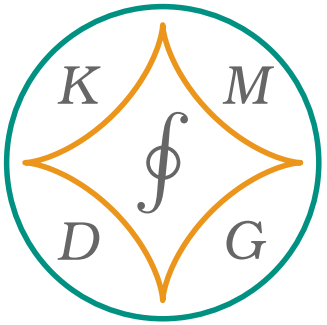
Numerická matematika: Pracovní listy

Pavel Ludvík, Zuzana Morávková

Katedra matematiky a deskriptivní geometrie

VŠB - Technická univerzita Ostrava



2 - Interpolace polynomy

1 Interpolace a aproximace

Úloha interpolace

Jsou dány vzájemně různé uzly x_i a funkční hodnoty y_i, $i = 0, \ldots, n$. Hledáme polynom splňující systém interpolačních rovností

$$p_n(x_i) = y_i,\ i = 0, \ldots, n\,,$$

tedy polynom, jehož graf budete zadanými uzly procházet.

Existuje právě jeden interpolační polynom stupně nejvýše n. Dále popíšeme tři různé způsoby, jak tento polynom nalézt.

Interpolační polynom v základním tvaru

Jsou dány vzájemně různé uzly x_i a funkční hodnoty y_i, $i = 0, \ldots, n$. Dosazením obecného tvaru polynomu

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

do interpolačních rovností dostaneme soustavu lineárních rovnic

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n = y_i, \quad i = 0, \ldots, n,$$

kterou lze zapsat maticově jako

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Vyřešením této soustavy lineárních rovnic nalezneme koeficienty $a_0, a_1, \ldots, a_n \in \mathbb{R}$ hledaného interpolačního polynomu.

Interpolační polynom v Lagrangeově tvaru

Interpolační polynom v Lagrangeově tvaru je určen předpisem

$$p(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + \cdots + y_n\varphi_n(x),$$

kde $\varphi_0(x)$, $\varphi_1(x)$, $\varphi_2(x)$ jsou polynomy Lagrangeovy báze dané úlohy:

$$\varphi_i(x) = \frac{(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}.$$

Interpolační polynom v Newtonově tvaru

Interpolační polynom v Newtonově tvaru je určen předpisem

$$\begin{aligned} p(x) =& y_0 + f[x_1, x_0](x - x_0) + f[x_2, x_1, x_0](x - x_0)(x - x_1) + \\ &+ f[x_3, x_2, x_1, x_0](x - x_0)(x - x_1)(x - x_2) + \cdots + \\ &+ f[x_n, \ldots, x_0](x - x_0)(x - x_1)\cdots(x - x_{n-1}), \end{aligned}$$

kde $f[x_1, x_0]$ je poměrná diference 1. řádu., $f[x_2, x_1, x_0]$ je poměrná diference 2. řádu, až $f[x_n, \ldots, x_0]$ je poměrná diference řádu n.

Poměrné diference se spočtou jako

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

obecně: $$f[x_n, \ldots, x_0] = \frac{f[x_n, \ldots, x_1] - f[x_{n-1}, \ldots, x_0]}{x_n - x_0}.$$

3 - Interpolační polynom v základním tvaru

Pro uzly x_i a funkční hodnoty y_i dané následující tabulkou

	i=0	i=1	i=2
x_i	0	3	4
y_i	2	1	5

sestavte interpolační polynom v základním tvaru.

Nejprve zadáme do vektoru `x` uzly x_i a do vektoru `y` funkční hodnoty y_i.

```
>> x = [0; 3; 4]
>> y = [2; 1; 5]
```

Koeficienty polynomu spočítáme jako řešení soustavy lineárních rovnic.

```
>> M = [ones(3,1) x x.^2]
>> a = M\y
```

Koeficienty a_i jsou zjevně racionální čísla a proto si je vypíšeme ve tvaru zlomku.

```
>> format rat
>> a
```

Nalezený interpolační polynom je

$$p_2(x) = 2 - \frac{43}{12}x + \frac{13}{12}x^2 .$$

Polynom nejprve uložíme do proměnné `p2` a pak vykreslíme jeho graf na intervalu $[x_0, x_2] = [0, 4]$.

```
>> plot(x,y,'o')
>> grid on, hold on
>> p2 = @(x)a(1)+a(2)*x+a(3)*x.^2;
>> fplot(p2, [0 4], 'r')
>> legend('uzly','obecny polynom')
```

4 - Interpolační polynom v Lagrangeově tvaru

Pro uzly x_i a funkční hodnoty y_i dané následující tabulkou

	i=0	i=1	i=2
x_i	0	3	4
y_i	2	1	5

sestavte interpolační polynom v Lagrangeově tvaru.

Zadáme uzly.

```
>> x = [0; 3; 4]
>> y = [2; 1; 5]
```

Hledaný polynom má tvar

$$p_2(x) = \frac{1}{6}(x-3)(x-4) - \frac{1}{3}x(x-4) + \frac{5}{4}x(x-3)\,.$$

Vykreslíme zadané body a nalezený polynom.

```
>> plot(x,y,'o')
>> grid on, hold on
>> p = @(x) 1/6*(x-3).*(x-4)-1/3*x.*(x-4)+5/4*x.*(x-3)
>> fplot(p2,[0 4], 'r')
>> legend('uzly','Lagrangeuv polynom')
```

5 - Interpolační polynom v Newtonově tvaru

Pro uzly x_i a funkční hodnoty y_i dané následující tabulkou

	i=0	i=1	i=2
x_i	0	3	4
y_i	2	1	5

sestavte interpolační polynom v Newtonově tvaru.

