

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

NUMERICKÉ ŘEŠENÍ OKRAJOVÉ ÚLOHY METODOU
KONĚČNÝCH DIFERENCÍ

SEMESTRÁLNÍ PROJEKT
SEMESTRAL PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JURAJ REPČÍK

Brno 2017

OBSAH

Úvod	1
1 Metóda konečných diferencii	2
2 Používanie programu	6
2.1 Vzorové príklady	8
2.1.1 Príklad 1	8
2.1.2 Príklad 2	10
2.1.3 Príklad 3	10
2.1.4 Príklad 4	11
2.1.5 Príklad 5	13
3 Analýza zdrojového kódu	15
3.1 Okrajové podmienky, 2 typy	15
3.2 Koeficienty k	16
3.3 Funkcia over jednoznačnosť	16
3.4 Metódy riešenia sústavy rovníc	16
4 Zdrojový kód	18
Záver	29
Literatúra	30

ÚVOD

Zadaním úlohy bolo vytvoriť program na počítanie diferenciálnych rovníc (okrajovej úlohy) metódou konečných diferencií. Program dokáže vyhodnotiť výsledok ako vektor hodnôt \mathbf{x} a vektor funkčných hodnôt $\mathbf{y}(\mathbf{x})$. Rovnica musí byť druhého rádu a okrajové podmienky sú zadané v všeobecnej forme, tzv. Sturmova.

Text je rozdelený do štyroch hlavných kapitol. Prvá sa venuje všeobecnému popisu metódy konečných diferencií, druhá prezentuje program a predstavuje návod na jeho používanie. Tretia kapitola analyzuje vybrané časti zdrojového kódu. V štvrtej kapitole je ukázaný celý zdrojový kód.

1 METÓDA KONEČNÝCH DIFERENCIÍ

V okrajovej úlohe je výsledkom vypočítať priebeh funkcie $y(x)$, keď je zadaná diferenciálna rovnica g druhého rádu napríklad v tvare:

$$y'' = g(x, y, y'). \quad (1.1)$$

V tomto projekte je spracované riešenie diferenciálnej rovnice v upravenom tvare:

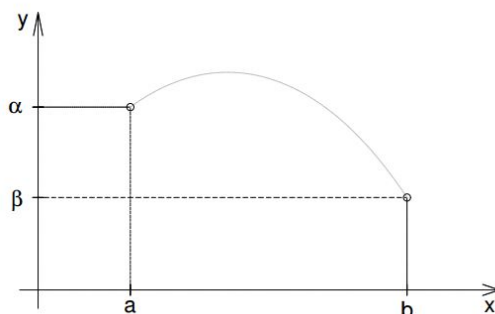
$$b(x) * y'' + c(x) * y' + d(x) * y + e(x) = f(x). \quad (1.2)$$

S úlohou musia byť zadané okrajové podmienky, ktoré definujú funkčnú hodnotu alebo deriváciu alebo ich lineárnu kombináciu na začiatku a konci intervalu riešenia $< a, b >$.

Ak sú zadané funkčné hodnoty, podmienky sa nazývajú *Dirichletove* [1] a sú zadané v nasledujúcom tvare:

$$y(a) = \alpha, \quad y(b) = \beta. \quad (1.3)$$

Názorná ukážka takéhoto zadania je na obrázku 1.1.



Obr. 1.1: Okrajová úloha, dirichletove podmienky [2]

Okrajové podmienky môžu byť zadané aj vo forme derivácii funkcie y :

$$y'(a) = \alpha, \quad y'(b) = \beta, \quad (1.4)$$

vtedy sa nazývajú *Neumannove*. Všeobecne sa dajú okrajové podmienky definovať ako *Sturm* nasledujúcim zápisom:

$$\alpha_1 * y(a) + \alpha_2 * y'(a) = \gamma_1, \quad \beta_1 * y(b) + \beta_2 * y'(b) = \gamma_2, \quad (1.5)$$

kde aspoň jeden koeficient α a aspoň jeden koeficient β musí byť nenulový.

