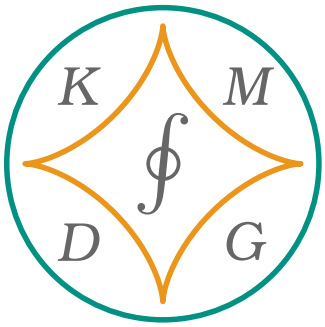
Numerická matematika
Řešené příklady s Matlabem a aplikované úlohy

Pavel Ludvík, Zuzana Morávková

Katedra matematiky a deskriptivní geometrie
Vysoká škola báňská – Technická Univerzita Ostrava



ISBN 978-80-248-3892-2

OBSAH

Předmluva

Studijní materiály tvořící tato skripta jsou určeny převážně pro studenty kombinované i prezenční formy fakulty strojní Vysoké školy báňské – Technické univerzity Ostrava navštěvující předmět Numerická matematika.

Naše skripta jsou určena jako přirozený doplněk skript Radka Kučery *Numerické metody*, která jsou zaměřená na teoretický výklad základních partií numerické matematiky. V předloženém materiálu jsme se soustředili na praktickou stránku a student zde nalezne řadu řešených úloh pokrývajících celou šíři kurzu. Síla numerických metod se projeví zejména ve spolupráci s počítačem. Jedním z hlavních cílů těchto skript proto bylo usnadnit studentům přechod od teoretického pochopení metody k jejímu využití v praxi a její implementaci v matematickém softwaru MATLAB. Další snahou autorů bylo ukázat, že numerická matematika není izolovaným vědním oborem, ale aplikovanou vědou *par excellence* s širokým využitím v inženýrské praxi.

Skripta jsou rozčleněna do tří logických celků. První část tvoří stručný úvod do programu MATLAB, který může sloužit buď nezávisle jako manuál pro začínající uživatele programu, nebo jako „servisní“ kapitola pro další části knihy, které s MATLABem intenzivně pracují. Druhá část obsahuje řešení základních typů úloh z numerické matematiky. Řešení je provedeno nejen teoreticky, ale souběžně také s pomocí MATLABu. U řady úloh se čtenář seznámí s několika různými implementacemi řešení a může porovnat jejich výhody a nevýhody. Úlohy jsou přehledně uspořádány do tématických celků respektujících běh kurzu Numerická matematika. Při práci s MATLABem se odkazujeme na první část, kde je použití programu podrobně vysvětleno.
Ve třetí části čtenář nalezne deset aplikovaných úloh ilustrující možnosti využití numerických metod v inženýrských oborech. Součástí řešení úlohy je vždy nalezení vhodné fyzikální interpretace problému, sestavení numerické úlohy a její vyřešení pomocí MATLABu. V této kapitole se čtenář dozví také o pokročilejších schopnostech MATLABu, které v jednodušších úlohách v předchozích částech skript nebylo nutné použít.

Autorem první části MATLAB, kapitol *Řešení nelineárních rovnic* 6, *Interpolace* 9, *Numerická derivace* 10.2. a *Obyčejné diferenciální rovnice: počáteční úlohy* 11 a aplikovaných úloh 12.2., 12.3., 12.6., 12.9., 12.10. je Zuzana Morávková. Autorem kapitol *Soustavy lineárních rovnic: přímé metody* 7, *Soustavy lineárních rovnic: iterační metody* 8, *Aproximace funkcí* 9.4., *Numerické integrování* 10 a aplikovaných úloh 12.1., 12.4., 12.5., 12.7., 12.8. je Pavel Ludvík.

Rádi bychom upozornili na webovou stránku `http://mdg.vsb.cz/portal/nm`, kde je umístěn nejen tento text, ale také řada dalších souvisejících studijních materiálů.

Tento studijní text vznikl za finanční podpory projektu IRP-FRVŠ 158/2015 *Inovace předmětu Numerická matematika na Fakultě strojní Vysoké školy báňské - Technické univerzitě Ostrava* a Katedry matematiky a deskriptivní geometrie VŠB-TUO.

Příjemně strávený čas s numerickou matematikou přeje kolektiv autorů.

I. MATLAB

KAPITOLA

1

ZÁKLADY PRÁCE S MATLABEM

1.1. Proměnné

Jména proměnných

V MATLABu se ve jménech proměnných rozlišují velká a malá písmena. Jméno proměnné může obsahovat písmena, číslice a podtržítka (_), maximálně však 31 znaků. První znak musí být písmeno. Pro přiřazení se používá symbol rovnítko (=). Pokud chybí přiřazení, zavede se automaticky proměnná `ans`, do které se uloží odeslaná hodnota.
Příkazy můžeme psát po jednom na řádek, nebo více příkazů na řádek oddělené čárkou (,) nebo středníkem (;). Symbol středník (;) slouží k potlačení výstupu, tj. příkaz se vykoná, ale nezobrazí se výsledek. Chceme-li znát hodnotu dané proměnné, zjistíme ji napsáním jména proměnné.

Příklad: Spočítáme hodnotu $55 + 17$, výsledek se automaticky uloží do proměnné `ans`.

```
>> 55+17
ans =
    72
```

Příklad: Do proměnné a přiřadíme hodnotu 12 a do proměnné b druhou mocninu a^2. Příkazy oddělíme čárkou.

```
>> a=12, b=a^2
a =
    12
b =
   144
```

Příkazy pro práci s proměnnými

Pro různé formáty výpisu nebo pro jejich smazání nám poslouží tyto příkazy.

`format`	formátování výstupu `long`, `short`, `rat`
`clear`	smazání proměnné

Příkazem `format long` nastavíme výpis čísel obvykle na 14 desetinných míst, příkazem `format short` na 4 místa. Mění se pouze formát výstupu na obrazovku, přesnost vnitřních výpočtů se nemění. Příkazem `format rat` zvolíme výpis ve tvaru zlomků.

```
>> format rat
>> 100/30
ans =
    10/3
>> format short
>> 100/30
ans =
    3.3333
>> format long
>> 100/30
ans =
   3.33333333333333
```

1.2. Čísla

MATLAB pracuje s typem matice. Vektory jsou matice typu $1 \times n$ nebo $n \times 1$. Číslo je čtvercová matice matice typu 1.

Zadávání čísel

Reálná čísla zadáváme s desetinnou tečkou (`.`), čísla lze také zadávat v exponenciálním tvaru například číslo 0.000014 zadáme takto `1.4e-5`, číslo 532300 takto `53.23e4`. Pro exponent můžeme používat symbol `E` nebo `e`. Při zadávání nesmíme dělat v čísle mezeru.

Příklad: Do proměnné `x` přiřadíme hodnotu 2.34 a do r hodnotu 0.0000532.

```
>> x=2.34
x =
    2.3400
>> r=0.532e-4
r =
  5.3200e-005
```

Příklad: Třemi způsoby přiřadíme hodnotu 123 000 do proměnné a.

```
>> a=123000
a =
       123000
>> a=123e3
a =
       123000
>> a=1.23e5
a =
       123000
```

Operace s čísly

Pro operace s čísly používáme následující symboly.

+	sčítání
-	odčítání
*	násobení
/	dělení
^	mocnina

Příklad: Vyčíslíme výraz $4^2 - 7 \cdot 12$.

```
>> 4^2-7*12
ans =
   -68
```

Jak jsme zvyklí z matematiky, provádějí se operace v předepsaném pořadí, nejprve operace umocnění, pak násobení a dělení a nakonec sčítání a odčítání. Například, když napíšeme $5 + 7 \cdot 2$, tak se nejprve provede násobení $7 \cdot 2 = 14$ a poté se provede sčítání $5 + 14 = 19$. Pokud chceme změnit pořadí v jakém se bude výraz počítat, například chceme-li spočítat $(5 + 7) \cdot 2$, použijeme závorky. Takže se nejprve provede sčítání $5 + 7 = 12$ a pak násobení $12 \cdot 2 = 24$.
Stejně tak v MATLABu platí stejná priorita operací a používáme kulaté závorky (žádné jiné pro tento účel nelze použít).

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

Příklad: Spočítáme hodnotu $\frac{111+171}{57-10}$. Ručně budeme počítat takto: $\frac{111+171}{57-10} = \frac{282}{47} = 6$. Na počítači zadáme:

```
>> (111+171)/(57-10)
ans =
     6
```

Vidíme, že jak čitatele $111 + 171$, tak jmenovatele $57 - 10$ jsme dali do závorek, aby se nejprve provedlo sčítání, odčítání a pak dělení. Ukážeme, co by se stalo, kdybychom tak neučinili.

```
>>  111+171/57-10
ans =
   104
```

A vidíme, že je výsledek špatně. MATLAB spočítal `111+171/57-10=111+3-10=104`.

KAPITOLA

2

MATICE A VEKTORY

2.1. Vektory

Vektory zadáváme do hranatých závorek a jednotlivé prvky oddělujeme čárkou nebo mezerou. K jednotlivým prvkům přistupujeme pomocí kulatých závorek.

`length(v)`	počet prvků vektoru v
`end`	index posledního prvku

Příklad: Zadáme vektor $u = (1, 8, -3.2, -1.3, 0.4)$, spočítáme počet jeho prvků a vypíšeme jeho čtvrtý a poslední prvek.

```
>> u=[1 8 -3.2 -1.3 0.4]
u =
    1.0000    8.0000   -3.2000   -1.3000    0.4000
>> length(u)
ans =
     5
>> u(4)
ans =
   -1.3000
>> u(end)
ans =
   0.4000
```

2.2. Vygenerování pravidelného vektoru

Potřebujeme-li vygenerovat vektor čísel, která jsou prvky aritmetické posloupnosti, použijeme dvojtečku (:).

```
prvni_clen:diference:mezni_clen
```

Příklad: Potřebujeme vektor (4, 7, 10, 13, 16, 19, 22), tedy čísla od 4 do 22 s krokem 3.

```
>> 4:3:22
ans =
     4     7    10    13    16    19    22
```

Příklad: Pokud se diference vynechá, MATLAB ji bere jako rovnu 1.

```
>> 2:10
ans =
     2     3     4     5     6     7     8     9    10
```

Příklad: Diference může být i záporné číslo.

```
>> 13:-3:0
ans =
    13    10     7     4     1
```

Příklad: Diference může být i desetinné číslo.

```
>> 5:0.2:6
ans =
    5.0000    5.2000    5.4000    5.6000    5.8000    6.0000
```

2.3. Matice

Matici zadáváme v hranatých závorkách, prvky na řádku oddělujeme mezerou nebo čárkou. Jednotlivé řádky pak středníkem nebo klávesou *Enter*.

Příklad: Zadáme matici $\boldsymbol{A} = \begin{pmatrix} 3 & 5 \\ 0 & 9 \\ -3 & 2 \end{pmatrix}$, prvky na řádku oddělíme mezerou a jednotlivé řádky středníkem.

```
>> A=[3 5; 0 9; -3 2]
A =
     3     5
     0     9
    -3     2
```

Matici lze vytvořit spojením jiných matic nebo vektorů, se kterými zacházíme jako s prvky. Matice „vedle sebe“ jsou jako prvky na řádku a matice „nad sebou“ jsou jako řádky v matici.

Příklad: Pomocí matic $\begin{pmatrix} 4 & 5 \\ 6 & 2 \end{pmatrix}$ a $\begin{pmatrix} 8 \\ 0 \end{pmatrix}$ vytvoříme matici $\begin{pmatrix} 4 & 5 & 8 \\ 6 & 2 & 0 \end{pmatrix}$. Umístíme tedy matice vedle sebe, oddělíme je mezerou jako prvky na řádku.

```
>> a=[4 5; 6 2], b=[8;0]
a =
     4     5
     6     2
b =
     8
     0
>> c=[a b]
c =
     4     5     8
     6     2     0
```

Příklad: Pomocí vektorů $v = (0, 0, 1)$ a $u = (2, 3, 4)$ vytvoříme matici $\begin{pmatrix} 0 & 0 & 1 \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix}$. Umístíme tedy vektory nad sebe, oddělíme je středníkem jako řádky v matici, a to v pořadí v, u, v.

```
>> u=[2 3 4]
u =
     2     3     4
>> v=[0 0 1]
v =
     0     0     1
>> [v;u;v]
ans =
     0     0     1
     2     3     4
     0     0     1
```

Příklad: Přidáme k dané matici $A = \begin{pmatrix} 3 & -8 & 67 & 1 \\ 1 & 0 & 0 & 23 \end{pmatrix}$ jako její poslední řádek vektor $\begin{pmatrix} 1 & 2 & -9 & 12 \end{pmatrix}$. Umístíme tedy vektor pod matici, oddělíme je od sebe středníkem.

```
>> A=[3 -8 67 1; 1 0 0 23]
A =
     3    -8    67     1
     1     0     0    23
>> A=[A; 1 2 -9 12]
A =
```

```
     3     -8     67      1
     1      0      0     23
     1      2     -9     12
```

2.4. Práce s částmi matic a vektorů

MATLAB umí pracovat s jednotlivými prvky, řádky nebo sloupci matice, tedy obecně se submaticemi.

`A(i,j)`	prvek a_{ij}
`A(i,:)`	i-tý řádek matice $\boldsymbol{A}$
`A(:,j)`	j-tý sloupec matice $\boldsymbol{A}$

Příklad: V tomto příkladě ukážeme práci s jednotlivými částmi (jsou označeny šedě) této matice.

$$\begin{pmatrix} 9 & 4 & 9 & 4 & 1 & 0 & 5 & 2 \\ 2 & 0 & 7 & 8 & 2 & 7 & 5 & 1 \\ 6 & 8 & 1 & 0 & 1 & 4 & 8 & 0 \\ 4 & 4 & 4 & 3 & 6 & 9 & 3 & 7 \\ 8 & 6 & 9 & 8 & 2 & 4 & 1 & 8 \\ 7 & 7 & 9 & 0 & 1 & 4 & 3 & 8 \\ 4 & 6 & 1 & 0 & 0 & 3 & 2 & 1 \end{pmatrix}$$

Nejprve matici zadáme.

```
>> A=[9 4 9 4 1 0 5 2;2 0 7 8 2 7 5 1;6 8 1 0 1 4 8 0;
4 4 4 3 6 9 3 7;8 6 9 8 2 4 1 8;7 7 9 0 1 4 3 8;4 6 1 0 0 3 2 1]
```

Vypíšeme první řádek matice A.

```
>> A(1,:)
ans =
     9     4     9     4     1     0     5     2
```

A vypíšeme sedmý sloupec.

```
>> A(:,7)
ans =
     5
     5
     8
     3
     1
     3
     2
```

Zobrazíme hodnotu prvku $a_{3,3}$.

```
>> A(3,3)
ans =
     1
```

A zobrazíme vyznačené submatice.

```
>> A(3:6,1)
ans =
     6
     4
     8
     7
>> A(4:7,3:5)
ans =
     4     3     6
     9     8     2
     9     0     1
```

Nakonec vypíšeme celou matici $\boldsymbol{A}$.

```
>> A
A    =

     9     4     9     4     1     0     5     2
     2     0     7     8     2     7     5     1
     6     8     1     0     1     4     8     0
     4     4     4     3     6     9     3     7
     8     6     9     8     2     4     1     8
     7     7     9     0     1     4     3     8
     4     6     1     0     0     3     2     1
```

Příklad: V matici $\boldsymbol{B}$ přepíšeme prvky v druhém sloupci na čísla 10.

```
>> B=[2 3 1 6;5 0 7 9;3 2 1 4]
B =
     2     3     1     6
     5     0     7     9
     3     2     1     4
>> B(:,2)=10
B =
     2    10     1     6
     5    10     7     9
     3    10     1     4
```

Smažeme první řádek tak, že ho nahradíme prázdnou matici `[ ]`.

```
>> B(1,:)=[ ]
B =
     5    10     7     9
     3    10     1     4
```

Příklad: Z matice C vypíšeme jen sudé sloupce.

```
>> C=[0.5 2 0.45 2.4 9 0 4.9 ;9 3.4 5.2 8 9 0 1]
C =
 0.5000    2.0000    0.4500    2.4000    9.0000         0    4.9000
 9.0000    3.4000    5.2000    8.0000    9.0000         0    1.0000
>> C(:,1:2:end)
ans =
    0.5000    0.4500    9.0000    4.9000
    9.0000    5.2000    9.0000    1.0000
```

2.5. Funkce pro tvorbu matic

Matice se speciálními prvky lze vytvořit i pomocí následujících příkazů.

`zeros(m,n)`	nulová matice typu $m \times n$
`ones(m,n)`	matice jedniček typu $m \times n$
`eye(m,n)`	jednotková matice typu $m \times n$
`rand(m,n)`	matice náhodných čísel z intervalu $(0,1)$ typu $m \times n$

Pokud se použijí funkce `zeros`, `ones`, `eye`, `rand` pouze s jedním parametrem (například `zeros(3)`), vznikne čtvercová matice příslušného řádu.

2.6. Operace s maticemi a vektory

Pro součet, rozdíl a transpozici matice nebo vektoru používáme následující symboly.

`+`	součet matic (např. `A+B` je matice s prvky $a_{ij} + b_{ij}$)
`-`	rozdíl matic (např. `A-B` je matice s prvky $a_{ij} - b_{ij}$)
`'`	transponovaná matice (např. `A'` je matice s prvky a_{ji})
`inv(A)`	inverzní matice $\boldsymbol{A}^{-1}$

Z matematiky známe součin matic, který se provádí tzv. „řádek krát sloupec“(například $\boldsymbol{A} \cdot \boldsymbol{B}$). Připomeňme, že násobit můžeme pouze matici typu $m \times n$ krát matici typu $n \times p$ a výsledkem násobení je matice typu $m \times p$. Dále známe součin čísla a matice (například $c \cdot \boldsymbol{A}$). Pro obě tyto operace používáme symbol hvězdička (*).

`*`	součin matic „řadek krát sloupec“
`/`	pravé maticové dělení (např. `A/B` je matice $\boldsymbol{A} \cdot \boldsymbol{B}^{-1}$)
`\`	levé maticové dělení (např. `A\B` je matice $\boldsymbol{A}^{-1} \cdot \boldsymbol{B}$)
`^`	mocnina matic (např. `A^k` je $\boldsymbol{A} \cdot \boldsymbol{A} \cdot \ldots \cdot \boldsymbol{A}$ (k-krát))

Příklad: Spočítáme $\boldsymbol{A} + \boldsymbol{B}$, $-4\boldsymbol{A}$, $\boldsymbol{A} \cdot \boldsymbol{B}$, $\boldsymbol{A}^2$.

```
>> A=[2 3; 0 9],B=[1 0; -1 5]
A =
     2     3
     0     9
B =
     1     0
    -1     5
>> A+B
ans =
     3     3
    -1    14
>> -4*A
ans =
    -8   -12
     0   -36
>> A*B
ans =
    -1    15
    -9    45
>> A^2
ans =
     4    33
     0    81
```

2.7. Operace „prvek po prvku”

Symbolem tečka a hvězdička (`.*`) označuje MATLAB tzv. součin „prvek po prvku“, který se počítá tak, že se vynásobí prvky obou matic na stejných pozicích. Takto můžeme násobit pouze matice stejných typů.

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

Příklad: Na maticích $\boldsymbol{A}, \boldsymbol{B}$ z předchozího příkladu ukážeme násobení „prvek po prvku“ (.∗), kdy se spolu vynásobí prvky na stejných pozicích.

```
>> A=[2 3; 0 9],B=[1 0; -1 5]
A =
     2     3
     0     9
B =
     1     0
    -1     5
>> A.*B
ans =
     2     0
     0    45
```

Příklad: Spočítáme hodnoty $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, $\frac{1}{5}$, $\frac{1}{6}$ a 2^2, 3^2, 4^2, 5^2, 6^2

```
>> x=2:6
x =
     2     3     4     5     6
>> 1./x
ans =
    0.5000    0.3333    0.2500    0.2000    0.1667
>> x.^2
ans =
     4     9    16    25    36
```

2.8. Součet, součin, minimum, maximum

Pro výpočet součtu, součinu nebo pro nalezení největšího, nejmenšího čísla mezi prvky vektoru (bez ohledu na to, zda je vektor sloupcový nebo řádkový) nebo ve sloupcích matice máme k dispozici funkce.

`max( )`	maximum ve vektoru nebo ve sloupcích matice
`min( )`	minimum ve vektoru nebo ve sloupcích matice
`sum( )`	součet ve vektoru nebo ve sloupcích matice
`prod( )`	součin ve vektoru nebo ve sloupcích matice

Příklad: Spočítáme maximum z prvků vektoru $(2, -1, 4, 9, 2)$.

```
>> max([2 -1 4 9 2])
ans =
     9
```

Příklad: Spočítáme maximum ve sloupcích matice $\begin{pmatrix} 2 & 5 & 6 \\ 3 & 0 & 9 \\ 1 & 8 & -1 \\ 4 & 9 & 2 \end{pmatrix}$.

```
>> matice=[2 5 6; 3 0 9; 1 8 -1; 4 9 2]
matice =
     2     5     6
     3     0     9
     1     8    -1
     4     9     2
>> max(matice)
ans =
     4     9     9
```

A určíme maximum ze všech prvků matice.

```
>> maximum=max(max(matice))
maximum =
     9
```

Spočítáme součiny ve sloupcích matice.

```
>> soucins=prod(matice)
soucins =
    24     0  -108
```

A nakonec spočítáme součiny v řádcích matice, a to pomocí matice transponované (tj. řádky se stanou sloupci).

```
>> soucinr=prod(matice')
soucinr =
    60     0    -8    72
```

2.9. Řešení soustav lineárních rovnic

Soustavu lineárních rovnic lze vyřešit několika způsoby. Jedním z nich je použít dělení zprava, tedy pro soustavu $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ najdeme řešení příkazem `x=A\b`. A to jak v případě, kdy má soustava jedno řešní, tak v případě, kdy má nekonečně mnoho řešení závislých na parametru či parametrech. Pak dostaneme jedno z těchto řešení.
Chceme-li najít všechna řešení soustavy, která má nekonečně mnoho řešení, použijeme příkaz pro Gaussovu-Jordanovu eliminaci.

`x=A\b`	řešení soustavy lineárních rovnic
`rref([A,b])`	Gaussova-Jordanova eliminace

Najděte řešení soustavy lineárních rovnic

$$\begin{aligned} 3x_1 + 7x_2 - 2x_3 &= 2, \\ 4x_1 + 3x_2 + 2x_3 &= 5, \\ 2x_1 + 11x_2 - 6x_3 &= -1, \\ -x_1 + 7x_2 + 3x_3 &= -1. \end{aligned}$$

Nejprve zadáme matici a vektor.

```
>> A=[ 3  7 -2; 4 3 2; 2 11 -6; -1 7 3]
A =
       3             7            -2
       4             3             2
       2            11            -6
      -1             7             3
>> b=[2;5;-1;-1]
b =
       2
       5
      -1
      -1
```

Pomocí zpětného lomítka najdeme řešení soustavy.

```
>> x=A\b
x =
    1.1714
   -0.1200
    0.3371
```

Zkontrolujeme chybu výpočtu.

```
>> A*x-b
ans =
  1.0e-014 *
   -0.0444
   -0.1776
    0.0888
    0.2220
```

Řešení lze nalézt i převedením rozšířené matice na trojúhelníkový tvar.

```
>> rref([A,b])
ans =
    1.0000         0         0    1.1714
         0    1.0000         0   -0.1200
         0         0    1.0000    0.3371
```

```
0    0    0    0
```

Soustava má řešení

$$x_1 = 1.1714\,, \quad x_2 = -0.1200\,, \quad x_3 = 0.3371\,.$$

KAPITOLA

3

FUNKCE

3.1. Základní matematické funkce

Konstanty

`pi`	Ludolfovo číslo $\pi = 3.1415\ldots$
`exp(1)`	Eulerovo číslo $e = 2.7183\ldots$

Nyní uvedeme seznam funkcí. Funkce můžeme aplikovat i na matice, funkce se pak vyčíslí pro jednotlivé prvky matice.

Goniometrické a cyklometrické funkce

MATLAB počítá v radiánech, tedy argumenty goniometrických funkcí a hodnoty cyklometrických funkcí jsou v radiánech.

`sin(x)`	sinus $y = \sin(x)$
`cos(x)`	kosinus $y = \cos(x)$
`tan(x)`	tangens $y = \tan(x)$
`cot(x)`	kotangens $y = \cot(x)$
`asin(x)`	arkussinus $y = \arcsin(x)$
`acos(x)`	arkuskosinus $y = \arccos(x)$
`atan(x)`	arkustangens $y = \arctan(x)$
`acot(x)`	arkuskotangens $y = \operatorname{arccot}(x)$

Příklad: Spočítáme hodnotu funkce $y = \dfrac{\tan(x) + 2\pi}{\sin(x - \pi/2)}$ v bodě $x = 0$.

```
>> x=0;y=(tan(x)+2*pi)/sin(x-pi/2)
y =
    -6.2832
```

Exponenciální a logaritmické funkce

`log(x)`	přirozený logaritmus $y = \ln(x)$
`log10(x)`	dekadický logaritmus $y = \log(x)$
`log2(x)`	logaritmus při základu 2, $y = \log_2(x)$
`exp(x)`	exponeciální funkce $y = e^x$
`pow2(x)`	exponenciální funkce při základu 2, $y = 2^x$

Příklad: Spočítáme hodnotu $\log_2(8)$.

```
>> log2(8)
ans =
      3
```

Příklad: Spočítáme hodnotu $\log_3(81)$. MATLAB nemá k dispozici funkci pro logaritmus se základem 3, avšak z matematiky známe vzorec $\log_a(x) = \frac{\ln(x)}{\ln(a)}$, který použijeme.

```
>> log(81)/log(3)
ans =
     4
```

A provedeme kontrolu.

```
>>3^4
ans =
    81
```

Ostatní funkce

`sqrt(x)`	druhá odmocnina $y = \sqrt{x}$
`abs(x)`	abolutní hodnota $y = \lvert x \rvert$
`sign(x)`	funkce signum $y = \text{sign}(x)$

Příklad: Spočítáme hodnotu funkce $y = \frac{1}{x} + e^{-x}$ v bodě $x = 4$.

```
>> x=4
x =
```

```
     4
>> 1/x+exp(-x)
ans =
    0.2683
```

Příklad: Spočítáme hodnotu funkce $y = \sqrt[3]{x-2}$ v bodě $x = 241.5$.

```
>> x=241.5; y=(x-2)^(1/3)
>> y =
    6.2101
```

Příklad: Spočítáme absolutní hodnotu čísel $-5,\ 9,\ 0,\ 3,\ 4,\ -8$.

```
>> posl=[-5 9 0 3 4 -8]
posl =
    -5     9     0     3     4    -8
>> vysledek=abs(posl)
vysledek =
     5     9     0     3     4     8
```

3.2. Definice vlastní funkce (anonymní funkce)

Vlastní matematickou funkci nadefinujeme následujícím způsobem.

```
jméno=@(proměnné)předpis
```

Příklad: Nadefinujeme funkci $f = \sin(x^2) - \sqrt{3-x}$ a vyčíslíme v bodě 1.

```
>> f=@(x)sin(x.^2)-sqrt(3-x)
f =
     f(x) = @(x)sin(x.^2)-sqrt(3-x)
>> f(1)
ans =
   -0.5727
```

Příklad: V definici funkce v předchozím příkladě jsme mocninu zapsali pomocí operace (.^). Je-li potřeba spočítat hodnoty funkce ve více bodech, je nezbytné použít operace „prvek po prvku".

```
>> f=@(x)1./x
f =
     f(x) = @(x)1./x
>> x=1:5
x =
     1     2     3     4     5
```

```
>> f(x)
ans =
    1.0000    0.5000    0.3333    0.2500    0.2000
>> [x',f(x)']
ans =
    1.0000    1.0000
    2.0000    0.5000
    3.0000    0.3333
    4.0000    0.2500
    5.0000    0.2000
```

KAPITOLA

4

GRAFY

4.1. Vykreslení grafu

Ke grafickému znázornění dat nebo vykreslení grafu funkce máme v MATLABu k dispozici příkaz `plot`. Vstupem příkazu `plot` jsou souřadnice bodů, které jsou spojeny lomenou čarou. Chceme-li vykreslit graf funkce, lze použít i příkaz `fplot`, prvním parametrem je funkční předpis v apostrofech a druhým je interval, na kterém chceme graf vykreslit.

`plot(x,y)`	graf bodů o souřadnicích (x, y)
`fplot('f',[a,b])`	graf funkce f na intervalu $\langle a, b \rangle$

Příklad: Vykreslíme graf funkce $y = x^2$ na intervalu $\langle -4, 4 \rangle$ pomocí příkazu `plot` nebo `fplot`. Ukážeme čtyři způsoby, výsledný graf je zobrazen na obrázku 4.1.

1. Nejprve do proměnné `x` uložíme souřadnice x, které vygenerujeme jako čísla mezi -4 a 4 s krokem 0.1, tedy čísla -4, -3.9, -3.8, $\ldots, 3.9$, 4. Pak vytvoříme vektor `y` jako funkční hodnoty funkce $y = x^2$ v bodech `x`. A vykreslíme graf.

```
>> x=-4:0.1:4;
>> y=x.^2;
>> plot(x,y)
```

2. Do proměnné `x` uložíme souřadnice x, které vygenerujeme jako body mezi -4 a 4. A vykreslíme graf.

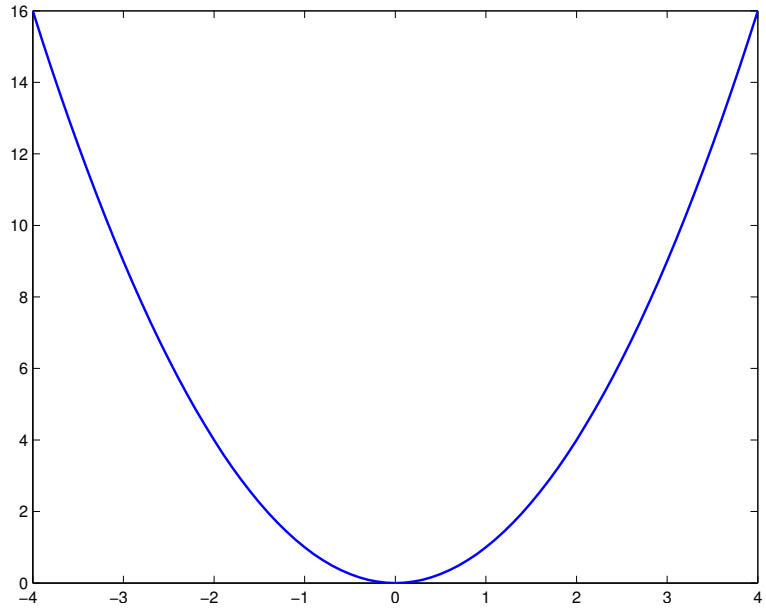
```
>> x=-4:0.1:4;
>> plot(x,x.^2)
```

3. Vykreslíme graf funkce $f(x) = x^2$ na intervalu $\langle -4, 4 \rangle$.

```
>> fplot('x^2',[-4,4])
```

4. Nadefinujeme vlastní funkci $f(x) = x^2$ a vykreslíme graf na intervalu $\langle -4, 4 \rangle$.

```
>> f=@(x)x^2
>> fplot(f,[-4,4])
```



Obrázek 4.1: Graf funkce $y = x^2$ na intervalu $\langle -4, 4 \rangle$

Specifikace grafu

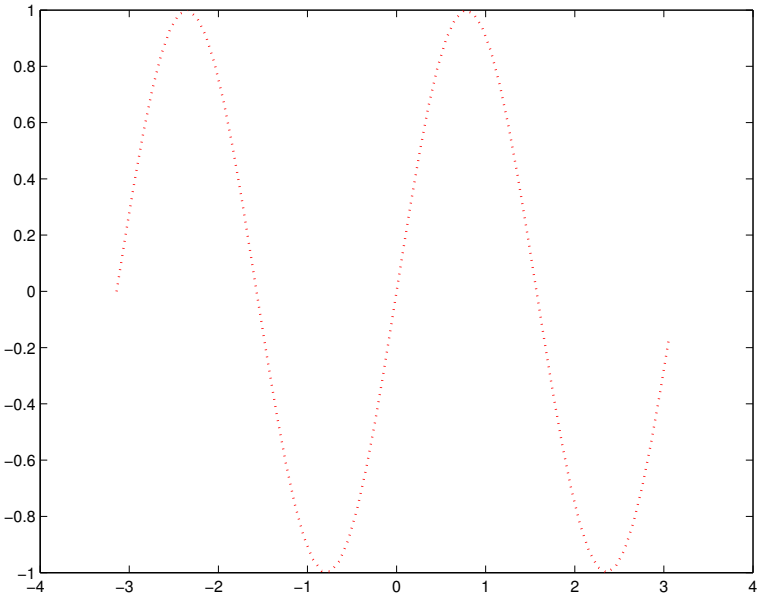
Chceme-li graf vykreslit jinak než modrou plnou čárou, použijeme tzv. specifikaci grafu. Pro specifikaci můžeme použít libovolnou kombinaci voleb z prvního, druhého a třetího sloupce tabulky (v uvedeném pořadí), přičemž některou lze vynechat. Specifikaci uvádíme jako řetězec, tj. do apostrofů (`'`), a je to nepovinný třetí parametr příkazu `plot` a `fplot`. Například červenou čárkovanou čáru vytvoříme specifikací `'r-'`, zelené kroužky `'go'`, žlutou čáru s trojúhelníčky `'yv-'`.