Spočteme poměrné diference:

i	x_i	y_i	1.řád	2.řád
0	0	$\boxed{\mathbf{2}}$	$\boxed{-\frac{1}{3}}$	$\boxed{\frac{13}{12}}$
1	3	1	4	
2	4	5		

Pomocí tučně zvýrazněných poměrných diferencí v prvním řádku tabulky sestavíme interpolační polynom:

$$p_2(x) = 2 - \frac{1}{3}(x-0) + \frac{13}{12}(x-0)(x-3) = 2 - \frac{1}{3}x + \frac{13}{12}x(x-3)\,.$$

Zadáme uzly.

```
>> x = [0; 3; 4]
>> y = [2; 1; 5]
```

Vykreslíme zadané body a nalezený polynom.

```
>> plot(x,y,'o')
>> grid on, hold on
>> p2 = @(x) 2-1/3*x+13/12*x*(x-3);
>> fplot(p2, [0 4], 'r')
>> legend('uzly','Newtonuv polynom')
```

6 - Aproximace metodou nejmenších čtverců

Úloha aproximace

Jsou dány vzájemně různé uzly x_i a funkční hodnoty y_i, $i = 1, \ldots, n$. Hledáme funkce splňující

$$\varphi(x_i) \approx y_i,\ i = 0, \ldots, n\,,$$

tedy funkci, jejíž graf budete procházet „blízko“ zadaných uzlů.

Aproximace metodou nejmenších čtverců

Nechť jsou dány funkce $\varphi_1(x)$ a $\varphi_2(x)$. Chceme nalézt hodnoty $c_1, c_2 \in \mathbb{R}$ tak, aby funkce tvaru $\varphi(x) = c_1\varphi_1(x) + c_2\varphi_2(x)$ byla nejlepší aproximací dat ve smyslu nejmenších čtverců.
K tomu je třeba nejprve sestavit normální rovnice. Ty mají tvar

$$\begin{aligned} c_1 \sum_{i=1}^{n} (\varphi_1(x_i))^2 + c_2 \sum_{i=1}^{n} \varphi_1(x_i) \cdot \varphi_2(x_i) &= \sum_{i=1}^{n} y_i \cdot \varphi_1(x_i), \\ c_1 \sum_{i=1}^{n} \varphi_2(x_i) \cdot \varphi_1(x_i) + c_2 \sum_{i=1}^{n} (\varphi_2(x_i))^2 &= \sum_{i=1}^{n} y_i \cdot \varphi_2(x_i). \end{aligned}$$

Takovou soustavu je pak snadné vyřešit a nalézt tak ty správné koeficienty $c_1, c_2 \in \mathbb{R}$.

Poznámka

Budeme-li hledat přímku, tedy lineární funkci $\varphi(x) = c_1 + c_2 x$. Pak $\varphi_1(x) = 1$, $\varphi_2(x) = x$ a soustava normálních rovnic má tvar

$$\begin{aligned} c_1 \sum_{i=1}^{n} 1 + c_2 \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} y_i\,, \\ c_1 \sum_{i=1}^{n} x_i + c_2 \sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} y_i \cdot x_i\,. \end{aligned}$$

7 - Metoda nejmenších čtverců: přímka

Aproximujte následující data

	i=0	i=1	i=2	i=3	i=4
x_i	2	3	5	7	8
y_i	5	4	4	3	0

lineární funkcí

$$\varphi(x) = c_1 + c_2 x$$

metodou nejmenších čtverců.

Nejprve zadáme data.

```
>> x=[2 3 5 7 8]
>> y=[5 4 4 3 0]
```

Pro výpočet budeme potřebovat také matici soustavy normálních rovnic **G** a vektor pravé strany **d**.

```
>> G(1,1) = 5
>> G(1,2) = sum(x)
>> G(2,1) = sum(x)
>> G(2,2) = sum(x.^2)
>> d(1,1) = sum(y)
>> d(2,1) = sum(x.*y)
```

Soustavu normálních rovnic vyřešíme.

```
>> c = G\d
```

Hledaná aproximace nejlepší ve smyslu nejmenších čtverců má tedy tvar (koeficienty zaokrouhlujeme na čtyři desetinná místa)

$$\varphi(x) = 6.46923 - 0.65385\,x\,.$$

Získanou aproximaci nyní uložíme do proměnné `f`.

```
>> f = @(x) c(1)+c(2)*x
```

a vykreslíme její graf společně se znázorněním zadaných bodů.

```
>> plot(x,y,'o')
>> grid on, hold on
>> fplot(f,[2 8],'r')
>> legend('zadana data','primka')
```

8 - Metoda nejmenších čtverců

Aproximujte následující data

x_i	-5.5	-4.1	-3.5	1.3	2.6	3.5	5.8	7.3	10.6	14.9
y_i	-34	-11	12	26	-15	-19	25	41	-27	-17

funkcí

$$\varphi(x) = c_1 \cos(x) + c_2 \frac{1}{e^x}$$

metodou nejmenších čtverců.