Je zrejmé, že *Sturm*ove podmienky zahrňujú aj špeciálne prípady, rovnice 1.3 a 1.4, takže sú všeobecne použiteľné. V programe sa teda budú zadávať koeficienty $\alpha_1, \alpha_2, \gamma_1, \beta_1, \beta_2, \gamma_2$ spolu s rozsahom riešenia a a b . Diferenciálna rovnica bude zadaná v tvare 1.2.

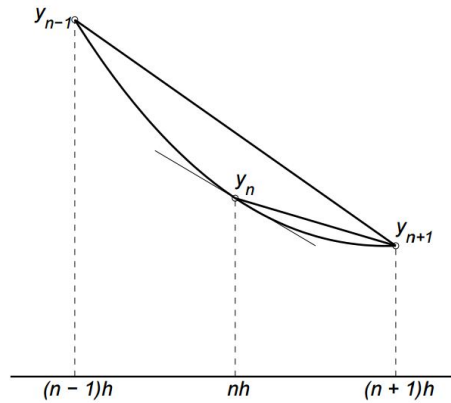
Metóda konečných diferencií spočíva v nahradení derivovaných členov diferenciálami. Existuje mnoho spôsobov ako sa dá aproximovať diferenciál. V texte [3] sa na strane 59 nachádza prehľadná tabuľka pre derivácie prvého a druhého rádu.

V tomto projekte sú použité nasledovné aproximácie [4]:

$$y'(x_i) = \frac{y_{i+1} - y_{i-1}}{2 * h} - \frac{h^2}{6} * y'''(x_i) - \dots \approx \frac{y_{i+1} - y_{i-1}}{2 * h}, \quad (1.6)$$

$$y''(x_i) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \frac{h^2}{12} * y^{(4)}(x_i) - \dots \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad (1.7)$$

kde h je krok medzi dvoma bodmi riešenia a zápis y_i je ekvivalentný zápisu $y(x_i)$. Aproximácia 1.6 je graficky znázornená na obrázku 1.2.



Obr. 1.2: Náhrada derivácie diferenciou [4]

V zanedbaných členoch rovníc 1.6 a 1.7 sa vyskytuje násobiteľ h^2 . To znamená, že ak bude krok $h < 1$ a bude sa zmenšovať, zanedbané členy budú mať menší vplyv na presnosť výsledku. Chyba sa pri $h < 1$ nebude akumulovať.

Ak nahradíme derivované členy aproximáciami v rovnici 1.2, dostaneme po úprave tento výsledok:

$$b(x_i) * (y_{i+1} - 2y_i + y_{i-1}) + c(x_i) * h * \frac{y_{i+1} - y_{i-1}}{2} + d(x_i) * y_i + e(x_i) = f(x_i) * h^2. \quad (1.8)$$

Sieť sa vytvorí z intervalu $< a, b >$ a počtu krokov n nasledovne:

$$x_i = a + i * h, \quad i = 0, 1, \dots, n, \quad h = \frac{b - a}{n}. \quad (1.9)$$

Zo zápisu vyplýva že $x_0 = a$, $x_n = b$.

Na zapísanie rovnice 1.8 do maticeového tvaru pre všetky i , je potrebné zistiť koeficienty k pri y_{i-1}, y_i, y_{i+1} . Prepísaná rovnica vyzerá nasledovne:

$$k_{-1}(x_i, h) * y_{i-1} + k_0(x_i, h) * y_i + k_1(x_i, h) * y_{i+1} = f(x_i) * h^2. \quad (1.10)$$

Rovnaké tri funkcie koeficientov sú použité v každom riadku matice A , s dosadenými hodnotami ktoré zodpovedajú bodu v sieti.

