovolené riziko α a pak z tabulky najdeme hodnotu *x* vyhovující vztahu

$$2 \cdot \Phi_0\left(x\right) = 1 - \alpha \ .$$

Jelikož náhodná veličina $\overline{\xi}_n$ je přibližně normální se směrodatnou odchylkou $\sigma = \sqrt{D\xi / n}$, nejčastěji se volí $1 - \alpha = 0{,}997$, čemuž odpovídá $x = 3$. Tato volba bývá v reálném případě zbytečně přísná a určí horní hranici chyby. Někdy používaná pravděpodobná chyba pracující s $\alpha = 0{,}5$ je rovna

$$\delta_{pravd} = 0{,}67 \cdot \sqrt{\frac{D\xi}{n}} \ .$$

Z uvedených odhadů vidíme, že chyba metody se s růstem počtu pokusů *n* zmenšuje jako $\frac{1}{\sqrt{n}}$. Existují sice speciální postupy, jež dovolují rychlost konvergence poněkud zvýšit, přesto však bývá vždy nižší než $\frac{1}{n}$. Proto k dosažení dostatečné přesnosti je třeba velkého počtu pokusů, což je základním omezením metody Monte Carlo.

Na počátku výpočtu nebývá rozptyl modelované veličiny ξ znám, je jej však možno odhadnout i v průběhu výpočtu. K tomu je nutno spolu s výrazem $\sum_i \xi_i$ počítat i $\sum_i (\xi_i)^2$. Pro dosti velká *n* platí ($n > 30$)

$$\frac{1}{n} \cdot \sum_{i=1}^{n} (\xi_i)^2 \approx E(\xi^2) \ ,$$

neboli pro odhad rozptylu dostaneme vztah

$$D\xi \approx \frac{1}{n} \cdot \sum_{i=1}^{n} (\xi_i)^2 - \left[\frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i \right]^2 \quad .$$

Chybu můžeme přibližně odhadnout v průběhu modelování i bez výpočtu rozptylu. K tomu stačí hledanou sumu $\sum_i \xi_i$ rozdělit na několik částí a dílčí součty spolu porovnat.

5.1. Generování náhodných čísel

Výpočetní schéma metody Monte Carlo předpokládá modelování náhodného procesu pomocí operací s náhodnými čísly. Tato čísla bývají získávána několika způsoby. K dispozici jsou různé tabulky náhodných čísel, je možno i zkonstruovat generátory náhodných čísel, založené na analýze náhodných fyzikálních pochodů (šumu elektronických prvků, radioaktivního rozpadu, apod.).

V současné době, kdy jsou úlohy metodou Monte Carlo téměř výhradně řešeny na samočinném počítači, rozhodujícím se stal způsob třetí – pomocí pseudonáhodných čísel. V pravém slova smyslu se nejedná o náhodná čísla, neboť jsou získávána přísně determinovaným způsobem, zato však je výpočet možno provádět přímo počítačem a získat tato čísla rychle a v dostatečném počtu.

Postupně byla vytvořena celá řada algoritmů, jejichž použití dává posloupnost pseudonáhodných čísel s dobrými statistickými vlastnostmi. Většina algoritmů vede na rekurentní vzorec prvního řádu:

$$\xi_{i+1} = g \cdot \xi_i \left(\mathrm{mod}\ M\right).$$

Pro hodnoty $\xi_0 = 1,\ g = 5^{17}\ a\ M = 2^{42}$ je pak perioda opakování posloupnosti pseudonáhodných čísel 2^{40}. Užívají se však i rekurentní vzorce vyšších řádů, kdy nové pseudonáhodné číslo ξ_{i+1} určíme na základě operací s *r* předchozími čísly

$$\xi_i,\ \xi_{i-1},\ \mathrm{K}\ ,\ \xi_{i-r+1}\ .$$

V algoritmech, kde pracujeme s náhodnými (i pseudonáhodnými) čísly, má konvergence pravděpodobnostní charakter. Existují však i speciální čísla (budeme jim říkat „kvazináhodná“), jejichž použití v algoritmech metody Monte Carlo přináší řadu výhod – konvergence je vždy zajištěna, nejsou nutné statistické testy a konvergence je téměř rychlostí $\frac{1}{n}$. Okruh problémů, pro které jsou kvazináhodná čísla vhodná, je ale omezen, a tato čísla se mohou používat pouze v zadaném pořadí. Příkladem takové posloupnosti kvazináhodných čísel rozdělených rovnoměrně v intervalu $(0,1)$ je

$$\frac{1}{2},\ \frac{1}{4},\ \frac{3}{4},\ \frac{1}{8},\ \frac{5}{8},\ \frac{3}{8},\ \frac{7}{8},\ \frac{1}{16},\ \mathrm{K}$$

nebo

$$\frac{1}{3},\ \frac{2}{3},\ \frac{1}{9},\ \frac{4}{9},\ \frac{7}{9},\ \frac{2}{9},\ \frac{5}{9},\ \frac{8}{9},\ \mathrm{K}\quad .$$

5.2. Transformace náhodné veličiny

Předpokládejme, že máme k dispozici náhodnou veličinu γ rovnoměrně rozdělenou na intervalu $(0, 1)$. Další parametry rozdělení jsou

$$E\gamma = \frac{1}{2} \qquad D\gamma = \frac{1}{12} \quad .$$

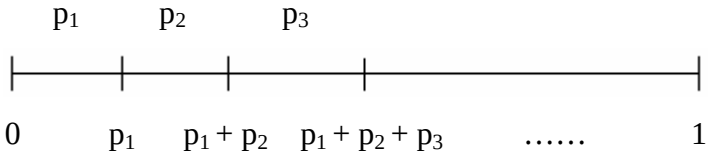
Při modelování náhodného procesu chceme pracovat s náhodnou veličinou ξ obecně nerovnoměrně rozdělenou na intervalu (a, b) s hustotou pravděpodobnosti $p_\xi(x)$ (s distribuční funkcí $F_\xi(x)$), případně může veličina ξ být i diskrétní. Je proto nutno veličinu γ na ξ transformovat – říká se též „rozehrát náhodnou veličinu ξ “. Metod transformace je několik:

1) Rozehrávání diskrétní náhodné veličiny

Veličinu ξ budeme mít zadánu tabulkou

$$\xi = \begin{pmatrix} x_1 & x_2 & \text{K} & x_n \\ p_1 & p_2 & \text{K} & p_n \end{pmatrix} \quad .$$

Vytvoříme vektor o *n* složkách: $p_1,\ p_1 + p_2,\ p_1 + p_2 + p_3, \text{K}\ , 1$.



Pak nagenerujeme jednu hodnotu γ a určíme, do kterého intervalu padne, to znamená, že budeme vyšetřovat podmínku

$$\gamma < \sum_{i=1}^{j} p_i \quad .$$

První interval *j*, pro který bude tato podmínka splněna, určí příslušnou hodnotu $\xi = x_j$.

2) Metoda inverzní funkce

Pro získání hodnoty spojité náhodné veličiny ξ musíme pro jednotlivé hodnoty γ vyřešit vzhledem ke ξ jednoznačně rovnici

$$\int_a^{\xi} p(x) \cdot dx = \gamma \quad . \qquad (1.9.)$$

Pokud je integrál řešitelný analyticky, získáme transformační vzorec typu $\xi = g(\gamma)$.