Nejprve zadáme data.

```
>> x = [-5.5 -4.1 -3.5 1.3 2.6 3.5 5.8 7.3 10.6 14.9]
>> y = [-34 -11 12 26 -15 -19 25 41 -27 -17]
```

Následně definujeme funkce $\varphi_1(x) = \cos x$ a $\varphi_2(x) = \frac{1}{e^x}$ pod proměnnými `f1, f2`.

```
>> f1 = @(x)cos(x)
>> f2 = @(x)1./exp(x)
```

Pro výpočet budeme potřebovat také matici soustavy normálních rovnic **G** a vektor pravé strany **d**.

```
>> G(1,1) = sum(f1(x).^2)
>> G(1,2) = sum(f1(x).*f2(x))
>> G(2,1) = sum(f2(x).*f1(x))
>> G(2,2) = sum(f2(x).^2)
>> d(1,1) = sum(f1(x).*y)
>> d(2,1) = sum(f2(x).*y)
```

Soustavu normálních rovnic vyřešíme.

```
>> c = G\d
```

Hledaná aproximace nejlepší ve smyslu nejmenších čtverců je

$$\varphi(x) = 18.1137 \cos x - 0.1630 \frac{1}{e^x}.$$

Získanou aproximaci nyní uložíme do proměnné `f`

```
>> f = @(x)c(1)*f1(x)+c(2)*f2(x)
```

a vykreslíme její graf společně se znázorněním zadaných bodů.

```
>> plot(x,y,'o')
>> grid on, hold on
>> fplot(f,[-5.5 14.9],'g')
>> legend('zadana data','nalezena funkce')
```

9 - Nelineární rovnice

2 Nelineární rovnice

Nelineární rovnice

Je dána spojitá funkce $f(x)$. Hledáme $x \in \mathbb{R}$, které je řešením rovnice

$$f(x) = 0.$$

Separace kořenů

Grafická separace: Z grafu funkce f najdeme polohu průsečíků s x-ovou osou.
Separace tabelací: Sestavíme tabulku funkčních hodnot funkce f a podle znaménkových změn určíme intervaly obsahující kořeny.

Metoda půlení intervalu

Bod x^k určíme jako střed intervalu $\langle a^k, b^k \rangle$ podle vzorce

$$x^k = \frac{a^k + b^k}{2}.$$

Další interval zvolíme podle znamének funkčních hodnot $f(a^k)$, $f(x^k)$, $f(b^k)$.

Je-li $f(a^k)f(x^k) < 0$, potom $a^{k+1} := a^k$, $b^{k+1} := x^k$;

A je-li $f(x^k)f(b^k) < 0$, potom $a^{k+1} := x^k$, $b^{k+1} := b^k$;

Intervaly tedy postupně půlíme a jejich středy tvořící posloupnost $\{x^k\}$ konvergují ke kořenu $\bar{x}$. Výpočet ukončíme při dosažení zadané přesnosti ε, tj. když platí

$$\frac{b^k - a^k}{2} \leq \varepsilon$$

a poslední střed x^k je pak aproximací kořene $\bar{x}$ s přesností ε.

Newtonova metoda

Nechť jsou splněny následující předpoklady:

1. f' nemění znaménko na intervalu $\langle a, b \rangle$;
2. f'' nemění znaménko na intervalu $\langle a, b \rangle$;
3. platí $f(a) \cdot f(b) < 0$;
4. platí $\left|\frac{f(a)}{f'(a)}\right| < b - a$ a $\left|\frac{f(b)}{f'(b)}\right| < b - a$.

Potom posloupnost $\{x^k\}$ počítaná podle vzorce

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

konverguje pro libovolnou počáteční aproximaci $x^1 \in \langle a, b \rangle$.
Výpočet ukončíme při dosažení zadané přesnosti ε, tj. když platí

$$|x^k - x^{k+1}| \leq \varepsilon\,.$$

10 - Metoda půlení intervalu

Určete všechny kořeny rovnice

$$2x + 2 - e^x = 0$$

s přesností $\varepsilon = 10^{-2}$ metodou půlení intervalu.

Provedeme separaci kořenů. Zadáme funkci a vykreslíme její graf.

```
>> f = @(x)2*x+2-exp(x)
>> fplot(f, [-5,5])
>> grid on
```

Do proměnných `a1` a `b1` zadáme meze intervalu, které jsme zjistili separací. V těchto bodech zjistíme funkční hodnoty.

```
>> a1 = -1
>> b1 = 0
>> f(a1)
>> f(b1)
```

Spočítáme `x1` jako polovinu intervalu (a_1, b_1) a v tomto bodě spočítáme funkční hodnotu.

```
>> x1 = (a1+b1)/2
>> f(x1)
```

Spočítáme chybu výpočtu, a pokud je větší než ε, výpočet pokračuje dál.

```
>> Chyba = abs(b1-a1)/2
```

Podle znamének $f(a_1)$, $f(x_1)$, $f(b_1)$ určíme nový interval (a_2, b_2).

```
>> a2 = a1
>> b2 = x1
```

Spočítáme `x2` jako polovinu intervalu (a_2, b_2) a v tomto bodě spočítáme funkční hodnotu. Spočítáme chybu výpočtu, a pokud je větší než ε, výpočet pokračuje dál.

```
>> x2 = (a2+b2)/2
>> f(x2)
>> Chyba = abs(b2-a2)/2
```

Podle znamének $f(a_2)$, $f(x_2)$, $f(b_2)$ určíme nový interval (a_3, b_3).

```
>> a3 = a2
>> b3 = x2
```

Spočítáme `x3` jako polovinu intervalu (a_3, b_3) a v tomto bodě spočítáme funkční hodnotu. Spočítáme chybu výpočtu, a pokud je větší než ε, výpočet pokračuje dál.

```
>> x3 = (a3+b3)/2
>> f(x3)
>> Chyba = abs(b3-a3)/2
```

Podle znamének $f(a_3)$, $f(x_3)$, $f(b_3)$ určíme nový interval (a_4, b_4).

```
>> a4 = x3
>> b4 = b3
```

Výpočet pokračuje dál, pokud je chyba větší než ε.