$$A = \begin{bmatrix} k_{-1}(x_0, h) & k_0(x_0, h) & k_1(x_0, h) & 0 & \dots & 0 \\ 0 & k_{-1}(x_1, h) & k_0(x_1, h) & k_1(x_1, h) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & k_{-1}(x_{n-1}, h) & k_0(x_{n-1}, h) & k_1(x_{n-1}, h) & 0 \\ 0 & \dots & 0 & k_{-1}(x_n, h) & k_0(x_n, h) & k_1(x_n, h) \end{bmatrix} \quad (1.11)$$

$$A * \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} f(x_0) * h^2 \\ f(x_1) * h^2 \\ \vdots \\ f(x_{n-1}) * h^2 \\ f(x_n) * h^2 \end{bmatrix} \quad (1.12)$$

Ak sa jedná o príklad so zadanými Dirichletovými podmienkami (rovnica 1.3), v sústava sa zjednoduší, lebo riešenie pre $y(x_0) = \alpha$ a $y(x_n) = \beta$ je známe. Po preformulovaní bude sústava vyzeráť takto (pri podmienkach z rovnice 1.3):

$$A_d = \begin{bmatrix} k_0(x_0, h) & k_1(x_0, h) & 0 & \dots & 0 \\ k_{-1}(x_1, h) & k_0(x_1, h) & k_1(x_1, h) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & k_{-1}(x_{n-1}, h) & k_0(x_{n-1}, h) & k_1(x_{n-1}, h) \\ 0 & \dots & 0 & k_{-1}(x_n, h) & k_0(x_n, h) \end{bmatrix} \quad (1.13)$$

$$A_d * \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} f(x_1) * h^2 - k_{-1}(x_0, h) * \alpha \\ f(x_2) * h^2 \\ \vdots \\ f(x_{n-2}) * h^2 \\ f(x_{n-1}) * h^2 - k_1(x_n, h) * \beta \end{bmatrix} \quad (1.14)$$

Ak je príklad zadaný so všeobecnými Strumovými podmienkami (rovnica 1.5), alebo Neumannovými, do sústavy rovníc sa pripíše jedna rovnica navyše pre každú okrajovú podmienku. Rovnica pre začiatočnú podmienku je [1]:

$$\alpha_1 * y_0 + \frac{\alpha_2}{2h} * (-3 * y_0 + 4 * y_1 - y_2) = \gamma_1, \quad (1.15)$$

upravená na:

$$\left(\alpha_1 - \frac{3\alpha_2}{2h}\right) * y_0 + \frac{4\alpha_2}{2h} * y_1 - \frac{\alpha_2}{2h} * y_2 = \gamma_1, \quad (1.16)$$

Pre koncovú podmienku platí podobne [1]:

$$\beta_1 * y_n + \frac{\beta_2}{2h} * (3 * y_n - 4 * y_{n-1} + y_{n-2}) = \gamma_2, \quad (1.17)$$

upravená na:

$$\left(\beta_1 + \frac{3\beta_2}{2h}\right) * y_n - \frac{4\beta_2}{2h} * y_{n-1} + \frac{\beta_2}{2h} * y_{n-2} = \gamma_2, \quad (1.18)$$

Rovnice 1.16 a 1.18 sa pridajú do sústavy rovníc 1.12. Matica A sa potom jednoduchými riadkovými operáciami upraví na tridiagonálny tvar.

Podmienky riešiteľnosti sústavy rovníc pre metódu konečných diferencií zodpovedajú podmienkam pre riešiteľnosť danej diferenciálnej rovnice v danom intervale. Dá sa dokázať, že matica A je regulárna [5], [6]. V prípade že A je diagonálne dominantná alebo neostro diagonálne dominantná, kde aspoň jedna rovnica spĺňa ostrú diagonálnu dominantnosť, je zaručené že Jacobiho aj Gauss-Seidelova iteračná metóda na riešenie sústav lineárnych rovníc bude konvergovať pre ľubovoľný počiatočný stav [1] a sústava bude mať jedno jednoznačné riešenie.