3) Metoda výběru (metoda Neumannova):

V řadě případů nelze integrál (1.9.) vyřešit. Pak použijeme následující metodu: Stačí nám předpoklad, že hustota $p_\xi(x)$ je omezená na intervalu (a, b). Zvolíme M tak, aby platilo $p(x) \le M$, $x \in (a, b)$. Nagenerujeme dvě hodnoty náhodné veličiny γ: γ_1 *a* γ_2 a vytvoříme čísla

$$\eta_1 = a + \gamma_1 \cdot (b - a)$$
$$\eta_2 = M \cdot \gamma_2 \quad .$$

Bude-li bod P o souřadnicích (η_1, η_2) ležet pod křivkou $\gamma = p_\xi(x)$, tj. bude-li $\eta < p_\xi(\eta_1)$. Položíme $\xi = \eta_1$. Nebude-li podmínka splněna, nagenerujeme novou dvojici γ_1 *a* γ_2 a postup opakujeme. Účinnost metody závisí na hodnotě M, kterou je nutno zvolit co nejmenší

$$M = \sup_{a<x<b} p_\xi(x) \quad .$$

4) Metoda superpozice

Budeme hledat náhodnou veličinu ξ s distribuční funkcí $F(x) = \sum_{i=1}^{m} c_i \cdot F_i(x)$,

kde $F_i(x)$ jsou též distribuční funkce, $c_1 + \mathrm{K} + c_m = 1$ a všechna $c_i > 0$.

Zavedeme diskrétní náhodnou veličinu η s rozdělením

$$\eta = \begin{pmatrix} 1 & 2 & \ldots & m \\ c_1 & c_2 & \ldots & c_m \end{pmatrix} \quad .$$

Platí: nagenerujeme-li dvě nezávislé hodnoty γ_1 *a* γ_2 veličiny γ, rozehrajeme číslem γ_1 hodnotu $\eta = k$ a pak z rovnice $F_k(\xi) = \gamma_2$ určíme ξ, bude distribuční funkce veličiny ξ rovna $F(x)$.

Této metody můžeme použít, bude-li výpočet pomocí metody inverzní funkce příliš složitý.

5) Modelování n-rozměrného náhodného bodu

Budou-li souřadnice bodu nezávislé, můžeme každou z nich modelovat zvlášť pomocí dříve uvedených metod.

Pokud máme generovat body z oblasti složitého průběhu, je vhodné zobecnit metodu Neumannovu. Zvolíme jednoduchou oblast, která v sobě obsahuje zadanou oblast. Body pak generujeme v této nové větší oblasti
a testujeme, padnou-li do naší původní oblasti. Pouze takové body zachováme a ostatní vyloučíme.

6) Transformace typu $\xi = g(\gamma_1, \mathrm{K}\ , \gamma_n)$

Kromě zatím uvedených transformačních metod existuje množství dalších algoritmů, které konstruují náhodnou veličinu ξ pomocí několika $(n \geq 2)$ hodnot veličiny γ.

Příklad:

Nalezněte náhodnou veličinu ξ definovanou na intervalu $(0, 2)$ s hustotou pravděpodobnosti $p(x) = \frac{5}{12} \cdot \left[1 + (x-1)^4\right]$.

Použijeme-li metodu inverzní funkce, budeme muset vyřešit vzhledem ke ξ rovnici $(\xi - 1)^5 + 5 \cdot \xi = 12 \cdot \gamma - 1$. Proto dáme přednost metodě superpozice. Hustotu pravděpodobnosti $p(x)$ složíme ze dvou hustot

$$p_1(x) = \frac{1}{2} \qquad p_2(x) = \frac{5}{2} \cdot (x-1)^4 \qquad p(x) = \frac{5}{6} \cdot p_1(x) + \frac{1}{6} \cdot p_2(x) \quad .$$

Metoda nám dá následující jednoduchý algoritmus pro určení ξ

$$\xi = \begin{cases} 2 \cdot \gamma_2 & \gamma_1 < \frac{5}{6} \\ 1 + \sqrt[5]{2 \cdot \gamma_2 - 1} & \gamma_1 \geq \frac{5}{6} \end{cases} \quad .$$

5.3. Výpočet určitých integrálů

Metoda Monte Carlo je vhodná i pro řešení úloh, které mají obvyklé determinované algoritmy. Teorie je rozpracována pro výpočet jednoduchých i
násobných integrálů, pro řešení soustav lineárních algebraických rovnic, pro operace s maticemi, pro řešení diferenciálních a integrálních rovnic, apod.

Úkolem bude vypočítat integrál z funkce $f(x)$ na vlastním intervalu $\langle a, b \rangle$

$$I = \int_a^b f(x) \cdot dx \quad .$$

K dispozici máme dvě jednoduché metody.

1) Výpočet střední hodnoty funkce

Metoda určení hodnoty integrálu I spočívá v tom, že vezmeme náhodnou veličinu ξ rovnoměrně rozdělenou na intervalu $\langle a, b \rangle$ a zavedeme náhodnou veličinu $\eta = f(\xi)$, jejíž matematické očekávání $E\eta$ je rovno průměrné hodnotě funkce $f(x)$ na intervalu $\langle a, b \rangle$. Hodnotu $E\eta$ odhadneme pomocí aritmetického průměru *n* nezávislých realizací veličiny η. Potom

$$I \doteq (b-a) \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} f(\xi_i) \quad .$$

2) Geometrická metoda

V této metodě je počítána plocha pod křivkou $y = f(x)$. Budeme předpokládat, že funkce $f(x)$ je na celém intervalu $\langle a, b \rangle$ omezená. Vezmeme dvě náhodné veličiny: $\xi \in \langle a, b \rangle$ a $\eta \in \langle 0, c \rangle$ rovnoměrně rozdělené na příslušných intervalech. Zkonstruujeme *n* náhodných bodů (ξ, η) a budeme určovat, kolik z nich leží pod křivkou $y = f(x)$. Jejich počet označíme n'. Potom

$$I \doteq c \cdot (b-a) \cdot \frac{n'}{n} \quad .$$

Srovnání obou metod ukazuje, že geometrická metoda má větší (nebo v optimálním případě stejný) rozptyl než metoda založená na výpočtu střední hodnoty integrované funkce a bývá proto většinou méně přesná.

Při volbě metody dáváme přednost postupu s jehož použitím rychleji nalezneme výsledek se zadanou chybou. Pouhé zjištění velikosti rozptylu $D\xi$ k tomu nestačí, musíme ještě vzít v úvahu rychlost výpočtu jedné hodnoty integrované funkce τ. Kritériem vhodnosti metody pak bude součin $\tau \cdot D\xi$.

Příklad:

Porovnejte vhodnost obou jednoduchých metod integrace pro integrál

$$I = \int_0^1 \sqrt[5]{x} \cdot dx \quad .$$

Tento integrál je roven $\frac{5}{6}$. Algoritmus založený na určení střední hodnoty funkce dává pro výpočet I vztah

$$I \approx \frac{1}{n} \cdot \sum_{i=1}^{n} \sqrt[5]{\gamma_i} \quad .$$

Rozptyl náhodné veličiny $\eta = (\gamma)^{\frac{1}{5}}$ bude

$$D \cdot \eta = \int_0^1 x^{\frac{2}{5}} \cdot dx - \left(\frac{5}{6}\right)^2 = \frac{5}{252} \quad .$$

Při použití geometrické metody s $c = 1$ dostaneme pro I vztah

$$I \approx \frac{1}{n} \cdot \sum_{i=1}^{n} \xi_i \quad ,$$

kde

$$\xi_i = 1 \qquad pro \ \gamma_i' < \gamma_i^{\frac{1}{5}}$$
$$= 0 \qquad pro \ \gamma_i' \geq \gamma_i^{\frac{1}{5}} \quad .$$

Rozptyl

$$D \cdot \xi = c \cdot I - I^2 = \frac{5}{36} \quad .$$

Pouhé srovnání rozptylů dává jednoznačně přednost metodě založené na určení střední hodnoty funkce $f(x)$. Budeme-li však uvažovat i počet operací potřebných pro získání jednoho členu v součtech tvořících odhady *I*, bude situace složitější. V první metodě musíme nagenerovat náhodné číslo γ_i a počítat $\gamma_i^{\frac{1}{5}}$. V druhé metodě musíme nagenerovat dvě náhodná čísla γ_i a γ_i' a studovat podmínku $\gamma_i' < \gamma_i^{\frac{1}{5}}$.

Pokud ji převedeme na ekvivalentní podmínku $(\gamma_i')^5 < \gamma_i$, bude kvůli obvykle pomalému postupu výpočtu páté odmocniny počítačem i přes větší rozptyl geometrická metoda vhodnější. Z uvedeného příkladu vidíme, že volba nejvhodnějšího algoritmu v metodě Monte Carlo není vždy věcí jednoduchou, dosažené úspory času však mohou být značné.