Kořen je -0.77 ± 10^{-2}. Ostatní kořeny se spočtou analogicky.

11 - Newtonova metoda

Určete všechny kořeny rovnice

$$x - 4\cos^2(x) = 0$$

s přesností $\varepsilon = 10^{-8}$ Newtonovou metodou.

Provedeme separaci kořenů.

```
>> f = @(x)x-4*cos(x).^2
>> fplot(f, [-5,5])
>> grid on
```

Zadáme meze nalezeného intervalu.

```
>> a = 3
>> b = 4
```

Zadáme první a druhou derivaci.

```
>> df = @(x)1+8*cos(x).*sin(x)
>> ddf = @(x)-8*sin(x).^2+8*cos(x).^2
```

Ověříme předpoklady.

```
>> x = a:0.1:b
>> df(x)
>> ddf(x)
>> f(a)*f(b)
>> abs(f(a)/df(a))
>> abs(f(b)/df(b))
```

Předpoklady nejsou splněny, a tak je potřeba nalézt menší interval, ve kterém leží kořen. Pak ověříme předpoklady znovu.

```
>> fplot(f,[3,4])
>> a = 3.4
>> b = 3.6
>> x = a:0.01:b;
>> df(x)
>> ddf(x)
>> f(a)*f(b)
>> abs(f(a)/df(a))
>> abs(f(b)/df(b))
```

Zadáme počáteční aproximaci.

```
>> format long
>> x0 = a
```

Vypočítáme další aproximaci a chybu, a pokud není menší než ε, výpočet pokračuje dál.

```
>> x1 = x0-f(x0)/df(x0)
>> Chyba = abs(x0-x1)
```

Vypočítáme další aproximaci a chybu, a pokud není menší než ε, výpočet pokračuje dál.

```
>> x2 = x1-f(x1)/df(x1)
>> Chyba = abs(x1-x2)
```

Výpočet pokračuje dál, dokud chyba není menší než ε.

Kořen je 3.50214739 ± 10^{-8}.
Ostatní kořeny spočítáme obdobným způsobem.

12 - Soustavy lineárních rovnic: iterační metody

3 Soustavy lineárních rovnic: iterační metody

Soustava lineárních rovnic

Je dána soustava lineárních rovnic

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

s regulární čtvercovou maticí **A**. Pak má soustava lineárních rovnic právě jedno řešení **x**.

Konvergenční kritérium

Soustavu lineárních rovnic $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ upravíme pomocí řádkově ekvivalentních úprav na tvar s ostře diagonální maticí.
Poté soustavu přepíšeme do iteračního tvaru

$$\mathbf{x} = \mathbf{C} \cdot \mathbf{x} + \mathbf{d}\,.$$

Jacobiho metoda

Nechť $\mathbf{x}^{(0)}$ je daná počáteční aproximace. Iterační výpočet provádíme podle rekurentního vzorce

$$\mathbf{x}^{(k+1)} = \mathbf{C} \cdot \mathbf{x}^{(k)} + \mathbf{d}, \quad k = 0, 1, 2, \ldots.$$

Rozepíšeme maticové násobení po prvcích

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i, \quad i = 1, \ldots, n \quad k = 0, 1, 2, \ldots.$$

Jestliže posloupnost vektorů $\{\mathbf{x}^k\}$ konverguje k vektoru **x**, pak **x** je řešením $\mathbf{x} = \mathbf{C} \cdot \mathbf{x} + \mathbf{d}$ a tedy i $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$.
Vektor $\mathbf{x}^{(0)}$ můžeme zvolit libovolně. Výpočet ukončíme, jestliže je splněno ukončovací kritérium

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \varepsilon,$$

kde $\varepsilon > 0$ je dané malé číslo a $\|\cdot\|$ je řádková norma.

Gaussova-Seidelova metoda

Nechť $\mathbf{x}^{(0)}$ je daná počáteční aproximace. Iterační výpočet provádíme podle rekurentního vzorce

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} c_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} c_{ij} x_j^{(k)} + d_i, \quad i = 1, \ldots, n$$

Vektor $\mathbf{x}^{(0)}$ můžeme zvolit libovolně. Výpočet ukončíme, jestliže je splněno ukončovací kritérium

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \varepsilon.$$

.

13 - Jacobiho metoda

Vyřešte soustavu lineárních rovnic

$$\begin{aligned} -x_1 - 6x_2 + 7x_3 &= 16, \\ 4x_1 - 5x_2 + 3x_3 &= 8, \\ 3x_1 - x_2 + x_3 &= 11 \end{aligned}$$

pomocí Jacobiho iterační metody s přesností $\varepsilon = 10^{-2}$.