`plot(x,y,'specifikace')`	graf bodů o souřadnicích (x, y)
`fplot('f',[a,b],'specifikace')`	graf funkce f na intervalu $\langle a, b \rangle$

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

Příklad: Ukážeme graf funkce $y = \sin(2x)$ na intervalu $\langle -\pi, \pi \rangle$ červenou tečkovanou čarou, viz obrázek 4.2.

```
>> fplot('sin(2*x)',[-pi,pi],'r:')
```

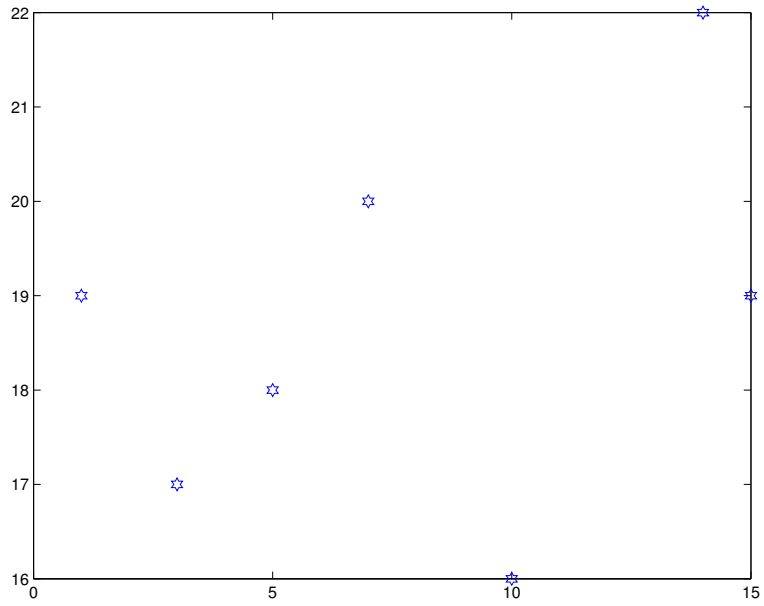


Obrázek 4.2: Graf funkce $y = \sin(2x)$

Příklad: Specifikace pomocí symbolů se často používá, chceme-li zobrazit diskrétní data. Máme zadané hodnoty v tabulce, a zobrazíme je jako hvězdičky, viz obrázek 4.3.

t	1	3	5	7	10	14	15
s	19	17	18	20	16	22	19

```
>> t=[1 3 5 7 10 14 15];
>> s=[19 17 18 20 16 22 19];
>> plot(t,s,'h')
```



Obrázek 4.3: Graf diskrétních dat

4.2. Více grafů najednou

`hold`	vykreslení grafů do jednoho obrázku, nastavením `on` tuto vlastnost zapneme, `off` vypneme

Příklad: Máme naměřené hodnoty v tabulce.

x_i	1	3	7	9	10
y_i	9	6	0	32	89

Víme, že předpokládané chování této veličiny je dáno funkcí

$$y = x^2 - xe^{\frac{x}{10}} + 1\,.$$

Chceme do jednoho grafu zobrazit naměřené i teoretické hodnoty. Nejprve zadáme data z tabulky do proměnných `xi, yi` a příkazem `hold on` otevřeme okno (zatím prázdné), do kterého se budou zobrazovat všechny grafy, které budeme vykreslovat.

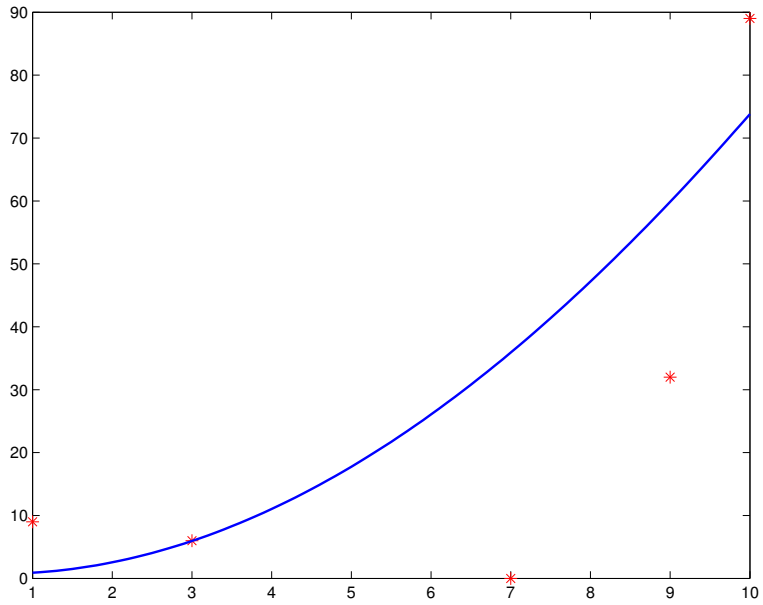
```
>> xi=[1 3 7 9 10];
>> yi=[9 6 0 32 89];
>> hold on
```

Vykreslíme data z tabulky jako červené hvězdičky.

```
>> plot(xi,yi,'r*')
```

A přikreslíme graf teoretických hodnot.

```
>> fplot('x^2-x*exp(x/10)+1',[1,10])
```



Obrázek 4.4: Graf funkce

4.3. Nastavení grafu

Do grafu můžeme přidat popisy os, nadpis nebo změnit rozsah os.

`axis([x1,x2,y1,y2])`	rozsah os pro první proměnnou od `x1` do `x2`, pro druhou od `y1` do `y2`
`axis equal`	osy v poměru 1:1
`grid`	zobrazení mřížky do grafu, zapnout `on`, vypnout `off`
`title('text')`	titulek obrázku
`xlabel('text')`	popis x-ové osy
`ylabel('text')`	popis y-ové osy
`legend('text1','text2','text3')`	legenda ke grafům

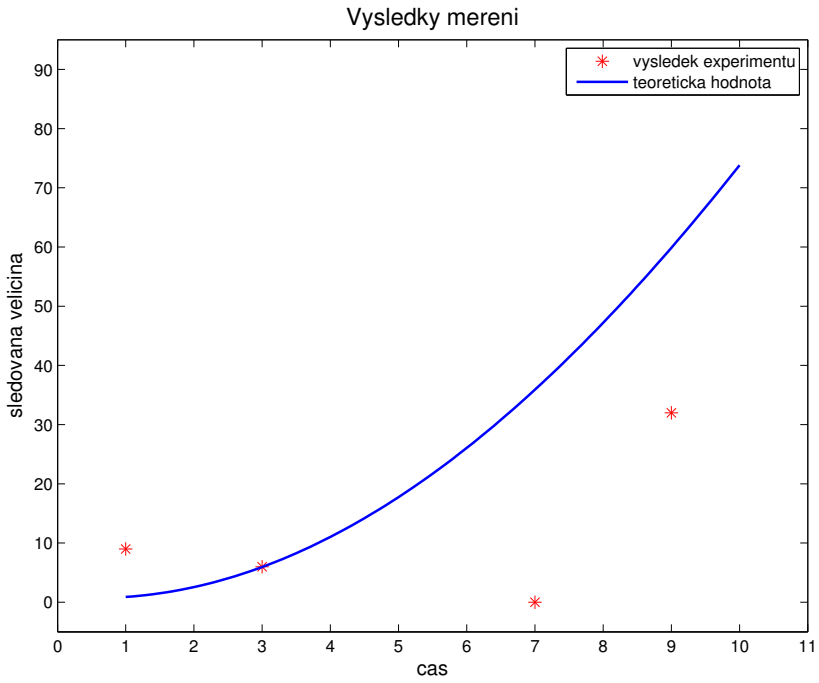
Příklad: Nejprve vykreslíme graf stejně jako v předchozím příkladě.

```
>> xi=[1 3 7 9 10];
>> yi=[9 6 0 32 89];
```

```
>> hold on
>> plot(xi,yi,'r*')
>> fplot('x^2-x*exp(x/10)+1',[-5,10])
```

Grafu změníme rozsah os, přidáme legendu, nadpis a popis obou os.

```
>> axis([0,11,-5,95])
>> legend('vysledek experimentu','teoreticka hodnota')
>> title('Vysledky mereni')
>> xlabel('cas')
>> ylabel('sledovana velicina')
```



Obrázek 4.5: Graf funkce

KAPITOLA

5

PROGRAMOVÁNÍ V MATLABU

5.1. Logická proměnná

Pravda, nepravda

V MATLABu není speciální datový typ pro logické proměnné, *pravda* má hodnotu `1` a *nepravda* má hodnotu `0`.

Relační operátory

Relační operátory lze použít na čísla, vektory i matice, kde se pak srovnávají prvek po prvku. Výsledkem je jedna (platí) nebo nula (neplatí).

`==`	rovnost $=$
`~=`	nerovnost $\neq$
`<`	menší než $<$
`>`	větší než $>$
`<=`	menší nebo roven $\leq$
`>=`	větší nebo roven $\geq$

Příklad: Zjistíme, která čísla z vektoru $\boldsymbol{a} = (3,\ 9,\ 4,\ 1,\ -5,\ 3)$ jsou větší než 2 a pak porovnáme prvky vektoru $\boldsymbol{a}$ s prvky vektoru $\boldsymbol{b} = (4,\ 4,\ 0,\ 9,\ 3,\ 0)$. MATLAB vrátí vektor o stejné délce, na pozicích prvků, pro které daná podmínka platí je `1` a na pozicích, kde neplatí je `0`.

```
>> a=[3 9 4 1 -5 3], b=[4 4 0 9 3 0]
a =
     3     9     4     1    -5     3
b =
     4     4     0     9     3     0
>> a>2
ans =
     1     1     1     0     0     1
>> a>b
ans =
     0     1     1     0     0     1
```

Logické operátory

Připomeňme logické operátory negace ($\neg$), konjunkce ($\wedge$) a disjunkce ($\vee$) a jejich zápis v MATLABu.

A	B	not(A) nebo ~A	and(A,B) nebo A&B	or(A,B) nebo A|B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Příklad: Zjistíme, která čísla z vektoru $\boldsymbol{a} = (3,\ 9,\ 4,\ 1,\ -5,\ 3)$ jsou větší než 2 a zároveň menší než 5.

```
>> a=[3 9 4 1 -5 3]
a =
     3     9     4     1    -5     3
>> a>2 & a<5
ans =
     1     0     1     0     0     1
```

5.2. Rozhodovací blok `if`

Chceme-li provést výpočet popsaný *blokem příkazů* pouze v případě, že je splněna *podmínka*, použijeme rozhodovací blok. Jednotlivé příkazy v bloku oddělujeme čárkami nebo středníky. Píšeme-li rozhodovací blok na jeden řádek, tak i za podmínku píšeme čárku.

```
if podmínka
    blok příkazů
end
```

V případě, že je potřeba větvení podle více podmínek, použijeme následující strukturu. Blok `elseif` *podmínka , blok příkazů* můžeme uvést i vícekrát.

```
if podmínka 1
    blok příkazů 1
elseif podmínka 2
    blok příkazů 2
...
else
    blok příkazů 3
```

Příklad: Spočítáme kořeny kvadratické rovnice. Zadáme koeficienty, spočítáme diskriminant a pokud je nezáporný, tak vypočítáme kořeny. Vyzkoušíme pro rovnici $x^2 - x - 12 = 0$.

```
>> a=1; b=-1; c=-12;
>> D=b^2-4*a*c
D =
    49
>> if D>=0, x(1)=(-b+sqrt(D))/(2*a),x(2)=(-b-sqrt(D))/(2*a),end
x =
     4     -3
```

Ošetříme případ, kdy má rovnice jeden kořen, tedy má nulový diskriminant. Stejné dva příkazy (výpočet diskriminantu a rozhodovací blok) vyzkoušíme pro tři různé příklady $x^2 + 2x + 1 = 0$, $x^2 + 3 = 0$, $x^2 - x - 12 = 0$.

```
>> a=1; b=2; c=1;
>> D=b^2-4*a*c
D =
     0
>> if D>0, x(1)=(-b+sqrt(D))/(2*a),x(2)=(-b-sqrt(D))/(2*a),
   elseif D==0, x(1)=-b/(2*a),end
x =
    -1
>> a=1; b=0; c=3;
>> D=b^2-4*a*c
D =
    -12
>> if D>0, x(1)=(-b+sqrt(D))/(2*a),x(2)=(-b-sqrt(D))/(2*a),
   elseif D==0, x(1)=-b/(2*a),end
>> a=1; b=-1; c=-12;
>> D=b^2-4*a*c
```

```
D =
    49
>> if D>0, x(1)=(-b+sqrt(D))/(2*a),x(2)=(-b-sqrt(D))/(2*a),
   elseif D==0, x(1)=-b/(2*a),end
x =
     4     -3
```

5.3. Cyklus `for` a `while`

Občas je potřeba vykonat blok příkazů vícekrát ze sebou. Pokud známe počet opakování použijeme cyklus se známým počtem opakování. Řídící proměnná nabývá postupně hodnot v zadaném rozsahu.

cyklus se známým počtem opakování

```
for řídící proměnná=rozsah hodnot
    blok příkazů
end
```

Příklad: Spočítáme faktoriál čísla 8.

```
>> fakt=1
fakt =
     1
>> for i=2:8, fakt=fakt*i, end
fakt =
     2
fakt =
     6
fakt =
    24
fakt =
   120

fakt =
   720
fakt =
        5040
fakt =
       40320
```

cyklus s podmínkou

V případě, že neznáme počet opakování, pouze známe *podmínku*, při jejíž splnění se *blok příkazů* má stále opakovat, použijeme následující strukturu.

```
while podmínka
    blok příkazů
end
```

Příklad: Budeme postupně sčítat čísla z posloupnosti 1, 3, 2, 4, 5, 6, 3, 4, 5, 3, 6, 7, 8, dokud nebude součet větší než 20.

```
>> a=[1 3 2 4 5 6 3 4 5 3 6 7 8 ];
>> soucet=0
soucet =
     0
>> i=1
i =
     1
>>  while soucet<=20, soucet=soucet+a(i), i=i+1; end
soucet =
     1
soucet =
     4
soucet =
     6
soucet =
    10
soucet =
    15
soucet =
    21
```

II. Řešené příklady v Matlabu

KAPITOLA

6

ŘEŠENÍ NELINEÁRNÍCH ROVNIC

Nelineární rovnice

Je dána spojitá funkce $f(x)$. Hledáme $\bar{x} \in \mathbb{R}$, 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.

6.1. 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 $a^{k+1} := a^k$, $b^{k+1} := x^k$.

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.

Příklad 1: Určete všechny kořeny rovnice

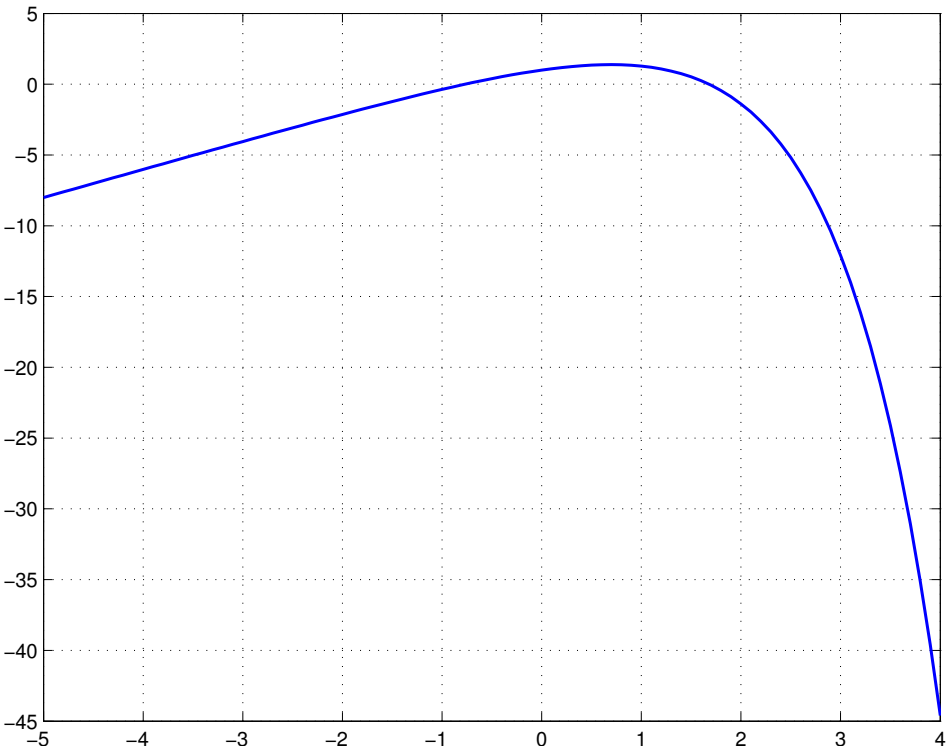
$$2x + 2 - e^x = 0$$

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

Provedeme separaci kořenů. Zadáme funkci a vykreslíme její graf na vhodném intervalu, který určíme podle definičního oboru funkce f. Funkce $f(x) = 2x + 2 - e^x$ má definiční obor $D_f = \mathbb{R}$.

```
>> f=@(x)(2*x+2-exp(x))
f =
    @(x)(2*x+2-exp(x))
>> fplot(f,[-5,4])
>> grid on
```

anonymní funkce str. 24, příkazy `fplot` str. 26, `grid` str. 30



Lze snadno vidět, že graf funkce má dva průsečíky s osou x, ležících v intervalech $\langle -1, 0\rangle$ a $\langle 1, 2\rangle$.
Ukážeme dvě implementace metody půlení intervalu.

Implementace A

Začneme s výpočtem kořene z intervalu $\langle -1, 0 \rangle$. Do proměnných a^1 a b^1 zadáme meze intervalu, který jsme zjistili separací a vypočítáme funkční hodnoty $f(a^1)$ a $f(b^1)$.

```
>> a1=-1
a1 =
    -1
>> b1=0
b1 =
     0
>> f(a1)
ans =
   -0.3679
>> f(b1)
ans =
     1
```

Spočítáme x^1 jako polovinu intervalu (a^1, b^1) a v tomto bodě spočítáme funkční hodnotu.

```
>> x1=(a1+b1)/2
x1 =
   -0.5000
>> f(x1)
ans =
    0.3935
```

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

```
>> abs(b1-a1)/2
ans =
    0.5000
```

Nalezaná data zapíšeme do tabulky.

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k - a^k \rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$

Připomeňme, že funkce sign je funkce znaménko, tj. kladným číslům přiřadí hodnotu 1 a záporným hodnotu -1.

Podle znamének funkčních hodnot $f(a^1)$, $f(x^1)$, $f(b^1)$ určíme interval v druhém kroku (a^2, b^2). Jelikož $f(a^1) < 0$ a $f(x^1) > 0$, tedy platí $f(a^1)f(x^1) < 0$, bude v druhém kroku $a^2 = a^1 = -1$ a $b^2 = x^1 = -0.5$.

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

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k-a^k\rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$
2	-1	$-$			-0.5	$+$	

Spočítáme x^2 jako polovinu intervalu (a^2, b^2) a v tomto bodě spočítáme funkční hodnotu. Spočítáme chybu výpočtu, a dokud je větší než ε, výpočet pokračuje dál.

```
>> x2=(a2+b2)/2
x2 =
    -0.7500
>> f(x2)
ans =
     0.0276
>> abs(b2-a2)/2
ans =
     0.2500
```

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k-a^k\rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$
2	-1	$-$	-0.75	$+$	-0.5	$+$	$0.25 > 10^{-2}$

Podle znamének $f(a^2)$, $f(x^2)$, $f(b^2)$ určíme interval v dalším kroku (a^3, b^3). Jelikož $f(a^2) < 0$ a $f(x^2) > 0$, tedy platí $f(a^2)f(x^2) < 0$, bude v druhém kroku $a^3 = a^2 = -1$ a $b^3 = x^2 = -0.75$.

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

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k-a^k\rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$
2	-1	$-$	-0.75	$+$	-0.5	$+$	$0.25 > 10^{-2}$
3	-1	$-$			-0.75	$+$	

Ve výpočtu pokračujeme dál.

```
>> x3=(a3+b3)/2
x3 =
    -0.8750
>> f(x3)
ans =
    -0.1669
>> abs(b3-a3)/2
ans =
     0.1250
```

Podle znamének $f(a^3)$, $f(x^3)$, $f(b^3)$ určíme nový interval (a^4, b^4). Jelikož $f(x^3) < 0$ a $f(b^3) > 0$, tedy platí $f(x^3)f(b^3) < 0$, bude v druhém kroku $a^4 = x^3 = -0.825$ a $b^4 = b^3 = -0.75$.

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

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k-a^k\rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$
2	-1	$-$	-0.75	$+$	-0.5	$+$	$0.25 > 10^{-2}$
3	-1	$-$	-0.875	$-$	-0.75	$+$	$0.125 > 10^{-2}$
4	-0.875	$-$			-0.75	$+$	

Výpočet pokračuje dál, dokud je chyba větší než ε.
Data zapisujeme do tabulky.

k	a^k	$\text{sign}(f(a^k))$	x^k	$\text{sign}(f(x^k))$	b^k	$\text{sign}(f(b^k))$	$\frac{\lvert b^k-a^k\rvert}{2}$
1	-1	$-$	-0.5	$+$	0	$+$	$0.5 > 10^{-2}$
2	-1	$-$	-0.75	$+$	-0.5	$+$	$0.25 > 10^{-2}$
3	-1	$-$	-0.875	$-$	-0.75	$+$	$0.125 > 10^{-2}$
4	-0.875	$-$	-0.8125	$-$	-0.75	$+$	$0.0625 > 10^{-2}$
5	-0.8125	$-$	-0.7813	$-$	-0.75	$+$	$0.0313 > 10^{-2}$
6	-0.7813	$-$	-0.7656	$+$	-0.75	$+$	$0.0156 > 10^{-2}$
7	-0.7813	$-$	-0.7734	$-$	-0.7656	$+$	$0.0078 \leq 10^{-2}$

Chyba 0.0078 je již menší nebo rovna než zadaná přesnost $\varepsilon = 10^{-2}$. Přibližnou hodnotu kořene $x^7 = -0.7734$ zaokrouhlíme na dvě desetinná místa (neboť zadaná přesnost je na dvě desetinná místa) a zapíšeme následujícím způsobem.

Kořen rovnice $2x + 2 - e^x = 0$ je -0.77 ± 10^{-2}.

Ostatní kořeny spočítáme obdobným způsobem.

Implementace B
Výpočteme kořen z intervalu $\langle 1, 2\rangle$.
Do proměnných `a` a `b` zadáme meze intervalu, který jsme zjistili separací. A nastavíme inicializační hodnotu indexu `k`.

```
>> k=0; a=1; b=2;
```

V každém kroku zvýšíme index `k` o jedna, spočítáme `x(k)` a chybu. Volbu intervalu do dalšího kroku provedeme pomocí rozhodovacího bloku `if`.

```
>> k=k+1
k =
     1
>> x(k)=(a(k)+b(k))/2
x =
    1.5000
>> chyba=(b(k)-a(k))/2
chyba =
    0.5000
>> if f(a(k))*f(x(k))<0, a(k+1)=a(k);b(k+1)=x(k);
else a(k+1)=x(k); b(k+1)=b(k);end
```

rozhodovací blok str. 33

Následující čtyři řádky opakujeme, dokud je chyba větší než zadaná přesnost.

```
>> k=k+1
>> x(k)=(a(k)+b(k))/2
>> chyba=(b(k)-a(k))/2
>> if f(a(k))*f(x(k))<0, a(k+1)=a(k);b(k+1)=x(k);
else a(k+1)=x(k); b(k+1)=b(k);end
```

rozhodovací blok str. 33

V sedmém kroku je dosažena zadaná přesnost.

```
>> k=k+1
k =
     7
>> x(k)=(a(k)+b(k))/2
x =
    1.5000    1.7500    1.6250    1.6875    1.6563    1.6719
   1.6797
>> chyba=(b(k)-a(k))/2
chyba =
    0.0078
```

Přibližnou hodnotu kořene zaokrouhlíme na dvě desetinná místa.

Kořen rovnice $2x + 2 - e^x = 0$ je 1.68 ± 10^{-2}.

6.2. Newtonova metoda

Nechť jsou splněny následující předpoklady:

1. f' je nenulová na intervalu $\langle a, b \rangle$;
2. f'' nemění znaménko v intervalu (a, b);
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 $x^0 \in \langle a, b\rangle$.
Výpočet ukončíme při dosažení zadané přesnosti ε, tj. když platí

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

Příklad 2: Určete všechny kořeny rovnice

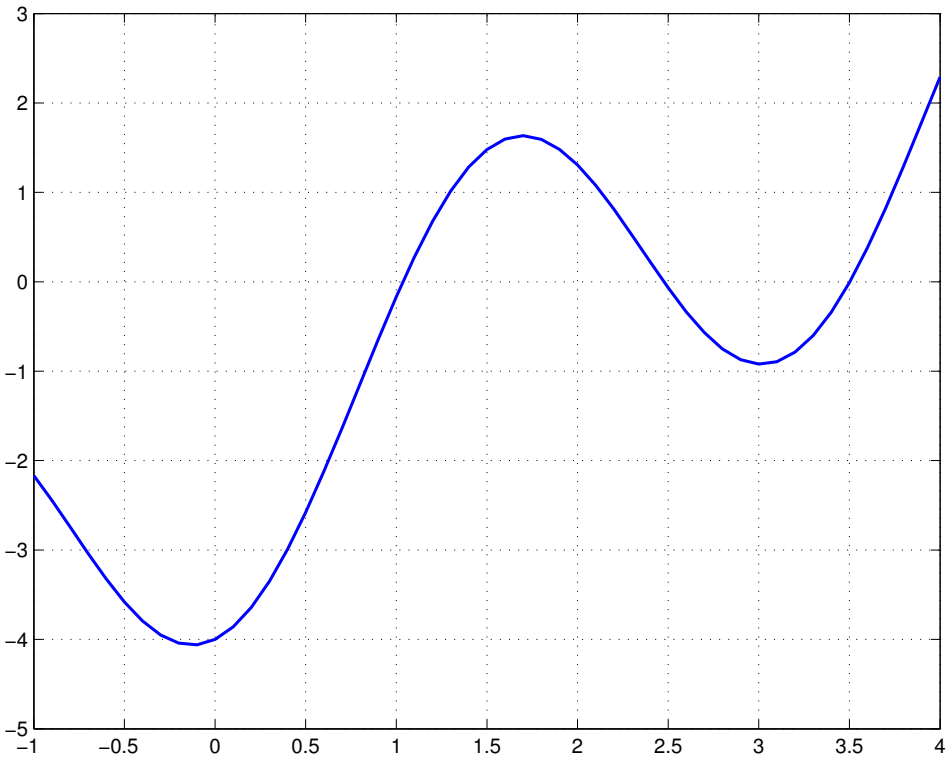
$$x - 4\cos^2(x) = 0$$

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

Provedeme separaci kořenů. Zadáme funkci na levé straně rovnice $f(x) = x - 4\cos^2(x)$ a vykreslíme její graf.

```
>> f=@(x)x-4*cos(x).^2
f =
    @(x)x-4*cos(x).^2
>> fplot(f,[-1,5])
>> grid on
```

anonymní funkce str. 24, příkazy `fplot` str. 26, `grid` str. 30



Z grafu vidíme, že rovnice má tři kořeny (průsečíky s osou x) a to v intervalech $\langle 1,2\rangle$, $\langle 2,3\rangle$ a $\langle 3,4\rangle$.

Implementace A
Nejprve budeme počítat kořen z intervalu $\langle 3,4\rangle$. Zadáme meze tohoto intervalu.

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

Spočítáme první a druhou derivaci.

$$f' = 1 + 8\cos(x)\sin(x) \qquad f'' = -8\sin^2(x) + 8\cos^2(x)$$

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

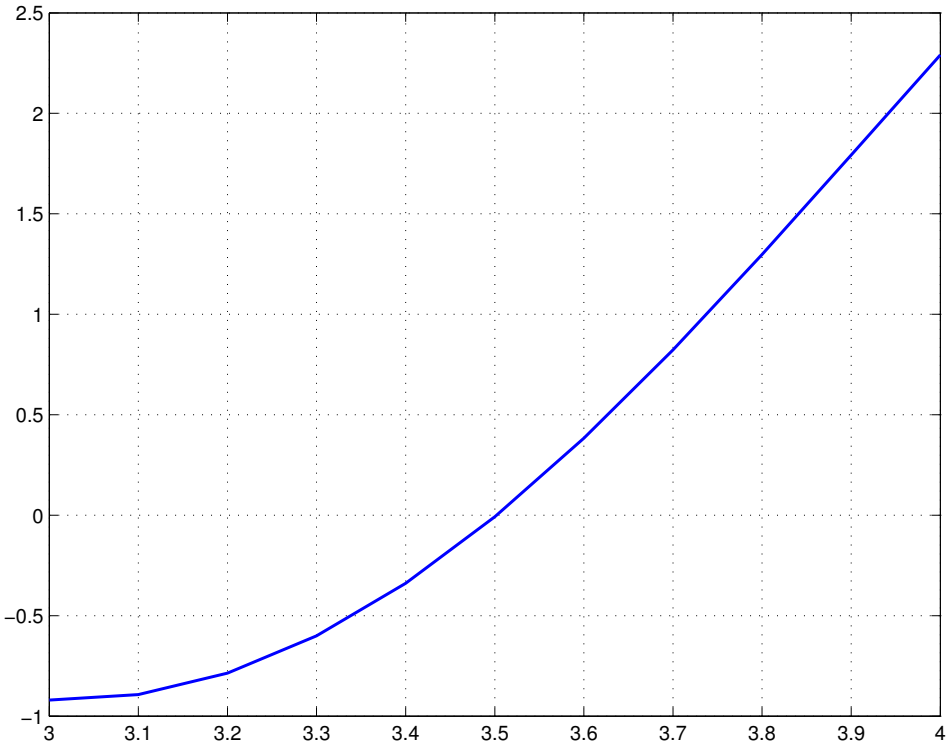
anonymní funkce str. 24

Ověříme předpoklady. Pro ověření, zda hodnoty první derivace nemění znaménko v intervalu $\langle 3,4\rangle$, vytvoříme body x z tohoto intervalu, krok volíme 0.1.

```
>> x=a:0.1:b
x =
  Columns 1 through 6
    3.0000    3.1000    3.2000    3.3000    3.4000    3.5000
  Columns 7 through 11
    3.6000    3.7000    3.8000    3.9000    4.0000
>> df(x)
ans =
  Columns 1 through 6
   -0.1177    0.6676    1.4662    2.2462    2.9765    3.6279
  Columns 7 through 11
    4.1747    4.5948    4.8717    4.9942    4.9574
```

dvojtečková konvence str. 12

Vidíme, že hodnoty první derivace se mění na $\langle 3,4\rangle$ a předpoklady tedy nejsou splněny. Je potřeba nalézt menší interval, na kterém leží kořen. Na tomto menším intervalu opět ověříme předpoklady. Vykreslíme graf na $\langle 3,4\rangle$.



A vidíme, že kořen leží na $\langle 3.4, 3.6\rangle$. Na tomto intervalu opět ověříme předpoklady. Zadáme meze nově nalezeného intervalu.

```
>> a=3.4;
>> b=3.6;
```

Pro hodnoty z intervalu $\langle 3.4, 3.6\rangle$ vyčíslíme první a druhou derivaci.

```
>> x=a:0.01:b;
>> df(x)
ans =
  Columns 1 through 6
    2.9765    3.0456    3.1139    3.1814    3.2480    3.3138
  Columns 7 through 12
    3.3785    3.4424    3.5053    3.5671    3.6279    3.6877
  Columns 13 through 18
    3.7464    3.8040    3.8605    3.9159    3.9701    4.0230
  Columns 19 through 21
    4.0748    4.1254    4.1747
```

dvojtečková konvence str. 12

Vidíme, že hodnoty první derivace se nemění na celém $\langle 3.4, 3.6\rangle$.

```
>> ddf(x)
ans =
  Columns 1 through 6
    6.9552    6.8747    6.7915    6.7056    6.6170    6.5258
  Columns 7 through 12
    6.4320    6.3355    6.2366    6.1351    6.0312    5.9249
```

```
  Columns 13 through 18
    5.8162    5.7052    5.5919    5.4764    5.3587    5.2388
  Columns 19 through 21
    5.1168    4.9928    4.8668
```

Vidíme, že hodnoty druhé derivace se nemění na celém $\langle 3.4, 3.6\rangle$.
Dále ověříme platnost jednoduché podmínky $f(a) \cdot f(b) < 0$, která zajistí existenci kořene na daném intervalu.

```
>> f(a)*f(b)
ans =
   -0.1299
```

Předpoklad je splněn, neboť hodnota je záporná.
Nakonec ověříme platnost následujících podmínek $\left|\frac{f(a)}{f'(a)}\right| < b-a$ a $\left|\frac{f(b)}{f'(b)}\right| < b-a$.

```
>> abs(f(a)/df(a))
ans =
    0.1138
>> abs(f(b)/df(b))
ans =
    0.0918
```

Obě hodnoty jsou menší než $b - a = 3.6 - 3.4 = 0.2$.
Na intervalu $\langle 3.4, 3.6\rangle$ jsou všechny předpoklady splněny a je tedy zajištěna konvergence aproximací ke kořeni rovnice.
Pro výpočet s přesností 10^{-8} je potřeba znát hodnoty na více desetinných míst, a proto nastavíme delší výpis příkazem `format long`. Zadáme nejprve počáteční aproximaci, kterou můžeme volit jako hodnotu z daného intervalu $\langle 3.4, 3.6\rangle$. Zvolíme a.

```
>> format long
>> x0=a
x0 =
   3.40000000000000
```

příkaz format str. 8

Vypočítáme další aproximaci a chybu. Pokud je chyba větší než ε, výpočet pokračuje dál.

$$x^1 = x^0 - \frac{f(x^0)}{f'(x^0)} = 3.4 - \frac{f(3.4)}{f'(3.4)} = 3.4 - \frac{3.4 - 4\cos^2(3.4)}{1 + 8\cos(3.4)\sin(3.4)} = 3.5138$$

```
>> x1=x0-f(x0)/df(x0)
x1 =
   3.51382505776211
>> abs(x0-x1)
ans =
   0.11382505776211
```

k	x^k	$\lvert x^{k-1} - x^k \rvert$
0	3.4	—
1	3.51382505776211	$0.11382505776211 > 10^{-8}$

Vypočítáme další aproximaci a chybu. Pokud je chyba větší než ε, výpočet pokračuje dál.

```
>> x2=x1-f(x1)/df(x1)
x2 =
   3.50225628403900
>> abs(x1-x2)
ans =
   0.01156877372312
>> x3=x2-f(x2)/df(x2)
x3 =
   3.50214740099497
>> abs(x2-x3)
ans =
    1.088830440254540e-004
>> x4=x3-f(x3)/df(x3)
x4 =
   3.50214739121355
>> abs(x3-x4)
ans =
    9.781422338761558e-009
```

Data zapisujeme do tabulky.

k	x^k	$\lvert x^{k-1} - x^k \rvert$
0	3.4	—
1	3.51382505776211	$0.11382505776211 > 10^{-8}$
2	3.50225628403900	$0.01156877372312 > 10^{-8}$
3	3.50214740099497	$1.088830440254540e-004 > 10^{-8}$
4	3.50214739121355	$9.781422338761558e-009 \leq 10^{-8}$

Chyba je menší nebo rovna než $\varepsilon = 10^{-8}$, proto výpočet ukončíme. Hodnotu x^4 zaokrouhlíme na osm desetinných míst a zapíšeme následujícím způsobem.