Konvergence obvyklé metody Monte Carlo je dosti pomalá. Je proto vhodné hledat postupy zvyšující přesnost výpočtu jiným způsobem než růstem počtu pokusů *n*. Existuje řada metod snižujících rozptyl:

1) Nalezení hlavní části

Integrál se budeme snažit rozdělit na dvě části – jednu dávající hlavní vklad do výsledku a druhou jako upřesňující korekci. Hlavní část se vypočítá analyticky nebo jinou numerickou metodou a korekci určíme pomocí metody Monte Carlo.

Pro výpočet integrálu najdeme funkci $g(x)$ blízkou funkci $f(x)$, jejíž integrál známe

$$I_1 = \int_a^b g(x) \cdot dx \quad .$$

Dále budeme pracovat s náhodnou veličinou

$$\eta = (a-b) \cdot [f(\xi) - g(\xi)] + I_1 \quad .$$

Pak

$$I \approx \frac{b-a}{n} \cdot \sum_{i=1}^{n} [f(\xi_i) - g(\xi_i)] + I_1 \quad .$$

Rozptyl veličiny η bude

$$D \cdot \eta = (b-a) \cdot \int_a^b [f(x) - g(x)]^2 \cdot dx - (I - I_1)^2 \quad .$$

2) Metoda váženého výběru

Při této metodě nebudeme volit náhodná čísla ξ v intervalu $\langle a, b \rangle$ rozdělená rovnoměrně, ale v oblasti přispívající více k hodnotě integrálu I je budeme generovat s větší hustotou. V nejjednodušším případě rozdělíme interval $\langle a, b \rangle$ na několik částí a v každé z nich vezmeme jinou náhodnou veličinu rovnoměrně rozdělenou pouze v tom dílčím intervalu. Hustotu rozdělení volíme podle velikosti integrované funkce v dílčím

intervalu. Výsledný odhad pro integrál *I* dostaneme jako součet výsledků z jednotlivých podintervalů. Této metodě se také říká metoda výběru po skupinách.

V obecnějším postupu volíme náhodnou veličinu ξ s hustotou pravděpodobnosti $p(x)$ spojitě se měnící v intervalu $\langle a, b \rangle$. Hledaný integrál upravíme do tvaru

$$I = \int_a^b f(x) \cdot dx = \int_a^b \frac{f(x)}{p(x)} \cdot p(x) \cdot dx \quad .$$

Zavedeme-li novou náhodnou veličinu $\eta = \frac{f(\xi)}{p(\xi)}$, bude pro ni platit $E\eta = 1$.

Jako odhad integrálu *I* proto můžeme vzít

$$I \approx \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{f(\xi_i)}{p(\xi_i)} \quad .$$

Vhodnou volbou hustoty pravděpodobnosti $p(x)$ nyní můžeme podstatně zmenšit rozptyl. Rozptyl náhodné veličiny η bude

$$D\eta = \int_a^b \frac{f^2(x)}{p(x)} \cdot dx - I^2 \quad .$$

Optimální volba hustoty pravděpodobnosti veličiny ξ je

$$p(x) = \frac{|f(x)|}{\int_a^b |f(x)| \cdot dx} \quad ,$$

kdy může rozptyl klesat dokonce až na nulu. V reálném případě budeme volit takovou náhodnou veličinu ξ, kterou ještě můžeme dostatečně jednoduše rozehrát a jejíž hustota pravděpodobnosti se bude co nejvíce podobat $|f(x)|$.

3) Metoda symetrizace integrované funkce

Obecně platí, že rozptyl bude tím menší, čím pomaleji se bude integrovaná funkce na intervalu $\langle a, b \rangle$ měnit. Budeme se proto snažit integrovanou funkci upravit tak, aby rozptyl poklesl. Konkrétní úprava závisí na tvaru funkce $f(x)$.

Je-li funkce $f(x)$ monotónní, je jí vhodné symetrizovat

$$g(x) = \frac{1}{2} \cdot [f(x) + f(a+b-x)] \quad .$$

Integrál pak budeme přibližně počítat podle vztahu

$$I \approx \frac{1}{2 \cdot n} \cdot \sum_{i=1}^{n} [f(\xi_i) + f(a+b-\xi_i)] \quad .$$

Doba výpočtu jednoho členu se prodlouží přibližně dvakrát, rozptyl však klesne ještě více. V případě funkcí složitějšího průběhu je nutno symetrizaci provádět též složitěji. Pro úspěšné použití uvedené metody je proto třeba nejprve získat co nejvíce informací o integrované funkci.

6. MATLAB

Název Matlab je zkratka odvozená z názvu MATrix LABoratory. Systém obsahuje vlastní interpret jazyku *MATLAB,* ve kterém lze připravit jak dávkové soubory, tak definovat i nové funkce. Tyto funkce mohou být interpretovány buď přímo z textové podoby souborů, nazývaných *m-file* nebo z předzpracované podoby *p-file*. Systém dále umožňuje přidávat moduly (soubory *mex-file*) zkompilované do strojového kódu procesoru. Jazykem zdrojových souborů může být jazyk *C*, *C++*, *Fortran* nebo po použití *mcc* (Matlab to C Compiler) i funkce uložená v *m-file*.

Asi nejdůležitější částí instalace **Matlab**u jsou "knihovny" funkcí (ve skutečnosti adresáře s *m* a *mex* soubory), které jsou nazývány *toolboxy*. *Toolboxy* obsahují vždy uceleným způsobem včetně dokumentace a příkladů zpracovaný určitý obor numerické matematiky, analytické matematiky, statistiky, systémového přístupu k regulacím a další obory, ve kterých nachází **Matlab** uplatnění.
Systém má uplatnění v typických oblastech:

- Matematické výpočty
- Vývoj algoritmů
- Modelování a simulace
- Analýza dat a vizualizace
- Vědecká a inženýrská grafika
- Vývoj aplikací včetně uživatelského interface

Funkce a jazyk Matlabu

Matlab podporuje funkce s různým počtem vstupních i výstupních parametrů. Při definici funkce jsou nadefinována formální jména vstupních a výstupních parametrů. Při volání funkce není nutné naplnit všechny vstupní parametry a ukládat hodnoty všech výstupních parametrů. V těle volané funkce jsou pak nepoužité vstupní parametry nedefinovány a funkce nesmí číst hodnoty těchto parametrů. K informaci o počtech parametrů slouží funkce **nargin** a **nargout** .

Důležitá upozornění: **Matlab** považuje veškerý obsah řádky za znakem **%** za komentář. Definice funkce musí být obsažena v první řádce souboru. Souvislý blok komentáře začínající od druhé řádky a označený znakem **%** v prvním sloupci bude

vypsán příkazem help *jméno_funkce* . Příkazy obsahující přiřazení nebo volání funkcí je ve funkcích vhodné ukončovat znakem **;** , jinak při každém provedení příkazu dojde k vypsání hodnot plněných proměnných.

Grafika

Nejjednodušším příkazem pro zobrazení závislosti dvou proměnných nebo průběhu jedné proměnné je příkaz **plot** . Zobrazovaná data jsou předávána ve formě sloupcových vektorů. Obecně má příkaz podobu **plot(X,Y,S,...)** , kde **X** a **S** mohou být vynechány. Vykreslení dvou period funkce sinus zajistí příkaz **plot(sin(0:0.01:4*pi)')** . Transpozice na sloupcový vektor není v tomto případě nezbytně nutná, plot je inteligentní. V takto vykresleném grafu bude mít osa x význam hodnoty indexu v předaném vektoru. Proto by bylo vhodnější nadefinovat proměnnou X=**[0:0.01:4*pi]'** dále proměnnou **Y=sin(X)** a použít delší formy příkazu **plot** . Textový parametr **S** specifikuje barvu a způsob vykreslení průběhu. Skupinu parametrů **X,Y,S** lze i několikrát opakovat Zároveň parametr Y nemusí být jen sloupcový vektor. Více sloupcových vektorů složených do matice provede vykreslení několika průběhů.