Jedna z možných úprav na soustavu s ostře diagonálně dominantní maticí.

$$\begin{pmatrix} 3 & -1 & 1 \\ 1 & -4 & 2 \\ -2 & -2 & 5 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 11 \\ -3 \\ 19 \end{pmatrix}$$

Rekurentní vzorce napíšeme v maticové formě a definujeme matici $\mathbf{C}$ a vektor $\mathbf{d}$.

$$\mathbf{x}^{(k+1)} = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{2}{5} & \frac{2}{5} & 0 \end{pmatrix} \cdot \mathbf{x}^{(k)} + \begin{pmatrix} \frac{11}{3} \\ \frac{3}{4} \\ \frac{19}{5} \end{pmatrix} = \mathbf{C} \cdot \mathbf{x}^{(k)} + \mathbf{d}$$

Začneme definicí potřebných objektů.

```
>> C = [0 1/3 -1/3;1/4 0 1/2;2/5 2/5 0]
>> d = [11/3;3/4;19/5]
>> x = [0;0;0]
```

Výpočet nového vektoru, chyby a uložení nového vektoru do proměnné `xnovy` provádíme zde:

```
>> xnovy = C*x+d
>> Chyba = max(abs(xnovy-x))
>> x = xnovy
```

Předchozí tři příkazy opakujeme, dokud chyba bude větší než 10^{-2}.

Hodnoty zaokrouhlíme na 2 desetinná místa a řešení zapíšeme jako

$$x_1 = 3 \pm 10^{-2},\ x_2 = 5 \pm 10^{-2},\ x_3 = 7 \pm 10^{-2}.$$

14 - Gaussova-Seidelova metoda

Vyřešte soustavu lineárních rovnic

$$\begin{array}{rrrcr} -x_1 & -6x_2 & +7x_3 & = & 16, \\ 4x_1 & -5x_2 & +3x_3 & = & 8, \\ 3x_1 & -x_2 & +x_3 & = & 11 \end{array}$$

pomocí Jacobiho iterační metody s přesností $\varepsilon = 10^{-2}$.

Jedna z možných úprav na soustavu s ostře diagonálně dominantní maticí.

$$\begin{pmatrix} 3 & -1 & 1 \\ 1 & -4 & 2 \\ -2 & -2 & 5 \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} 11 \\ -3 \\ 19 \end{pmatrix}$$

Soustavu přepíšeme do poboby, která se nám hodila již u Jacobiho metody a definujeme matici $\mathbf{C}$ a vektor $\mathbf{d}$.

$$\mathbf{x} = \begin{pmatrix} 0 & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{2}{5} & \frac{2}{5} & 0 \end{pmatrix} \cdot \mathbf{x} + \begin{pmatrix} \frac{11}{3} \\ \frac{3}{4} \\ \frac{19}{5} \end{pmatrix} = \mathbf{C} \cdot \mathbf{x} + \mathbf{d}$$

U Gaussovy-Seidelovy metody jsou rekurentní rovnice jiné a v maticové podobě si je nebudeme uvádět. Spokojíme se s faktem, že na rozdíl od Jacobiho metody se užívají nejnovější hodnoty aproximací, které jsou k dispozici. Začneme definicí potřebných objektů.

```
>> C = [0 1/3 -1/3;1/4 0 1/2;2/5 2/5 0]
>> d = [11/3;3/4;19/5]
>> x = [0;0;0]
```

Z technických důvodů inicializujeme proměnnou `xnovy` hodnotou proměnné `x`.

```
>> xnovy = x
```

Výpočet nového vektoru, chyby a uložení nového vektoru do proměnné `xnovy` provádíme zde:

```
>> for i=1:3, xnovy(i)=C(i,:)*xnovy+d(i), end
>> Chyba = max(abs(xnovy-x))
>> x = xnovy
```

Tyto příkazy opakujeme, dokud není chyba menší než 10^{-2}. Hodnoty zaokrouhlíme na dvě desetinná místa a řešení zapíšeme jako $x_1 = 3 \pm 10^{-2}$, $x_2 = 5 \pm 10^{-2}$, $x_3 = 7 \pm 10^{-2}$.

15 - Numerické integrování

4 Numerické integrování

Určitý integrál

Počítáme hodnotu určitého integrálu

$$\int_a^b f(x)\,dx.$$

Integrační formule

Jednoduchá lichoběžníková formule
Funkci f interpolujeme lineární funkci. Po její integraci je

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right).$$

Jednoduchá Simpsonova formule
Funkci f interpolujeme kvadratickou funkci v uzlech a $x_0 = a$, $x_1 = \frac{a+b}{2}$, $x_2 = b$. Dostáváme

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right).$$

Složená lichoběžníková formule

Chceme-li integrovat funkci f na intervalu $\langle a,b\rangle$ složenou lichoběžníkovou formulí, musíme nejprve zadaný interval rozdělit na $n \in \mathbb{N}$ stejně dlouhých dílků. Dílky pak budou mít velikost $h = (b-a)/n$ a dostaneme uzly $x_i = a + ih$, $i = 0, 1, \ldots, n$. Složená lichoběžníková formule pro krok h pak má tvar

$$I_h = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right].$$

Složená Simpsonova formule

Chceme-li integrovat funkci f na intervalu $\langle a,b\rangle$ složenou Simpsonovou formulí, musíme nejprve zadaný interval rozdělit na n stejně dlouhých dílků, kde $n \in \mathbb{N}$ je sudé. Dílky pak budou mít velikost $h = (b-a)/n$ a dostaneme lichý počet uzlů $x_i = a + ih, i = 0, 1, \ldots, n$. Složená Simpsonova formule pro krok h pak má tvar

$$I_h = \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{n/2} f(x_{2i-1}) + 2\sum_{i=1}^{n/2-1} f(x_{2i}) + f(x_n)\right].$$