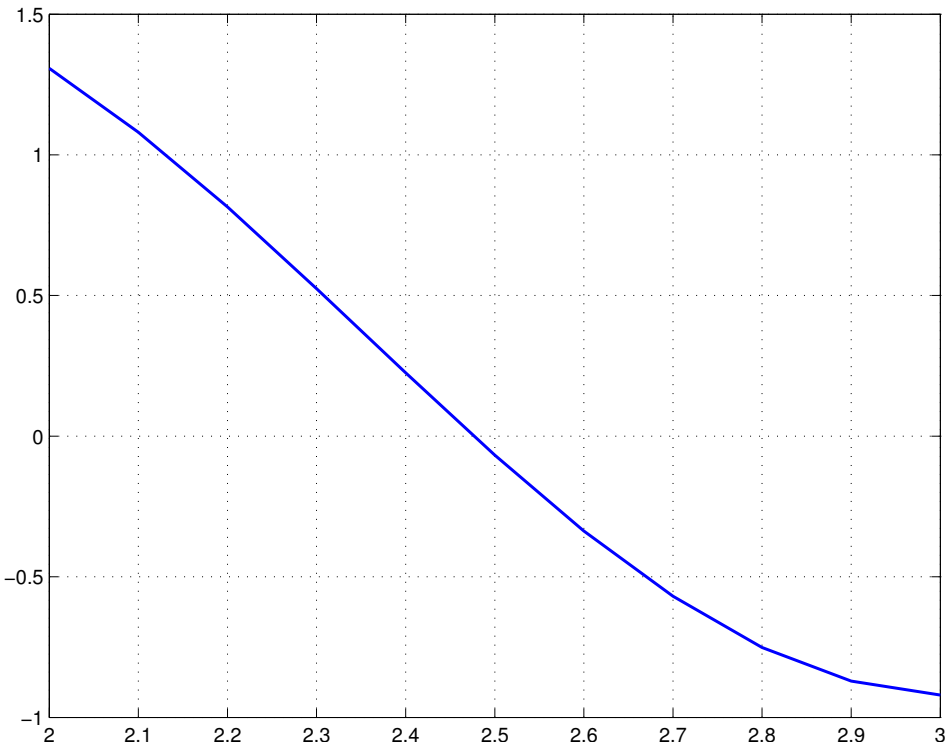
Kořen rovnice $x - 4\cos^2(x) = 0$ je 3.50214739 ± 10^{-8}.

Ostatní kořeny spočítáme obdobným způsobem.

Implementace B

Najdeme vhodný interval, ve kterém leží další kořen a na tomto intervalu ověříme předpoklady.

```
>> fplot(f,[2,3])
```



Z grafu vidíme, že kořen rovnice (průsečík s osou x) leží v intervalu $\langle 2.4, 2.6\rangle$. Ověříme předpoklady Newtonovy metody.

```
>> a=2.4; b=2.6;
>> x=a:0.05:b
x =
    2.4000    2.4500    2.5000    2.5500    2.6000
>>  df(x)
ans =
   -2.9847   -2.9298   -2.8357   -2.7033   -2.5338
>>  ddf(x)
ans =
    0.7000    1.4921    2.2693    3.0238    3.7481
>>  f(a)*f(b)
ans =
   -0.2293
>>  abs(f(a)/df(a))
ans =
    0.1674
>>  abs(f(b)/df(b))
ans =
    0.1811
```

dvojtečková konvence str. 12

Nastavíme dlouhý výpis a zvolíme počáteční aproximaci.

```
>> format long
>> x=a
x =
   2.40000000000000
```

příkaz format str. 8

Vypočítáme další aproximaci, kterou uložíme do proměnné xnovy. Spočítáme chybu jako absolutní hodnotu rozdílu mezi x a xnovy.

```
>> xnovy=x-f(x)/df(x)
xnovy =
   2.47538619175203
>> chyba=abs(x-xnovy)
chyba =
   0.07538619175203
```

Pokud je chyba větší než zadaná přesnost budeme opakovat následující tři příkazy, tj. uložení předchozí aproximace do proměnné x, výpočet nové aproximace a chyby.

```
>> x=xnovy;
>> xnovy=x-f(x)/df(x)
xnovy =
   2.47646766323749
>> chyba=abs(x-xnovy)
chyba =
   0.00108147148546
>> x=xnovy;
>> xnovy=x-f(x)/df(x)
xnovy =
   2.47646804730806
>> chyba=abs(x-xnovy)
chyba =
    3.840705731228411e-007
>> x=xnovy;
>> xnovy=x-f(x)/df(x)
>> xnovy =
   2.47646804730811
>> chyba=abs(x-xnovy)
chyba =
    4.884981308350689e-014
```

Výpočet ukončíme a poslední aproximaci zaokrouhlíme na osm desetinných míst.

Kořen rovnice $x - 4\cos^2(x) = 0$ je 2.47646805 ± 10^{-8}.

Příkaz `fzero`

Matlab má vestavěnou funkci pro hledání kořene rovnice `fzero`. Vstupem tohoto příkazu je předpis funkce nebo její název a hodnota poblíž hledeného kořene.

```
>> f=@(x)x-4*cos(x).^2
f =
    @(x)x-4*cos(x).^2
>> fzero(f,1)
ans =
    1.0367
>> fzero(f,2)
ans =
    2.4765
>> fzero(f,4)
ans =
    3.5021
```

příkaz `fzero`, Matlab help

KAPITOLA

7

SOUSTAVY LINEÁRNÍCH ROVNIC: PŘÍMÉ METODY

Uvažujme soustavu lineárních rovnic

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

s regulární čtvercovou maticí řád n a předpokládejme, že $\boldsymbol{P}$, $\boldsymbol{L}$ a $\boldsymbol{U}$ jsou matice, které tvoří LU-rozklad $\boldsymbol{PA} = \boldsymbol{LU}$. To znamená, že $\boldsymbol{P}$ je permutační matice, $\boldsymbol{L}$ je dolní trojúhelníková matice a $\boldsymbol{U}$ je horní trojúhelníková matice. Dosazením pak získáme následující ekvivalence:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \Leftrightarrow \quad \boldsymbol{PA}\boldsymbol{x} = \boldsymbol{P}\boldsymbol{b} \quad \Leftrightarrow \quad \boldsymbol{LU}\boldsymbol{x} = \boldsymbol{P}\boldsymbol{b}.$$

Předpokládejme, že má soustava řešení $\boldsymbol{x}$. Zavedeme-li proměnnou $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{x}$, převedeme tím zadanou soustavu na rovnici

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{b}.$$

Tu vyřešíme a nalezené $\boldsymbol{y}$ dosadíme zpět do rovnice $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{x}$, z níž vypočteme $\boldsymbol{x}$.

Příklad 3: Pomocí LU-rozkladu $\boldsymbol{PA} = \boldsymbol{LU}$ řešte soustavu

$$\begin{pmatrix} -1 & 0 & 3 \\ 3 & -3 & -6 \\ 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ -66 \\ 16 \end{pmatrix}.$$

Výpočet LU-rozkladu s permutační maticí pomocí Gaussovy eliminační metody s výběrem hlavního prvku můžeme spočítat buď ručně, jak jsme to ukázali v Kapitole . Nyní si ukážeme, jak toho dosáhnout pomocí matematického softwaru MATLAB. Bude vhodné pracovat s zlomky, takže nejprve příslušným způsobem nastavíme příkaz `format`. Pak vytvoříme proměnnou `A`, do níž uložíme zadanou matici soustavy.

```
>> format rat % nastaveni zlomku na vystup
>> A=[-1 0 3;3 -3 -6;1 1 2];
```

příkaz format str. 8

Následně využijeme příkazu `lu`, který LU-rozklad $\boldsymbol{PA} = \boldsymbol{LU}$ spočte, matice $\boldsymbol{P}$, $\boldsymbol{L}$ a $\boldsymbol{U}$ uložíme do příslušných proměnných a necháme je zobrazit.

```
>> [L U P]=lu(A)
L =
        1             0             0
        1/3           1             0
       -1/3          -1/2           1

U =

        3            -3            -6
        0             2             4
        0             0             3

P =
        0             1             0
        0             0             1
        1             0             0
```

příkaz `lu`, MATLAB help

Výpočtem můžeme ověřit, že skutečně $\boldsymbol{PA} = \boldsymbol{LU}$, neboli $\boldsymbol{A} = \boldsymbol{P}^\top \boldsymbol{LU}$.

```
>> P'*L*U
ans =
       -1             0             3
        3            -3            -6
        1             1             2
```

Dále definujeme (sloupcový) vektor pravé strany soustavy $\boldsymbol{b}$.

```
>> b=[24;-66;16];
```

Řešení $\boldsymbol{x}$ nalezneme ve dvou krocích. Nejprve vyřešíme soustavu $\boldsymbol{Ly} = \boldsymbol{Pb}$ a jakmi-le budeme znát vektor $\boldsymbol{y}$, vyřešíme soustavu $\boldsymbol{Ux} = \boldsymbol{y}$. Pro řešení soustav lineárních rovnic nám MATLAB nabízí řadu prostředků. Každou ze soustav proto vyřešíme jiným způsobem a čtenář sám se může rozhodnout, který je mu bližší.

```
>> y=inv(L)*P*b % nasobeni inverzni matici k L zleva
y =
     -66
      38
      21

>> x=U\y % taktez nasobeni inverzni matici k U zleva
x =
      -3
       5
       7
```

řešení soustavy lineárních rovnic str. 19

Řešením úlohy je proto trojice:

$$x_1 = -3, \quad x_2 = 5, \quad x_3 = 7.$$

Příklad 4: Nalezněte LU-rozklad $\boldsymbol{A} = \boldsymbol{LU}$ matice

$$\begin{pmatrix} -1 & 0 & 3 \\ 3 & -3 & -6 \\ 1 & 1 & 2 \end{pmatrix}$$

pomocí Gaussovy eliminační metody bez výběru hlavního prvku.

Ačkoli je při ručním počítání LU-rozklad bez permutační matice jednodušší úlohou než LU-rozklad s permutační maticí, při spolupráci s MATLABem se situace poněkud otáčí. Příkaz `lu` totiž automaticky provádí výběr hlavního prvku, s jeho pomocí tudíž vždy provádíme rozklad $\boldsymbol{PA} = \boldsymbol{LU}$ namísto zde požadovaného $\boldsymbol{A} = \boldsymbol{LU}$. Matice $\boldsymbol{L}$, $\boldsymbol{U}$ jsou obecně v obou rozkladech různé.
Je samozřejmě možné si v MATLABu algoritmus pro LU-rozklad bez výběru hlavního prvku naprogramovat. S využitím určitého triku však můžete pro dosažení stejného cíle použít i standardních prostředků MATLABu. Zadanou matici $\boldsymbol{A}$ uložíme ve formě tzv. *řídké matice* (anglicky *sparse matrix*; jde o matice obsahující velké množství nulových položek), protože příkaz `lu` pro takové matice dovoluje přesněji specifikovat metodu rozkladu.

```
>> format rat
>> A=sparse([-1 0 3;3 -3 -6;1 1 2]) % definujeme matici A jako
   ridkou
A =
```

```
   (1,1)          -1
   (2,1)           3
   (3,1)           1
   (2,2)          -3
   (3,2)           1
   (1,3)           3
   (2,3)          -6
   (3,3)           2
```

příkaz `sparse`, MATLAB help

Z výstupu MATLABu si můžete všimnou, že řídké matice jsou v MATLABu reprezentovány nestandardním způsobem. Ačkoli zadaná matice $\boldsymbol{A}$ řídká není, my ji jako takovou budeme reprezentovat. Příkaz, který z matice reprezentované jako řídká vyrobí opět matici reprezentovanou standardním způsobem, je `full`.

```
>> [L,U]=lu(sparse(A),0); % parametr 0 zakazuje vyber hlavniho
   prvku
>> L=full(L),U=full(U) % matice L a U prevedeme na standardni
   matice
L =
        1             0             0
       -3             1             0
       -1          -1/3             1
U =

       -1             0             3
        0            -3             3
        0             0             6
```

příkaz `full`, MATLAB help

Všimněte si, že jsme v příkazu `lu` použili druhý parametr 0, což je možné pouze u řídkých matic. Tento parametr ovlivňuje výběr hlavního prvku při Gaussově eliminaci. Pokud je roven 0, k výběru hlavního prvku nedochází.
Matice z LU-rozkladu jsou tedy

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1/3 & 1 \end{pmatrix}, \quad \boldsymbol{P} = \begin{pmatrix} -1 & 0 & 3 \\ 0 & -3 & 3 \\ 0 & 0 & 6 \end{pmatrix}.$$

Povšimněte si, že jde o zcela jiné matice, než jaké jsme obdrželi v předchozí úloze, ačkoli matice $\boldsymbol{A}$ je totožná.

KAPITOLA

8

SOUSTAVY LINEÁRNÍCH ROVNIC: ITERAČNÍ METODY

Uvažujme soustavu lineárních rovnic $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ s regulární čtvercovou maticí $\boldsymbol{A} = (a_{ij})$, vektorem pravé strany $\boldsymbol{b} = (b_i)$ a vektorem neznámých $\boldsymbol{x} = (x_i)$. Soustavu převedeme na ekvivalentní soustavu v *iteračním tvaru*

$$\boldsymbol{x} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{d},$$

kde $\boldsymbol{C}$ je *iterační matice* a $\boldsymbol{d}$ je sloupcový vektor.
Nechť $\boldsymbol{x}^{(0)}$ je daná počáteční aproximace. Výpočet provádíme podle rekurentního vzorce

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

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

8.1. Jacobiho metoda

Přepokládejme, že je matice $\boldsymbol{A}$ soustavy lineárních rovnic *ostře diagonálně dominantní*. Soustavu $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ můžeme zapsat jako

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \quad i = 1, \ldots, n.$$

Z i-té rovnice vyjádříme i-tou neznámou a *Jacobiho metoda* je pak určena rekurentními vzorci

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

pro $k = 0, 1, 2, \ldots$.
Díky podmínce ostré diagonální dominance matice $\boldsymbol{A}$ posloupnost vektorů $\boldsymbol{x}^{(k+1)}$ konverguje k řešení.

Příklad 5: 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 $\varepsilon = 10^{-2}$.

Zadanou soustavu napíšeme ve tvaru rozšířené matice a několika ekvivalentními úpravami dosáhneme toho, že bude matice soustavy ostře diagonálně dominantní. To je totiž podmínka postačující pro konvergenci Jacobiho iterační metody.

$$\left(\begin{array}{ccc|c} -1 & -6 & 7 & 16 \\ 4 & -5 & 3 & 8 \\ 3 & -1 & 1 & 11 \end{array}\right) \overset{\leftarrow\,\rfloor}{\leftarrow\,\rfloor} \sim \left(\begin{array}{ccc|c} 3 & -1 & 1 & 11 \\ 4 & -5 & 3 & 8 \\ -1 & -6 & 7 & 16 \end{array}\right) \overset{\rfloor\,-1}{\leftarrow\,\rfloor_{+}}$$

$$\sim \left(\begin{array}{ccc|c} 3 & -1 & 1 & 11 \\ 1 & -4 & 2 & -3 \\ -1 & -6 & 7 & 16 \end{array}\right) \overset{\rfloor\,-1}{\leftarrow\,\rfloor_{+}} \sim \left(\begin{array}{ccc|c} 3 & -1 & 1 & 11 \\ 1 & -4 & 2 & -3 \\ -2 & -2 & 5 & 19 \end{array}\right)$$

Matice soustavy

$$\begin{pmatrix} 3 & -1 & 1 \\ 1 & -4 & 2 \\ -2 & -2 & 5 \end{pmatrix}$$

je ostře diagonálně dominantní, neboť $3 > 1+1$, $4 > 1+2$ a $5 > 2+2$.
Upravenou soustavu převedeme na iterační tvar

$$\begin{aligned} x_1 &= \tfrac{1}{3}(11 + x_2 - x_3), \\ x_2 &= -\tfrac{1}{4}(-3 - x_1 - 2x_3), \\ x_3 &= \tfrac{1}{5}(19 + 2x_1 + 2x_2), \end{aligned}$$

z nichž doplněním indexů snadno vytvoříme rekurentní vzorce pro Jacobiho metodu:

$$\begin{aligned} x_1^{(k+1)} &= \tfrac{1}{3}\left(11 + x_2^{(k)} - x_3^{(k)}\right), \\ x_2^{(k+1)} &= -\tfrac{1}{4}\left(-3 - x_1^{(k)} - 2x_3^{(k)}\right), \\ x_3^{(k+1)} &= \tfrac{1}{5}\left(19 + 2x_1^{(k)} + 2x_2^{(k)}\right). \end{aligned}$$

Následně zvolíme (libovolnou) počáteční aproximaci, například

$$\boldsymbol{x^{(0)}} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0,0,0).$$

Dosazením do rekurentních vzorců spočteme následníka inicializačního vektoru

$$\begin{aligned} x_1^{(1)} &= \tfrac{1}{3}\left(11 + x_2^{(0)} - x_3^{(0)}\right) = \tfrac{11}{3}, \\ x_2^{(1)} &= -\tfrac{1}{4}\left(-3 - x_1^{(0)} - 2x_3^{(0)}\right) = \tfrac{3}{4}, \\ x_3^{(1)} &= \tfrac{1}{5}\left(19 + 2x_1^{(0)} + 2x_2^{(0)}\right) = \tfrac{19}{5}. \end{aligned}$$

Chyba po prvním kroku má velikost

$$\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|_R = \max\{|\tfrac{11}{3} - 0|, |\tfrac{3}{4} - 0|, |\tfrac{19}{5} - 0|\} = \tfrac{19}{5} = 3.8 > 10^{-2}.$$

Protože jsme nedosáhli požadované přesnosti, spočteme další krok.

$$\begin{aligned} x_1^{(2)} &= \tfrac{1}{3}\left(11 + x_2^{(1)} - x_3^{(1)}\right) = \tfrac{1}{3}\left(11 + \tfrac{3}{4} - \tfrac{19}{5}\right) = \tfrac{53}{20}, \\ x_2^{(2)} &= -\tfrac{1}{4}\left(-3 - x_1^{(1)} - 2x_3^{(1)}\right) = -\tfrac{1}{4}\left(-3 - \tfrac{11}{3} - 2\cdot\tfrac{19}{5}\right) = \tfrac{107}{30}, \\ x_3^{(2)} &= \tfrac{1}{5}\left(19 + 2x_1^{(1)} + 2x_2^{(1)}\right) = \tfrac{1}{5}\left(19 + 2\cdot\tfrac{11}{3} + 2\cdot\tfrac{3}{4}\right) = \tfrac{167}{30}. \end{aligned}$$

Chyba má po druhém kroku velikost

$$\|\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}\|_R = \max\{|\tfrac{53}{20} - \tfrac{11}{3}|, |\tfrac{107}{30} - \tfrac{3}{4}|, |\tfrac{167}{30} - \tfrac{19}{5}|\} = \tfrac{169}{60} \approx 2.8167 > 10^{-2}.$$

Tento postup opakujeme, dokud je chyba větší než 10^{-2}.
MATLAB nám může ruční výpočet výrazně usnadnit. Jacobiho metodu můžeme implementovat mnoha způsoby. Některé základní implementace si ukážeme.

Implementace A
Definujeme inicializační vektor $\boldsymbol{x}^{(0)} = (0, 0, 0)$.

```
>> x = [0 0 0];
```

Přímým přepisem rekurentních vzorců do MATLABu spočteme následníka a necháme si zobrazit jeho prvky.

```
>> xnovy(1)=1/3*(11+x(2)-x(3));
>> xnovy(2)=-1/4*(-3-x(1)-2*x(3));
>> xnovy(3)=1/5*(19+2*x(2)+2*x(3)) % chybejici strednik na konci
    prikazu zpusobi, ze MATLAB vypise hodnoty promenne xnovy
xnovy =
    3.6667    0.7500    3.8000
```

Dále spočteme velikost chyby.

```
>> max(abs(xnovy-x))
ans =
    3.8000
```

Díky předchozímu ručnímu výpočtu pro nás samozřejmě není žádným překvapením, že je velikost chyby větší než požadovaná přesnost 10^{-2} a je tedy třeba ve výpočtu pokračovat. Další krok spočívá ve vyvolání stejného kódu v MATLABu. Předtím však je nutné uložit do proměnné x nově spočtenou hodnotu xnovy.

```
>> x = xnovy;
```

Ve druhém kroku tedy všechny výše uvedené příkazy zopakujeme. Pro praktický výpočet je výhodné napsat všechny na jeden řádek.

```
>> xnovy(1)=1/3*(11+x(2)-x(3));xnovy(2)=-1/4*(-3-x(1)-2*x(3));
   xnovy(3)=1/5*(19+2*x(2)+2*x(3)),max(abs(xnovy-x)),x=xnovy;
xnovy =

    2.6500    3.5667    5.6200
ans =

    2.8167
```

Spočetli jsme tedy, že $\boldsymbol{x}^{(2)} = (2.6500, 3.5667, 5.6200)$ a velikost chyby je $\|\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}\|_R = 2.8167 > 10^{-2}$. Uvedené příkazy budeme v MATLABu vyvolávat tak dlouho, dokud nedosáhneme požadované přesnosti. Spočtené hodnoty jsou uvedeny v tabulce.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_R$
0	0	0	0	—
1	3.6667	0.7500	3.8000	3.8000
2	2.6500	3.5667	5.5667	2.8167
3	3.0000	4.1958	6.2867	0.7200
4	2.9697	4.6433	6.6783	0.4475
5	2.9883	4.8316	6.8452	0.1883
6	2.9955	4.9316	6.9280	0.0881
7	2.9972	4.9629	6.9661	0.0432
8	2.9989	4.9823	6.9840	0.0195
9	2.9994	4.9918	6.9925	0.0094

Hodnoty zaokrouhlíme na dvě desetinná místa a řešení zapíšeme jako

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

Implementace B

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

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

Tyto objekty nyní vytvoříme v MATLABu. Znovu také definujeme inicializační vektor $\boldsymbol{x}^{(0)}$.

```
>> 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];
```

Snadno si rozmyslíme, že každý z rekurentních vzorců Jacobiho metody můžeme zapsat jako součet prvku vektoru $\boldsymbol{d}$ a skalárního součinu řádku matice $\boldsymbol{C}$ a vektoru $\boldsymbol{x}^{(k)}$. Zapsáno kódem:

```
>> xnovy(1)=C(1,:)*x'+d(1);
>> xnovy(2)=C(2,:)*x'+d(2);
>> xnovy(3)=C(3,:)*x'+d(3)
xnovy =
    3.6667    0.7500    3.8000
```

Všimněte si, že počítáme s transponovaným vektorem `x'`. Důvodem je to, že součin řádkového vektoru a slopcového vektoru je roven právě skalárnímu součinu těchto vektorů. Chybu spočteme stejným způsobem jako v Implementaci A. První krok uzavřeme uložením nově spočteného vektoru nacházejícího se v proměnné `xnovy` do proměnné `x`.

```
>> max(abs(xnovy-x))
ans =
    3.8000

>> x=xnovy;
```

Nyní bychom se vrátili k rekurentním vzorcům a spočetli opět hodnotu vektoru `novy` a chyby. Celý postup bychom opakovali, dokud by chyba nedosáhla požadované hodnoty. Získané hodnoty jsou uvedeny v tabulce u Implementace A.

Implementace C

V poslední představené implementaci opět využijeme maticového zápisu rekurentních vzorců Jacobiho metody. 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] % pro potreby teto implementace je vyhodne pracovat
   se sloupcovymi vektory
x =
     0
     0
     0
```

Vektor `xnovy` tentokrát nebudeme vytvářet po složkách, ale najednou. Výpočet nového vektoru, chyby a uložení nového vektoru do proměnné `xnovy` provádíme zde:

```
>> xnovy=C*x+d
xnovy =
    3.6667
    0.7500
    3.8000

>> max(abs(xnovy-x))
ans =
    3.8000

>> x=xnovy;
```

Předchozí příkazy bychom opakovali, dokud by chyba byla větší než 10^{-2}. Hodnoty, které bychom získali, jsou uvedeny v tabulce u Implementace A.

8.2. Gaussova-Seidelova metoda

Přepokládejme, že je matice $\boldsymbol{A}$ soustavy lineárních rovnic *ostře diagonálně dominantní*. Soustavu $\boldsymbol{Ax} = \boldsymbol{b}$ můžeme zapsat jako

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \quad i = 1, \ldots, n.$$

Z i-té rovnice vyjádříme i-tou neznámou a *Jacobiho metoda* je pak určena rekurentními vzorci

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

pro $k = 0, 1, 2, \ldots$.
Díky podmínce ostré diagonální dominance matice $\boldsymbol{A}$ posloupnost vektorů $\boldsymbol{x}^{(k+1)}$ konverguje k řešení.

Příklad 6: Vyřešte soustavu lineárních rovnic

$$\begin{array}{rrrrrr} -7x_1 & +x_2 & +x_3 & +3x_4 & -x_5 = & 14, \\ x_1 & +8x_2 & -2x_3 & -x_4 & +3x_5 = & -5, \\ 2x_1 & +3x_2 & -9x_3 & -2x_4 & +x_5 = & -7, \\ -x_1 & -x_2 & +2x_3 & +8x_4 & -2x_5 = & 7, \\ -2x_1 & +3x_2 & -2x_3 & +x_4 & -10x_5 = & -18, \end{array}$$

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

Matice soustavy

$$\begin{pmatrix} -7 & 1 & 1 & 3 & -1 \\ 1 & 8 & -2 & -1 & 3 \\ 2 & 3 & -9 & -2 & 1 \\ -1 & -1 & 2 & 8 & -2 \\ -2 & 3 & -2 & 1 & -10 \end{pmatrix}$$

je ostře diagonálně dominantní, protože $7 > 1+1+3+1$, $8 > 1+2+1+3$, $9 > 2+3+2+1$, $8 > 1+1+2+2$ a $10 > 2+3+2+1$. Konvergence Gaussovy-Seidelovy metody je proto zaručena. Soustavu nyní převedeme na iterační tvar.

$$
\begin{aligned}
x_1 &= -\frac{1}{7}\left(14 - x_2 - x_3 - 3x_4 + x_5\right), \\
x_2 &= \frac{1}{8}\left(-5 - x_1 + 2x_3 + x_4 - 3x_5\right), \\
x_3 &= -\frac{1}{9}\left(-7 - 2x_1 - 3x_2 + 2x_4 - x_5\right), \\
x_4 &= \frac{1}{8}\left(7 + x_1 + x_2 - 2x_3 + 2x_5\right), \\
x_5 &= -\frac{1}{10}\left(-18 + 2x_1 - 3x_2 + 2x_3 - x_4\right).
\end{aligned}
$$

Doplněním indexů získáme rekurentní vzorce pro Gaussovu-Seidelovu metodu. Mějte na paměti, že na rozdíl od Jacobiho metody doplňujeme na pravé straně rovnosti index (k) nad diagonálu a index $(k+1)$ pod diagonálu.

$$
\begin{aligned}
x_1^{(k+1)} &= -\frac{1}{7}\left(14 - x_2^{(k)} - x_3^{(k)} - 3x_4^{(k)} + x_5^{(k)}\right), \\
x_2^{(k+1)} &= \frac{1}{8}\left(-5 - x_1^{(k+1)} + 2x_3^{(k)} + x_4^{(k)} - 3x_5^{(k)}\right), \\
x_3^{(k+1)} &= -\frac{1}{9}\left(-7 - 2x_1^{(k+1)} - 3x_2^{(k+1)} + 2x_4^{(k)} - x_5^{(k)}\right), \\
x_4^{(k+1)} &= \frac{1}{8}\left(7 + x_1^{(k+1)} + x_2^{(k+1)} - 2x_3^{(k+1)} + 2x_5^{(k)}\right), \\
x_5^{(k+1)} &= -\frac{1}{10}\left(-18 + 2x_1^{(k+1)} - 3x_2^{(k+1)} + 2x_3^{(k+1)} - x_4^{(k+1)}\right).
\end{aligned}
$$

Inicializační vektor zvolíme jako $\boldsymbol{x}^{(0)} = (0,0,0,0,0)$. Následný vektor spočteme dosazením do rekurentních vzorců.

$$
\begin{aligned}
x_1^{(1)} &= -\frac{1}{7}\left(14 - x_2^{(0)} - x_3^{(0)} - 3x_4^{(0)} + x_5^{(0)}\right) = -2, \\
x_2^{(1)} &= \frac{1}{8}\left(-5 + 2\right) = -\frac{3}{8}, \\
x_3^{(1)} &= -\frac{1}{9}\left(-7 + 2\cdot 2 + 3\cdot\frac{3}{8}\right) = \frac{5}{24}, \\
x_4^{(1)} &= \frac{1}{8}\left(7 - 2 - \frac{3}{8} - 2\cdot\frac{5}{24}\right) = \frac{101}{192}, \\
x_5^{(1)} &= -\frac{1}{10}\left(-18 - 2\cdot 2 + 3\cdot\frac{3}{8} + 2\cdot\frac{5}{24} - \frac{101}{192}\right) = \frac{1343}{640}.
\end{aligned}
$$

Velikost chyby spočteme jako

$$
\begin{aligned}
\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}\|_R &= \max\left\{\left|-2-0\right|, \left|-\tfrac{3}{8}-0\right|, \left|\tfrac{5}{24}-0\right|, \left|\tfrac{101}{192}-0\right|, \left|\tfrac{1343}{640}-0\right|\right\} = \\
&= \tfrac{1343}{640} \approx 2.0984 > 10^{-2}.
\end{aligned}
$$

Ve výpočtu posloupnosti bychom takto pokračovali, dokud je chyba větší než 10^{-2}. Obdrželi bychom hodnoty uvedené v tabulce.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$	$x_5^{(k)}$	$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_R$
0	0	0	0	0	0	—
1	−2.0000	−0.3750	0.2083	0.5260	2.0984	2.0984
2	−2.0981	−1.0318	0.0839	0.9874	1.9921	0.6568
3	−1.9968	−0.9780	0.0099	0.9987	2.0038	0.1013
4	−1.9966	−0.9995	0.0016	1.0010	1.9992	0.0215
5	−1.9991	−0.9993	0.0001	1.0000	2.0000	0.0026

Hodnoty zaokrouhlíme na dvě desetinná místa a řešení zapíšeme jako $x_1 = -2 \pm 10^{-2}$, $x_2 = -1 \pm 10^{-2}$, $x_3 = 0 \pm 10^{-2}$, $x_4 = 1 \pm 10^{-2}$, $x_5 = 2 \pm 10^{-2}$.

Implementace A

První způsob implementace v MATLABu, který si ukážeme, je velice přímočarý. Definujeme nejprve počáteční aproximaci.

```
>> x = [0 0 0 0 0];
```

Poté přepíšeme rekurentní vzorce.

```
>> xnovy(1)=-1/7*(14-x(2)-x(3)-3*x(4)+x(5));
>> xnovy(2)=1/8*(-5-xnovy(1)+2*x(3)+x(4)-3*x(5));
>> xnovy(3)=-1/9*(-7-2*xnovy(1)-3*xnovy(2)+2*x(4)-x(5));
>> xnovy(4)=1/8*(7+xnovy(1)+xnovy(2)-2*xnovy(3)+2*x(5));
>> xnovy(5)=-1/10*(-18+2*xnovy(1)-3*xnovy(2)+2*xnovy(3)-xnovy(4)
   )
xnovy =
   -2.0000    -0.3750     0.2083     0.5260     2.0984
```

Spočteme velikost chyby.

```
>> max(abs(xnovy-x))
ans =
    2.0984
```

Vidíme, že chyba je větší než požadovaná přesnost, proto je třeba ve výpočtu pokračovat. Než znovu vyvoláme kód obsahující rekurentní vzorce, uložíme nově spočtenou hodnotu do proměnné x.

```
>> x=xnovy;
```

Výpočtem vychom získaly hodnoty uvedené v tabulce výše.

Implementace B

Druhá implementace Gaussovy-Seidelovy metody je o něco sofistikovanější. Nejprve zapíšeme iterační tvar soustavy v maticovém tvaru.

$$\boldsymbol{x} = \begin{pmatrix} 0 & \frac{1}{7} & \frac{1}{7} & \frac{3}{7} & -\frac{1}{7} \\ -\frac{1}{8} & 0 & \frac{1}{4} & \frac{1}{8} & -\frac{3}{8} \\ \frac{2}{9} & \frac{1}{3} & 0 & -\frac{2}{9} & \frac{1}{9} \\ \frac{1}{8} & \frac{1}{8} & -\frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{1}{10} & 0 \end{pmatrix} \cdot \boldsymbol{x} + \begin{pmatrix} -2 \\ -\frac{5}{8} \\ \frac{7}{9} \\ \frac{7}{8} \\ \frac{9}{5} \end{pmatrix} = \boldsymbol{C} \cdot \boldsymbol{x} + \boldsymbol{d}.$$

Přirozeným způsobem jsme definovali matici $\boldsymbol{C}$ a vektor $\boldsymbol{d}$. Totéž nyní provedeme v MATLABu.