Matlab umožňuje vykreslit více grafů do jednoho okna, a to jak přes sebe (**hold on**) tak i vedle a pod sebe (**subplot**). Je možné libovolně měnit měřítka os, používat logaritmické a semilogaritmické zobrazení. Funkce **mesh** vykresluje z matic **3-D** grafy.

Aproximace dat polynomem *n*-tého stupně

Funkce LINREGRESE (resp. LINEST v anglické verzi Excelu) pomocí metody nejmenších čtverců vypočítá koeficienty přímky (y = m1x1 + m2x2 + ... + b), která nejlépe odpovídá zadaným datům (y, x1, x2...). Kromě vypočtených koeficientů funkce umí vracet i další regresní statistiky. V MATLABu lze samozřejmě data aproximovat také. V případě, že chceme zadaná (naměřená) data aproximovat polynomem *n*-tého stupně, můžeme to jednoduše provést pomocí funkce polyfit, která používá metodu nejmenších čtverců:

$$p = polyfit(x, y, n) \quad ,$$

kde

x je vektor hodnot nezávisle proměnné,

y je vektor hodnot závisle proměnné,

n je stupeň polynomu, jímž chceme aproximovat body $[x_i, y_i]$ a

p je vektor koeficientů výsledného polynomu $P(x)$, přičemž

$$P(x) = p(1)\cdot x^n + p(2)\cdot x^{n-1} + \mathrm{K} + p(n)\cdot x + p(n+1)$$

Poznámky:

- volba stupně polynomu většinou závisí na fyzikální podstatě problému
- pokud pro data délky *N* použijeme polynom stupně $N-1$, provádíme aproximaci interpolací (tj. polynom bude procházet všemi body $[x_i, y_i]$)
- pro polynom stupně *N* (a vyšších) úloha samozřejmě nemá jednoznačné řešení a funkce polyfit hlásí "Warning"
- pokud potřebujeme aproximovat jinou funkcí než polynomem, lze si M-soubor funkce polyfit upravit (a uložit pod jiným názvem) nebo použít metodu nejmenších čtverců, která vede k řešení soustavy lineárních rovnic. Uvedené postupy lze samozřejmě použít pouze pro modely, které jsou lineární v parametrech.

Protože nás kromě koeficientů aproximačního polynomu většinou zajímají také hodnoty polynomu $P(x)$ ve všech prvcích vektoru x (např. kvůli grafickému znázornění), existuje také funkce *polyval*:

$$y\_aprox = polyval(p, x) \quad ,$$

kde

p je vektor koeficientů aproximačního polynomu,
x je vektor hodnot nezávisle proměnné a
y_aprox je vektor hodnot aproximačního polynomu.

Součet čtverců odchylek pak vypočteme snadno - buď pomocí cyklu for (přes všechny prvky *y_aprox* a *y*), anebo jednoduše s využitím toho, že nás zajímá součet prvků vektoru, který vznikl umocněním každého prvku vektoru rozdílů mezi *y_aprox* a *y*:

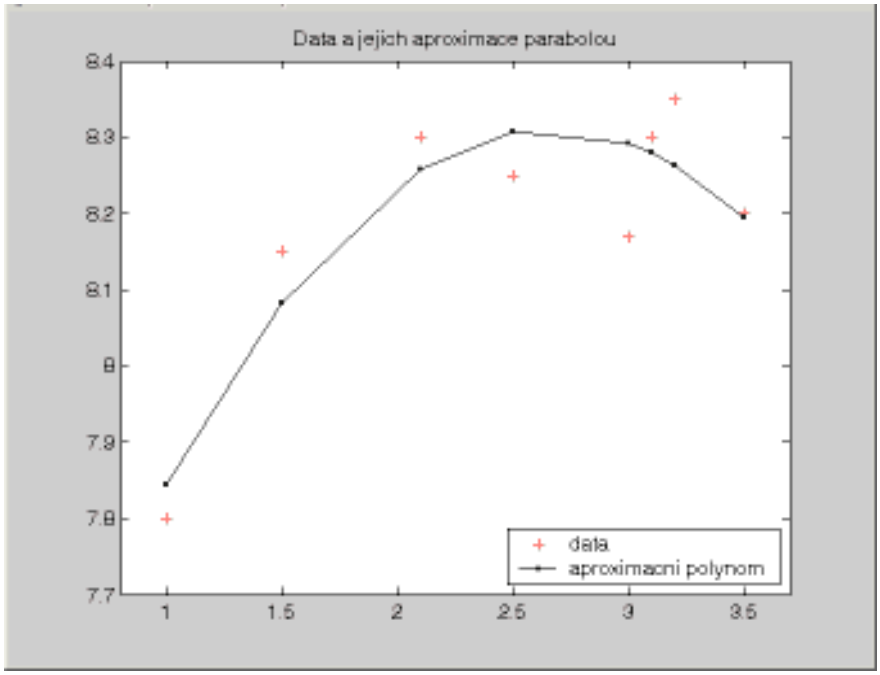
$$S = sum((y\_aprox - y).\hat{}2)$$

Příklad:

Chceme aproximovat 8 naměřených hodnot *u* a *v* polynomem 2. stupně:

```
>> u=[1 1.5 2.1 2.5 3 3.1 3.2 3.5]; % namerena data, nezav. prom.
>> v=[7.8 8.15 8.3 8.25 8.1 8.3 8.35 8.2]; % namerena data, zav. prom.
>> p=polyfit(u,v,2) % koeficienty polynomu 2. stupne pro 'u' a 'v'
p = -0.1684    0.8977    7.1150
>> v_aprox=polyval(p,u); % hodnoty polynomu v 'u'
>> S=sum((v_aprox-v).^2) % soucet ctvercu odchylek
S = 0.0345
>> plot(u,v,'r+') % graf, puvodni data jako cervene krizky
>> hold on % prikreslime dalsi
>> plot(u,v_aprox,'k.-') % graf polynomu jako cerne body spojene carou
>> axis([0.8 3.7 7.7 8.4]) % uprava os
>> title('Data a jejich aproximace parabolou')% nazev grafu
>> legend('data','aproximacni polynom',4) % zobrazime legendu (4 = vpravo dole)
```

A výsledek:



Stručný popis oken

Command Window – příkazové okno pro zadávání příkazů v jazyku Matlabu.
Workspace – zde se zobrazuje obsah paměti; je možné jednotlivé proměnné editovat.
Command History – dříve zadané příkazy; označení datem.
Current Directory – prohlížeč souborů a adresářů.
Launch Pad – rychlý přístup k hlavním součástem MATLABu.
Help / Matlab Help v hlavní nabídce - zobrazí nápovědu.
File / Set Path nastaví cesty k používaným adresářům.
File / Preferences – základní nastavení komponent Matlabu.