Výpočet integrálu se zadanou přesností

Spočítáme hodnotu pomocí integrační formule pro krok h tj. I_h. Dále spočítáme hodnotu pro poloviční krok $h/2$, tj. $I_{h/2}$. Výpočet ukončíme, pokud platí

$$|I_h - I_{h/2}| \leq \varepsilon.$$

16 - Složená lichoběžníková formule

Vypočtěte integrál

$$\int_{-1}^{3} \mathrm{e}^{x^2}\,dx$$

složenou lichoběžníkovou formulí pro $n = 8$.

Nejprve definujeme funkci `f` a integrační meze uložíme do proměnných `a,b`.

```
>> f = @(x)exp(x.^2);
>> a = -1;
>> b = 3;
```

Zadáme $n = 8$ a spočítáme velikost kroku h. Uzly uložíme jako vektor do proměnné `x`.

```
>> n = 8;
>> h = (b-a)/n;
>> x = a:h:b;
```

Numerickou hodnotu integrálu spočteme a uložíme do proměnné `I`.

```
>> I=h/2*(f(x(1))+2*sum(f(x(2:n)))+f(x(n+1)))
```

Integrál má hodnotu 2320.64.

17 - Složená Simpsonova formule

Vypočtěte integrál

$$\int_1^e \frac{\ln x}{\sqrt{9-x^2}}\,dx$$

složenou Simpsonovou formulí se zadanou přesností $\varepsilon = 10^{-8}$.

Nejprve definujeme funkci `f` a integrační meze uložíme do proměnných `a`, `b`.

```
>> f = @(x)log(x)./sqrt(9-x.^2);
>> a = 1;
>> b = exp(1);
```

V prvním kroku zvolíme $n = 2$. Uzly uložíme jako vektor do proměnné `x`.

```
>> n = 2;
>> h = (b-a)/n;
>> x = a:h:b;
```

Numerickou hodnotu integrálu spočteme a uložíme do proměnné `Inovy`.

```
>> Inovy = h/3*(f(x(1))+4*sum(f(x(2:2:n)))+2*sum(f(x(3:2:n)))+f(x(n+1)))
```

Spočtenou hodnotu integrálu pouze uložíme do proměnné `I`. Pak zdvojnásobíme hodnotu n a celý výpočet zopakujeme. Nakonec spočteme chybu $|I_h - I_{2h}|$

```
>> I = Inovy;
>> n = 2*n
>> h = (b-a)/n
>> x = a:h:b;
>> Inovy = h/3*(f(x(1))+4*sum(f(x(2:2:n)))+2*sum(f(x(3:2:n)))+f(x(n+1)))
>> Chyba = abs(Inovy-I)
```

Předchozích šest příkazů budeme opakovat, dokud bude chyba větší než 10^{-8}.
Protože cílíme na přesnost 10^{-8}, nebudou nám samozřejmě stačit čtyři desetinná místa, která MATLAB zobrazuje při formátu `short`. Je třeba přepnout formát výstupu na `long`.

```
>> format long
```

Výsledek zaokrouhlíme na osm desetinných míst a zapíšeme jako

$$\int_1^e \frac{\ln x}{\sqrt{9-x^2}}\,dx = 0.50661191 \pm 10^{-8}.$$

18 - Počáteční úlohy pro ODR

5 Počáteční úlohy pro ODR

Počáteční úloha pro obyčejnou diferenciální rovnici

Hledáme funkci $y = y(x)$, která na intervalu $\langle a, b\rangle$ vyhovuje rovnici

$$y'(x) = f(x, y(x))$$

a počáteční podmínce

$$y(a) = c.$$

Eulerova metoda

Interval $\langle a, b\rangle$ rozdělíme na ekvidistantní uzly s krokem h:

$$x_i = a + ih\,, \quad i = 0, 1, \ldots, n\,,$$

kde $n = \frac{b-a}{h}$.
Pak spočítáme čísla $y_0 = c, y_1, y_2, \ldots, y_n$, která aproximují hodnoty přesného řešení $y(x_0), y(x_1), y(x_2), \ldots, y(x_n)$:

$$\begin{aligned} y_0 &= c, \\ y_{i+1} &= y_i + hf(x_i, y_i), \quad i = 0, 1, \ldots, n-1. \end{aligned}$$

Rungeova-Kuttova metoda

Interval $\langle a, b\rangle$ rozdělíme na ekvidistantní uzly s krokem h:Rungeova-Kuttova metoda

$$x_i = a + ih\,, \quad i = 0, 1, \ldots, n\,,$$

kde $n = \frac{b-a}{h}$. Pak spočítáme čísla $y_0 = c, y_1, y_2, \ldots, y_n$, která aproximují hodnoty přesného řešení $y(x_0), y(x_1), y(x_2), \ldots, y(x_n)$:

$$y_0 = c,$$

pro $i = 0, 1, \ldots, n-1$ počítáme hodnotu y_{i+1} pomocí vzorce

$$\begin{aligned} y_{i+1} &= y_i + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= hf(x_i, y_i), \\ k_2 &= hf(x_i + \tfrac{h}{2}, y_i + \tfrac{k_1}{2}), \\ k_3 &= hf(x_i + \tfrac{h}{2}, y_i + \tfrac{k_2}{2}), \\ k_4 &= hf(x_{i+1}, y_i + k_3). \end{aligned}$$

19 - Eulerova metoda

Počáteční úlohu

$$y' = y - x^2 + 2, \quad y(0) = -1\,,$$

řešte na intervalu $\langle 0, 2\rangle$ pomocí Eulerovy metody s krokem $h = 0.5$.