```
>> C=[0 1/7 1/7 3/7 -1/7
      -1/8 0 1/4 1/8 -3/8
      2/9 1/3 0 -2/9 1/9
      1/8 1/8 -1/4 0 1/4
      -1/5 3/10 -1/5 1/10 0];
>> d=[-2;-5/8;7/9;7/8;9/5];
```

Definujeme nejprve počáteční aproximaci $\boldsymbol{x}^{(0)} = (0,0,0,0,0)$.

```
>> xnovy=[0 0 0 0 0];
```

Z technických důvodů uložíme do proměnné `x` hodnotou proměnné `xnovy`.

```
>> x=xnovy;
```

Můžete si rozmyslet, že iterační rovnice Gaussovy-Seidelovy metody jsou (obdobně jako u Jacobiho metody) součtem prvku vektoru $\boldsymbol{d}$ a skalárního součinu řádku matice $\boldsymbol{C}$ a vektoru složeného z části z prvků vektoru $\boldsymbol{x}^{(k)}$ a $\boldsymbol{x}^{(k+1)}$. V MATLABu to můžeme zapsat následovně.

```
>> xnovy(1)=C(1,:)*xnovy'+d(1);
>> xnovy(2)=C(2,:)*xnovy'+d(2);
>> xnovy(3)=C(3,:)*xnovy'+d(3);
>> xnovy(4)=C(4,:)*xnovy'+d(4);
>> xnovy(5)=C(5,:)*xnovy'+d(5);
>> xnovy % vysledek po vypoctu nechame zobrazit
   -2.0000   -0.3750    0.2083    0.5260    2.0984
```

Protože pracujeme s poměrně velkým počtem neznámých, mohli bychom s výhodou použít cyklus `for`. Předchozí kód bychom pak nahradili tímto.

```
>> for i=1:5, xnovy(i)=C(i,:)*xnovy'+d(i);end
>> xnovy % vysledek po vypoctu nechame zobrazit
   -2.0000   -0.3750    0.2083    0.5260    2.0984
```

příkaz `for` str. 35

Standardním způsobem spočteme velikost chyby.

```
>> max(abs(xnovy-x))
ans =
    2.0984
```

Posledních několik příkazů (počínaje `x=xnovy`) opakujeme, dokud je chyba větší než 10^{-2}. Získané hodnoty opět odpovídají hodnotám v tabulce výše.

KAPITOLA

9

INTERPOLACE A APROXIMACE FUNKCÍ

Úloha interpolace

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

9.1. 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

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

Vyřešením této soustavy lineárních rovnic najdeme koeficienty hledaného interpolačního polynomu.

Příklad 7: 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 `xu` uzly x_i, do vektoru `yu` funkční hodnoty y_i.

```
>> xu=[0; 3; 4]
xu =
     0
     3
     4
>> yu=[2; 1; 5]
yu =
     2
     1
     5
```

Hledaný interpolační polynom pro $n = 2$ má tvar $p_2(x) = a_0 + a_1 x + a_2 x^2$. Koeficienty polynomu spočítáme jako řešení soustavy lineárních rovnic.

$$\begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 5 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 5 \end{pmatrix}$$

Sestavíme matici soustavu lineárních rovnic a tuto soustavu vyřešíme.

```
>> M=[ones(3,1) xu xu.^2]
M =
     1     0     0
     1     3     9
     1     4    16

>> a=M\yu
a =
    2.0000
   -3.5833
    1.0833
```

příkaz ones str. 16, řešení soustavy lineárních rovnic str. 19

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

```
>> format rat
>> a
a =
           2
      -43/12
       13/12

>> format short
>> p=@(x)a(1)+a(2)*x+a(3)*x.^2
p =
    @(x)a(1)+a(2)*x+a(3)*x.^2
```

příkaz format str. 8

Nalezený interpolační polynom je

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

Pro vykreslení grafu nalezeného polynomu $p_2(x)$ si vytvoříme body xg z intervalu $\langle x_0, x_2 \rangle$ a spočítáme hodnotu interpolačního polynomu $p_2(x) = a_0 + a_1 x + a_2 x^2$ v těchto bodech.

```
>> xg=xu(1):0.01:xu(end)
xg =
         0    0.0100    0.0200   .   .   .   3.9900    4.0000
>> yg=p(xg)
yg =
    2.0000    1.9643    1.9288   .   .   .   4.9493    5.0000
```

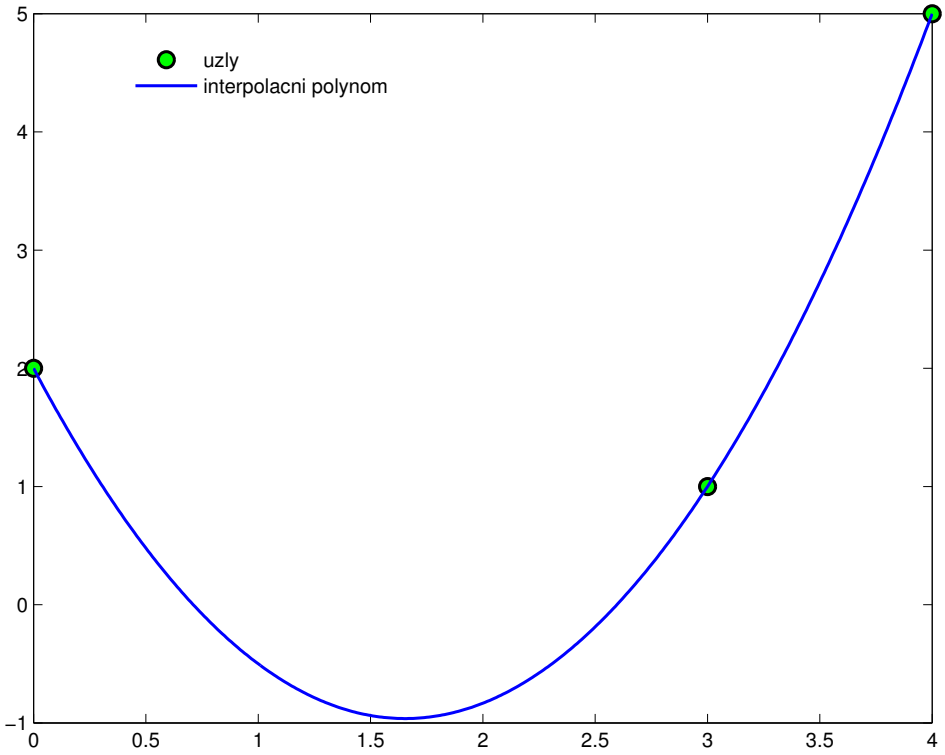
dvojtečková konvence str. 12

Vykreslíme graf nalezeného polynomu.

```
>> hold on
```

```
>> plot(xu,yu,'o')
>> plot(xg,yg)
>> legend('uzly','interpolacni polynom')
```

příkazy plot str. 26, legend str. 30, hold str. 29



9.2. 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 $\varphi_i(x)$ jsou polynomy Lagrangeovy báze dané úlohy

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

Příklad 8: Pro uzly x_i a funkční hodnoty f_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.

Sestavíme Lagrangeovy báze k jednotlivým uzlům.

$$\begin{aligned}
\varphi_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-3)(x-4)}{(0-3)(0-4)} = \frac{1}{12}(x-3)(x-4)\\
\varphi_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0)(x-4)}{(3-0)(3-4)} = -\frac{1}{3}x(x-4)\\
\varphi_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)(x-3)}{(4-0)(4-3)} = \frac{1}{4}x(x-3)
\end{aligned}$$

Dohromady pak dostáváme

$$p(x) = y_0\varphi_0(x) + y_1\varphi_1(x) + y_2\varphi_2(x) = 2\cdot\frac{1}{12}(x-3)(x-4) + 1\cdot\left(-\frac{1}{3}x(x-4)\right) + 5\cdot\frac{1}{4}x(x-3).$$

Hledaný polynom má tvar

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

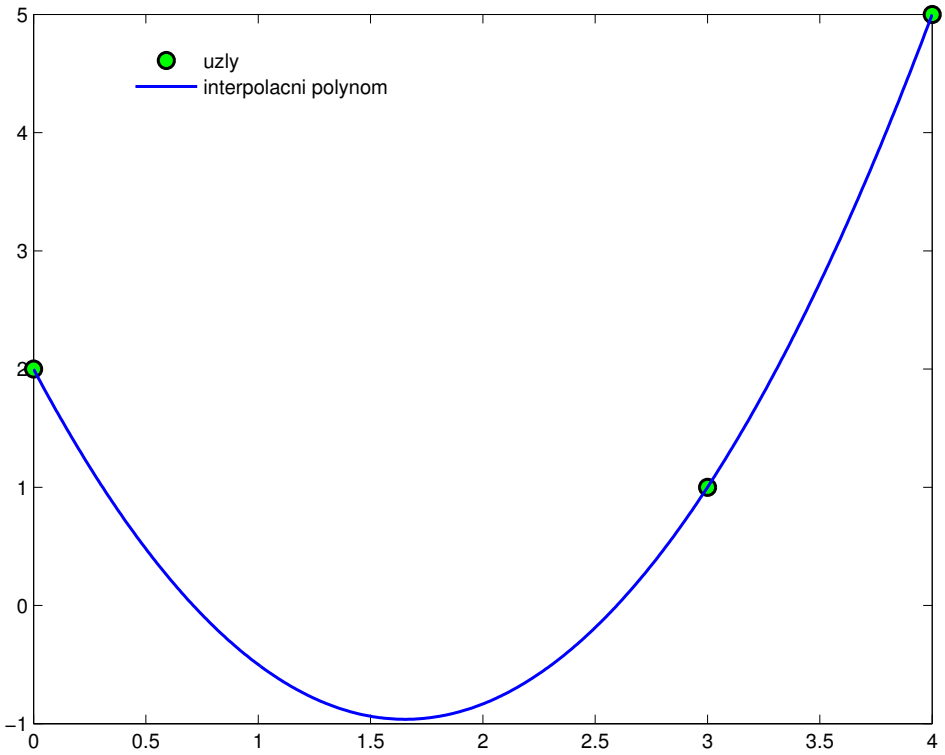
Zadáme uzly.

```
>> xu=[0; 3; 4]
xu =
     0
     3
     4
>> fu=[2; 1; 5]
fu =
     2
     1
     5
```

Pro vykreslení grafu vytvoříme body xg a v těchto bodech spočítáme hodnotu nalezeného interpolačního polynomu.

```
>> p=@(x)1/6*(x-3).*(x-4)-1/3*x.*(x-4)+5/4*x.*(x-3)
p =
    @(x)1/6*(x-3).*(x-4)-1/3*x.*(x-4)+5/4*x.*(x-3)
>> xg=0:0.01:4
xg =
         0    0.0100    0.0200   .   .   .   3.9900    4.0000
>>  yg=p(xg)
yg =
    2.0000    1.9643    1.9288   .   .   .   4.9493    5.0000
>> plot(xu,yu,'go',xg,yg,'b')
>> legend('uzly','polynom ')
```

anonymní funkce str. 24, příkazy `plot` str. 26, `legend` str. 30



V předchozím postupu jsme interpolační polynom sestavili „ručně“. Ukážeme postup, jak sestavit bázové polynomy a interpoleční polynom v MATLABu.
Sestavíme bázové polynomy. Připomeňme, že ve vektoru `xu` jsou uzly.

```
>> phi1=@(x)(x-xu(2)).*(x-xu(3))./((xu(1)-xu(2))*(xu(1)-xu(3)))
phi1 =
    @(x)(x-xu(2)).*(x-xu(3))./((xu(1)-xu(2))*(xu(1)-xu(3)))

>> phi2=@(x)(x-xu(1)).*(x-xu(3))./((xu(2)-xu(1))*(xu(2)-xu(3)))
phi2 =
    @(x)(x-xu(1)).*(x-xu(3))./((xu(2)-xu(1))*(xu(2)-xu(3)))

>> phi3=@(x)(x-xu(1)).*(x-xu(2))./((xu(3)-xu(1))*(xu(3)-xu(2)))
phi3 =
    @(x)(x-xu(1)).*(x-xu(2))./((xu(3)-xu(1))*(xu(3)-xu(2)))

>> p=@(x)fu(1)*phi1(x)+fu(2)*phi2(x)+fu(3)*phi3(x)
p =
    @(x)fu(1)*phi1(x)+fu(2)*phi2(x)+fu(3)*phi3(x)
```

anonymní funkce str. 24

Předpis interpolačního polynomu v MATLABu je ve formě anonymní funkce, který použijeme k vykreslení grafu a případně výpočtu hodnoty interpolační polynomu pro určitou hodnotu.
Pro vykreslení grafu vytvoříme body `xg` a v těchto bodech spočítáme hodnotu nalezeného interpolačního polynomu.

```
>> xg=0:0.01:4
xg =
         0    0.0100    0.0200   .   .   .   3.9900    4.0000
>>  yg=p(xg)
yg =
    2.0000    1.9643    1.9288   .   .   .   4.9493    5.0000
>> plot(xu,yu,'go',xg,yg,'b')
>> legend('uzly','polynom ')
```

dvojtečková konvence str. 12, příkazy `plot` str. 26, `legend` str. 30

Příklad 9: Pro uzly x_i a funkční hodnoty tří různých funkcí f_i, h_i a g_i dané následující tabulkou

	i=0	i=1	i=2
x_i	0	3	4
f_i	2	1	5
g_i	2.2	0.9	4.8
h_i	1.9	1.2	5.1

sestavte interpolační polynom v Lagrangeově tvaru.

Sestavení bázových funkcí v MATLABu bylo zdlouhavé a posloužilo nám pouze k vykreslení grafu interpolačního polynomu. V následujícím příkladě ukážeme výhodnost sestavování polynomu v Lagrangeově tvaru.
Pro hodnoty x_i sestavíme bázové polynomy pouze jednou a poté je použijeme k vytvoření předpisu interpolačních polynomů.

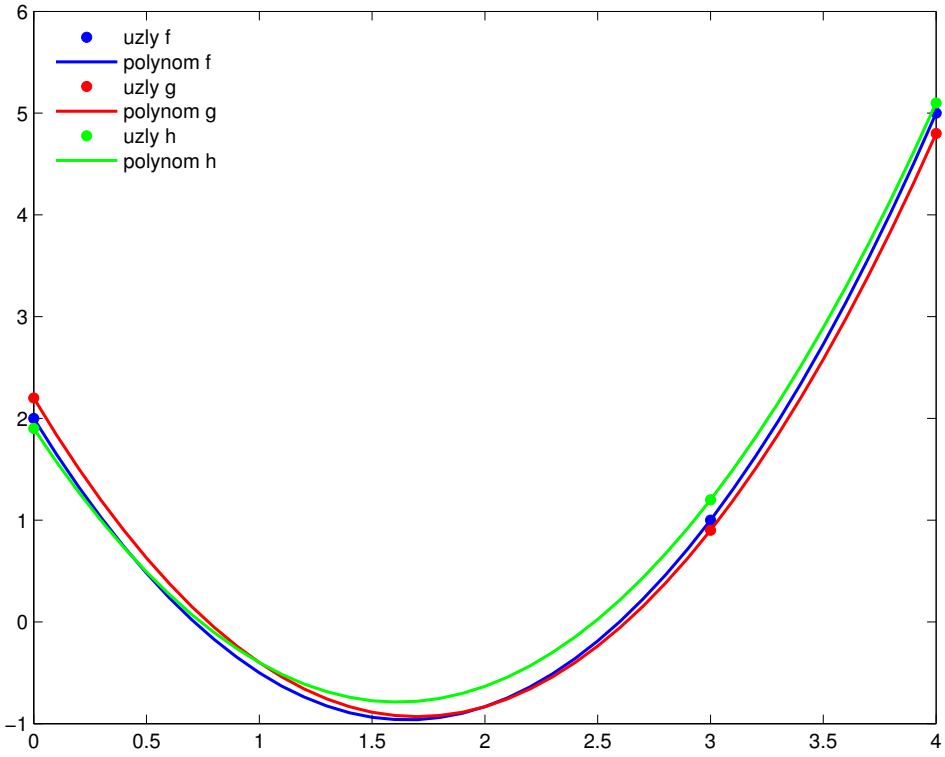
```
>> xu=[0; 3; 4]
xu =
     0
     3
     4
>> phi1=@(x)(x-xu(2)).*(x-xu(3))./((xu(1)-xu(2))*(xu(1)-xu(3)))
phi1 =
    @(x)(x-xu(2)).*(x-xu(3))./((xu(1)-xu(2))*(xu(1)-xu(3)))
>> phi2=@(x)(x-xu(1)).*(x-xu(3))./((xu(2)-xu(1))*(xu(2)-xu(3)))
phi2 =
    @(x)(x-xu(1)).*(x-xu(3))./((xu(2)-xu(1))*(xu(2)-xu(3)))
>> phi3=@(x)(x-xu(1)).*(x-xu(2))./((xu(3)-xu(1))*(xu(3)-xu(2)))
phi3 =
    @(x)(x-xu(1)).*(x-xu(2))./((xu(3)-xu(1))*(xu(3)-xu(2)))
```

anonymní funkce str. 24

Vykreslíme grafy jednotlivých nalezených interpolačních polynomů.

```
>> xg=0:0.1:4
xg =
         0    0.1000    0.2000   .   .   .   3.9000    4.0000
>> fu=[2; 1; 5];
>>  fg=fu(1)*phi1(xg)+fu(2)*phi2(xg)+fu(3)*phi3(xg)
fg =
    2.0000    1.6525    1.3267   .   .   .   4.5025    5.0000
>> gu=[2.2; 0.9; 4.8] ;
>>  gg=gu(1)*phi1(xg)+gu(2)*phi2(xg)+gu(3)*phi3(xg)
gg =
   2.2000    1.8425    1.5067   .   .   .4.3125    4.8000
>> hu=[1.9; 1.2; 5.1];
>>  hg=hu(1)*phi1(xg)+hu(2)*phi2(xg)+hu(3)*phi3(xg)
hg =
    1.9000    1.5770    1.2747   .   .   .   4.6170    5.1000
>> plot(xu,fu,'bo',xg,fg,'b',xu,gu,'ro',xg,gg,'r',xu,hu,'go',xg,
   hg,'g')
>> legend('uzly f','polynom f','uzly g','polynom g','uzly h','
   polynom h')
```

dvojtečková konvence str. 12, příkazy `plot` str. 26, `legend` str. 30



9.3. 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.

Příklad pro $n = 4$.
Výpočet poměrných diferencí pro $n = 4$ je proveden v následující tabulce:

i	x_i	y_i	1.řád $f[x_{i+1}, x_i]$	2.řád $f[x_{i+2}, x_{i+1}, x_i]$	3.řád $f[x_{i+3}, x_{i+2}, x_{i+1}, x_i]$	4.řád $f[x_{i+4}, x_{i+3}, x_{i+2}, x_{i+1}, x_i]$
0	x_0	f_0	$f[x_1,x_0] = \frac{f_1-f_0}{x_1-x_0}$	$f[x_2,x_1,x_0] = \frac{f[x_2,x_1]-f[x_1,x_0]}{x_2-x_0}$	$f[x_3,x_2,x_1,x_0] = \frac{f[x_3,x_2,x_1]-f[x_2,x_1,x_0]}{x_3-x_0}$	$f[x_4,x_3,x_2,x_1,x_0] = \frac{f[x_4,x_3,x_2,x_1]-f[x_3,x_2,x_1,x_0]}{x_4-x_0}$
1	x_1	f_1	$f[x_2,x_1] = \frac{f_2-f_1}{x_2-x_1}$	$f[x_3,x_2,x_1] = \frac{f[x_3,x_2]-f[x_2,x_1]}{x_3-x_1}$	$f[x_4,x_3,x_2,x_1] = \frac{f[x_4,x_3,x_2]-f[x_3,x_2,x_1]}{x_4-x_1}$	
2	x_2	f_2	$f[x_3,x_2] = \frac{f_3-f_2}{x_3-x_2}$	$f[x_4,x_3,x_2] = \frac{f[x_4,x_3]-f[x_3,x_2]}{x_4-x_2}$		
3	x_3	f_3	$f[x_4,x_3] = \frac{f_4-f_3}{x_4-x_3}$			
4	x_4	f_4				

Interpolační polynom v Newtonově tvaru pro $n = 4$ je určen předpisem

$$\begin{aligned} p(x) =& y_0 + f[x_1, x_0](x - x_0) + f[x_2, x_1, x_0](x - x_0)(x - x_1) + \\ &+ f[x_3, x_2, x_1, x_0](x - x_0)(x - x_1)(x - x_2) + \\ &+ f[x_4, x_3, x_2, x_1, x_0](x - x_0)(x - x_1)(x - x_2)(x - x_3)\,. \end{aligned}$$

Příklad 10: Pro uzly x_i a funkční hodnoty f_i dané následující tabulkou

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

sestavte interpolační polynom v Newtonově tvaru.

Výpočet poměrných diferencí je proveden v následující tabulce:

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

Pomocí hodnot vypočtených v prvním řádku tabulky sestavíme interpolační polynom.

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

$$p(x) = 2 - \frac{1}{3}x + \frac{13}{12}x(x-3)$$

```
>> x=[0; 3; 4]
x =
     0
     3
     4
>> y=[2; 1; 5]
y =
     2
     1
     5
>> format rat
>> n=length(x);
```

příkazy `format` str. 8, `length` str. 11

Diference nultého řádu jsou funkční hodnoty v uzlech, tj. y_i. Diference prvního řádu se počítá podle vzorce $f[x_{i+1}, x_i] = \frac{f_{i+1}-f_i}{x_{i+1}-x_i}$.

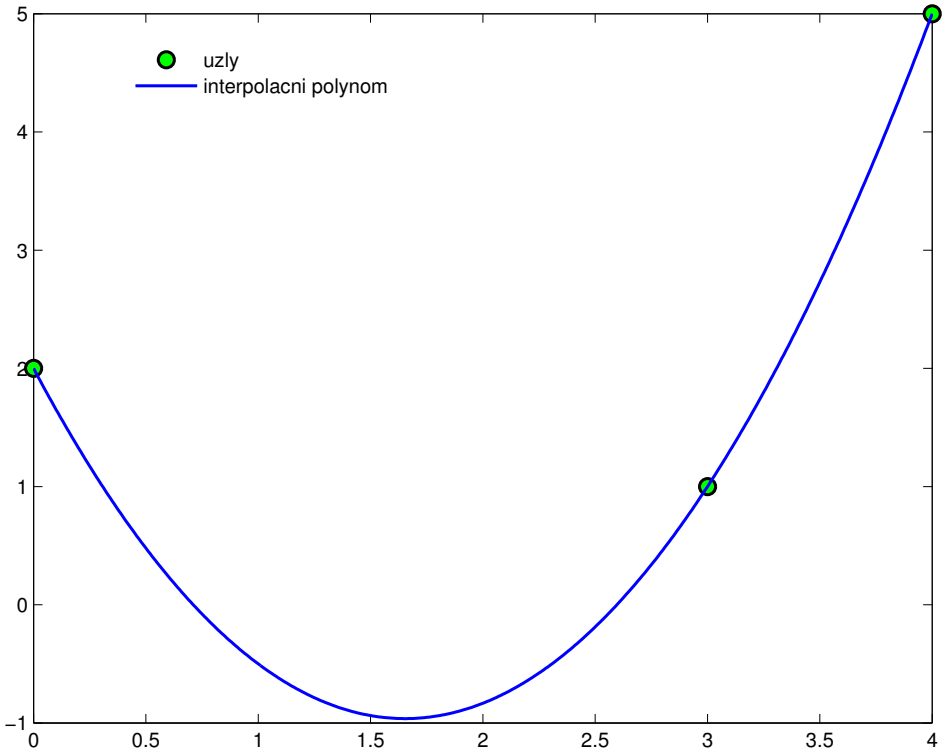
```
>> df0=y
df0 =
       2                1                5
>> for i=1:n-1, df1(i)=(df0(i+1)-df0(i))/(x(i+1)-x(i)), end
df1 =
      -1/3
df1 =
      -1/3             4
>> for i=1:n-2, df2(i)=(df1(i+1)-df1(i))/(x(i+2)-x(i)), end
df2 =
      13/12
```

cyklus `for` str. 35

Sestavíme interpolační polynom a vykreslíme jeho graf společně se zadanými uzly.

```
>> format short
>> xg=x(1):0.01:x(3)
xg =
         0    0.0100    0.0200   .   .   .   3.9900    4.0000
>> yg=df0(1)+df1(1)*(xg-x(1))+df2(1)*(xg-x(1)).*(xg-x(2))
yg =
    2.0000    1.9643    1.9288   .   .   .   4.9493    5.0000
>> plot(x,y,'go')
>> hold on
>> plot(xg,yg)
>> legend('uzly','interpolacni polynom')
```

dvojtečková konvence str. 12, příkaz `format` str. 8, `plot` str. 26, `hold` str. 29, `legend` str. 30



Příklad 11: K datům z předchozího příkladu přidejte uzel $x_3 = 1, y_3 = \frac{3}{2}$ a najděte interpolační polynom.

Do tabulky poměrných diferencí přidáme uzel a dopočítáme diferenci 1.řádu $f[x_3, x_2]$, 2. řádu $f[x_3, x_2, x_1]$ a spočítáme poměrou diferenci 3. řádu $f[x_3, x_2, x_1, x_0]$.

	x_i	f_i	1.řád	2.řád	3.řád
$i = 0$	0	2	$-\frac{1}{3}$	$\frac{13}{12}$	$\frac{1}{3}$
$i = 1$	3	1	4	$\frac{17}{12}$	
$i = 2$	4	5	$\frac{7}{6}$		
$i = 3$	1	$\frac{3}{2}$			

K interpolačnímu polynomu přidáme další člen.

$$p(x) = 2 - \frac{1}{3}(x-0) + \frac{13}{12}(x-0)(x-3) + \frac{1}{3}(x-0)(x-3)(x-4).$$

Nalezený interpolační polynem je

$$p(x) = 2 - \frac{1}{3}x + \frac{13}{12}x(x-3) + \frac{1}{3}x(x-3)(x-4).$$

Přidáme hodnoty nového uzly k vektorům x a y.

```
>> x=[x; 1]
x =
     0
     3
     4
     1
>> y=[y;1.5]
y =
    2.0000
    1.0000
    5.0000
    1.5000
```

Dopočítáme diferenci 1.řádu $f[x_3, x_2]$ 2. řádu. $f[x_3, x_2, x_1]$.

```
>> format rat
>> n=length(x);
>> df0=y
df0 =
       2
       1
       5
       3/2
>> for i=1:n-1, df1(i)=(df0(i+1)-df0(i))/(x(i+1)-x(i)), end
df1 =
      -1/3
df1 =
      -1/3            4
df1 =
      -1/3            4              7/6
>> for i=1:n-2, df2(i)=(df1(i+1)-df1(i))/(x(i+2)-x(i)), end
df2 =
      13/12
df2 =
      13/12          17/12
```

cyklus `for` str. 35, příkazy `format` str. 8, `length` str. 11

Spočítáme poměrou diferenci 3. řádu $f[x_3, x_2, x_1, x_0]$.

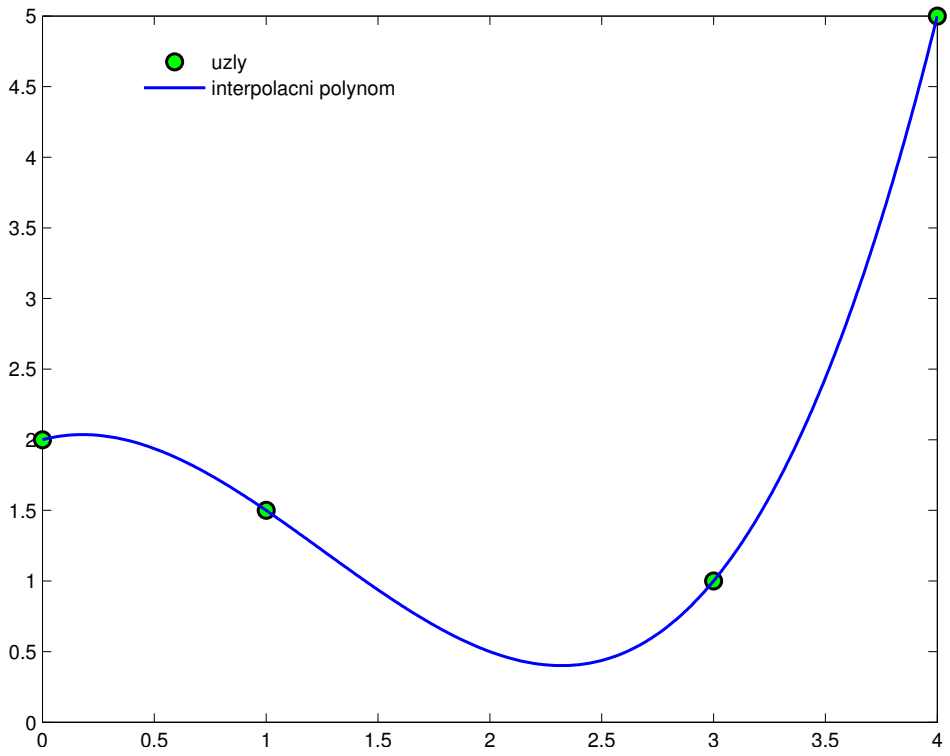
```
>> for i=1:n-3, df3(i)=(df2(i+1)-df2(i))/(x(i+3)-x(i)), end
df3 =
       1/3
```

cyklus `for` str. 35

Rozšíříme interpolační polynomu o další člen a spočítáme jeho hodnoty v bodech xg.

```
>> xg=min(x):0.01:max(x);
>> yg=df0(1)+df1(1)*(xg-x(1))+df2(1)*(xg-x(1)).*(xg-x(2))+df3(1)
   *(xg-x(1)).*(xg-x(2)).*(xg-x(3))
yg =
    2.0000    2.0040    2.0078  ...  4.9361    5.0000
>> plot(x,y,'go')
>> hold on
>> plot(xg,yg)
>> legend('uzly','interpolacni polynom')
```

příkazy`format` str. 8, `plot` str. 26, `hold` str. 29, `legend` str. 30, `max, min` str. 18



9.4. Aproximace metodou nejmenších čtverců

Nechť jsou dány vzájemně různé uzly x_i a funkční hodnoty y_i, $i = 1, \ldots, n$. Nechť jsou dány také 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 $c_1, c_2 \in \mathbb{R}$.

Příklad 12: Aproximujte následující data

	i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9
x_i	-2.3	-1.3	0.6	1.5	2.8	3.3	4.6	5.9	7.8	9.3
y_i	-51	-15	8	31	-47	-11	-101	-110	-223	-307

metodou nejmenších čtverců

a) funkcí
$$\varphi(x) = c_1 \sin(x) + c_2 x^2,$$

b) lineární funkcí
$$\zeta(x) = d_1 x + d_2.$$

Získané výsledky porovnejte jak graficky, tak číselně.

Případ a
V našem případě je $\varphi_1(x) = \sin x$ a $\varphi_2(x) = x^2$. Soustavu můžeme s výhodou zapsat v maticovém tvaru. Definujeme také pomocnou matici $\boldsymbol{G_a}$ a vektor $\boldsymbol{d_a}$.

$$\begin{pmatrix} \sum_{i=1}^{n} (\varphi_1(x_i))^2 & \sum_{i=1}^{n} \varphi_1(x_i) \cdot \varphi_2(x_i) \\ \sum_{i=1}^{n} \varphi_2(x_i) \cdot \varphi_1(x_i) & \sum_{i=1}^{n} (\varphi_2(x_i))^2 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i \cdot \varphi_1(x_i) \\ \sum_{i=1}^{n} y_i \cdot \varphi_2(x_i) \end{pmatrix}$$
$$\boldsymbol{G_a} \cdot \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \boldsymbol{d_a}$$

Vyřešením této soustavy lineárních rovnic získáme hodnoty koeficientů c_1, c_2, pro něž je funkce $\varphi(x)$ nejlepší aproximací zadaných dat ve smyslu nejmenších čtverců.
V MATLABu nejprve zadáme data.

```
>> x=[-2.3 -1.3 0.6 1.5 2.8 3.3 4.6 5.9 7.8 9.3];
>> y=[-51 -15 8 31 -47 -11 -101 -110 -223 -307];
```

Následně definujeme funkce $\varphi_1(x) = \sin x$ a $\varphi_2(x) = x^2$ v MATLABu pod proměnnými `f1`, `f2`.

```
>> f1=@(x)sin(x);
>> f2=@(x)x.^2;
```

tečková notace 17

Pro výpočet budeme potřebovat také matici soustavy normálních rovnic $\boldsymbol{G_a}$ a vektor pravé strany $\boldsymbol{d_a}$.

```
>> Ga=[sum(f1(x).^2) sum(f1(x).*f2(x));sum(f2(x).*f1(x)) sum(f2(
   x).^2)]
Ga =
  1.0e+004 *
    0.0005    0.0035
    0.0035    1.3058

>> da=[sum(f1(x).*y);sum(f2(x).*y)]
da =
  1.0e+004 *
   -0.0045
   -4.6797
```

příkaz sum str. 18

Soustavu normálních rovnic vyřešíme jednou z mnoha metod, které MATLAB nabízí.

```
>> c=Ga\da
   16.2406
   -3.6277
```

řešení soustavy lineárních rovnic str. 19

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

$$\varphi(x) = 16.2406 \sin x - 3.6277x^2.$$

Případ b

Postup výpočtu je obdobný jako výše, budeme proto stručnější a soustředíme se zejména na popis odlišností. Definujeme funkce $\zeta_1(x) = 1$ a $\zeta_2(x) = x$ v MATLABu pod proměnnými `p1`, `p2`. Pak matici soustavy normálních rovnic $\boldsymbol{G_b}$ a vektor pravé strany $\boldsymbol{d_b}$.