Některé matematické funkce

cos	kosinus
sin	sinus
tan	tangens
exp	exponenciála o zakladu e
log, log10, log2	logaritmus přirozený, o základu 10 a 2
sqrt	(druhá) odmocnina
abs	absolutní hodnota
imag, real, conj	imaginární a reálná část komplexního čísla, komplexně sdružené číslo
round, floor, ceil	zaokrouhlení, zaokrouhlení doleva, doprava

Operace s maticemi

u = [7 8 9]	řádkový vektor
v = [3; 4; 5]	sloupcový vektor
v = u’	transpozice vektoru
w = 2:8	ektor [2 3 4 5 6 7 8]
y = 2:3:20	vektor [2 5 8 11 14 17 20]
y = 3: **end**	prvky vektoru *y* s indexy od 3. do posledního
X = **linspace**(a, b, N)	řádkový vektor N čísel ekvidistantně rozložených mezi *a* a *b*, $X(1)=a,\ X(N)=b$
X = **logspace**(a, b, N)	řádkový vektor N čísel exponenciálně rozložených mezi *a* a *b*, $X(1)=a,\ X(N)=b$

A = [1 2 3; 5 6 7]	matice o 2 řádcích a 3 sloupcích
B = **sparse**(A)	řídká matice
eye(3)	jednotková matice 3x3
ones(3)	matice jedniček
zeros(3)	matice nul
diag(w)	diagonální matice s vektorem w na diagonále
diag(A)	diagonála matice *A* (vektor)
spdiags(B, d, m, n)	řídká matice typu *m x n* vytvořená ze sloupců matice *B* umístěných na pozice určené vektorem *d*
full(A)	zobrazení řídké matice v „plném“ tvaru
spy(A)	schematické zobrazení „zaplnění“ matice *A*
rand(3,4)	matice typu 3 x 4 náhodných čísel mezi 0 a 1 (rovnoměrné rozdělení)
rand('state',0)	nastavení generátoru náhodných čísel do stavu 0
randn(3)	matice náhodných čísel s normálním rozdělením (střední hodnota je 0, rozptyl je 1)
ones(4, 8)	obdélníková matice jedniček
zeros(**size**(B))	nulová matice stejného typu jako *B*
M = **zeros**(3, 4, 5)	trojrozměrné pole nul

Kreslení

fplot('sin', [-2 10])	vykreslí funkci
x = 0 : pi/100 : 2*pi; y = sin(x); **plot**(x, y)	vykreslí body o souřadnicích *xi*, *yi* spojené lomenou čarou
plot(y)	vykreslí body o souřadnicích *i*, *yi* spojené lomenou čarou
plot(x, y, ':', x, sin(x-0.3), 'b--')	vykreslí dva grafy do stejného obrázku, první tečkovaně, druhý čárkovaně modře
plot(A)	vykreslí sloupce matice (několik sérií bodů spojených lomenou čarou)
plot(Z)	je-li *Z* vektor komplexních čísel, vykreslí se body o souřadnicích *real(Z)* a *imag(Z)*
t=0:0.2:2*pi; **plot**(cos(t)+i*sin(t), '*')	kreslení komplexního vektoru (kružnice)
t = 0 : pi/50 : 10*pi; **plot3**(sin(t), cos(t), t)	parametrické 3-D kreslení

t = 0:.01:2*pi; **polar**(t, abs(sin(2 * t) .* cos(2 * t)));	parametrické kreslení
[x, y, z] = **pol2cart**(theta, r, z)	převedení z polárních nebo cylindrických souřadnic do kartézských
[theta, r] = **cart2pol**(x, y)	převedení z kartézských souřadnic do polárních nebo cylindrických
[x, y, z] = **sph2cart**(theta, phi, r)	převedení ze sférických souřadnic do kartézských
[theta, phi, r] = **cart2sph**(x, y, z)	převedení z kartézských souřadnic do sférických
xlabel('rychlost [m/s]')	označení vodorovné osy
ylabel('čas [s]')	označení svislé osy
title('Pohyb kyvadla')	nadpis grafu
axis square, **axis**([-pi pi -1 1])	nastavení os souřadnic
grid on, **grid** off	zapnutí / vypnutí zobrazení souřadnic
text(1, 1.5,'{\itNalezněte lokální maxima.}')	umístění textu do grafu, používá se jazyka TeX
hold on	výstup do existujícího (posledního vytvořeného) grafu
hold off	překreslení předchozího grafického výstupu
subplot(m, n, p)	rozdělení aktuálního grafického okna na *mxn* částí v *m* řádcích a *n* sloupcích a nastavení grafického výstupu do p-tého z nich (počítáno po řádcích)
cla	vymazání aktivního grafu
clf	vymazání aktivního grafického okna

Polynomy

p = [1 –2 8]	reprezentace polynomu pomocí vektoru jeho koeficientů (v pořadí od nejvyšší mocniny)
polyval(p, A)	vyčíslení polynomu *p* v prvcích matice *A*
polyvalm(p, A)	vyčíslení polynomu v matici *A* (dosazení *A* za nezávisle proměnnou)
roots(p)	kořeny polynomu *p*
p = **polyfit**(x, y, n)	aproximace dat $[x, y]$ polynomem stupně *n* ve smyslu nejmenších čtverců

Optimalizační funkce

p = **polyfit**(x, y, n)	aproximace dat $[x, y]$ polynomem stupně *n* ve smyslu nejmenších čtverců
yk = **pchip**(x, y, xk)	interpolace po částech kubickým Hermiteovým polynomem
yk = **spline**(x, y, xk)	interpolace kubickou spline funkcí
lsqcurvefit(@F, [x0], xdata, ydata)	nelineární interpolace (metodou nejmenších čtverců); řeší úlohu najít *x*, pro které výraz *norm(F(x,xdata) – ydata, 2)^2* nabývá svého minima
x = **fsolve**(@F, x0)	řešení rovnice $F(x) = 0$; počáteční přiblížení x0

Funkce pro přibližné řešení diferenciálních rovnic

[T, Y] = **ode45**(@F, [0 1], [a b])	metoda Runge-Kutta 4.řádu pro soustavy diferenciálních rovnic
odeset('AbsTol', 1e-6)	nastavení parametrů pro řešiče diferenciálních rovnic

7. Příklady řešené pomocí MATLABU

Příklad 1:

Harmonický oscilátor, řešíme pomocí diferenciálních rovnic 2. řádu pomocí Matlabu.

$$\frac{d^2x}{dt^2} - \omega^2 \cdot x \qquad \left(\omega = \sqrt{\frac{k}{m}} \;\; vlastní \;\; frekvence \;\; oscilátoru \right)$$

Počáteční podmínky:

$$x(0) = x_0 \,, \; v(0) = v_0$$

Převedeme rovnici na soustavu 2 rovnic 1. řádu pomocí substituce $y_1 = x(t), \; y_2 = x'(t)$.

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -\omega^2 \cdot y_1 \end{aligned}$$

pak

$$\begin{aligned} y_1(0) &= x_0 \\ y_2(0) &= v_0 \quad . \end{aligned}$$

Vytvoříme funkci *harm_osc.m*, vracející pravou stranu soustavy diferenciálních rovnic:

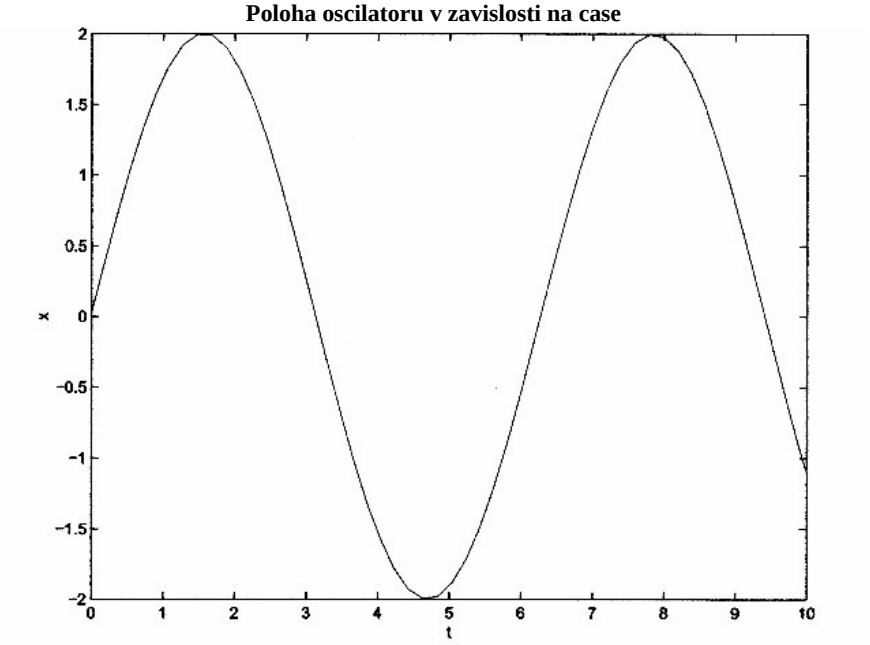
```
function dydt = harm_osc ( t, y)
omega2 = 1;
dydt = [y(2); -omega2 * y(1)];
```