Iteračná metóda začne s krokom menším ako 1 a vypočíta sa druhá iterácia s polovičným krokom. Absolútne hodnoty bodov $y(x)$ kde je známe riešenie z oboch iterácií sa odčítajú a absolútna hodnota výsledku sa považuje za presnosť výsledku metódy. Ak nie je presnosť v tomto kroku dosiahnutá, vypočíta sa ďalšia iterácia s polovičným krokom od posledne použitého kroku. Dva posledné výpočty sa potom znovu používajú na vyhodnotenie presnosti.

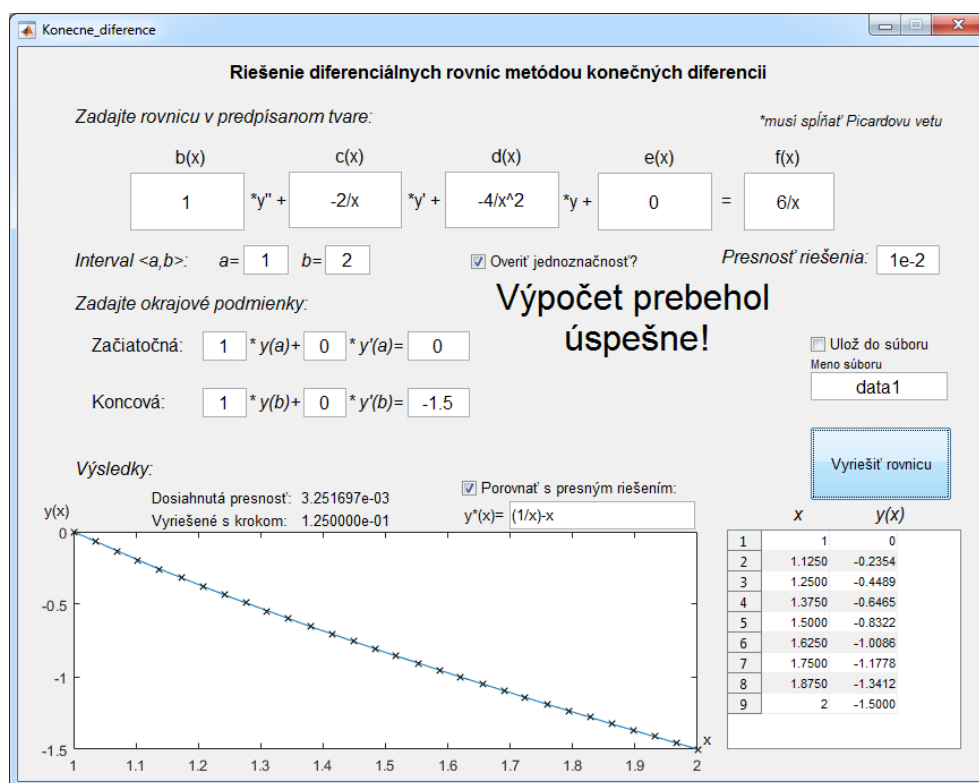
2 POUŽÍVANIE PROGRAMU

Na spustenie programu je nutné mať na počítači nainštalovaný program Matlab vo verzii 2014 alebo vyššiu s nainštalovanou súčasťou Symbolic toolbox a GUI Layout toolbox.

Archív s odovzdaným programom obsahuje súbory:

```
/.....Hlavný adresár, odovzdané súbory
|_GaussSeidel.m.....Gauss-Seidelova iteračná metóda
|_kondiff_calc.m.....Staršia verzia algoritmu pre kon. diferencie
|_kondiff_calc_v2.m.....Algorismus pre konečné diferencie
|_Konecne_diference.fig.....GUI definície
|_Konecne_diference.m.....<— hlavný spúšťač súbor, kód k obsluhu GUI
|_over_jednoznacost.m.....Overenie riesitelnosti matice A
|_save_data.m.....kód na uloženie dát do súboru
|_xx_porovnanie_chyb.m....skript na zobrazenie chyby posledného výpočtu
```

Program sa spustí pomocou **Konecne_diference.m**. Následne sa objaví grafické prostredie ako na obrázku 2.1 (na obrázku je stav po vypočítaní príkladu). Toto prostredie bude popísané v nasledujúcom texte.