```
>> p1=@(x)x.^0;
>> p2=@(x)x;
>> Gb=[sum(p1(x).^2) sum(p1(x).*p2(x));sum(p2(x).*p1(x)) sum(p2(
   x).^2)]
Gb =
   10.0000   32.2000
   32.2000  231.6200
>> db=[sum(p1(x).*y);sum(p2(x).*y)]
db =
  1.0e+003 *
   -0.8260
   -5.6879
```

tečková notace, příkaz sum str. 18

Při definici funkce p2 v MATLABu jsme se museli uchýlit k drobnému technickému triku a využít rovnosti $1 = x^0$. Kdybychom tak neučinili, museli bychom prvek v prvním řádku a prvním sloupci matice G_b definovat ručně jako $\sum_1^n \zeta_2^2(x) = \sum_1^n 1 = 10$ (což je počet bodů v tabulce).
Vyřešíme soustavu normálních rovnic.

```
>> d=Gb\db
    -6.3842
   -23.6695
```

řešení soustavy lineárních rovnic str. 19

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

$$\zeta(x) = -6.3842 - 23.6695x.$$

Porovnání aproximací

Získané aproximace nyní uložíme do proměnných f a p. Připomeňme, že koeficienty c_1, c_2 (resp. d_1, d_2) jsme právě spočetli a uložili do proměnné c (resp. d); jejich hodnoty jsou tudíž c(1) a c(2) (resp. d(1), d(2)).

```
>> f=@(x)c(1)*f1(x)+c(2)*f2(x);
>> p=@(x)d(1)*p1(x)+d(2)*p2(x);
```

Nyní porovnáme součty čtverců vzdáleností nalezených aproximací od zadaných dat, tj.

$$\sum_{i=1}^{10}(\varphi(x_i) - y_i)^2, \quad \sum_{i=1}^{10}(\zeta(x_i) - y_i)^2.$$

Výpočet provedeme v MATLABu.

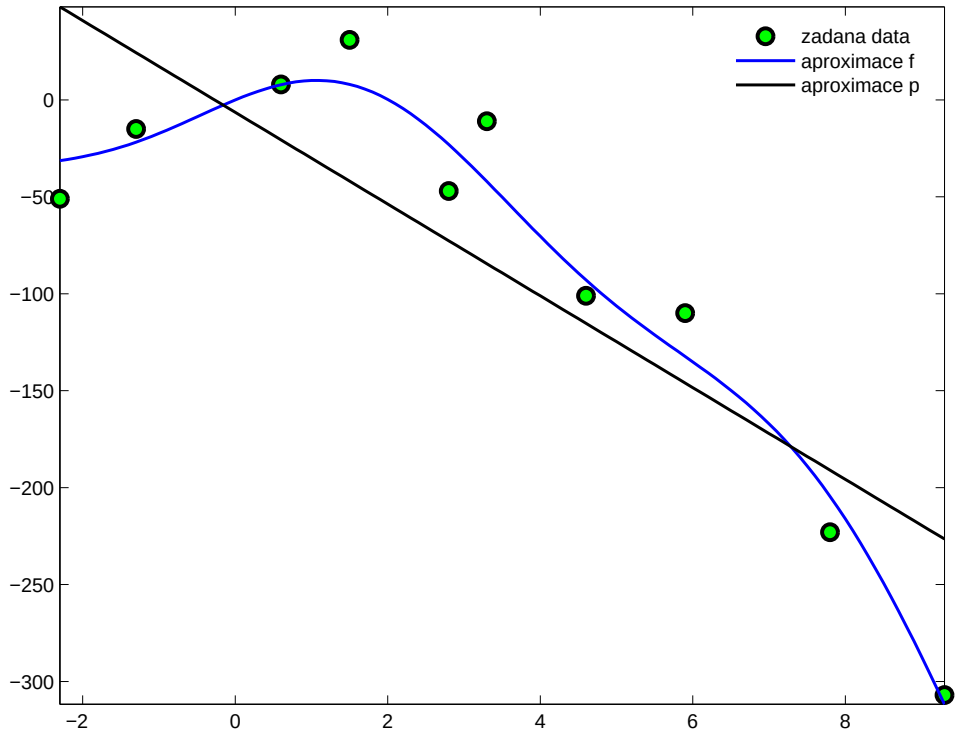
```
>> sum((f(x)-y).^2)
ans =
  3.4327e+003
>> sum((p(x)-y).^2)
ans =
  3.2557e+004
```

Protože je $3432 < 32557$, je funkce $\varphi(x)$ lepší aproximací než $\zeta(x)$.
Nakonec necháme MATLAB vykreslit grafy získaných aproximací $\varphi(x)$, $\zeta(x)$ společně se znázorněním zadaných bodů. Také z grafů je patrné, že lepší apriximaci dat poskytuje funkce $\varphi(x)$.

```
>> x1=-2.3:0.1:9.3; % pro potreby grafu vytvorime vektor
   pokryvajici vsechny zadane body, jako krok volime standardne
   0.1
```

```
>> plot(x,y,'go',x1,f(x1),'b-',x1,p(x1),'k-') % data vykreslime
   jako zelena kolecka a nalezene aproximace jako modrou, resp.
   cernou caru
>> legend('zadana data','aproximace f','aproximace p') % ke
   grafu zobrazime legendu
```

příkaz `plot` str. 26



KAPITOLA

10

NUMERICKÉ INTEGROVÁNÍ A DERIVOVÁNÍ

10.1. Výpočet integrálu se zadanou přesností

Složené Simpsonovo pravidlo

Pravidlo slouží k aproximaci hodnoty určitého integrálu

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

Rozdělíme interval $\langle a, b\rangle$ na $2m$ stejně dlouhých dílků, $m \geq 2$, s krokem $h = (b-a)/(2m)$, takže budeme mít lichý počet uzlů $x_i = a + ih$, $i = 0, 1, \ldots, 2m$. Pak

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4\sum_{i=1}^{m} f(x_{2i-1}) + 2\sum_{i=1}^{m-1} f(x_{2i}) + f(x_{2m})\right].$$

Příklad 13: Vypočtěte integrál

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

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); % nezapomente na teckovou notaci,
    zde je nutna
```

```
>> a=1;
>> b=exp(1);
```

Připomeňme, že přirozený logaritmus v MATLABu naleznete jako `log` a Eulerovo číslo byste pod proměnnou `e` zase hledali zbytečně – je třeba definovat pomocí exponenciální funkce.
Dále budeme počítat integrál složenou Simpsonovou formulí pro $n = 2, 4, 8, 16, \ldots$, dokud bude chyba větší než zadaná přesnost, tj. $|I_h - I_{2h}| > \varepsilon$.

Chceme-li integrovat funkci f na intervalu $\langle a, b\rangle$ složenou Simpsonovou formulí, musíme nejprve zadaný interval rozdělit na $2m$ stejně dlouhých dílků, kde $m \in \mathbb{N}$. Dílky pak budou mít velikost $h = (b-a)/(2m)$ a dostaneme lichý počet uzlů $x_i = a + ih$, $i = 0, 1, \ldots, 2m$. Mějte na paměti, že složená Simpsonova formule pro krok h pak má tvar

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

Abychom výpočet trochu urychlili, bude v prvním kroku $m = 2$. Interval tedy rozdělíme na $n = 2m = 4$ stejně velké dílky velikosti $h = (b-a)/n$ s uzlovými body x_i, $i = 0, 1, \ldots, n$. Délku kroku i uzlové body můžeme snadno spočíst:

$$\begin{aligned} h &= \frac{b-a}{n} = \frac{e-1}{4} \approx 0.4296, \\ x_0 &= a = 1, \\ x_1 &= a + h \approx 1.4296, \\ x_2 &= a + 2h \approx 1.8591, \\ x_3 &= a + 3h \approx 2.2887, \\ x_4 &= a + 4h = e \approx 2.7183. \end{aligned}$$

Totéž nyní provedeme v MATLABu. Uzly uložíme jako vektor do proměnné `x`.

```
>> m=2;
>> n=2*m;
>> h=(b-a)/n
h =
    0.4296

>> x=a:h:b
x =
    1.0000    1.4296    1.8591    2.2887    2.7183
```

Klíčem k elegantnímu výpočtu v MATLABu je umět jednoduše zapsat sumy v Simpsonově formuli. Jak si si poradit například se sumou $\sum_{i=1}^{m} f(x_{2i-1})$? Jednoduše – pro $m = 2$ sčítáme

$$\sum_{i=1}^{2} f(x_{2i-1}) = f(x_1) + f(x_3),$$

tj. funční hodnoty funkce f v *druhém* a *čtvrtém* prvku vektoru $(x_0, x_1, x_2, x_3, x_4)$, který jsme v MATLABu uložili do proměnné `x`. Sčítáme tedy fukční hodnoty prvků na *sudých* pozicích

tohoto vektoru. Připomeneme si nyní, jak v MATLABu z vektoru vybrat prvky na sudých pozicích, jak spočíst jejich funkční hodnoty a jak je sečíst.

```
>> x(2:2:4)
ans =
    1.4296    2.2887

>> sum(f(x(2:2:4)))
ans =
    0.5624
```

příkaz sum str. 18

V tuto chvíli by pro vás měl být přepis celého vzorce Simpsonovy formule do MATLABu hračkou. 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)))
Inovy =
    0.5104
```

Spočtenou hodnotu bychom měli porovnat s hodnotou spočtenou v předchozím kroku (tj. při dvojnásobné velikosti kroku). Protože jsme však s výpočtem právě začali, nemáme s čím porovnávat a je třeba pokračovat. Spočtenou hodnotu integrálu pouze uložíme do proměnné `I`. Pak zdvojnásobíme hodnotu m na $m = 4$ a celý výpočet zopakujeme.

```
>> I=Inovy;
>> m=2*m;
>> n=2*m;
>> 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)))
Inovy =
    0.5071
```

V tuto chvíli můžem odhadnout chybu. Spočteme $|I_h - I_{2h}| \approx |0.5104 - 0.5071| = 0.0033$. V MATLABu:

```
>> abs(Inovy-I)
ans =
    0.0033
```

Jelikož $0.0033 > 10^{-8}$, je třeba ve výpočtu pokračovat. Zdvojnásobíme opět počet dílků, tentokrát na $m = 8$. Všechny příkazy, které budeme opakovat, dokud nedosáhneme požadované přesnosti, tentokrát napíšeme najednou.

```
>> I=Inovy;
>> m=2*m;
>> n=2*m;
```

```
>> h=(b-a)/n;
>> x=a:h:b;
>> Inovy=h/3*(f(x(1))+4*sum(x(2:2:n))+2*sum(x(3:2:n))+f(x(n+1)))
>> abs(Inovy-I)
Inovy =
    0.5067
ans =
    1.0000e-004
```

Opět $10^{-4} > 10^{-8}$, je proto nutné pokračovat. 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
```

příkaz `format` str. 8

V tuto chvíli stačí napsat všechny výše uvedené příkazy na jeden řádek a opakovat je, dokud je chyba větší než 10^{-4}. Výpočtem byste získali následující hodnoty.

$n = 2m$	h	I_h	$\lvert I_h - I_{2h}\rvert$	$\varepsilon = 10^{-3} = 0.001$
4	0.42957046	0.51036199	—	
8	0.21478523	0.50708297	0.00327902	$> 10^{-8}$
16	0.10739261	0.50665442	0.00042855	$> 10^{-8}$
32	0.05369631	0.50661499	0.00003943	$> 10^{-8}$
64	0.02684815	0.50661211	0.00000288	$> 10^{-8}$
128	0.01342408	0.50661192	0.00000019	$> 10^{-8}$
256	0.00671204	0.50661191	0.00000001	$> 10^{-8}$
512	0.00335602	0.50661191	0.00000000	$\leq 10^{-8}$

Výsledek zapíšeme jako

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

Příkaz quad

V MATLABu přítomen příkaz `quad` pro numerické integrování. Příkaz používá adaptivní rekurzivní Simpsonovo pravidla. Jedná se o metodu, kterou jsme se zde nezabývali. Příkaz však můžeme použít k ověření správnosti předchozího výpočtu.

```
>> f=@(x)log(x)./sqrt(9-x.^2); % definujeme funkci
>> format long % format vystupu zmenime na presnejsi 'long'
>> quad(f,1,exp(1),1e-8) % spocteme integral z f od 1 do exp(1)
   s presnosti 1e-8
ans =
   0.50661191096231
```

Všimněme si, že se hodnota získaná příkazem quad skutečně na prvních osmi desetinných místech shoduje s výsledkem, který jsme získali složeným Simpsonovým pravidlem.

10.2. Numerické derivování

Příklad 14: Pro funkci

$$f(x) = \sin(x^2)$$

vypočtěte přibližně její první derivaci na intervalu $\langle 0, 2\rangle$ pomocí vzorce

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

s krokem h = 0.25.

Výpočet hodnot x_i je snadný $x_1 = 0,\ x_2 = 0.25,\ x_3 = 0.5, \ldots x_8 = 1.75,\ x_9 = 2.$
Vypočítáme přibližnou hodnotu derivace $f'(x_2)$

$$f'(x_2) = \frac{f(x_3) - f(x_1)}{x_3 - x_1} = \frac{\sin(x_3^2) - \sin(x_1^2)}{x_3 - x_1} = \frac{\sin(0.5^2) - \sin(0^2)}{0.5 - 0} = 0.4948$$

hodnotu derivace $f'(x_3)$

$$f'(x_3) = \frac{f(x_4) - f(x_2)}{x_4 - x_2} = \frac{\sin(x_4^2) - \sin(x_2^2)}{x_4 - x_2} = \frac{\sin(0.75^2) - \sin(0.25^2)}{0.75 - 0.25} = 0.9417.$$

Obdobně se spočítají všechny hodnoty $f'(x_i)$ pro $i = 4, \ldots, 8$. Zadáme velikost kroku h a vytvoříme hodnoty s krokem h na intervalu $\langle 0, 2\rangle$ pomocí dvojtečky.

```
>> h=0.25
h =
    0.2500
>> x=0:h:2
x =
  Columns 1 through 6
         0    0.2500    0.5000    0.7500    1.0000    1.2500
  Columns 7 through 9
    1.5000    1.7500    2.0000
```

dvojtečková konvence str. 12

Zadáme funkci f.

```
>> f=@(x)sin(x.^2)
f =
    @(x)sin(x.^2)
```

anonymní funkce str. 24

Nejprve spočítáme počet n hodnot x_i. Pomocí cyklu spočítáme přibližné hodnoty derivace podle vzorce v zadání. Připomeňme, že hodnoty počítáme pouze ve vnitřních bodech, tedy pro $x_2, \ldots, x_8$.

```
>> n=length(x)
n =
     9
>> for i=2:n-1, fd(i)=(f(x(i+1))-f(x(i-1)))/(2*h); end
>> fd
fd =
  Columns 1 through 6
         0    0.4948    0.9417    1.1881    0.9333   -0.1268
  Columns 7 through 8
   -1.8419   -3.0698
```

příkaz `length` str. 11, cyklus `for` str. 35

Všimněme si, že při vytváření vektoru `fd` v cyklu jsme do něj zapisovali hodnoty `fd(2)` ... `fd(8)`. MATLAB však do `fd(1)` zapsal hodnotu `0` pouze z technických důvodů, neboť nemohl nechat `fd(1)` prázdné.

Přibližné hodnoty derivace.

x_i	0	0.2500	0.5000	0.7500	1.0000	1.2500	1.5000	1.7500	2.0000
$f'(x_i)$	—	0.4948	0.9417	1.1881	0.9333	-0.1268	-1.8419	-3.0698	—

KAPITOLA

11

OBYČEJNÉ DIFERENCIÁLNÍ ROVNICE: POČÁTEČNÍ ÚLOHY

Počáteční úloha pro obyčejnou diferenciální rovnici

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

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

11.1. 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 $y(x_0)$, $y(x_1), y(x_2), \ldots, y(x_n)$:

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

Příklad 15: Počáteční úlohu

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

řešte na intervalu $\langle 0, 2\rangle$. Použijte Eulerovu metodu 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
a =
     0
>> b=2
b =
     2
>> c=-1
c =
    -1
>> f=@(x,y)(y-x.^2+2)
f =
     @(x,y)(y-x.^2+2)
```

anonymní funkce str. 24

Zadáme velikost kroku h a vypočítáme počet dílů dělení $n = \frac{b-a}{h} = \frac{2-0}{0.5} = 4$.

```
>> h=0.5
h =
    0.5000
>> n=(b-a)/h
n =
     4
```

Vypočítáme hodnoty $x_i = a + ih$ pro $i = 0, \ldots, n$

$$
\begin{aligned}
x_0 &= a = 0, \\
x_1 &= a + h = 0 + 0.5 = 0.5, \\
x_2 &= a + 2h = 0 + 2 \cdot 0.5 = 1, \\
x_3 &= a + 3h = 0 + 3 \cdot 0.5 = 1.5, \\
x_4 &= a + 4h = 0 + 4 \cdot 0.5 = 2.
\end{aligned}
$$

Pomocí dvojtečkové konvence spočítáme x_i v MATLABu.

```
>> x=a:h:b
x =
         0    0.5000    1.0000    1.5000    2.0000
```

dvojtečková konvence str. 12

Zadáme hodnotu $y_0 = c$ a spočítáme ostatní hodnoty y.

$$
\begin{aligned}
y_0 &= c = -1 \\
y_1 &= y_0 + h \cdot f(x_0, y_0) = -1 + 0.5 \cdot (-1 - 0^2 + 2) = -0.5 \\
y_2 &= y_1 + h \cdot f(x_1, y_1) = -0.5 + 0.5 \cdot (-0.5 - 0.5^2 + 2) = 0.125
\end{aligned}
$$

$$y_3 = y_2 + h \cdot f(x_2, y_2) = 0.125 + 0.5 \cdot (0.125 - 1^2 + 2) = 0.6875$$
$$y_4 = y_3 + h \cdot f(x_3, y_3) = 0.6875 + 0.5 \cdot (0.6875 - 1.5^2 + 2) = 0.9036$$

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)`.

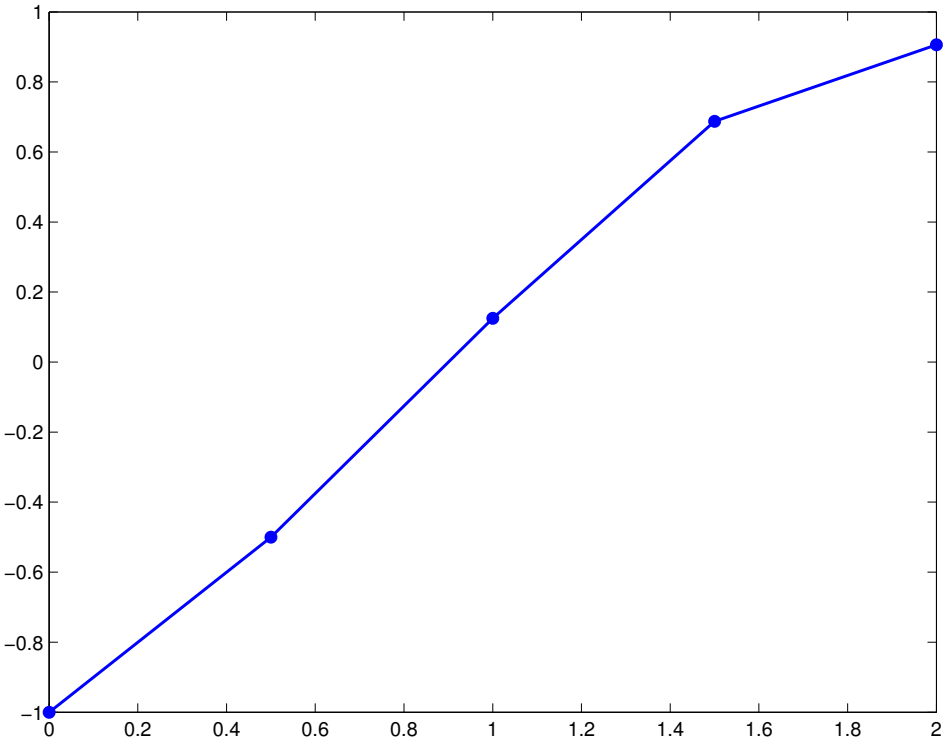
```
>> y(1)=c
y =
    -1
>> for i=1:n, y(i+1)=y(i)+h*f(x(i),y(i)), end
y =
   -1.0000   -0.5000
y =
   -1.0000   -0.5000    0.1250
y =
   -1.0000   -0.5000    0.1250    0.6875
y =
   -1.0000   -0.5000    0.1250    0.6875    0.9063
```

příkaz `for` str. 35

Hodnoty numerického řešení vypíšeme do tabulky a vykreslíme graf.

```
>> [x;y]
ans =
         0    0.5000    1.0000    1.5000    2.0000
   -1.0000   -0.5000    0.1250    0.6875    0.9063
>> plot(x,y,'b.-')
```

příkaz `plot` str. 26



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

Příklad 16: Porovnejte řešení předchozí úlohy s řešení pro krok $h = 0.1$.

Nejprve uložíme do proměnných `x05` a `y05` řešení pro krok $h = 0.5$ a pro další výpočet smažeme proměnné `x`, `y`, `n`, `h`.

```
x05=x; y05=y;
clear x y n h
```

příkaz `clear` str. 8

Zadáme velikost kroku h a vypočítáme počet dílů dělení n.

```
>> h=0.1
h =
    0.1000
>> n=(b-a)/h
n =
    20
```

Pomocí dvojtečkové konvence vypočítáme hodnoty x_i.

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

dvojtečková konvence str. 12

Zadáme hodnotu první hodnotu y_0 a spočítáme ostatní hodnoty y.

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

příkaz `for` str. 35

Hodnoty numerického řešení vypíšeme do tabulky.

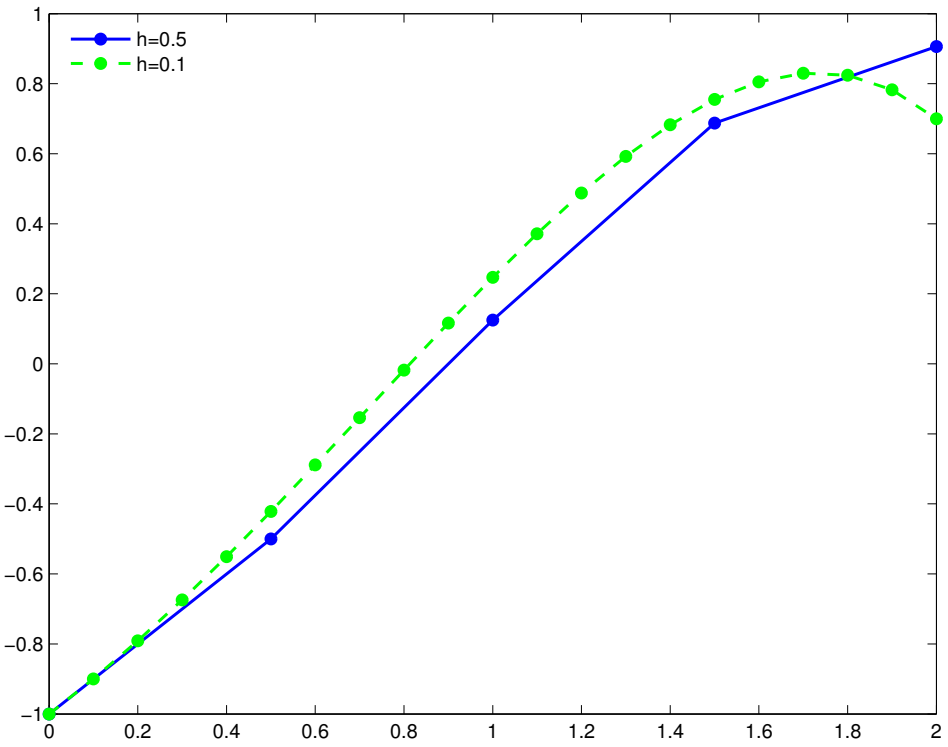
```
>> [x;y]
ans =
  Columns 1 through 6
         0    0.1000    0.2000    0.3000    0.4000    0.5000
   -1.0000   -0.9000   -0.7910   -0.6741   -0.5505   -0.4216
  Columns 7 through 12
    0.6000    0.7000    0.8000    0.9000    1.0000    1.1000
   -0.2887   -0.1536   -0.0179    0.1163    0.2469    0.3716
  Columns 13 through 18
```

```
    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
    0.4877    0.5925    0.6828    0.7550    0.8055    0.8301
  Columns 19 through 21
    1.8000    1.9000    2.0000
    0.8241    0.7825    0.6998
```

Vykreslíme numerické řešení pro krok $h = 0.5$ (které jsme uložili do proměnných x05, y05) a pro krok $h = 0.1$.

```
>> x01=x; y01=y;
>> plot(x05,y05,'b.-',x01,y01,'g.--')
>> legend('h=0.5','h=0.1')
```

příkazy plot str. 26, legend str. 30

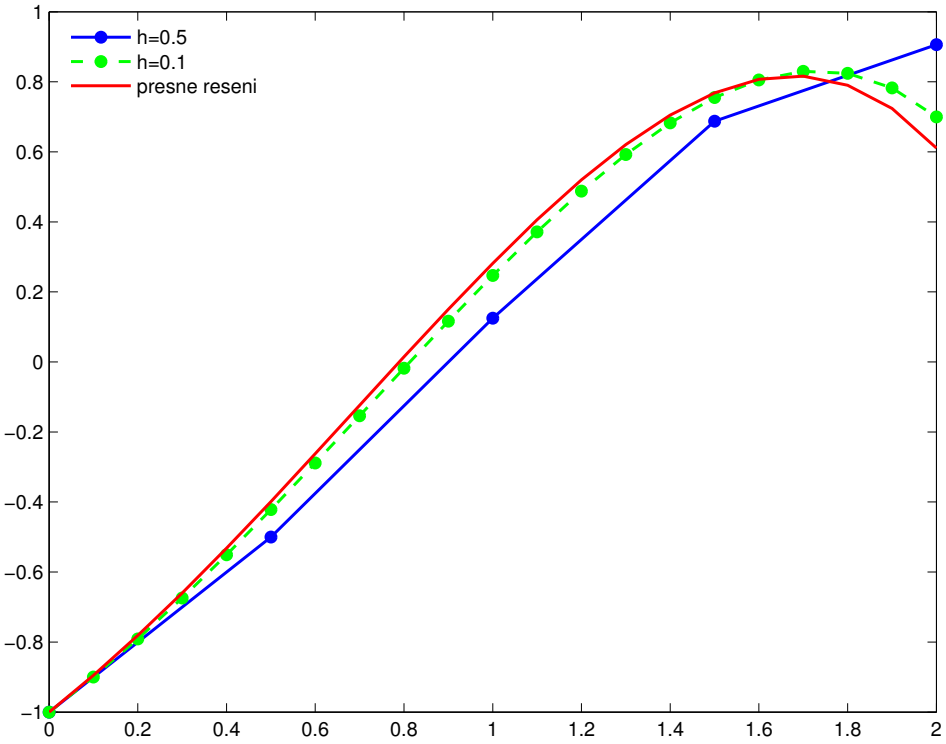


Příklad 17: Srovnejte řešení předchozí úlohy s přesným řešením.

Graficky srovnáme numerická řešení s přesným řešením $y = x^2 + 2x - e^x$ počáteční úlohy.

```
>> plot(x05,y05,'b.-',x01,y01,'g.--')
>> hold on
>> fplot('x.^2+2*x-exp(x)',[0,2],'r')
>> legend('h=0.5','h=0.1','presne reseni')
```

příkazy plot str. 26, fplot str. 26 , legend str. 29, hold str. 30



11.2. Metoda Runge-Kutta

Interval $\langle a, b\rangle$ rozdělíme ekvidistantními 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 $y(x_0), y(x_1), y(x_2), \ldots, y(x_n)$

$$\begin{aligned} y_0 &= c, \\ y_{i+1} &= y_i + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= hf(x_i, y_i), \\ k_2 &= hf(x_i + \tfrac{h}{2}, y_i + \tfrac{k_1}{2}), \\ k_3 &= hf(x_i + \tfrac{h}{2}, y_i + \tfrac{k_2}{2}), \\ k_4 &= hf(x(i+1), y_i + k_3), \quad i = 0, 1, \ldots, n-1. \end{aligned}$$

Příklad 18: Počáteční úlohu

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

řešte na intervalu $\langle -1, 4\rangle$ pomocí metody Runge-Kutta s krokem $h = 1$. Srovnejte s řešením pro krok $h = 0.5$ a $h = 0.1$.

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

anonymní funkce str. 24

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

```
>> h=1, n=(b-a)/h
h =
     1
n =
      5
```

Pomocí dvojtečkové konvence vypočítáme hodnoty x_i.

```
>> x=a:h:b
x =
    -1     0     1     2     3     4
```

dvojtečková konvence str. 12

Nyní spočítáme hodnoty y_i. První hodnota $y_0 = 1$ je známa z počáteční podmínky. Výpočet y_i pro $i = 1, \ldots, n$ je poněkud složitější. Pro každé i je potřeba určit k_1, k_2, k_3, k_4 a poté hodnotu y_i. Výpočet pro y_1.

$$k_1 = h \cdot f(x_0, y_0) = 1 \cdot (\sin(-1) + \cos(3 \cdot 1)) = -1.8315$$

$$k_2 = h \cdot f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right) = 1 \cdot \left(\sin(-1 + \frac{1}{2}) + \cos(3(1 + \frac{-1.8315}{2}))\right) = 0.4888$$

$$k_3 = h \cdot f\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right) = 1 \cdot \left(\sin(-1 + \frac{1}{2}) + \cos(3(1 + \frac{0.4888}{2}))\right) = -1.3095$$

$$k_4 = h \cdot f\left(x_1, y_0 + k_3\right) = 1 \cdot (\sin(0) + \cos(3(1 - 1.3095))) = 0.5991$$

$$y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1 + \frac{1}{6}(-1.8315 + 2 \cdot 0.4888 + 2 \cdot (-1.3095) + 0.5991) =$$
$$= 0.5210$$

Zadáme hodnotu první hodnotu y_0 a spočítáme ostatní hodnoty y. Připomeňme, že v MATLABu se indexuje od 1. V každém kroku se počítají hodnoty k_1, k_2, k_3, k_4 a hodnota y_{i+1}.

```
>> y(1)=c
y =
     1
```

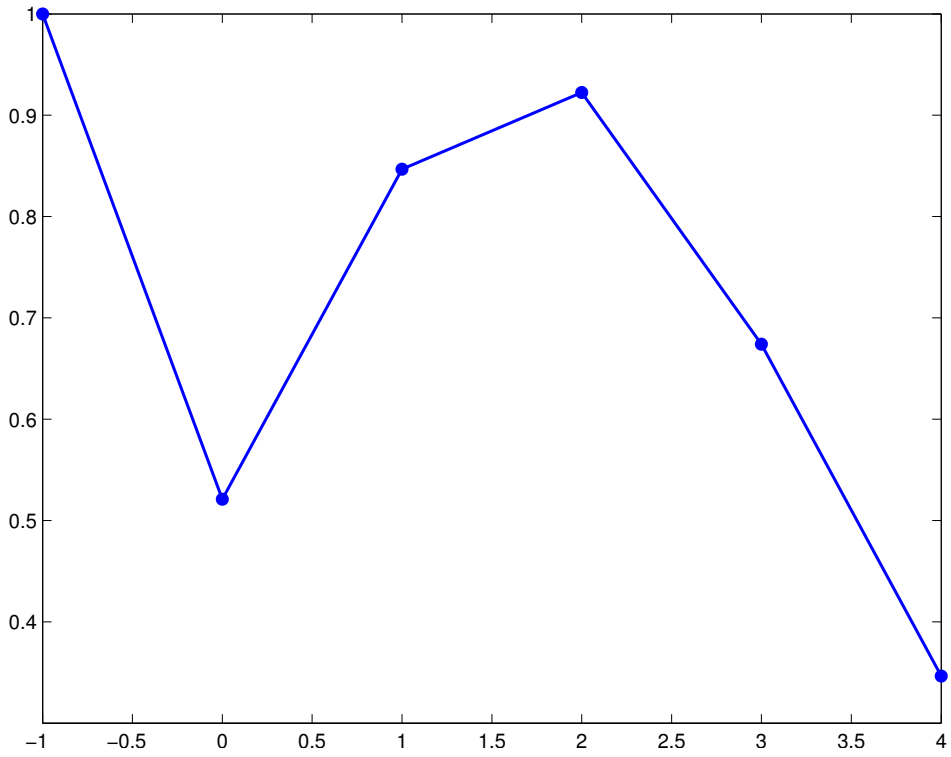
```
>> for i=1:n,
     k1=h*f(x(i),y(i));
     k2=h*f(x(i)+h/2,y(i)+k1/2);
     k3=h*f(x(i)+h/2,y(i)+k2/2);
     k4=h*f(x(i+1),y(i)+k3);
     y(i+1)=y(i)+1/6*(k1+2*k2+2*k3+k4),
 end
y =
    1.0000    0.5210
y =
    1.0000    0.5210    0.8468
y =
    1.0000    0.5210    0.8468    0.9223
y =
    1.0000    0.5210    0.8468    0.9223    0.6741
y =
    1.0000    0.5210    0.8468    0.9223    0.6741    0.3465
```

příkaz `for` str. 35

Hodnoty numerického řešení vypíšeme do tabulky a vykreslíme graf.

```
>> [x;y]
ans =
   -1.0000         0    1.0000    2.0000    3.0000    4.0000
    1.0000    0.5210    0.8468    0.9223    0.6741    0.3465
>> plot(x,y,'.-')
```

příkaz `plot` str. 26



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

Příklad 19: Porovnejte řešení předchozí úlohy pro krok $h = 0.5$ a $h = 0.1$.

Nejprve uložíme do proměnných `x1` a `y1` řešení pro krok $h = 1$ a smažeme proměnné `x`, `y`, `n`, `h`.