V příkazovém okně definujeme časový interval, počáteční podmínky a soustavu řešíme:

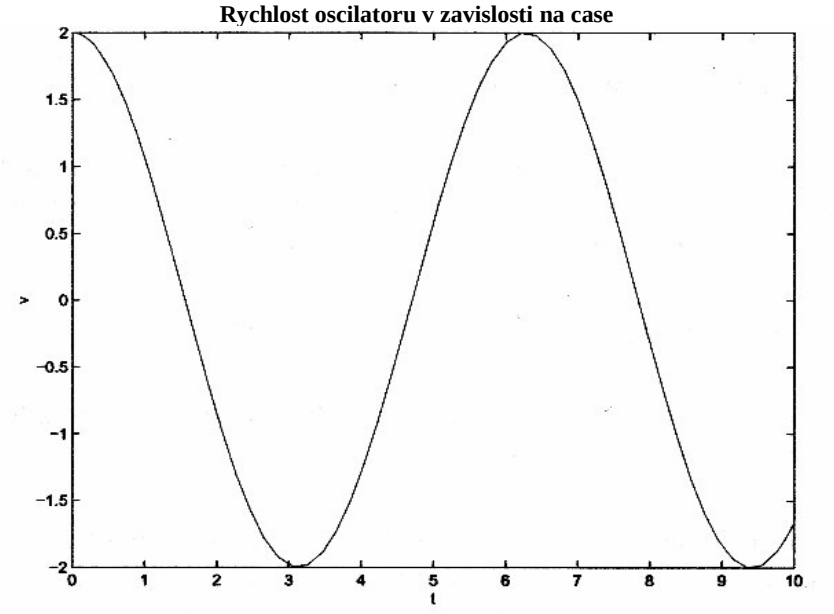
```
>> tinterval = [0 10];
>> y0 = [0;2];
>> [t, y] = ode45(@harm_osc, tinterval, y0);
```

Grafy *x(t), v(t)*zobrazime takto:

```
>> tinterval = [010];
>> y0 = [0;2];
>> [t, y] = ode45(@harm_osc, tinterval, y0);
>> figure;
>> plot(t, y(:, 1));
>> xlabel(´t´); ylabel(´x´);
>> title(´Poloha oscilatoru v zavislosti na case´);
```




```
>> figure;
>> plot(t, y(:,2));
>> xlabel(´t´); ylabel(´v´);
>> title(´Rychlost oscilatoru v zavislosti na case´);
```



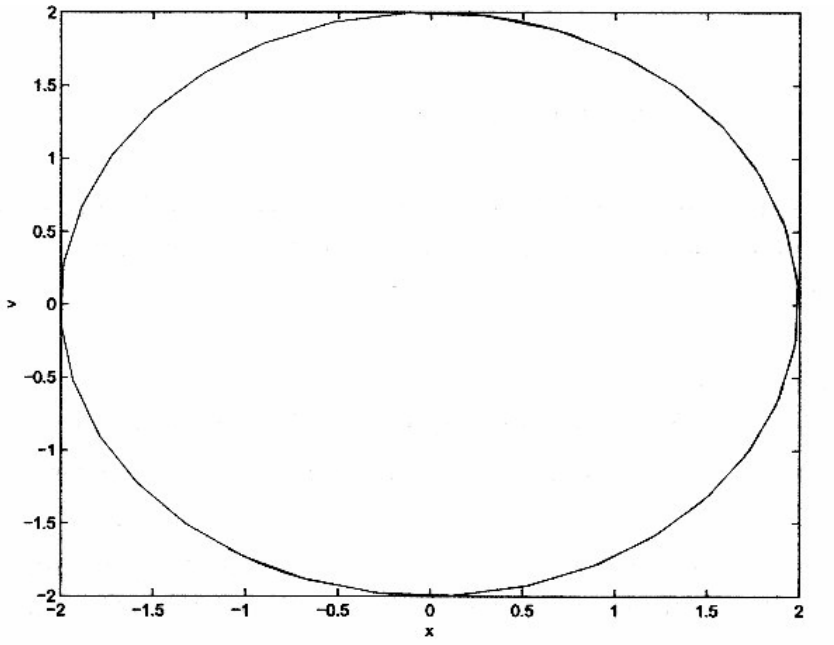
Vytvoříme fázový portrét, který dává do vzájemné souvislosti polohu *x* (vodorovná osa) s rychlostí *v* (svislá osa):

>> *figure*;

>> *plot*(*y*(:,1), *y*(:,2));

>> *xlabel*(´*x*´); *ylabel*(´*v*´);

>> *title*(´Fazovy portret harmonickeho oscilatoru´);



Příklad 2:

Z voltampérové charakteristiky diody jsme naměřili body dle tabulky.

U [V]	I [A]
0	0
0,332031	$3{,}1\cdot10^{-5}$
0,385742	$1{,}13\cdot10^{-4}$
0,405273	$2{,}00\cdot10^{-4}$
0,424804	$3{,}40\cdot10^{-4}$
0,43457	$4{,}65\cdot10^{-4}$
0,444336	$5{,}89\cdot10^{-4}$
0,454102	$7{,}32\cdot10^{-4}$
0,463868	$9{,}11\cdot10^{-4}$

Řešení:

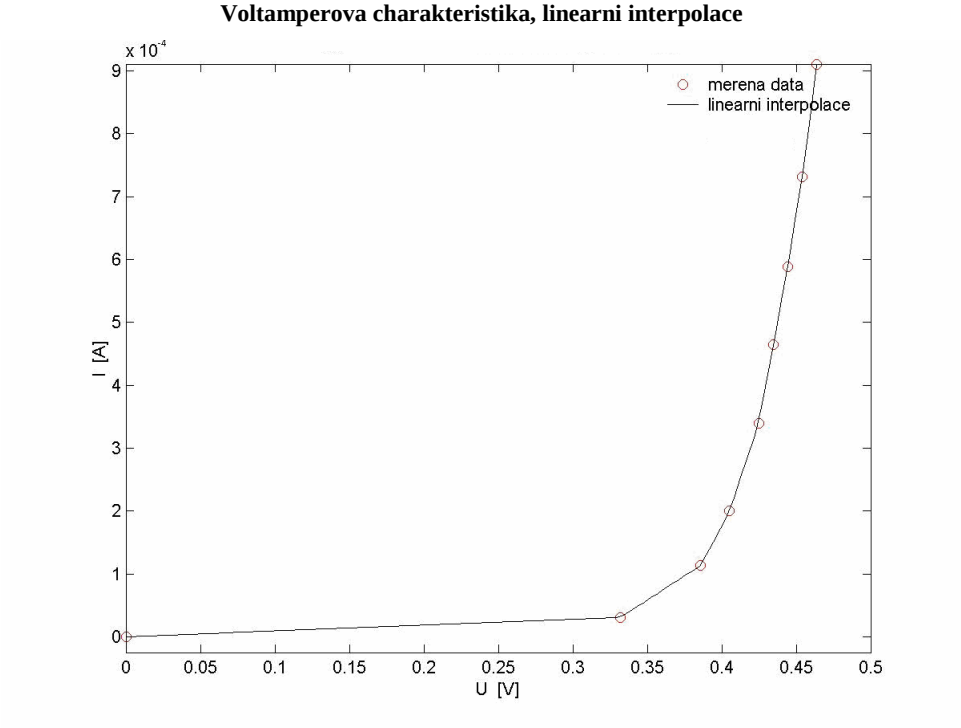
1) Řešíme pomocí lineární interpolace:

```
U = [ 0 0.332031 0.385742 0.405273 0.424804 ...          % [U] = V
    0.43457  0.444336 0.454102 0.463868 ];
I = [ 0 3.10e-5 1.13e-4 2.00e-4 3.40e-4 ...              % [I] = A
    4.65e-4 5.89e-4 7.32e-4 9.11e-4 ];
% jemne deleni napeti pro zobrazeni prokladanych krivek
UU = U(1) : (U(end)-U(1))/100 : U(end);

II = interp1(U,I,UU,'linear');
figure;
```