Obr. 2.1: Grafické prostredie programu

Program nemá implementované spoľahlivé overenie jednoznačnosti riešenia, takže pred počítaním príkladu v tomto programe sa doporučuje si overiť jednoznačnosť

riešenia ručne. Inak povedané, príklad s jeho okrajovými podmienkami musí spĺňať Picardovu vetu [7].

Najprv sa v programe vyplnia všetky potrebné vstupy a parametre riešenia. Pri zadávaní funkcii je nutné používať symbol “*” pri definovaní násobenia. Zápis “2x” by spôsobil chybu, preto je nutné napísať “2*x”. Tiež pri zadávaní desatinného čísla sa používa desatinná bodka (tečka) “.”. Zlomky sa definujú znamienkom lomeno “/”. Prirodzený logaritmus je možné napísať ako “**exp(1)**”. Podobne fungujú aj určité trigonometrické funkcie.

Rovnica sa zadáva do prostredia v tvare podľa zápisu 1.2. Do políček sa uvedú koeficienty pri deriváciách funkcie y, pri funkcii y, absolútny člen a funkcia na pravej strane rovnice. Celé funkcie nie je potrebné dávať do zátvoriek.

Číselné hodnoty sú akceptované v desatinnom alebo “exponenciálnom” formáte, napríklad “1.5” a “1e5”. Interval riešenia sa zadáva do číselných polí premenných a a b. Číselné pole “presnosť riešenia” udáva najväčší absolútny rozdiel v uzlových bodoch pri zjemňovaní siete na zastavenie výpočtu. Okrajové podmienky je možné zadať vo všeobecnom tvare ako Sturmova. Polia sú tiež číselného charakteru.

Zaškrtrávacie políčko “Overiť jednoznačnosť?” slúži na spustenie funkcie *over_jednoznacnost* ktorá vyhodnotí maticu A. Metóda na vyhodnotenie jednoznačnosti nezaručuje istotu že príklad bude jednoznačne riešiteľný aj keď tak vyhodnotí, pretože sa tento fakt overuje numericky, iba v konečnom počte bodov v rozsahu riešenia. Ak funkcia vyhodnotí že príklad nespĺňa dané požiadavky, výpočet neprebehne.

Zaškrtrávacie políčko “Ulož do súboru” slúži na uloženie výsledku do súboru *.csv s definovaným menom, ktoré sa vpíše do poľa “Meno súboru”.

Posledný prvok na nastavenie programu je porovnanie s presným riešením. Ak je známe presné riešenie diferenciálnej rovnice, užívateľ si môže overiť fungovanie programu napísaním tejto funkcie. Ak je zaškrtnuté políčko, tak sa po vypočítaní vykreslí okrem vypočítaného priebehu tiež 30 bodov presného riešenia na grafické porovnanie výsledkov. Presné riešenie sa v programe nikde nevyužíva, iba sa vykresľuje do grafu na názorné grafické porovnanie (v grafe čierne body zobrazené vybolom +).

Výsledné body z iteračného procesu sú vykreslené na grafe lineárne interpolovanými modrými úsečkami. Výsledok je tiež uvedený v tabuľke napravo od grafu.

Keď sú nastavené všetky parametre, kliknutím tlačidla **Vyriešiť rovnicu** sa zaháji výpočet. Veľkým písmom sa zobrazí, či výpočet prebieha a po ukončení sa ukáže správa o úspešnosti alebo neúspešnosti výpočtu.