```
>> x1=x; y1=y;
>> clear x y n h
```

příkaz `clear` str. 8

Zadáme velikost kroku h, vypočítáme počet dílů dělení n a hodnoty x_i.

```
>> h=0.5, n=(b-a)/h
h =
    0.1000
n =
    10
>> x=a:h:b;
```

dvojtečková konvence str. 12

Zadáme hodnotu první hodnotu y_0 a spočítáme ostatní hodnoty y.

```
>> 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
```

příkaz `for` str. 35

Hodnoty numerického řešení vypíšeme do tabulky.

```
>> [x;y]
ans =
  Columns 1 through 6
   -1.0000   -0.5000         0    0.5000    1.0000    1.5000
    1.0000    0.4994    0.4793    0.5953    0.7306    0.8430
  Columns 7 through 11
    2.0000    2.5000    3.0000    3.5000    4.0000
    0.8948    0.8425    0.6923    0.5188    0.3621
```

Nejprve uložíme do proměnných x05 a y05 řešení pro krok $h = 0.5$ a smažeme proměnné.

```
x05=x; y05=y;
clear x y n h
```

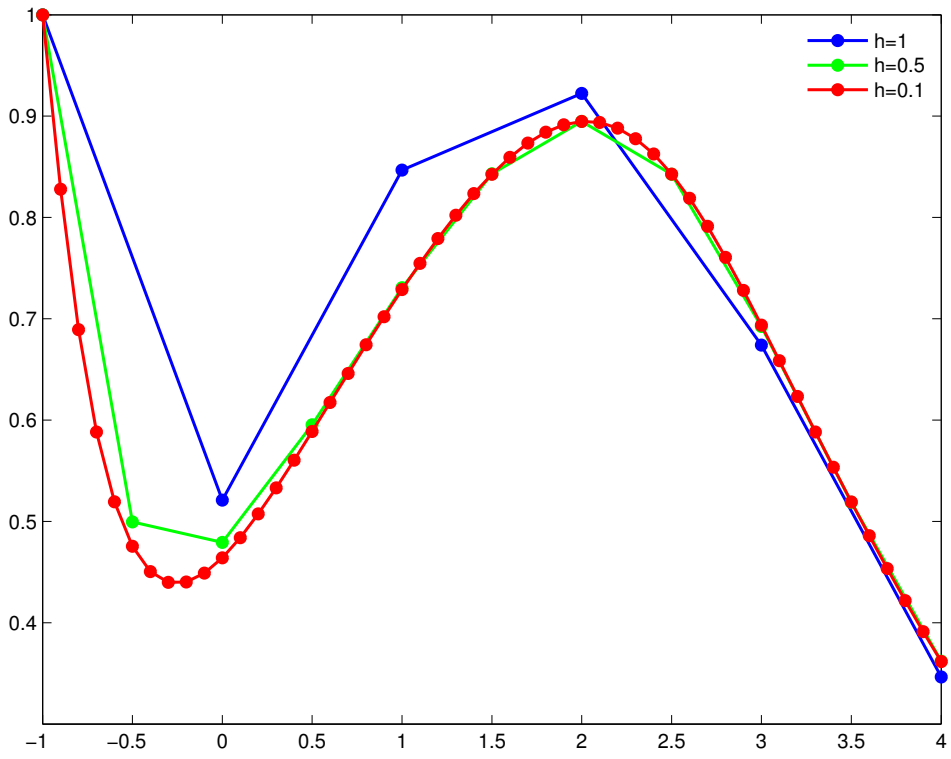
příkaz clear str. 8

```
>> h=0.1, n=(b-a)/h
h =
    0.1000
n =
    50
>> x=a:h:b;
>> 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
```

Vykreslíme numerické řešení pro $h = 1, 0.5, 0.1$.

```
>> x01=x; y01=y;
>> plot(x1,y1,'b.-',x05,y05,'g.-',x01,y01,'r.-')
>> legend('h=1', 'h=0.5','h=0.1')
```

příkazy plot str. 26, legend str. 30

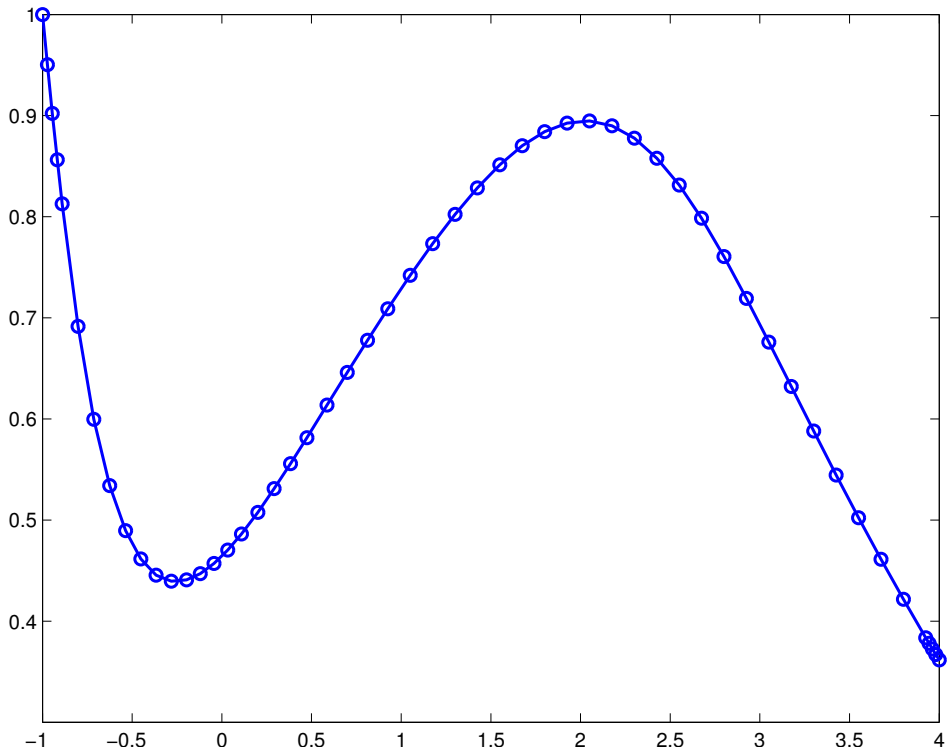


Příkaz `ode45`

Matlab má několik vestavěných funkcí řešících počáteční úlohy pro diferenciální rovnici prvního řádu (nebo soustavu diferenciálních rovnic prvního řádu) metodou Runge-Kutta. Jednou z nich je `ode45`. Použijeme-li příkaz bez výstupních parametrů, zobrazí se graf numerického řešení.

```
>> a=-1, b=4, c=1
a =
     -1
b =
      4
c =
     1
>> f=@(x,y)(sin(x)+cos(3*y))
f =
     @(x,y)(sin(x)+cos(3*y))
>> ode45(f,[a,b],c)
```

příkaz `ode45`, Matlab help



V případě uvedení výstupních parametrů (např. `x`, `y`) se do těchto proměnných uloží hodnoty numerického řešení.

```
>> [x,y]=ode45(f,[a,b],c)
x =
   -1.0000
   -0.9726
     .
```

```
      .
      .
    4.0000
y =
    1.0000
    0.9504
      .
      .
      .
    0.3616
```

příkaz ode45, Matlab help

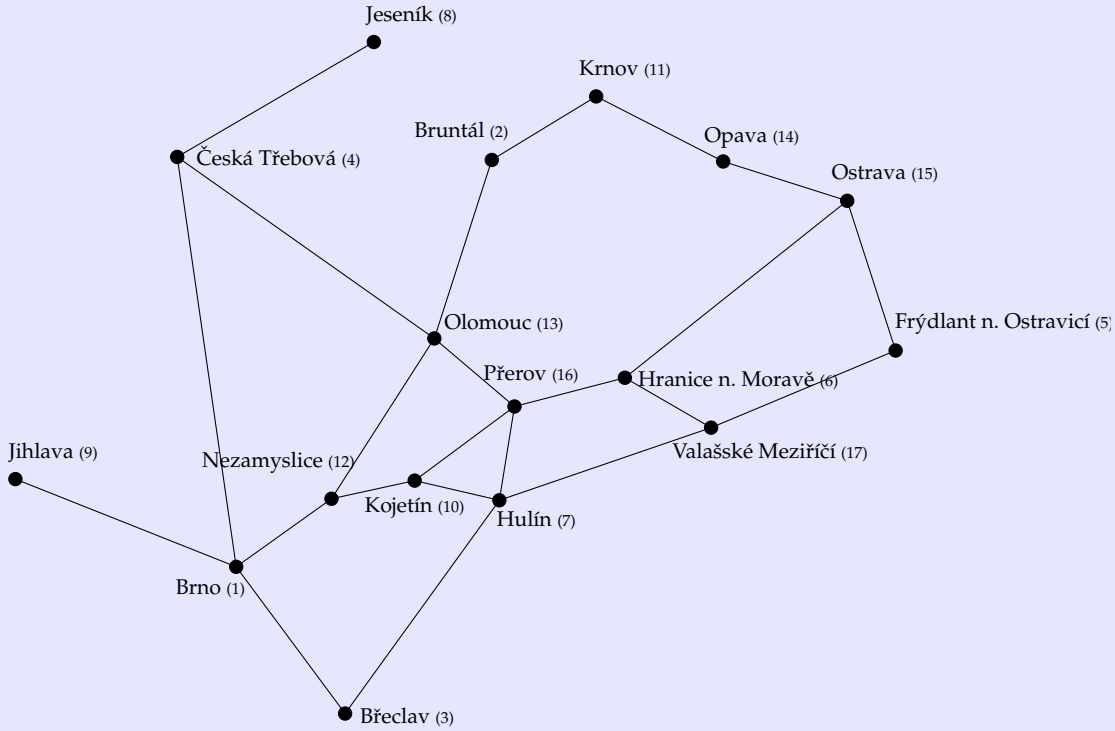
KAPITOLA

12

APLIKOVANÉ ÚLOHY

12.1. Dostupnost v dopravní síti

Na Obrázku 12.1 vidíte schematicky vyobrazenu železniční síť v Moravskoslezském kraji. Rozhodněte, které z měst je nejlépe dostupné železniční dopravou. Seřaďte města podle jejich dopravní dostupnosti.



Obrázek 12.1: Schéma železniční sítě na Moravě

Pro každé město definujeme *index dostupnosti* d_i vzhledem k dané železniční síti, viz Tabulka 12.1.

Index	Město	Index	Město
d_1	Brno	d_{10}	Kojetín
d_2	Bruntál	d_{11}	Krnov
d_3	Břeclav	d_{12}	Nezamyslice
d_4	Česká Třebová	d_{13}	Olomouc
d_5	Frýdlant n. Ostravicí	d_{14}	Opava
d_6	Hranice n. Moravě	d_{15}	Ostrava
d_7	Hulín	d_{16}	Přerov
d_8	Jeseník	d_{17}	Valašské Meziříčí
d_9	Jihlava		

Tabulka 12.1: Indexy dostupnosti měst

Nejdůležitější částí řešení bude vhodně definovat pojem *indexu dostupnosti*. Stanovme si nejprve své požadavky na index.

- Index je nezáporné reálné číslo.
- Index musí zohledňovat počet tratí, které do města vedou. Větší počet příchozích tratí by měl znamenat růst indexu.
- Trať vedoucí z města, které má vysoký index dostupnosti, musí mít vyšší váhu než trať z města, které je málo dostupné.

Náš přístup ilustrujeme na příkladu města Brna. Jeho index dostupnosti d_1 by měl být určen indexy dostupnosti měst, z nichž do Brna vedou tratě, tedy indexy d_3 (Břeclav), d_4 (Česká Třebová), d_9 (Jihlava) a d_{12} (Nezamyslice). Přesněji

$$d_3 + d_4 + d_9 + d_{12} = k \cdot d_1 \quad \text{(Brno)},$$

kde k je nějaká kladná konstanta. Role této konstanty je ryze technická. Bylo by možné se bez ní obejít, ale museli bychom vhodně normovat indexy dostupnosti a postup by se tím stal méně přehledným.
Obdobně zapíšeme podmínky pro indexy dostupnosti ostatních měst:

$$\begin{aligned}
d_{11} + d_{13} &= k \cdot d_2 && \text{(Bruntál)},\\
d_1 + d_7 &= k \cdot d_3 && \text{(Břeclav)},\\
d_1 + d_8 + d_{13} &= k \cdot d_4 && \text{(Česká Třebová)},\\
d_{15} + d_{17} &= k \cdot d_5 && \text{(Frýdlant n. Ostravicí)},\\
d_{15} + d_{16} + d_{17} &= k \cdot d_6 && \text{(Hranice n. Moravě)},\\
d_3 + d_{10} + d_{16} + d_{17} &= k \cdot d_7 && \text{(Hulín)},\\
d_4 &= k \cdot d_8 && \text{(Jeseník)},\\
d_1 &= k \cdot d_9 && \text{(Jihlava)},\\
d_7 + d_{12} + d_{16} &= k \cdot d_{10} && \text{(Kojetín)},\\
d_2 + d_{14} &= k \cdot d_{11} && \text{(Krnov)},\\
d_1 + d_{10} + d_{13} &= k \cdot d_{12} && \text{(Nezamyslice)},\\
d_2 + d_4 + d_{12} + d_{16} &= k \cdot d_{13} && \text{(Olomouc)},\\
d_{11} + d_{15} &= k \cdot d_{14} && \text{(Opava)},\\
d_5 + d_6 + d_{14} &= k \cdot d_{15} && \text{(Ostrava)},\\
d_6 + d_7 + d_{10} + d_{13} &= k \cdot d_{16} && \text{(Přerov)},\\
d_5 + d_6 + d_7 &= k \cdot d_{17} && \text{(Valašské Meziříčí)}.
\end{aligned}$$

Soustavu rovnic zapíšeme maticově jako

$$\begin{pmatrix}
0&0&1&1&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1\\
0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&1&1&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0
\end{pmatrix}
\cdot
\begin{pmatrix} d_1\\d_2\\d_3\\d_4\\d_5\\d_6\\d_7\\d_8\\d_9\\d_{10}\\d_{11}\\d_{12}\\d_{13}\\d_{14}\\d_{15}\\d_{16}\\d_{17}\end{pmatrix}
= k \cdot
\begin{pmatrix} d_1\\d_2\\d_3\\d_4\\d_5\\d_6\\d_7\\d_8\\d_9\\d_{10}\\d_{11}\\d_{12}\\d_{13}\\d_{14}\\d_{15}\\d_{16}\\d_{17}\end{pmatrix}$$

Matici soustavy pojmenujeme jako $\boldsymbol{A}$ a vektor indexů dostupnosti $\boldsymbol{d}$. Soustava podmínek má pak tvar

$$\boldsymbol{A} \cdot \boldsymbol{d} = k \cdot \boldsymbol{d}.$$

Naším cílem je nalézt $k > 0$ a vektor $\boldsymbol{d}$, aby byla soustava splněna. Číslo K ovšem není nic jiného než vlastní číslo matice $\boldsymbol{A}$ a vektor $\boldsymbol{d}$ je vlastní vektor příslušný tomuto vlastnímu číslu. Výpočet vlastních čísel a vektorů provedeme pomocí nástrojů MATLABu.
Definujeme nejprve matici $\boldsymbol{A}$.

```
>> A =[
0 0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0
0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
];
```

Použijeme příkaz `eig` pro výpočet vlastních čísel a vektorů.

```
>> [V, D] = diag(A);
```

Proměnná `D` odkazuje na diagonální matici, na jejíž diagonále se nachází vlastní čísla matice A uspořádané od nejmenšího po největší. Proměnná `V` odkazuje na matici, jejíž sloupce jsou vlastními vektory (první sloupec je vlastním vektorem příslušným nejmenšímu vlastnímu číslu, atd.).
Z teorie (využívá se Perronova-Frobeniova věta a snadný fakt, že matice A je symetrická) lze odvodit, že vlastní vektor příslušný největšímu vlastnímu číslu má buď všechny složky záporné nebo kladné. Tvrzení si ověříme v MATLABu. Největší vlastní číslo je prvek matice `D` v pozici $(17, 17)$ a příslušný vlastní vektor je 17. sloupcem matice `V`.

```
>> D(17,17) % Nejvetsi vlastni cislo
ans =
    3.1472
>> V(:,17) % Prislusny vlastni vektor
ans =
   -0.2505
   -0.1257
   -0.2033
   -0.2071
   -0.1224
   -0.2573
   -0.3893
   -0.0658
   -0.0796
   -0.3534
   -0.0603
   -0.2985
   -0.3354
   -0.0639
   -0.1410
   -0.4243
   -0.2444
```

Připomeňme, že vlastní vektor příslušný vlastnímu číslu není určen jednoznačně. Každý nenulový násobek zobrazeného vektoru je také vlastním vektorem. Zobrazený vektor proto můžeme vynásobit faktorem (-1) a získáme tak hledané kladné indexy dostupnosti. Výsledky zapíšeme do Tabulky 12.2. Města seřazená podle indexu dostupnosti naleznete v Tabulce 12.3.

Index	Město
0.2505	Brno
0.1257	Bruntál
0.2033	Břeclav
0.2071	Česká Třebová
0.1224	Frýdlant n. Ostravicí
0.2573	Hranice n. Moravě
0.3893	Hulín
0.0658	Jeseník
0.0796	Jihlava
0.3534	Kojetín
0.0603	Krnov
0.2985	Nezamyslice
0.3354	Olomouc
0.0639	Opava
0.1410	Ostrava
0.4243	Přerov
0.2444	Valašské Meziříčí

Tabulka 12.2: Spočtené indexy dostupnosti měst

Pořadí	Index	Město
1.	*0.4243*	*Přerov*
2.	0.3893	Hulín
3.	0.3534	Kojetín
4.	0.3354	Olomouc
5.	0.2985	Nezamyslice
6.	0.2573	Hranice n. Moravě
7.	0.2505	Brno
8.	0.2444	Valašské Meziříčí
9.	0.2071	Česká Třebová
10.	0.2033	Břeclav
11.	0.1410	Ostrava
12.	0.1257	Bruntál
13.	0.1224	Frýdlant n. Ostravicí
14.	0.0796	Jihlava
15.	0.0658	Jeseník
16.	0.0639	Opava
17.	0.0603	Krnov

Tabulka 12.3: Města seřazená podle indexů dostupnosti

Dospěli jsme k závěru, že nejdostupnějším městem v rámci moravské železniční sítě je Přerov. Nejméně dostupným městem je naopak Krnov.

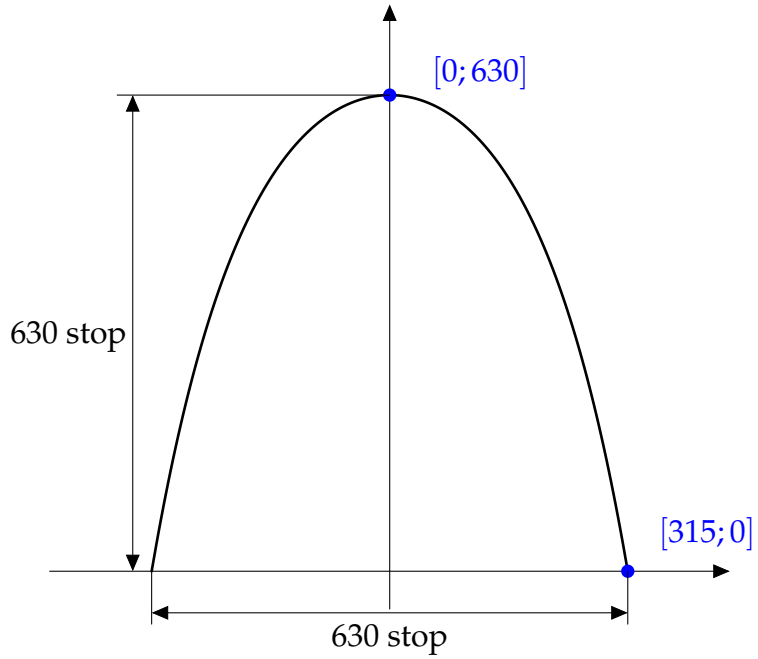
12.2. Známý památník Gateway Arch

Příklad 20: Známý památník Gateway Arch je symbolem Saint Louis (stát Missouri, USA). Památník navrhl finsko-americký architekt Eero Saarinen a německo-americký stavební inženýr Hannskarl Bandel v roce 1947. Stavba začala v roce 1963 a byla dokončena v roce 1965, celkové náklady byly 13 milionů dolarů.
Gateway Arch má tvar řetězovky, která má stejnou šířku při zemi a výšku, a to 630 stop. Památník Gateway Arch má tvar řetězovky

$$y = -a \cdot \cosh\left(\frac{x}{a}\right) + b\,.$$

Jaký je předpis křivky popisující památník?

Osu x umístíme na zem a osa y je totožná s osou symetrie památníku.



Hledanou křivkou je řetězovka

$$y = -a \cdot \cosh\left(\frac{x}{a}\right) + b\,. \tag{12.1}$$

Připomeňme, že funkce hyperbolický kosinus je definována $\cosh(x) = \frac{e^x + e^{-x}}{2}$ a její graf se nazývá řetězovka.
Vzhledem k umístění os, leží na křivce body $[0; 630]$ a $[315; 0]$, které dosadíme do vztahu (12.1) a dostaneme dvě rovnice

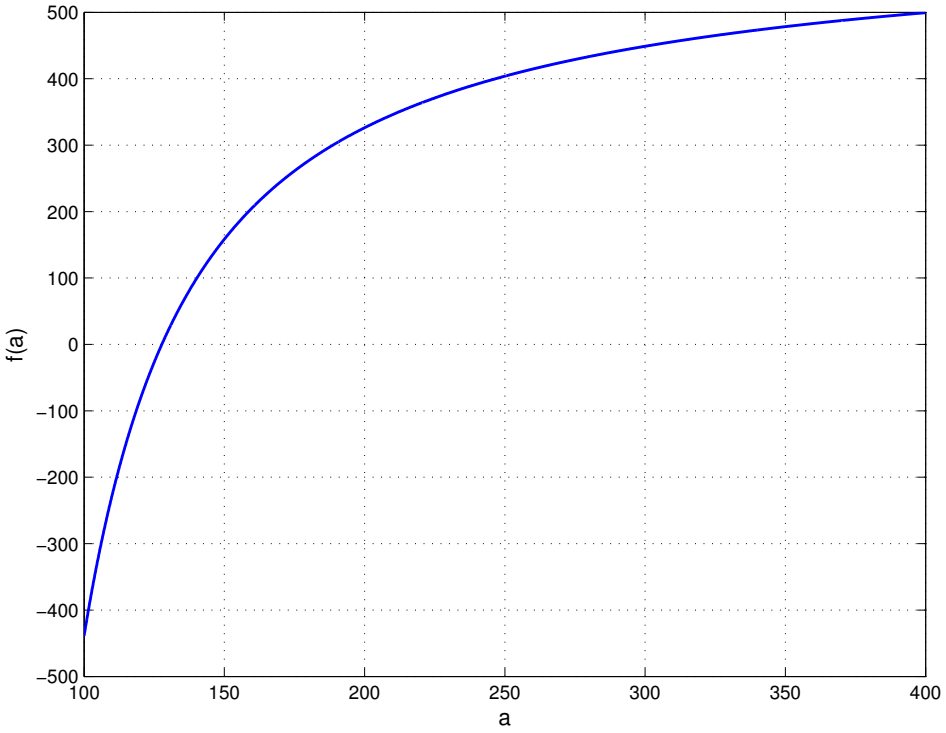
$$630 = -a \cdot \cosh\left(\frac{0}{a}\right) + b\,,$$
$$0 = -a \cdot \cosh\left(\frac{315}{a}\right) + b\,.$$

Z první rovnice vyjádříme $b = a + 630$ a dosadíme do druhé rovnice

$$-a \cdot \cosh\left(\frac{315}{a}\right) + a + 630 = 0\,. \tag{12.2}$$

Tuto rovnici lze vyřešit pomocí příkazu `fzero`, metodou půlení intervalu nebo Newtonovou metodou. Nejprve vykreslíme graf funkce na levé straně rovnice a určíme přibližnou polohu kořene.

```
>> f=@(a)(-a*cosh(315/a)+a+630)
f =
    @(a)(-a*cosh(315/a)+a+630)
>> fplot(f,[100,400]); grid on
```



Vidíme, že průsečík grafu s osou x (tj. kořen rovnice) leží mezi hodnotami 100 a 150. Najdeme řešení a pomocí příkazu `fzero` a dopočítáme hodnotu b.

```
>> a=fzero(f,150)
a =
   127.7115
>> b=a+630
b =
   757.7115
```

Řešení je $a = 127.7$, $b = 757.7$ a řetězovka má předpis

$$y = -127.7 \cdot \cosh\left(\frac{x}{127.7}\right) + 757.7\,,$$

kde $x \in \langle -315\,\text{stop}; 315\,\text{stop}\rangle$.

12.3. Likvidus pro slitinu cínu a hliníku

Čistý kov krystalizuje při konstantní teplotě. Slitina krystalizuje při různých teplotách v závislosti na poměru prvků. Pro dané složení slitiny se teplota počátku krystalizace slitiny nazývá teplota likvidu, teplota konce krystalizace se nazývá teplota solidu. Mezi těmito teplotami existuje tavenina s krystaly.

Je dána slitina cínu s příměsí hliníku (předpokládáme, že hliníku je ve slitině maximálně 40 At%). Likvidus je aproximován polynomem čtvrtého řádu. Přičemž víme, že cín má teplotu tání 168.5 °C. Ostatní koeficienty určíme z naměřených dat.

x_i [At% Al]	2.3	5.7	8.1	9.5	12.3	17.3	22.5	31.3
y_i [°C]	233.08	308.19	349.05	369	401.68	441.75	467.26	492.37

Pro jaké složení slitiny (s přesností na dvě desetinná místa) nastává krystalizace při teplotě 350 °C

Polynom čtvrtého řádu má tvar

$$t(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0,$$

kde x odpovídá At% příměsi hliníku ve slitině a $t(x)$ je teplota likvidu.
Teplota tání slitiny s 0 At% hliníku je 168.5, tedy platí $t(0) = 168.5$, po dosazení do předpisu $t(x)$ dostaneme hodnotu koeficientu $a_0 = 168.5$, tedy

$$t(x) = a_4x^4 + a_3x^3 + a_2x^2a_1x + 168.5.$$

Koeficienty polynomu a_4, a_3, a_2, a_1 najdeme metodou nejmenších čtverců. Hledáme minimum funkce

$$\Psi(a_4, a_3, a_2, a_1) = \sum_{i=1}^{8} (t(x_i) - y_i)^2 = \sum_{i=1}^{8} \left(a_4x_i^4 + a_3x_i^3 + a_2x_i^2 + a_1x_i + 168.5 - y_i\right)^2,$$

které najdeme jako řešení soustavy rovnic

$$\frac{\partial\Psi(a_4, a_3, a_2, a_1)}{\partial a_j} = 0 \qquad j = 1, 2, 3, 4.$$

Ukážeme výpočet jedné z parciálních derivací.

$$\frac{\partial\Psi(a_4, a_3, a_2, a_1)}{\partial a_1} = 0$$

$$\frac{\partial}{\partial a_1} \sum_{i=1}^{8} \left(a_4x_i^4 + a_3x_i^3 + a_2x_i^2 + a_1x_i + 168.5 - y_i\right)^2 = 0$$

$$\sum_{i=1}^{8} 2\left(a_4x_i^4 + a_3x_i^3 + a_2x_i^2 + a_1x_i + 168.5 - y_i\right) x_i = 0$$

$$\sum_{i=1}^{8}\left(a_4x_i^5+a_3x_i^4+a_2x_i^3+a_1x_i^2+168.5x_i-y_ix_i\right)=0$$

$$a_4\sum_{i=1}^{8}x_i^5+a_3\sum_{i=1}^{8}x_i^4+a_2\sum_{i=1}^{8}x_i^3+a_1\sum_{i=1}^{8}x_i^2=\sum_{i=1}^{8}\left(y_i-168.5\right)x_i$$

Obdobně spočítáme i ostatní parciální derivace a dostaneme soustavu lineárních rovnic

$$\begin{aligned}
a_4\sum_{i=1}^{8}x_i^8+a_3\sum_{i=1}^{8}x_i^7+a_2\sum_{i=1}^{8}x_i^6+a_1\sum_{i=1}^{8}x_i^5&=\sum_{i=1}^{8}x_i^4(y_i-168.5),\\
a_4\sum_{i=1}^{8}x_i^7+a_3\sum_{i=1}^{8}x_i^6+a_2\sum_{i=1}^{8}x_i^5+a_1\sum_{i=1}^{8}x_i^4&=\sum_{i=1}^{8}x_i^3(y_i-168.5),\\
a_4\sum_{i=1}^{8}x_i^6+a_3\sum_{i=1}^{8}x_i^5+a_2\sum_{i=1}^{8}x_i^4+a_1\sum_{i=1}^{8}x_i^3&=\sum_{i=1}^{8}x_i^2(y_i-168.5),\\
a_4\sum_{i=1}^{8}x_i^5+a_3\sum_{i=1}^{8}x_i^4+a_2\sum_{i=1}^{8}x_i^3+a_1\sum_{i=1}^{8}x_i^2&=\sum_{i=1}^{8}x_i(y_i-168.5).
\end{aligned}$$

Soustavu v MATLABu vyřešíme pomoci zpětného lomítka.

```
>> x=[2.3 5.7 8.1 9.5  12.3 17.3 22.5 31.3 ];
>> y=[233.08  308.19  349.05 369  401.68  441.75  467.26
   492.37];
>> A=[sum(x.^8) sum(x.^7) sum(x.^6) sum(x.^5);
      sum(x.^7) sum(x.^6) sum(x.^5) sum(x.^4);
      sum(x.^6) sum(x.^5) sum(x.^4) sum(x.^3);
      sum(x.^5) sum(x.^4) sum(x.^3) sum(x.^2)]
A =
  1.0e+011 *
    9.9552    0.3287    0.0110    0.0004
    0.3287    0.0110    0.0004    0.0000
    0.0110    0.0004    0.0000    0.0000
    0.0004    0.0000    0.0000    0.0000
>> b=[sum(x.^4.*(y-168.5)); sum(x.^3.*(y-168.5)); sum(x.^2.*(y
   -168.5)); sum(x.*(y-168.5))]
b =
  1.0e+008 *
    4.1979
    0.1548
    0.0062
    0.0003
>> a=A\b
a =
   -0.0002
    0.0250
   -1.2402
   30.8008
```

Rovnice likvidu má následující tvar

$$t(x) = -0.0002x^4 + 0.025x^3 - 1.24x^2 + 30.8x + 168.5.$$

Do rovnice pro likvidus dosadíme teplotu 350

$$350 = -0.0002x^4 + 0.025x^3 - 1.24x^2 + 30.8x + 168.5$$

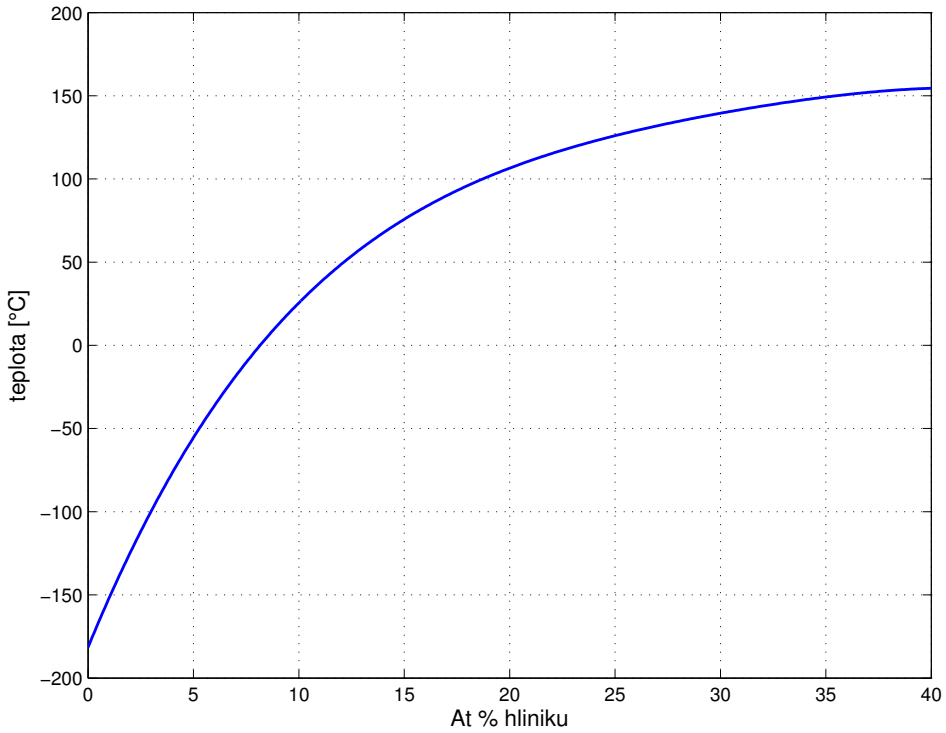
a tuto rovnici upravíme

$$-0.0002x^4 + 0.025x^3 - 1.24x^2 + 30.8x - 181.5 = 0\,.$$

Nejprve vykreslíme graf na levé straně rovnice a určíme přibližnou polohu kořene.

```
>> f=@(x)(-0.0002*x^4+0.025*x^3-1.24*x^2+30.8*x-181.5)
f =
    @(x)(-0.0002*x^4+0.025*x^3-1.24*x^2+30.8*x-181.5)
>> fplot(f,[0,40]); grid on
```

Vidíme, že kořen leží mezi hodnotami 5 a 10.



Pomocí příkazu `fzero` najdeme kořen této rovnice.