Nyní vytvoříme graf

```
plot(U,I,'or',UU,II,'k');
legend('merena data','linearni interpolace');
legend boxoff;
% protazeni osy y do zapornych hodnot
        c = axis;
if c(3) >= 0
  set(gca,'YLim',[c(3)-(c(4)-c(3))/40,Inf]);
end
```

```
xlabel('U  [V]');
ylabel('I  [A]');
title('Voltamperova charakteristika, linearni interpolace');
```



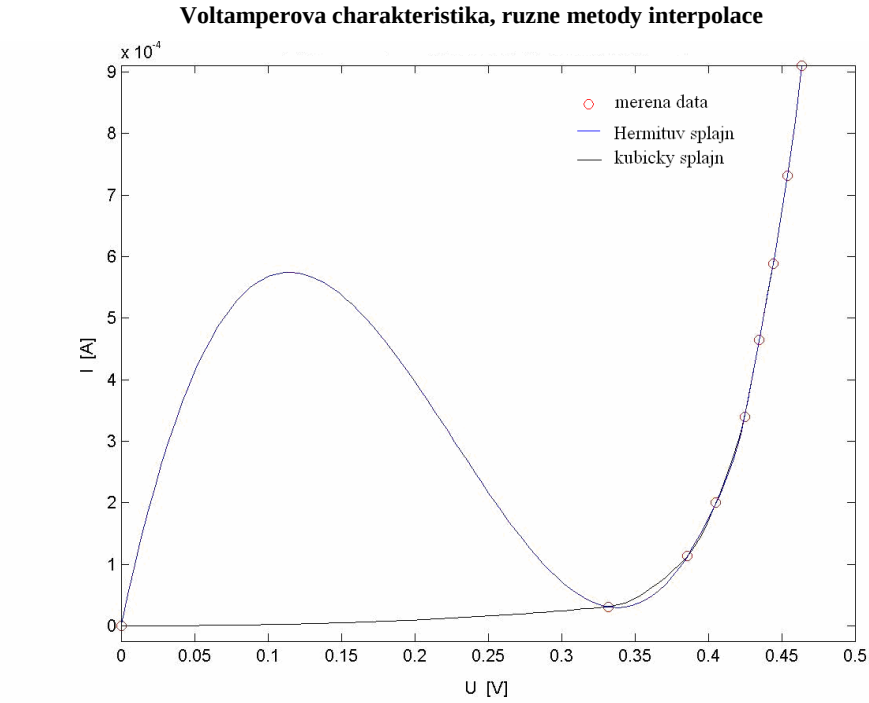
2) Řešíme různými metodami interpolace:

```
U = [ 0 0.332031 0.385742 0.405273 0.424804 ...          % [U] = V
      0.43457  0.444336 0.454102 0.463868 ];
I = [ 0 3.10e-5 1.13e-4 2.00e-4 3.40e-4 ...              % [I] = A
      4.65e-4 5.89e-4 7.32e-4 9.11e-4 ];
% jemne deleni napeti pro zobrazeni prokladanych krivek
UU = U(1) : (U(end)-U(1))/100 : U(end);


II1 = interp1(U,I,UU,'pchip');
II2 = interp1(U,I,UU,'spline');
figure;
```

Vytvoříme graf

```
plot(U,I,'or',UU,II1,'k',UU,II2,'b');
legend('merena data','Hermituv splajn','kubicky splajn');
```

```
legend boxoff;
% protazeni osy y do zapornych hodnot
c = axis;
if c(3) >= 0
  set(gca,'YLim',[c(3)-(c(4)-c(3))/40,Inf]);
end
xlabel('U  [V]');
ylabel('I  [A]');
title('Voltamperova charakteristika, ruzne metody interpolace');
```



3) Řešíme pomocí fitace kubickým splajnem:

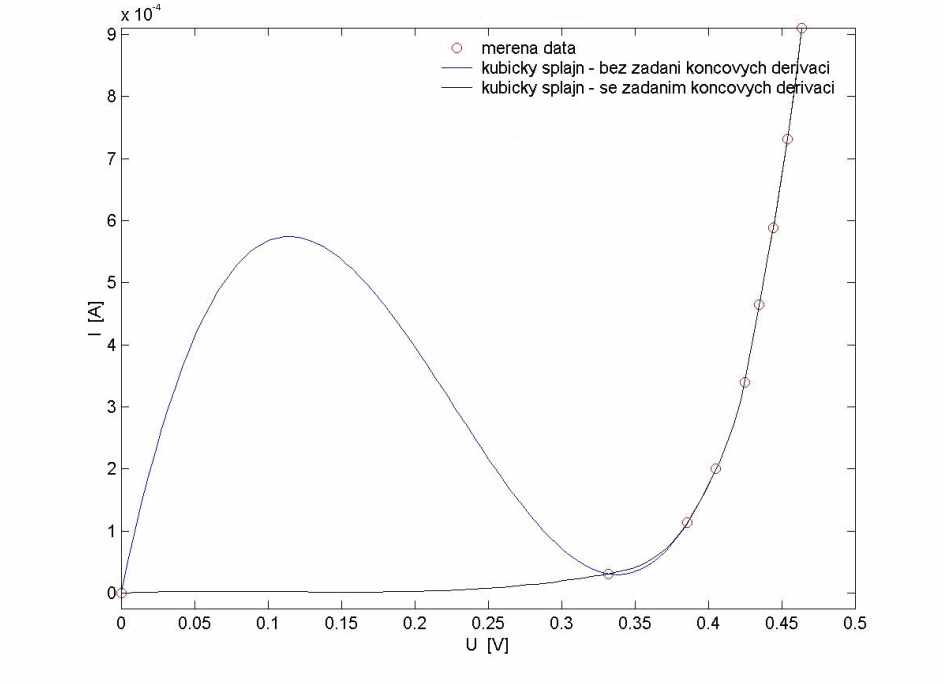
```
U = [ 0 0.332031 0.385742 0.405273 0.424804 ...          % [U] = V
      0.43457  0.444336 0.454102 0.463868 ];
I = [ 0 3.10e-5 1.13e-4 2.00e-4 3.40e-4 ...              % [I] = A
      4.65e-4 5.89e-4 7.32e-4 9.11e-4 ];
% jemne deleni napeti pro zobrazeni prokladanych krivek
UU = U(1) : (U(end)-U(1))/100 : U(end);
% interpolace kubickym splajnem, not-a-knot end condition
```

```
% 1) not-a-knot end condition (bez predepsanych derivaci splajnu na
%    pocatku a konci grafu)
p = spline(U,I);
II1 = ppval(p,UU);
% 2) interpolace s predepsanymi derivacemi dI/dU(zacatek), dI/dU(konec)
dIdU1 = (I(2)-I(1))/(U(2)-U(1));
dIdU2 = (I(end)-I(end-1))/(U(end)-U(end-1));
p = spline(U,[dIdU1,I,dIdU2]);
II2 = ppval(p,UU);
% grafy
figure;
```

Vytvoříme graf

```
plot(U,I,'or',UU,II1,'b',UU,II2,'k');
legend('merena data','kubicky splajn - bez zadani koncovych derivaci', ...
      'kubicky splajn - se zadanim koncovych derivaci');
legend boxoff;
% protazeni osy y do zapornych hodnot
c = axis;
if c(3) >= 0
  set(gca,'YLim',[c(3)-(c(4)-c(3))/40,Inf]);
end
xlabel('U  [V]');
ylabel('I  [A]');
title('Voltamperova charakteristika, fitace kubickym splajnem');
```