Zadáme krajní meze intervalu $\langle a, b\rangle$, hodnotu počáteční podmínky c a funkci pravé strany diferenciální rovnice f.

```
>> a = 0;
>> b = 2;
>> c = -1;
>> f = @(x,y)y-x.^2+2;
```

Zadáme velikost kroku h a vypočítáme počet dílů dělení $n = \frac{b-a}{h}$.

```
>> h = 0.5;
>> n = (b-a)/h;
```

Pomocí dvojtečkové konvence vypočítáme hodnoty $x_i = a + ih$ pro $i = 0, \ldots, n$.

```
>> x = a:h:b;
```

Zadáme hodnotu y_0 a spočítáme ostatní hodnoty y. Připomeňme, že v MATLABu se indexuje od 1, tedy hodnoty $y_0, y_1, \ldots, y_n$ se uloží do proměnných `y(1),y(2),...,y(n+1)`.

```
>> y(1) = c;
>> for i=1:n, y(i+1)=y(i)+h*f(x(i),y(i)); end
```

Hodnoty numerického řešení vypíšeme do tabulky a vykreslíme do grafu.

```
>> [x;y]
>> plot(x,y)
```

Nalezené numerické řešení diferenciální rovnice je

x_i	0	0.5	1	1.5	2
y_i	-1	-0.5	0.125	0.6875	0.9063

20 - Rungeova-Kuttova metoda

Počáteční úlohu

$$y' = \sin(x) + \cos(3y), \quad y(-1) = 1\,,$$

řešte na intervalu $\langle -1, 4\rangle$ pomocí Rungeovy-Kuttovy metody s krokem $h = 1$.

Zadáme krajní meze intervalu $\langle a, b\rangle$, hodnotu počáteční podmínky c a funkci pravé strany diferenciální rovnice f.

```
>> a = -1;
>> b = 4;
>> c = 1;
>> f = @(x,y)sin(x)+cos(3*y);
```

Zadáme velikost kroku h a vypočítáme počet dělení $n = \frac{b-a}{h}$.

```
>> h=1;
>> n=(b-a)/h;
```

Pomocí dvojtečkové konvence vypočítáme hodnoty x_i.

```
>> x=a:h:b;
```

Zadáme hodnotu první hodnotu y_0 a spočítáme ostatní hodnoty y. V každém kroku se počítají hodnoty k_1, k_2, k_3, k_4 a y_{i+1}.

```
>> y(1) = c;
>> for i=1:n,
     k1 = h*f(x(i),y(i));
     k2 = h*f(x(i)+h/2,y(i)+k1/2);
     k3 = h*f(x(i)+h/2,y(i)+k2/2);
     k4 = h*f(x(i+1),y(i)+k3);
     y(i+1) = y(i)+1/6*(k1+2*k2+2*k3+k4);
 end
```

Hodnoty numerického řešení vypíšeme do tabulky a vykreslíme do grafu.

```
>> [x;y]
>> plot(x,y)
```

Nalezené numerické řešení diferenciální rovnice je

x_i	-1	0	1	2	3	4
y_i	1	0.5210	0.8468	0.9223	0.6741	0.3465

21 - MATLAB operace a funkce

Operace

sčítání	`+`
odčítání	`-`
násobení	`*`
dělení	`/`
mocnina	`^`
závorky	`( )`

Priorita operací

priorita	operace
1.	`^`
2.	`* /`
3.	`+ -`

Matematické funkce

absolutní hodnota $\|x\|$	`abs( )`
druhá odmocnina $\sqrt{x}$	`sqrt( )`
exponenciální funkce e^x	`exp( )`
přirozený logaritmus $\ln(x)$	`log( )`
dekadický logaritmus $\log(x)$	`log10( )`
sinus $\sin(x)$	`sin( )`
kosinus $\cos(x)$	`cos( )`
tangens $\mathrm{tg}(x)$	`tan( )`
kotangens $\mathrm{cotg}(x)$	`cot()`
arkussinus $\arcsin(x)$	`asin( )`
arkuskosinus $\arccos(x)$	`acos( )`
arkustangens $\mathrm{arctg}(x)$	`atan( )`
arkuskotangens $\mathrm{arctg}(x)$	`acot( )`

Konstanty

Ludolfovo číslo $\pi = 3.14\ldots$	`pi`
nekonečno ∞	`inf`
neurčitý výraz	`NaN`

Definice vlastní matematické funkce

f=@(proměnné)*předpis funkce*

```
f=@(x)sin(x)-3*x
```

Méně používané funkce

signum	`sign( )`
logaritmus o základu 2	`log2( )`
hyperbolický sinus	`sinh( )`
hyperbolický kosinus	`cosh( )`
hyperbolický tangens	`tanh( )`
hyperbolický kotangens	`coth( )`