```
>> fzero(f,10)
ans =
    8.1627
```

Krystalizace slitiny cínu a hliníku při teplotě 350 °C nastává pro příměs 8.16 At % hliníku.

12.4. Kinetika enzymové aktivity I

Michaelisova-Mentenova rovnice popisující kinetiku enzymové aktivity má tvar

$$v(x) = \frac{a_1 x}{a_2 + x},$$

přičemž $v(x)$ je rychlost přeměny substrátu, zapříčiněné enzymovou katalýzou, v závislosti na koncentraci substrátu x. Experimentálně jsme získali rychlosti pro různé koncentrace substrátu:

x_i [10^{-4} mol · dm^{-3}]	2.267	2.703	3.575	4.098	4.098	3.974	4.211
v_i [10^{-8} mol · dm^{-3} · s^{-1}]	1.220	2.451	4.897	9.780	14.688	19.576	24.483

Michaelisovu-Mentenovu rovnici nejprve linearizujte a pak použijte metodu nejmenších čtverců k nalezení koeficientů a_1, a_2.

Rovnici

$$v = \frac{a_1 x}{a_2 + x}$$

můžeme přepsat do ekvivalentního tvaru

$$\frac{1}{v} = \frac{a_2 + x}{a_1 x},$$
$$\frac{1}{v} = \frac{a_2}{a_1} \cdot \frac{1}{x} + \frac{1}{a_1}.$$

Definujeme-li nové proměnné $V := \frac{1}{v}$, $X := \frac{1}{x}$, $A := \frac{1}{a_1}$ a $B := \frac{a_2}{a_1}$, transformuje se rovnice na lineární rovnici

$$V = BX + A.$$

Uvedenému procesu se říká linearizace. Tabulka dat se provedením substitucí také změní.

X_i	0.441	0.370	0.280	0.244	0.244	0.252	0.238
V_i	0.820	0.408	0.204	0.102	0.068	0.051	0.041

Nyní můžeme použít metodu nejmenších čtverců na modifikovanou úlohu. Budeme tedy minimalizovat funkci

$$\Psi(A, B) = \sum_{i=0}^{6} \Big[(BX_i + A)^2 - V_i \Big]^2.$$

Spočteme parciální derivace a položíme je rovny nule:

$$\frac{\partial \Psi}{\partial A} = 2 \sum_{i=0}^{6} (BX_i + A - V_i) = 0,$$
$$\frac{\partial \Psi}{\partial B} = 2 \sum_{i=0}^{6} (BX_i + A - V_i) X_i = 0.$$

Po úpravě obdržíme soustavu normálních rovnic:

$$7A + \left(\sum_{i=0}^{6} X_i\right) B = \sum_{i=0}^{6} V_i\,,$$
$$\left(\sum_{i=0}^{6} X_i\right) A + \left(\sum_{i=0}^{6} X_i^2\right) B = \sum_{i=0}^{6} X_i V_i\,,$$

což maticově zapsáno je

$$\begin{pmatrix} 7 & \sum_{i=0}^{6} X_i \\ \sum_{i=0}^{6} X_i & \sum_{i=0}^{6} X_i^2 \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{6} V_i \\ \sum_{i=0}^{6} X_i V_i \end{pmatrix}.$$

Soustavu snadno vyřešíme v MATLABu.

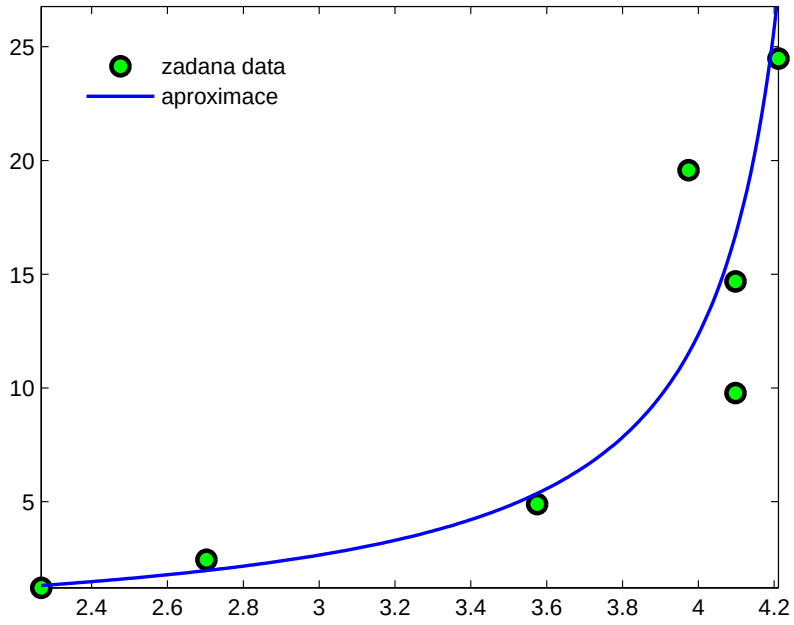
```
>> x = [ 2.267   2.703   3.575   4.098    4.098     3.974    4.211];
>> v = [1.220   2.451   4.897   9.780   14.688    19.576   24.483];
>> X = 1./x;
>> V = 1./v;
>> [7 sum(X); sum(X) sum(X.^2)]\[sum(V);sum(X.*V)]
ans =
   -0.8054
    3.5455
```

To znamená, že $A = -0.8054$ a $B = 3.5455$. Vrátíme-li se k původním neznámým, dostaneme

$$a_1 = \frac{1}{A} = -1.2416,$$
$$a_2 = \frac{B}{A} = -4.4022.$$

Získané hodnoty ověříme v MATLABu graficky, viz Obrázek 12.2.

```
>> a1 = 1/-0.8054
>> a2 = 3.5455/-0.8054
>> f = @(x)a1*x./(a2+x);
>> x1 = 2.2:0.01:4.3;
>> hold on
>> plot(x,v,'go',x1,f(x1),'b-')
>> legend('zadana data','aproximace')
```



Obrázek 12.2: Porovnání dat a získané aproximace Michaelisovy-Mentenovy rovnice

12.5. Kinetika enzymové aktivity II

Použijte data z předchozí úlohy a metodou nejmenších čtverců nalezněte hodnoty koeficientů a_1, a_2, aniž byste původní úlohu linearizovali. Výsledky této a předchozí úlohy porovnejte.

Připomeňme, že naším úkolem je nalézt nejlepší aproximaci (ve smyslu nejmenších čtverců) dat

x_i $[10^{-4}\ \mathrm{mol}\cdot\mathrm{dm}^{-3}]$	2.267	2.703	3.575	4.098	4.098	3.974	4.211
v_i $[10^{-8}\ \mathrm{mol}\cdot\mathrm{dm}^{-3}\cdot\mathrm{s}^{-1}]$	1.220	2.451	4.897	9.780	14.688	19.576	24.483

funkcí ve tvaru

$$v(x) = \frac{a_1 x}{a_2 + x} \quad ,$$

kde a, b jsou vhodné reálné konstanty.
Budeme tedy hledat minimum funkce

$$\Psi(a,b) = \sum_{i=0}^{6} \left(\frac{a_1 x_i}{a_2 + x_i} - v_i \right)^2 .$$

Nejprve spočteme parciální derivace funkce $\Psi(a,b)$ podle a a b a položíme je rovny nule:

$$0 = \frac{\partial E}{\partial a} = 2\sum_{i=0}^{6}\left(\frac{a_1 x_i}{a_2+x_i} - v_i\right)\frac{x_i}{a_2+x_i} = 2\sum_{i=0}^{6}\left(\frac{a_1 x_i^2}{(a_2+x_i)^2} - \frac{x_i v_i}{a_2+x_i}\right),$$
$$0 = \frac{\partial E}{\partial a_2} = 2\sum_{i=0}^{6}\left(\frac{a_1 x_i}{a_2+x_i} - v_i\right)\frac{-a_1 x_i}{(a_2+x_i)^2} = -2\sum_{i=0}^{6}\left(\frac{a_1^2 x_i^2}{(a_2+x_i)^3} - \frac{a_1 x_i v_i}{(a_2+x_i)^2}\right).$$

Rovnice ještě vydělíme konstantou 2 (resp. -2) a získáme tak soustavu dvou nelineárních rovnic. Funkce na levých stranách obou rovnic označíme po řadě jako $f_1(a_1, a_2)$ a $f_2(a_1, a_2)$:

$$f_1(a_1, a_2) = \sum_{i=0}^{6}\left(\frac{a_1 x_i^2}{(a_2+x_i)^2} - \frac{x_i v_i}{a_2+x_i}\right) = 0,$$
$$f_2(a_1, a_2) = \sum_{i=0}^{6}\left(\frac{a_1^2 x_i^2}{(a_2+x_i)^3} - \frac{a_1 x_i v_i}{(a_2+x_i)^2}\right) = 0.$$

K jejímu řešení využijeme Newtonovy metody pro soustavy rovnic. K tomu účelu potřebujeme spočítat Jakobiho matici funkce $f(a_1, a_2) = (f_1(a_1, a_2), f_2(a_1, a_2))^\top$, která je definována jako

$$\boldsymbol{J}_f(a_1, a_2) = \begin{pmatrix} \frac{\partial f_1}{\partial a_1}(a_1, a_2) & \frac{\partial f_1}{\partial a_2}(a_1, a_2) \\ \frac{\partial f_2}{\partial a_1}(a_1, a_2) & \frac{\partial f_2}{\partial a_2}(a_1, a_2) \end{pmatrix}.$$

Po spočtení derivací dostáváme

$$\boldsymbol{J}_f(a_1, a_2) = \begin{pmatrix} \sum_{i=0}^{6}\frac{x_i^2}{(a_2+x_i)^2} & \sum_{i=0}^{6}\frac{-2a_1 x_i^2}{(a_2+x_i)^3} + \frac{x_i v_i}{(a_2+x_i)^2} \\ \sum_{i=0}^{6}\frac{2a_1 x_i^2}{(a_2+x_i)^3} - \frac{x_i v_i}{(a_2+x_i)^2} & \sum_{i=0}^{6}\frac{-3a_1^2 x_i^2}{(a_2+x_i)^4} + \frac{2a_1 x_i v_i}{(a_2+x_i)^3} \end{pmatrix}.$$

Výpočet provedem v MATLABu. Nejprve definujeme vektory x a v, pak funkce f_1 a f_2. Všimněte si, že vstupní proměnnou obou funkcí je vektor `A`.

```
>> x = [2.267   2.703   3.575   4.098    4.098     3.974    4.211];
>> v = [1.220   2.451   4.897   9.780   14.688    19.576   24.483];
>> f1 = @(A)sum(A(1)*x.^2./(A(2)+x).^2-x.*v./(A(2)+x));
>> f2 = @(A)sum(A(1).^2*x.^2./(A(2)+x).^3-A(1)*x.*v./(A(2)+x)
   .^2);
```

Nyní je třeba definovat Jakobiho matici.

```
>> Jf = @(A)[sum(x.^2./(A(2)+x).^2), sum(-2*A(1)*x.^2./(A(2)+x)
   .^3+x.*v./(A(2)+x).^2); sum(2*A(1)*x.^2./(A(2)+x).^3-x.*v./(A
   (2)+x).^2), sum(-3*A(1)^2*x.^2./(A(2)+x).^4+2*A(1)*x.*v./(A
   (2)+x).^3)];
```

Jako počáteční aproximaci zvolíme výsledek předchozí úlohy, tj.

$$a_1 = -1.2416,$$
$$a_2 = -4.4022.$$

V MATLABu definujeme počáteční aproximaci a spočteme první krok Newtonovy metody.

```
>> X = [-1.2416;-4.4022];
>> Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.2411
   -4.4224
```

Nově získanou hodnotu `Xnovy` uložíme do `X` a postup zopakujeme.

```
>> X = Xnovy;
>> Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.2525
   -4.4244
```

Předchozí řádky budeme opakovat tak dlouho, dokud se číslice na vektoru `Xnovy` budou měnit.

```
>> X = Xnovy; Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.4524
   -4.4706
>> X = Xnovy; Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.5891
   -4.5034
>> X = Xnovy; Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.6382
   -4.5155
>> X = Xnovy; Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.6433
   -4.5168
>> X = Xnovy; Xnovy = X-inv(Jf(X))*[f1(X);f2(X)]
Xnovy =
   -1.6433
   -4.5168
```

Po několika málo iteracích jsme dospěli k výsledku

$$a_1 = -1.6433,$$
$$a_2 = -4.5168,$$

který je poměrně značně odlišný od výsledku předchozí úlohy, kde jsme se uchýlili k linearizaci.

Nyní v MATLABu definujeme aproximace získané metodou nejmenších čtverců z linearizované a původní úlohy.

```
>> a1lin = -1.2416;
>> a2lin = -4.4022;
>> flin = @(x)a1lin.*x./(a2lin+x);
>> a1 = -1.6433;
>> a2 = -4.5168;
>> f = @(x)a1.*x./(a2+x);
```

Dále porovnáme hodnoty obou aproximací s naměřenými daty.

```
>> [x; v; flin(x); (flin(x)-v).^2; f(x); (f(x)-v).^2]'
ans =
    2.2670    1.2200    1.3182    0.0097    1.6559    0.1900
    2.7030    2.4510    1.9751    0.2265    2.4489    0.0000
    3.5750    4.8970    5.3660    0.2199    6.2378    1.7979
    4.0980    9.7800   16.7261   48.2482   16.0799   39.6882
    4.0980   14.6880   16.7261    4.1538   16.0799    1.9373
    3.9740   19.5760   11.5229   64.8519   12.0311   56.9257
    4.2110   24.4830   27.3451    8.1915   22.6290    3.4375
```

Na závěr porovnáme kvadratické chyby obou aproximací.

```
>> sum((flin(x)-v).^2)
ans =
  125.9015
>> sum((f(x)-v).^2)
ans =
  103.9764
```

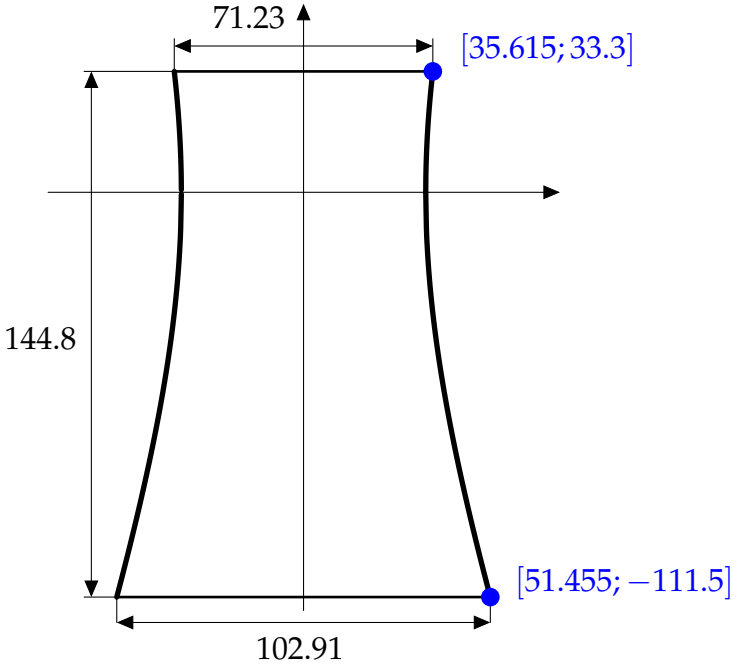
Linearizace úlohu do velké míry zjednodušila, ale z předchozích řádků je patrné, že to bylo na úkor přesnosti. Všechny spočtené hodnoty naleznete v Tabulce 12.4.

i	x_i	v_i	$\texttt{flin}(x_i)$	$(\texttt{flin}(x_i)-v_i)^2$	$\texttt{f}(x_i)$	$(\texttt{f}(x_i)-v_i)^2$
0	2.2670	1.2200	1.3182	0.0097	1.6559	0.1900
2	2.7030	2.4510	1.9751	0.2265	2.4489	0.0000
3	3.5750	4.8970	5.3660	0.2199	6.2378	1.7979
4	4.0980	9.7800	16.7261	48.2482	16.0799	39.6882
5	4.0980	14.6880	16.7261	4.1538	16.0799	1.9373
6	3.9740	19.5760	11.5229	64.8519	12.0311	56.9257
7	4.2110	24.4830	27.3451	8.1915	22.6290	3.4375
		$\sum_i(\texttt{flin}(\texttt{x}_\texttt{i})-\texttt{v}_\texttt{i})^2=$		125.9015	$\sum_i(\texttt{f}(\texttt{x}_\texttt{i})-\texttt{v}_\texttt{i})^2=$	103.9764

Tabulka 12.4: Porovnání aproximací

12.6. Objem a povrch chladící věže

Chladící věže jsou nezbytnou součástí elektráren. Slouží ke chlazení vody, která se používá při výrobním procesu elektrické energie ve strojovně. Nová věž s přirozeným tahem pro elektrárnu Ledvice je po chladících věžích na jaderné elektrárně Temelín druhá nejvyšší v ČR.
Je přesně 144.80 m vysoká a její průměr na patě bude 102.91 m a v koruně 71.23 m. Nejužší místo věže je ve výšce 111.5 m.
Chladící věž má tvar rotační hyperboloidu.
Jaký je předpis křivky, popisující tvar věže? Jaký je objem a povrch věže?



Osu x umístíme do nejužšího místa věže a osa y je totožná s osou symetrie věže. Hledanou křivkou je hyperbola se středem v počátku souřadnic

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1\,. \tag{12.3}$$

Na hyperbole leží body $[35.615;\ 33.3]$ a $[51.455;\ -111.5]$, které dosadíme do rovnice (12.3) a dostaneme dvě rovnice

$$\frac{35.615^2}{a^2} - \frac{33.3^2}{b^2} = 1,$$
$$\frac{51.455^2}{a^2} - \frac{(-111.5)^2}{b^2} = 1.$$

První rovnici vydělíme 35.615^2, druhou 51.455^2 a vyjádříme z obou $\frac{1}{a^2}$.

$$\frac{1}{a^2} - \frac{33.3^2}{35.615^2 b^2} = \frac{1}{35.615^2} \quad \Rightarrow \quad \frac{1}{a^2} = \frac{1}{35.615^2} + \frac{33.3^2}{35.615^2 b^2}$$
$$\frac{1}{a^2} - \frac{(-111.5)^2}{51.455^2 b^2} = \frac{1}{51.455^2} \quad \Rightarrow \quad \frac{1}{a^2} = \frac{(-111.5)^2}{51.455^2 b^2} + \frac{1}{51.455^2}$$

Členy $\frac{1}{a^2}$ dáme do rovnosti a vyjádříme b.

$$\frac{1}{35.615^2}+\frac{33.3^2}{35.615^2b^2}=\frac{1}{51.455^2}+\frac{(-111.5)^2}{51.455^2b^2}$$
$$51.455^2b^2+33.3^2\cdot 51.455^2=35.615^2b^2+(-111.5)^2\cdot 35.615^2$$
$$b=\sqrt{\frac{(-111.5)^2\cdot 35.615^2-33.3^2\cdot 51.455^2}{51.455^2-35.615^2}}$$

```
>> b=sqrt((111.5^2*35.615^2-33.3^2*51.455^2)/(51.455^2-35.615^2)
   )
b =
   96.4630
```

Dopočítáme hodnotu a.

$$\frac{1}{a^2}=\frac{1}{35.615^2}+\frac{33.3^2}{35.615^2b^2}$$
$$\frac{1}{a^2}=\frac{b^2+33.3^2}{35.615^2b^2}$$
$$a=\sqrt{\frac{35.615^2b^2}{b^2+33.3^2}}$$

```
>> a=sqrt(35.615^2*b^2/(b^2+33.3^2))
a =
   33.6654
```

Řešení zaokrouhlíme na dvě desetinná místa a výsledná křivka je dána předpisem

$$\frac{x^2}{33.66^2}-\frac{y^2}{96.46^2}=1,$$

kde $y\in\langle -111.5;\ 33.3\rangle$.

Pro výpočet objemu a povrchu rotačního tělesa je potřeba uvědomit si, že hyperboloid vznikne rotací části hyperboly kolem osy y a proto explicitně vyjádříme x z předpisu hyperboly

$$x=33.66\sqrt{1+\frac{y^2}{96.46^2}}. \tag{12.4}$$

Objem rotačního tělesa se spočítá podle vzorce

$$V=\pi\int_a^b x^2\,dy=\pi\int_{-111.5}^{33.3}33.66^2\left(1+\frac{y^2}{96.46^2}\right)\,dy.$$

```
>> format long
>> f=@(y)33.66^2*(1+y^2/96.46^2)
f =
  @(y) 33.66 ^ 2 * (1 + y ^ 2 / 96.46 ^ 2)
```

```
>> V=pi*quad(f,-111.5,33.3)
V =
    696872.492333375
```

Pro výpočet povrchu je potřeba spočítat derivaci funkce (12.4)

$$x' = \frac{33.66}{2} \frac{1}{\sqrt{1+\frac{y^2}{96.46^2}}} \frac{2y}{96.46^2} = \frac{33.66\,y}{96.46\sqrt{96.46^2+y^2}}.$$

Povrch rotačního tělesa se spočítá podle vzorce

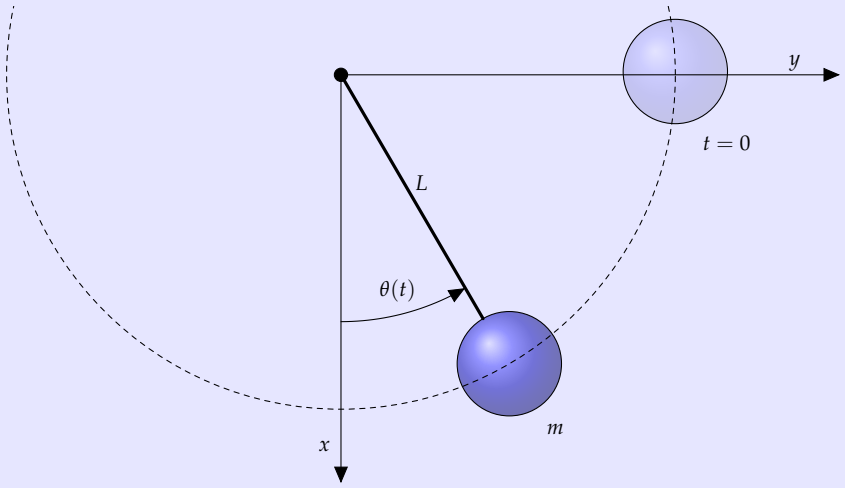
$$\begin{aligned} P =& 2\pi \int_a^b x\sqrt{1+(x')^2}\,dy = \\ =& 2\pi \int_{-111.5}^{33.3} 33.66\sqrt{1+\frac{y^2}{96.46^2}}\sqrt{1+\left(\frac{33.66\,y}{96.46\sqrt{96.46^2+y^2}}\right)^2}\,dy. \end{aligned}$$

```
>> f=@(y)33.66*sqrt(1+y^2/96.46^2)*sqrt(1+(33.66*y/(96.46*sqrt
   (96.16^2+y^2)))^2)
f =
    @(y) 33.66 * sqrt (1 + y ^ 2 / 96.46 ^ 2) * sqrt (1 + (33.66
    * y / (96.46 * sqrt (96.16 ^ 2 + y ^ 2))) ^ 2)
>> P=2*pi*quad(f,-111.5,33.3)
ans =
    35770.2170110041
```

Objem chladící věže je 696872 m^3 a povrch je 35770 m^2.

12.7. Tlumené kyvadlo

Mějme kyvadlo tvořené koulí hmotosti $m = 0.3$ kg zavěšenou na nehmotné tyči, která má délku $L = 1.3$ m. Koeficient tlumení nechť má hodnotu $c = 0.27\,\mathrm{N}\cdot\mathrm{s}\cdot\mathrm{m}^{-1}$. Označme jako $\theta(t)$ úhel kyvadla v čase t (tj. úhel sevřený tyčí kyvadla s vertikální osou). V čase $t = 0$ s je kyvadlo vypuštěno z polohy $\theta(0) = 90°$ s nulovou počáteční rychlostí, tj. $\frac{\mathrm{d}\theta}{\mathrm{d}t}(0) = 0\,\mathrm{rad}\cdot\mathrm{s}^{-1}$. Určete úhel kyvadla $\theta(t)$ po prvních 15 sekundách po vypuštění. Popis situace naleznete na Obrázku 12.3.

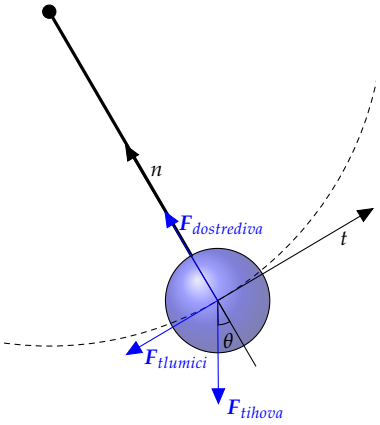


Obrázek 12.3: Popis kyvadla

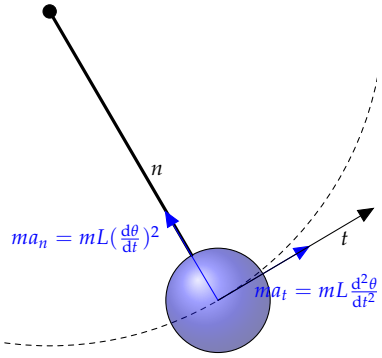
Podle Newtonova druhého pohybového zákona platí, že součet sil působících na těleso v určitém směru je roven součinu hmotnosti tělesa a jemu v tomto směru udělenému zrychlení. Připomeňme vztah, podle něhož je rychlost při pohybu po kružnici rovna součinu úhlové rychlosti a délky, tj. $v = \frac{\mathrm{d}\theta}{\mathrm{d}t}\cdot L$. Na zavěšenou kouli působí několik sil:

1. Dostředivá síla: Směřuje k bodu závěsu a její velikost je dána $F_{dostrediva} = mL\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2$.
2. Tíhová síla: Směřuje dolů rovnoběžně s osou x a její velikost je $F_{tihova} = mg$.
3. Tlumící síla: Působí proti směru pohybu a její velikost je $F_{tlumici} = cL\left|\frac{\mathrm{d}\theta}{\mathrm{d}t}\right|$. Velikost tlumící síly je totiž přímo úměrná rychlosti tělesa, tj. $L\frac{\mathrm{d}\theta}{\mathrm{d}t}$, a koeficientu tlumení c.

Síly působící na kouli kyvadla jsou znázorněny na Obrázku 12.4 a udělená zrychlení na Obrázku 12.4. Zakreslena je situace, kdy se koule kyvadla pohybuje prvním kvadrantem zleva doprava, tj. $\theta(t)$ je kladné a roste s časem. V jiných situacích by byl silový diagram obdobný.
Funkce úhlu $\theta(t)$, případně úhlové rychlosti, může nabývat také záporných hodnost. Záporné znaménko znamená, že je vektor v opačném směru.



Obrázek 12.4: Silový diagram



Obrázek 12.5: Tečná a normálová složka zrychlení

Z druhého pohybového zákona (sledujeme pouze tečné složky tíhové a tlumící síly a také zrychlení uvažujeme pouze ve směru tečny ke kružnici, po níž se koule kyvadla pohybuje) plyne, že

$$m\boldsymbol{a}_t = \boldsymbol{F}^t_{tihova} + \boldsymbol{F}_{tlumici}, \quad \text{neboli}$$
$$mL\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -cL\frac{\mathrm{d}\theta}{\mathrm{d}t} - mg\sin\theta.$$

Všimněte si záporných znamének před formulemi $cL\frac{\mathrm{d}\theta}{\mathrm{d}t}$ a $mg\sin\theta$. Ta nejsnáze vysvětlíme v situaci, kdy se koule kyvadla pohybuje zleva doprava, a $\theta(t)$ tedy roste a nachází se napravo od osy x, tedy $\theta(t) > 0$. Pak je $\frac{\mathrm{d}\theta}{\mathrm{d}t} > 0$
Po úpravě ekvivalentně

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{c}{m}\frac{\mathrm{d}\theta}{\mathrm{d}t} - \frac{g}{L}\sin\theta.$$

Jde o obyčejnou diferenciální rovnici druhého řádu, kterou převedeme na soustavu diferenciálních rovnic prvního řádu a vyřešíme. Zavedeme-li novou závisle proměnnou $v = \frac{\mathrm{d}\theta}{\mathrm{d}t}$ (rychlost zavěšené koule), platí pak $\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$. Diferenciální rovnici popisující pohyb kyvadla lze proto ekvivalentně zapsat jako soustavu diferenciálních rovnic

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = v, \quad \text{s počáteční podmínkou } \theta(0) = \tfrac{\pi}{2},$$
$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{c}{m}\frac{\mathrm{d}\theta}{\mathrm{d}t} - \frac{g}{L}\sin\theta, \quad \text{s počáteční podmínkou } v(0) = 0.$$

Pro nalezení řešení použijeme Eulerovu metodu modifikovanou pro soustavy diferenciálních rovnic prvního řádu. Její myšlenka je obdobná jako u obyčejné Eulerovy metody. Nechť je dána soustava dvou diferenciálních rovnic prvního řádu

$$\begin{aligned}\frac{\mathrm{d}y}{\mathrm{d}x} &= f_1(x,y,z),\\ \frac{\mathrm{d}z}{\mathrm{d}x} &= f_2(x,y,z),\end{aligned} \tag{12.5}$$

na intervalu $\langle a, b\rangle$ s počátečními podmínkami $y(a) = y_0$ a $z(a) = z_0$. Pak je Eulerova metoda s krokem h dána rekurentními vztahy

$$\begin{aligned} x_0 &= a, \\ x_{i+1} &= x_i + h, \\ y_{i+1} &= y_i + f_1(x_i, y_i, z_i)h, \\ z_{i+1} &= z_i + f_2(x_i, y_i, z_i)h. \end{aligned}$$

Výpočet provedeme s krokem $h = 0.1$ na intervalu $\langle 0, 15\rangle$. V MATLABu nejprve definujeme základní konstanty a hodnoty ze zadání. Funkci θ přitom označíme jako `th` a funkci v jako `v`.

```
>> m = 0.3; c = 0.27; L = 1.3; g = 9.81;
>> h = 0.1; t(1) = 0; th(1) = pi/2; v(1) = 0;
>> n = (15-0)/h;
```

Dále je třeba definovat funkce pravých stran obou diferenciálních rovnic (12.5). Pojmenujeme je po řadě jako `dthdt` a `dvdt`.

```
>> dthdt = @(t,th,v) v; dvdt = @(t,th,v) - c/m*v - g/L*sin(th);
```

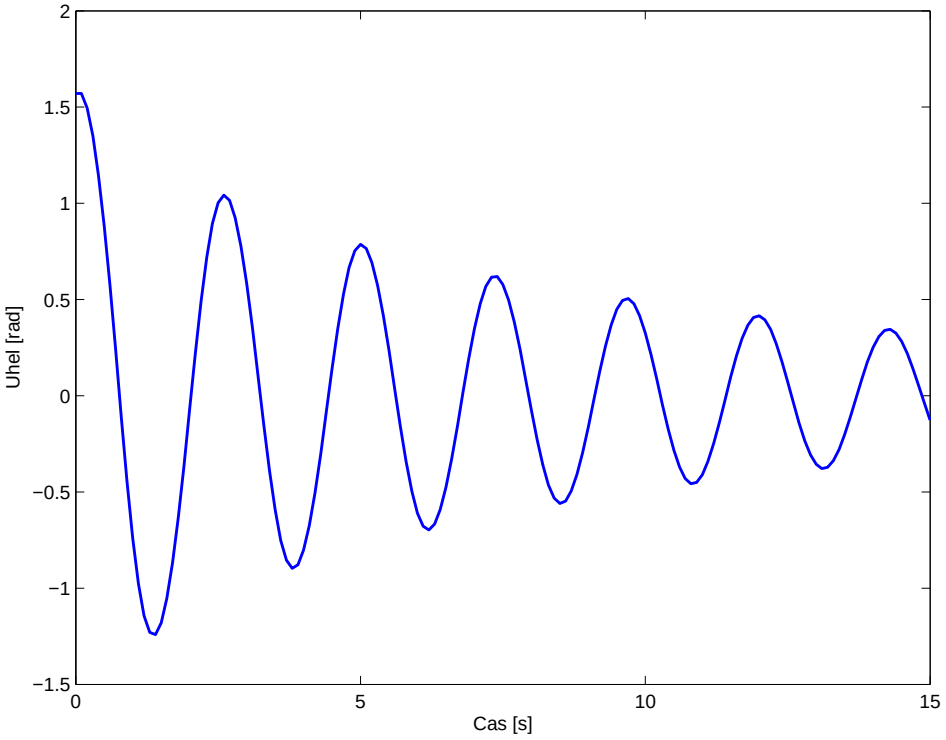
Nyní využijeme cyklu `for` pro aplikaci algoritmu Eulerovy metody.

```
>> for i = 1:n
 t(i+1) = t(i) + h;
 th(i+1) = th(i) + dthdt(t(i),th(i),v(i))*h;
 v(i+1) = v(i) + dvdt(t(i),th(i),v(i))*h;
end
```

Závěrem si necháme zobrazit hodnoty řešení v tabulce i graficky. Graf si můžete prohlédnout na Obrázku 12.6.