Voltamperova charakteristika, fitace kubickym splajnem



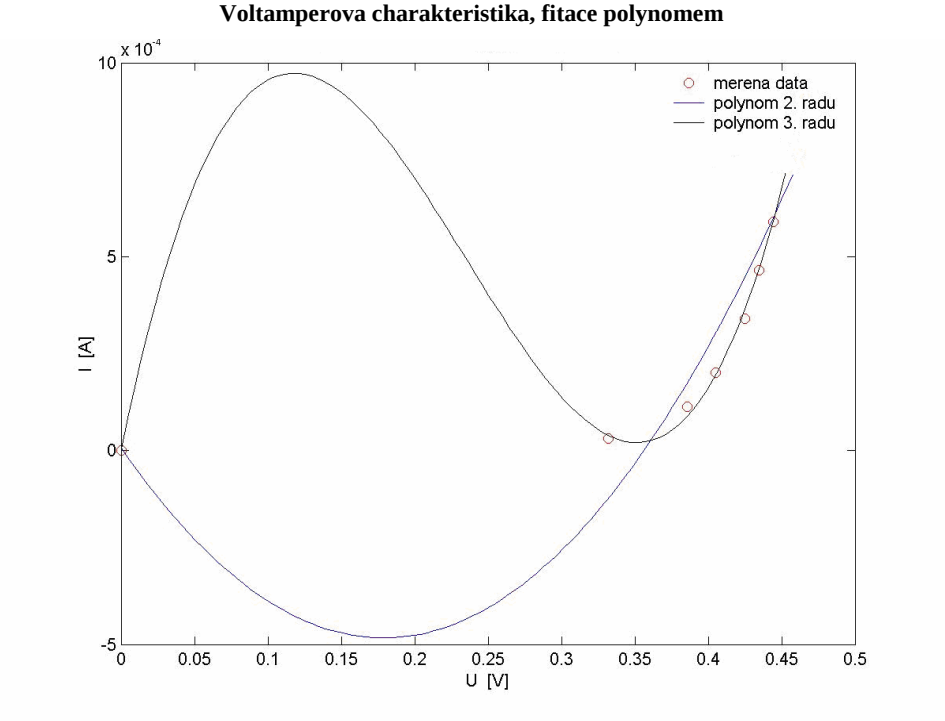
4) Řešíme pomocí fitace polynomem:

```
U = [ 0 0.332031 0.385742 0.405273 0.424804 ...          % [U] = V
      0.43457  0.444336 0.454102 0.463868 ];

I = [ 0 3.10e-5 1.13e-4 2.00e-4 3.40e-4 ...              % [I] = A
      4.65e-4 5.89e-4 7.32e-4 9.11e-4 ];
% jemne deleni napeti pro zobrazeni prokladanych krivek
UU = U(1) : (U(end)-U(1))/100 : U(end);
% aproximace (fitace) polynomem 2. radu, I = a2*U^2 + a1*U + a0
p = polyfit(U,I,2);
[II1] = polyval(p,UU);
p = polyfit(U,I,3);
[II2] = polyval(p,UU);
figure;
```

Vytvoříme graf

```
plot(U,I,'or',UU,II1,'b',UU,II2,'k');
legend('merena data','polynom 2. radu','polynom 3. radu');
```

```
legend boxoff;
xlabel('U  [V]');
ylabel('I  [A]');
title('Voltamperova charakteristika, fitace polynomem');
```



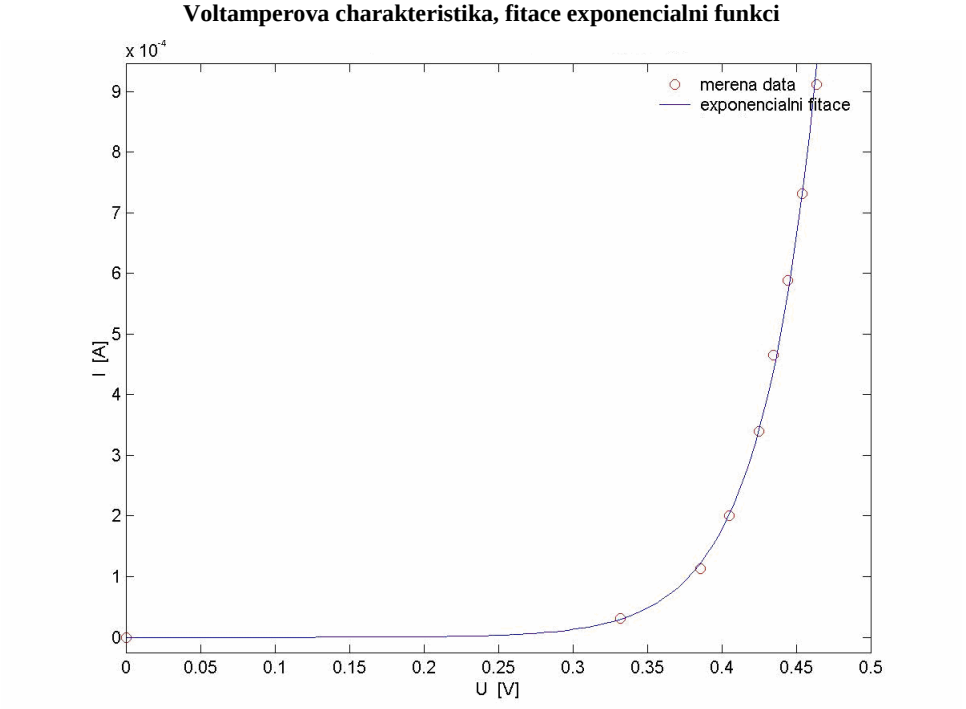
4)Řešíme pomocí fitace exponenciální funkcí:

```
U = [ 0 0.332031 0.385742 0.405273 0.424804 ...          % [U] = V
      0.43457  0.444336 0.454102 0.463868 ];
I = [ 0 3.10e-5 1.13e-4 2.00e-4 3.40e-4 ...              % [I] = A

      4.65e-4 5.89e-4 7.32e-4 9.11e-4 ];
% jemne deleni napeti pro zobrazeni prokladanych krivek
UU = U(1) : (U(end)-U(1))/100 : U(end);

% aproximace (fitace) exponencialni funkci,
% I = a*exp(b*U), ln(I) = ln(a) + b*U
% pred logaritmovanim proudu je treba vyloucit prvni hodnotu I = 0 !!!
lnI = log(I(2:end));
```

```
[p,s] = polyfit(U(2:end),lnI,1);
[lnII,delta] = polyval(p,UU,s);
figure;
```

Opět vytvoříme graf

```
plot(U,I,'or',UU,exp(lnII),'b');
legend('merena data','exponencialni fitace');
legend boxoff;
% protazeni osy y do zapornych hodnot
c = axis;
if c(3) >= 0
  set(gca,'YLim',[c(3)-(c(4)-c(3))/40,Inf]);
end
xlabel('U  [V]');
ylabel('I  [A]');
title('Voltamperova charakteristika, fitace exponencialni funkci');
```



8. Závěr:

Cílem diplomové práce bylo podrobněji specifikovat vybrané metody numerické matematiky, často užívané ve fyzice. Byly popsány a rozebrány metody řešení diferenciálních rovnic, metoda Monte Carlo, různé metody interpolace a aproximace. K většině metodám jsou uvedeny názorné příklady i jejich postup řešení.

V diplomové práci byly popsány a rozebrány základní funkce počítačového programu Matlab, který se často používá pro přehledné řešení příkladů pomocí numerických metod.

V poslední části práce je důkladně rozepsán postup řešení několika fyzikálních příkladů tak , jak je lze řešit v programu Matlab.

Byla bych ráda, kdyby moje práce byla přínosem pro všechny, kteří se o danou problematiku zajímají a aby přispěla ke zpestření výuky daných numerických metod a pomohla i při práci s Matlabem.

9. Seznam použité literatury

[1] HRACH, R. Numerické metody ve fyzikální elektronice, Praha: SPN, 1981

[2] MAROŠ, B., MAROŠOVÁ, M. Základy numerické matematiky, Brno: VUT, 1997

[3] NAVARA, M., NĚMEČEK, A.Numerické metody, Praha: ČVUT, 2005

[4] ENGLICH, J. Úvod do praktické fyziky I, Praha: MatFyzPress, 2006

[5] http://cmp.felk.cvut.cz/~pisa/Public/ST_matlab.html-33k

[6] http://oprt.vscht.cz/majerova/matlab/lekce9.html