Zaokrouhlování

zaokrouhlování na celé číslo	`round( )`
zaokrouhlování na nejbližší nižší celé číslo (dolů)	`floor( )`
zaokrouhlování na nejbližší vyšší celé číslo (nahoru)	`ceil( )`
zaokrouhlování na nejbližší celé číslo směrem k nule	`fix( )`

22 - MATLAB matice a vektory

Vektory a matice

`A(3,2)` prvek $a_{3,2}$ matice A
`A(3,:)` třetí řádek matice A
`A(:,2)` druh sloupec matice A

Základní informace o matici, vektoru

rozměr matice A (počet řádků , počet sloupců)	`size(A)`
počet prvků matice A	`numel(A`
počet prvků vektoru $\vec{v}$	`length(v)`

Příkazy lineární algebry

determinant matice A	`det(A)`
hodnost matice A	`rank(A)`
inverzní matice A^{-1}	`inv(A)`
převedení matice A na horní trojúhelníkový tvar pomocí eliminace	`rref(A)`
transponovaná atice k matici A	`A'`

Další operace pro matice

matice typu $m \times n$ náhodných čísel z intervalu $\langle 0, 1 \rangle$	`rand(m,n)`
matice nul typu $m \times n$	`zeros(m,n)`
matice jedniček typu $m \times n$	`ones(m,n)`
jednotková matice typu $m \times n$	`eyes(m,n)`
součet prvků ve sloupcích matice A	`sum(A)`
součin prvků ve sloupcích matice A	`prod(A)`
největší hodnota ve sloupcích matice A	`max(A)`
nejmenší hodnota ve sloupcích matice A	`min(A)`

Maticové operace

součet matic	`+`
(např. `A+B` je matice s prvky $a_{ij} + b_{ij}$)	
rozdíl matic	`-`
(např. `A-B` je matice s prvky $a_{ij} - b_{ij}$)	
součin matic	`*`
„řádek krát sloupec“	
pravé maticové dělení	`/`
(např. `A/B` je matice $A \cdot B^{-1}$)	
levé maticové dělení	`\`
(např. `A\B` je matice $A^{-1} \cdot B$)	
mocnina matic	`^`
(např. `A^k` je $A \cdot A \cdot \ldots \cdot A$ (k-krát)	

Operace „prvek po prvku“

součin matic „prvek po prvku“	`.*`
(např. `A.*B` je matice s prvky $a_{ij} b_{ij}$)	
pravé dělení „prvek po prvku“	`./`
(např. `A./B` je matice s prvky a_{ij} / b_{ij})	
levé dělení „prvek po prvku“	`.\`
(např. `A.\B` je matice s prvky b_{ij} / a_{ij})	
mocnina	`.^`
(např. `A.^k` je matice s prvky $(a_{ij})^k$)	

23 - MATLAB programování

Programování

záhlaví funkce

`function [`*výstupy*`] =` *jméno* `(`*vstupy*`)`

rozhodovací blok

`if` *podmínka 1*
blok příkazů
`end`

rozhodovací blok

`if` *podmínka 1*
blok příkazů
`elseif` *podmínka 2*
blok příkazů
...
`else`
blok příkazů `end`

cyklus se známým počtem opakování

`for` *rozsah hodnot*
blok příkazů `end`

cyklus s podmínkou

`while` *podmínka*
blok příkazů `end`

Relační operátory

je rovno $=$	`==`
není ovno $\neq$	`~=`
je menší $<$	`<`
je větší $>$	`>`
je menší nebo rovno $\leq$	`<=`
je větší nebo rovno $\geq$	`>=`

Logické operátory

a (konjunkce) $\wedge$	`and(a,b)` nebo `a & b`
nebo (disjunkce) $\vee$	`or(a,b)` nebo `a | b`
negace $\neg$	`not(a)` nebo `~a`

24 - MATLAB grafy a jiné

Grafy, ostatní příkazy

```
plot(x,y)
plot(x,y,'specifikace')
fplot(funkce,[a,b])
```

Specifikace grafu

Barvy		Symboly		Typy čar	
b	modrá	.	tečky	-	plná
g	zelená	o	kroužky	:	tečkovaná
r	červená	x	křížky ×	-.	čerchovaná
c	světle modrá	+	křížky +	--	tečkovaná
m	fialová	*	hvězdy		
y	žlutá	s	čtverce		
k	černá	d	kosočtverce		
		v	trojúhelníky (dolu)		
		^	trojúhelníky (nahoru)		
		<	trojúhelníky (vlevo)		
		>	trojúhelníky (vpravo)		
		p	pěticípé hvězdy		
		h	šesticípé hvězdy		

Úprava grafu

rozsah os	`axis([ , , , ])`
poměr os 1:1	`axis equal`
nadpis obrázku	`title('text')`
popis x-ové osy	`xlabel('text')`
popis y-ové osy	`ylabel('text')`
legenda	`legend('text1','text2',...)`
zobrazení mřížky do grafu	`grid on`
více grafů jednoho obrázku	`hold on`

Příkazy pro práci s proměnnými, programem

smazání proměnných	`clear`
zavření okna s obrázkem	`close`
smazání obrazovky	`clc`
uložení textu z command window	`diary 'soubor.txt'`
seznam proměnných	`who`
seznam proměnných s informacemi	`whos`
formát výpisu dlouhý	`format long`
formát výpisu krátký	`format short`
formát výpisu ve zlomku	`format rat`