```
>> [t' th' v']
ans =
         0    1.5708         0
    0.1000    1.5708   -0.7546
    0.2000    1.4953   -1.4413
    0.3000    1.3512   -2.0641
    0.4000    1.1448   -2.6148
    0.5000    0.8833   -3.0666
    0.6000    0.5767   -3.3738
    0.7000    0.2393   -3.4816
    0.8000   -0.1089   -3.3471
    0.9000   -0.4436   -2.9639
    1.0000   -0.7400   -2.3732
   ..........................
   14.0000    0.2520    0.5551
   14.1000    0.3075    0.3170
```

```
   14.2000    0.3392    0.0601
   14.3000    0.3452   -0.1964
   14.4000    0.3255   -0.4341
   14.5000    0.2821   -0.6364
   14.6000    0.2185   -0.7892
   14.7000    0.1396   -0.8817
   14.8000    0.0514   -0.9074
   14.9000   -0.0393   -0.8645
   15.0000   -0.1258   -0.7570
>> plot(t,th)
>> xlabel('Cas [s]')
>> ylabel('Uhel [rad]')
```



Obrázek 12.6: Graf numerické aproximace funkce $\theta(t)$

12.8. Parašutista

Parašutista hmotnosti $m = 75$ kg vyskočí z letadla a padá volným pádem. Působí na něj aerodynamická odporová síla velikosti $F_{odpor} = c_o(\frac{dy}{dt})^2$, kde y označuje vzdálenost, kterou padající parašutista urazil v čase t. Určete, jakou vzdálenost urazí za 9 s. Při výpočtu uvažujte koeficient $c_o = 0.2028$ kg · m a gravitační zrychlení $g = 8.1$ m · s^2.

Na parašutistu působí (proti směru pádu) odporová síla

$$F_{odpor} = c_o \left(\frac{dy}{dt}\right)^2$$

a dále tíhová síla

$$F_{tihova} = mg.$$

Podle Newtonova druhého pohybového zákona platí, že součet sil je roven součinu hmotnosti a zrychlení. Protože zrychlení je rovno druhé derivaci funkce polohy, tj. $a = \frac{d^2y}{dt^2}$, platí v našem případě

$$mg - c_o \left(\frac{dy}{dt}\right)^2 = F_{tihova} - F_{odpor} = ma = m\left(\frac{dy}{dt}\right)^2.$$

Po vydělení hmotností obdržíme rovnici

$$\left(\frac{d^2y}{dt^2}\right) = g - \frac{c_o}{m}\left(\frac{dy}{dt}\right)^2$$

s počáteční podmínkou $y(0) = 0$.
V rovnici se vyskytuje druhá derivace, jde tedy o diferenciální rovnici druhého řádu. Zavedeme-li však substituci $z = \frac{dy}{dt}$ (nová funkce hraje roli zrychlení parašutisty), dostaneme počáteční úlohu

$$\frac{dz}{dt} = g - \frac{c_o}{m} z^2, \quad z(0) = 0.$$

Tu nejprve vyřešíme a následně integrací získaného řešení z obdržíme y.
K vyřešení diferenciální rovnice použijeme MATLAB a Rungeovu-Kuttovu metodu čtvrtého řádu s krokem $h = 0.1$.
Nejprve definujeme konstanty a počáteční podmínky.

```
>> m = 75; c = 0.2028; g = 8.81; h = 0.1; % Konstanty
>> n = (9-0)/h;
>> f = @(t,z)g-c/m*z^2; % Funkce prave strany
>> t(1) = 0; z(1) = 0; % Pocatecni podminky
```

Následně využijeme Rungeovu-Kuttovu metodu implementovanou v interním příkazu MATLABu `ode45`. Vypíšeme pouze některé ze spočtených hodnot.

```
>> [t z] = ode45(f,0:h:9,0);
>> [t z]
```

```
ans =
         0          0
    0.1000     0.8809
    0.2000     1.7614
    0.3000     2.6411
    0.4000     3.5195
    0.5000     4.3963
    0.6000     5.2709
    0.7000     6.1431
    0.8000     7.0124
    0.9000     7.8784
    1.0000     8.7407
    ...............
    8.0000    48.1729
    8.1000    48.4232
    8.2000    48.6669
    8.3000    48.9043
    8.4000    49.1356
    8.5000    49.3607
    8.6000    49.5800
    8.7000    49.7934
    8.8000    50.0011
    8.9000    50.2034
    9.0000    50.4002
```

Řešení si můžeme nechat zobrazit také graficky, viz Obrázek 12.7.

```
>> plot(t,z)
>> xlabel('Cas [s]')
>> ylabel('Zrychleni [m.s^2]')
```

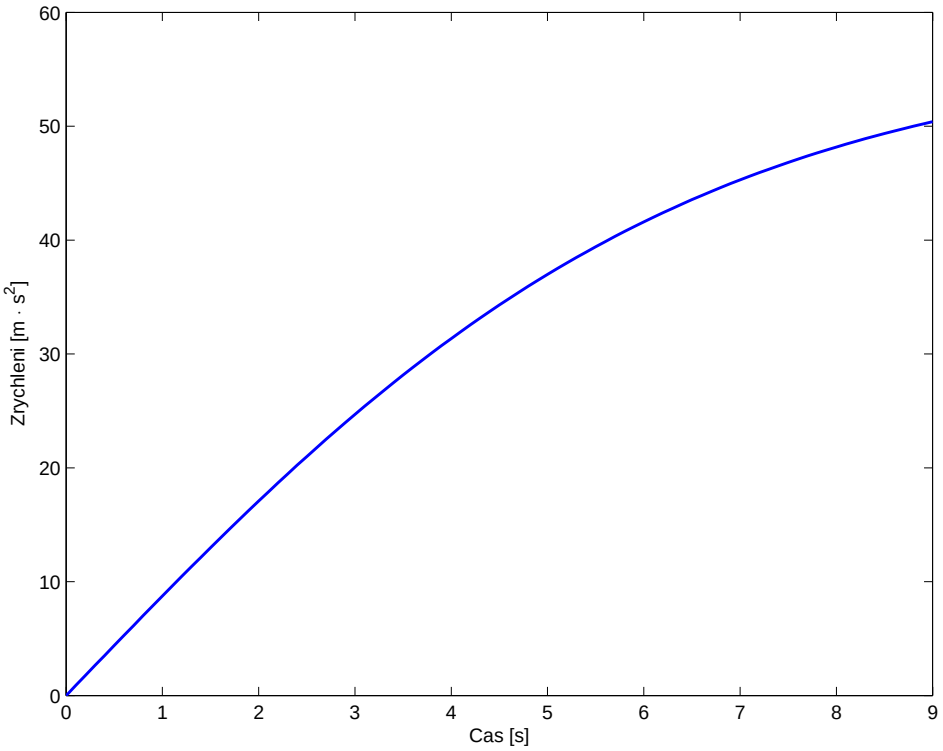
V druhé části úlohy přejdeme k řešení původního problému, totiž jakou vzdálenost urazí parašutista během 9 s. K tomu využijeme Newtonovy-Leibnizovy formule

$$\int_0^t z(t)\,\mathrm{d}t = \int_0^t \frac{\mathrm{d}y}{\mathrm{d}t}(t)\,\mathrm{d}t = y(t) - y(0) = y(t).$$

Naším úkolem je určit hodnotu $y(9) = \int_0^9 z(t)\,dt$. Protože neznáme předpis funkce z, nemůžeme použít interní funkce MATLABU quad. Známe však hodnoty funkce z v uzlových bodech intervalu $\langle 0,9\rangle$ s krokem $h = 0.1$, tj. v bodech $t_i = 0 + ih$, kde $i = 0,\ldots,n$. To pro numerický výpočet integrálu stačí. Použijeme složené Simpsonovo pravidlo, které má v našem případě tvar

$$\int_0^9 z(t)\,\mathrm{d}t \approx \frac{h}{3}\left[z(t_0) + 4\sum_{i=1}^{n/2} z(t_{2i-1}) + 2\sum_{i=1}^{n/2-1} z(t_{2i}) + z(t_n)\right].$$

Poté vzorec přepíšeme v MATLABu.



Obrázek 12.7: Graf numerické aproximace funkce $z(t) = \dot{y}(t)$

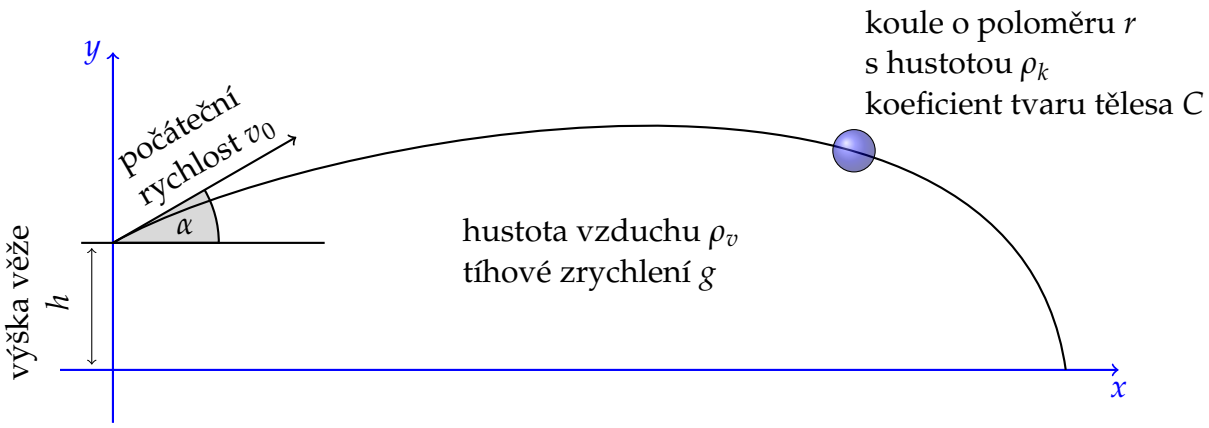
```
>> I = h/3*(z(1)+4*sum(z(2:2:n))+ 2*sum(z(3:2:n))+z(n+1)) %
   Pripomenme, ze v MATLABu jsou uzly indexovany od jednicky
I =
   279.6779
```

Parašutista tedy během volného pádu, bereme-li v úvahu aerodynamickou odporovou sílu, urazí během 9 s asi 270.66 m.

12.9. Šikmý vrh v odporovém prostředí

Dělová koule je vystřelena z věže výšky h pod úhlem α s počáteční rychlosti v_0. Dělová koule má poloměr r a hustotu ρ_k. Prostředí, které ovlivňuje dráhu letu střely je charakterizováno hustotou vzduchu ρ_v. Celou situaci ilustruje obr. 12.8.
Zadané hodnoty fyzikálních veličin.

elevační úhel	$\alpha = \pi/4$ rad
počáteční rychlost	$v_0 = 500\ \mathrm{m{\cdot}s^{-1}}$
výška věže	$h = 10$ m
poloměr koule	$r = 0.05$ m
hustota koule	$\rho_K = 7800\ \mathrm{kg{\cdot}m^{-3}}$
hustota vzduchu	$\rho_V = 1.2\ \mathrm{kg{\cdot}m^{-3}}$
koeficient tvaru	$C = 0.26$
tíhové zrychlení	$g = 9.81\ \mathrm{m{\cdot}s^{-2}}$



Obrázek 12.8: Popis úlohy

Fyzikální formulace

Při fyzikální interpretaci problému budeme vycházet ze skládání sil

$$\boldsymbol{F}_{\text{výsledná}} = -\boldsymbol{F}_{\text{odpor}} - \boldsymbol{F}_{\text{tíhová}}\,. \tag{12.6}$$

Odpor vzduchu působí silou $\boldsymbol{F}_{\text{odpor}}$. Dále musíme vzít v úvahu také sílu tíhovou $\boldsymbol{F}_{\text{tíhová}}$. Jednotlivé složky rovnice spočítáme podle následujících vztahů

$$\boldsymbol{F}_{\text{výsledná}} = m\boldsymbol{a} \qquad \boldsymbol{F}_{\text{odpor}} = \frac{1}{2}\,C\,\rho_v\,S\,v\,\boldsymbol{v} \qquad \boldsymbol{F}_{\text{tíhová}} = m\,\boldsymbol{g}\,,$$

kde $\boldsymbol{a}$ je zrychlení, $\boldsymbol{v}$ je vektor okamžité rychlosti a v je jeho velikost. Po jejich dosazení do rovnice (12.6) popisující rovnováhu sil obdržíme rovnost

$$m\,\boldsymbol{a} = -\frac{1}{2}\,C\,\rho_v\,S\,v\,\boldsymbol{v} - m\,\boldsymbol{g}\,.$$

Po úpravě dostaneme rovnici

$$\boldsymbol{a} = -k\,v\,\boldsymbol{v} - \boldsymbol{g},$$

kde

$$k = \frac{C\rho_v S}{2m} = \frac{C\rho_v \pi r^2}{2\rho_k \frac{4}{3}\pi r^3} = \frac{2C\rho_v}{8\rho_k r},$$

neboť v případě koule $S = \pi r^2$ a $m = \rho_k \frac{4}{3}\pi r^3$.
Zrychlení je derivací rychlost v čase, tedy $\boldsymbol{a} = \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}$. Dostáváme diferenciální rovnici

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -k\,v\,\boldsymbol{v} - \boldsymbol{g}.$$

Potřebujeme znát nejen okamžitou rychlost $\boldsymbol{v}$, ale i polohu koule $\boldsymbol{s}$. Rychlost je derivací polohy v čase

$$\frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} = \boldsymbol{v}.$$

Tento vztah je další diferenciální rovnicí, kterou budeme řešit.
Počáteční podmínky jsou dány počáteční polohou, tj. $\boldsymbol{s}(0) = (0, h)$ a počáteční rychlostí $\boldsymbol{v}(0) = (v_0 \cos\alpha, v_0 \sin\alpha)$.
Fyzikální formulace šikmého vrhu v odporovém prostředí tedy představuje soustavu diferenciálních rovnic s počátečními podmínkami.

$$\left.\begin{aligned} &\text{Hledáme polohu koule } \boldsymbol{s} \text{ a její rychlost } \boldsymbol{v}: \\ &\frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} = \boldsymbol{v}, \\ &\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -k\,v\,\boldsymbol{v} - \boldsymbol{g}, \\ &\boldsymbol{s}(0) = (0,h)\,, \quad \boldsymbol{v}(0) = (v_0\cos\alpha, v_0\sin\alpha) \end{aligned}\right\} \tag{12.7}$$

Matematická formulace

Poloha a rychlost střely závisí na dvou složkách a soustavu diferenciálních rovnic (12.7) můžeme převést na soustavu čtyř diferenciálních rovnic 1. řádu.

Přičemž označíme $\boldsymbol{s} = (s_x, s_y)$, $\boldsymbol{v} = (v_x, v_y)$ a platí $\boldsymbol{g} = (0, g)$ a $v = \sqrt{v_x^2 + v_y^2}$.

$$\left.\begin{aligned} s_x' &= v_x \\ v_x' &= -k\,v_x\sqrt{v_x^2+v_y^2} \\ s_y' &= v_y \\ v_y' &= -k\,v_y\sqrt{v_x^2+v_y^2} - g \\ s_x(0) = 0, &\quad v_x(0) = v_0\cos\alpha, \quad s_y(0) = h, \quad v_y(0) = v_0\sin\alpha \end{aligned}\right\} \tag{12.8}$$

Řešení této soustavy je poloha koule $[s_x(t), s_y(t)]$ a její rychlost $(v_x(t), v_y(t))$ v čase t.

Numerické řešení

K řešení problému použijeme systém MATLAB. Sestavíme program pro řešení soustavy diferenciálních rovnic s počátečními podmínkami. Použijeme vestavěnou funkci Matlabu `ode45` (Runge-Kutta 4.řádu). Připomeňme syntaxi tohoto příkazu: řešíme-li diferenciální rovnici $y' = f(x,y)$ s počáteční podmínkou $y(a) = c$ na intervalu $\langle a,b\rangle$, pak je syntaxe následující `[x,y]=ode45(f,[a,b],c)`.
Pro řešení našeho problému si nejprve sestavíme funkci, popisující pravé strany soustavy. Funkci nazveme `strela` a je potřeba si položit tyto otázky:

1. Vstupem funkce je vektor neznámých `X`, který má složky odpovídající s_x, s_y, v_x, v_y a nezávislá proměnná čas t. Výstupem funkce budou pravé strany soustavy (12.8), které označíme `dX`.

2. Využijeme možnost nadefinovat některé proměnné jako globální, pak jejich hodnota bude známa i uvnitř jednotlivých funkcí. Postačí v hlavním okně a jednotlivých funkcích použít příkaz `global` a jména proměnných, které mají být globální. K výpočtu pravých stran budeme potřebovat konstanty `g,k`.

```
function [dX]=strela(t,X);
%spocita prave strany soustavy
global g k

%prejmenujeme si promenne
sx=X(1);
vx=X(2);
sy=X(3);
vy=X(4);

%prave strany soustavy
dX(1,1)=vx;
dX(2,1)=-k*sqrt(vx^2+vy^2)*vx;
dX(3,1)=vy;
dX(4,1)=-k*sqrt(vx^2+vy^2)*vy-g;
```

Nejprve nastavíme hodnoty parametrů úlohy.

```
>> global g k
>> alfa=pi/4; v0=500; h=10; roK=7800;
>> r=0.05; c=0.26; roV=1.2; g=9.81;
>> k=(3*c*roV)/(8*roK*r);
```

Stěžejní otázkou je, na jakém intervalu hledáme řešení soustavy. Tedy jaký je čas dopadu. Zatím použijeme odhad z příkladu bez odporu vzduchu

$$t_{odhad} = \frac{1}{g}(v_0 \sin\alpha + \sqrt{v_0^2 \sin^2\alpha - 2gh}).$$

```
>> todhad=1/g*(v0*sin(alfa)+ sqrt(v0^2*sin(alfa)^2-2*h*g));
```

Spočítáme počáteční podmínky (12.8).

```
>> pp=[0,v0*cos(alfa),h,v0*sin(alfa)]
pp =
         0   353.5534    10.0000   353.5534
```

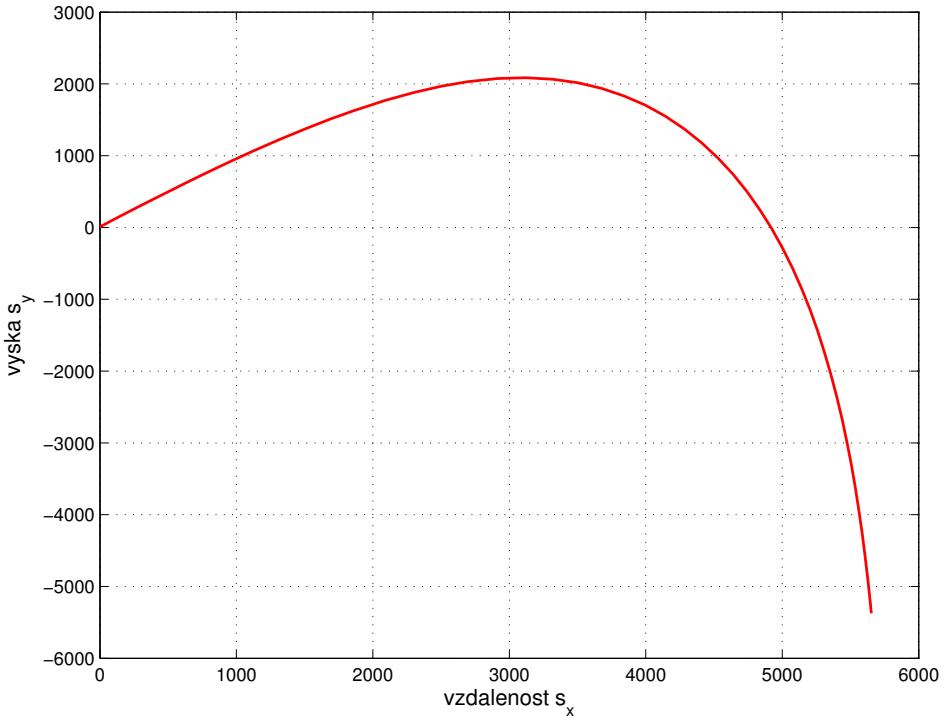
Najdeme řešení soustavy differenciálních rovnic (12.8). Výstupem příkazu `ode45` je vektor `cas` a matice `reseni`, jejíž první sloupec je hodnota s_x a třetím sloupcem je hodnota s_y.

```
>> [cas,reseni]=ode45(@strela ,[0,todhad],pp)
cas =
         0
    0.0000
    0.0000
       ...
   35.2071
   37.0084
   38.8097
   40.6110
   42.4123
   44.2136
       ...
   71.6425
   71.8472
   72.0519
reseni =
  1.0e+003 *

         0    0.3536    0.0100    0.3536
    0.0000    0.3536    0.0100    0.3536
    0.0000    0.3536    0.0100    0.3536
    ..................................
    4.6379    0.0590    0.7439   -0.1297
    4.7402    0.0546    0.5036   -0.1369
    4.8346    0.0503    0.2512   -0.1433
    4.9216    0.0463   -0.0120   -0.1488
    5.0015    0.0425   -0.2844   -0.1536
    5.0749    0.0390   -0.5648   -0.1577
    ..................................
    5.6493    0.0093   -5.2995   -0.1792
    5.6512    0.0092   -5.3362   -0.1793
    5.6531    0.0091   -5.3729   -0.1793
```

Dráhu střely zobrazíme do grafu (viz obr. 12.9). Vidíme, že skutečný čas dopadu je menší než je odhad, který jsme použili.

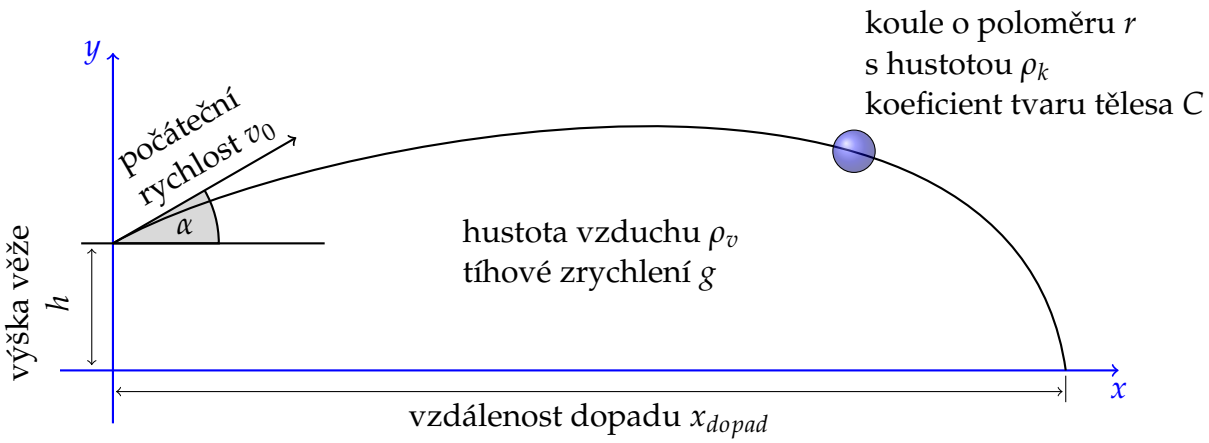
```
>> sx=reseni(:,1);
>> sy=reseni(:,3);
>> plot(sx,sy)
>> grid on
```



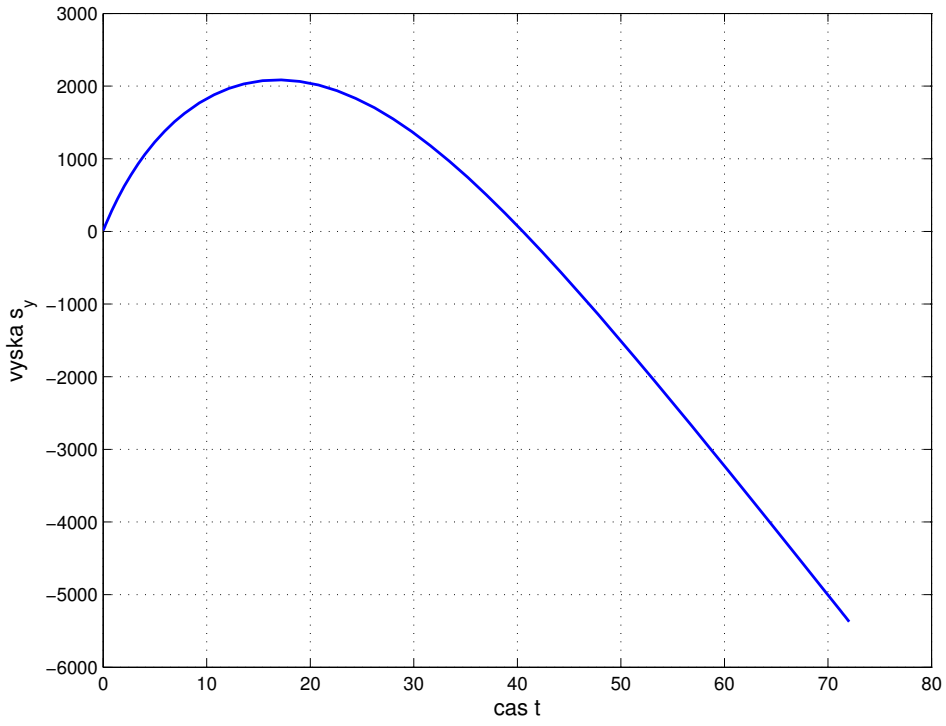
Obrázek 12.9: Dráha střely.

12.10. Vzdálenost dopadu u šikmého vrhu v odporovém prostředí

Jaká bude vzdálenost a čas dopadu u předchozí úlohy?



V předchozím příkladě jsem ukázali, že dráha střely je řešením soustavy diferenciálních rovnic (12.8). Pro výpočet vzdálenosti dopadu je potřeba znát čas dopadu. Na obrázku 12.10 je zobrazena výška střely $s_y(t)$ v závisloti na čase t.



Obrázek 12.10: Dráha střely a její výška v čase.

Vidíme, že čas dopadu bude řešení rovnice $s_y(t) = 0$.

$$\left.\begin{array}{l}\text{Hledáme } t \in (0, t_{odhad}) : \\ s_y(t) = 0\,, \\ \text{kde } s_y(t) \text{ je řešením (12.8).}\end{array}\right\} \tag{12.9}$$

K nalezení řešení této nelineární rovnice použijeme buď vestavěnou funkci Matlabu `fzero` nebo vlastní implementaci numerické metody na řešení rovnice, např. metodu půlení intervalu.
Všimněme si, že funkce $s_y(t)$ nemá explicitní vyjádření. Umíme pouze spočítat její hodnotu v čase t, a to tak, že hodnotu t zvolíme jako horní mez intervalu, na kterém řešení soustavu diferenciálních rovnic.
Sestrojíme funkci, která spočítá výšku v čase t. Připomeňme, že výška střely s_y je třetí složka řešení soustavy diferenciálních rovnic.
Pro výpočet je potřeba mít i funkce `strela` z předchozího příkladu.

```
function [sy]=vyska(t)
%funkce pocita vysku strely v case t
global g h k v0 alfa

%pocatecni podminky
pp=[0,v0*cos(alfa),h,v0*sin(alfa)];

%resime soustavu diferencialnich r. (od 0 do t)
[cas,reseni]=ode45(@strela,[0,t],pp);

%reseni je ve tvaru sx,vx,sy,vy
%vyska sy je treti slozka reseni
sy=reseni(end,3);
```

Nejprve nastavíme hodnoty parametrů úlohy.

```
>> global g h k v0 alfa
>> alfa=pi/4; v0=500; h=10; roK=7800;
>> r=0.05; c=0.26; roV=1.2; g=9.81;
>> k=(3*c*roV)/(8*roK*r);
```

A najdeme kořen pomocí funkce `fzero`, jako počáteční aproximaci použijeme odhad t_{odhad}.

```
>> todhad=1/g*(v0*sin(alfa)+ sqrt(v0^2*sin(alfa)^2-2*h*g));
>> [t_dopad]=fzero(@vyska,todhad)
t_dopad =
   40.5276
```

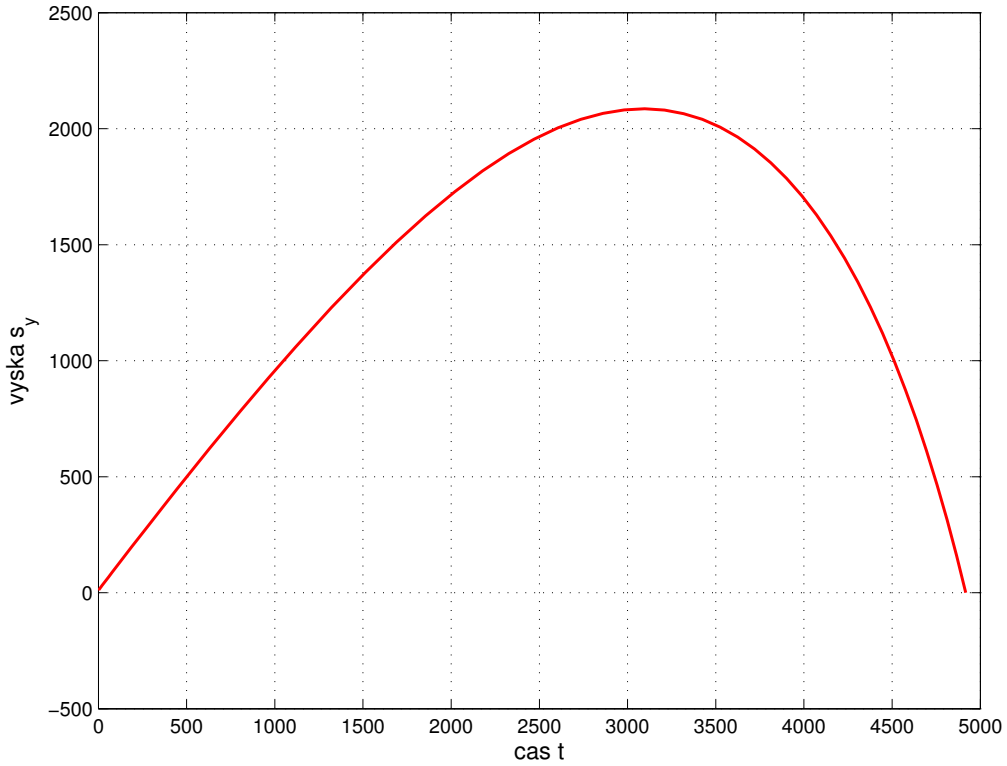
A najdeme řešení soustavy diferenciálních rovnic na intervalu $\langle 0, t_{dopad} \rangle$

```
>> pp=[0,v0*cos(alfa),h,v0*sin(alfa)];
>> [cas,reseni]=ode45(@strela,[0,t_dopad],pp);
```

```
>> sx_dopad=reseni(end,1)
sx_dopad =
   4.9172e+003
>> sy_dopad=reseni(end,3)
sy_dopad =
  -5.6843e-013
```

Střela dopadne na zem po 40.5 sekundách do vzdálenosti 4917 metrů.
Dráhu střely zobrazíme do grafu (viz obr. 12.11).

```
>> sx=reseni(:,1);
>> sy=reseni(:,3);
>> plot(sx,sy)
>> grid on
```



Obrázek 12.11: Dráha s časem dopadu

LITERATURA

Základní doporučená literatura – numerická matematika

[1] GILAT, Amos a Vish SUBRAMANIAM. *Numerical methods for engineers and scientists: an introduction with applications using MATLAB*. 2nd ed. Hoboken, N.J.: Wiley, 2011, xvi, 495 p. ISBN 9780470565155.

[2] KUČERA, Radek. *Numerické metody*. 1. vyd. Ostrava: VŠB - Technická univerzita, 2006, 132 s. ISBN 80-248-1198-7. Dostupné z: http://mdg.vsb.cz/portal/nm/nm.pdf.

[3] WOODFORD, Chris a Chris PHILLIPS. *Numerical methods with worked examples: Matlab edition*. 2nd ed. New York: Springer, 2012, x, 256 p. ISBN 9789400713666.

Základní doporučená literatura – práce s Matlabem

[4] DUŠEK, František. *MATLAB a SIMULINK: úvod do používání*. Vyd. 1. Pardubice: Univerzita Pardubice, 2000, 146 s., [4] s. barev. obr. příl. ISBN 80-7194-273-1.

[5] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB pro začátečníky*. 2. vyd. Praha: BEN - technická literatura, 2005, 151 s. ISBN 80-7300-175-6.

[6] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB: tvorba uživatelských aplikací*. 1. vyd. Praha: BEN - technická literatura, 2004, 215 s. ISBN 80-7300-133-0.

Doplňková literatura

[7] FIEDLER, Miroslav. *Speciální matice a jejich použití v numerické matematice*. 1. vyd. Praha: Státní nakladatelství technické literatury, 1981, 266 s. Teoretická knižnice inženýra.

[8] HAMMING, R. *Numerical methods for scientists and engineers*. 2nd ed. New York: Dover, 1973, ix, 721 p. ISBN 0486652416.

[9] MÍKA, Stanislav. *Numerické metody algebry*. 2., nezm. vyd. Praha: SNTL, 1985, 169 s. Matematika pro vysoké školy technické.

[10] PŘIKRYL, Petr. *Numerické metody matematické analýzy: vysokoškolská příručka pro vysoké školy technické*. 2. nezm. vyd. Praha: Státní nakladatelství technické literatury, 1988, 187 s. Matematika pro vysoké školy technické.

[11] RALSTON, Anthony. *Základy numerické matematiky: příručka pro univerzity ČSR*. 2. čes. vyd. Praha: Academia, 1978, 635, [1] s.

[12] SÜLI, Endre a David MAYERS. *An introduction to numerical analysis*. New York: Cambridge University Press, 2003, x, 433 p. ISBN 0521007941.

[13] TEODORESCU, P, Nicolae-Doru STĂNESCU a Nicolae PANDREA. *Numerical analysis with applications in mechanics and engineering*. Hoboken, NJ: Wiley, 2013, xi, 633 pages. ISBN 9781118077504.

[14] WALLISCH, Pascal. *MATLAB for neuroscientists: an introduction to scientific computing in MATLAB*. Burlington: Elsevier, 2009, xiv, 384 s., [8] s. obr. příl. ISBN 978-0-12-374551-4.

[15] VITÁSEK, Emil. *Numerické metody*. Praha: SNTL, 1987.

Název: Numerická matematika: Řešené příklady s Matlabem a aplikované úlohy

Katedra: Katedra matematiky a deskriptivní geometrie

Autoři: Pavel Ludvík, Zuzana Morávková

Místo, rok, vydání: Ostrava, 2016, 1. vydání

Počet stran: 137

Vydala: Vysoká škola báňská-Technická univerzita Ostrava

Neprodejné

ISBN 978-80-248-3892-2