Po vypočítaní úlohy sa tiež nad grafom funkcie y(x) zobrazí výsledná dosiahnutá presnosť a krok s ktorým je výsledok vypočítaný. Ak užívateľ chce výpočet s menším krokom, odporúča sa vypočítať rovnaký príklad s menším číslom presnosti (väčšia

presnosť).

2.1 Vzorové príklady

Táto kapitola sa venuje overeniu funkčnosti na príkladoch, ktoré boli vytvorené alebo prevzaté.

2.1.1 Príklad 1

Na účel odtestovania programu bol vytvorený nasledovný príklad:

$$\frac{1}{x}y'' + 4y' - 5y + 2x = 2x + 5e^{5x}\left(\frac{5}{x} + 3\right) + 25e^{5(3-x)}\left(\frac{1}{x} - 1\right). \quad (2.1)$$

Dirichletove okrajové podmienky sú:

$$y(1) = y(2) = 22174.87895 \quad (2.2)$$

Presné riešenie tohto príkladu je:

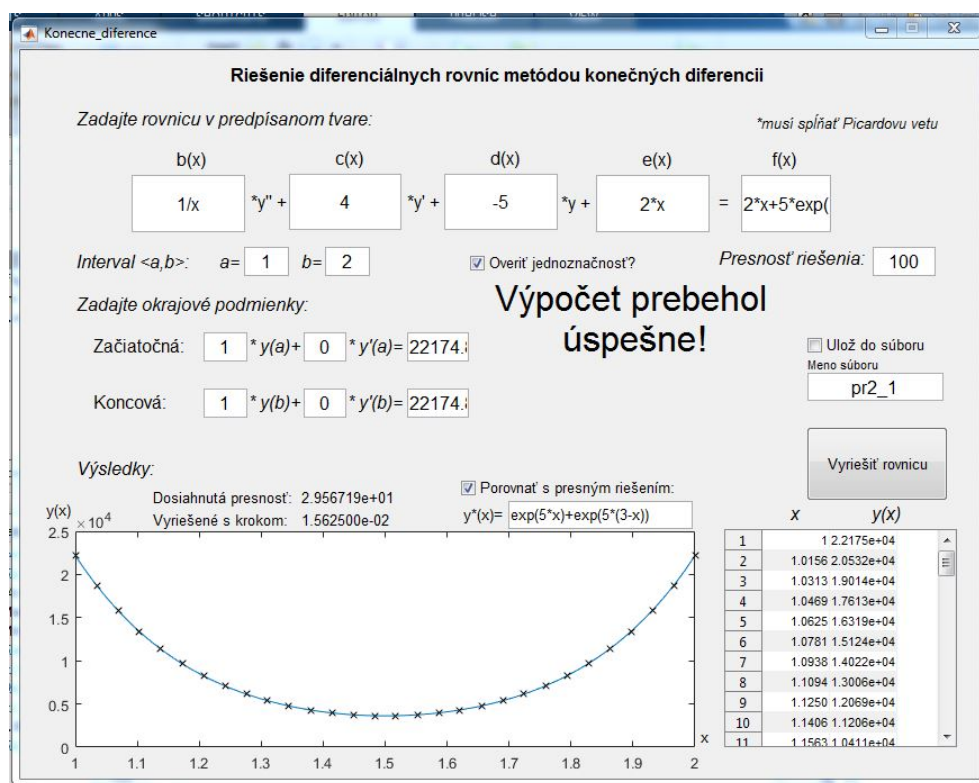
$$y^*(x) = e^{5x} + e^{5(3-x)}. \quad (2.3)$$

Program úspešne vypočítal tento príklad. Naprv overil jednoznačnosť a potom bolo potrebné 5 iterácií na dosiahnutie presnosti 10^2 . Program vypisuje na pozadí v Command Window v prostredí Matlab informácie o priebehu, ktoré v tomto prípade ukázali:

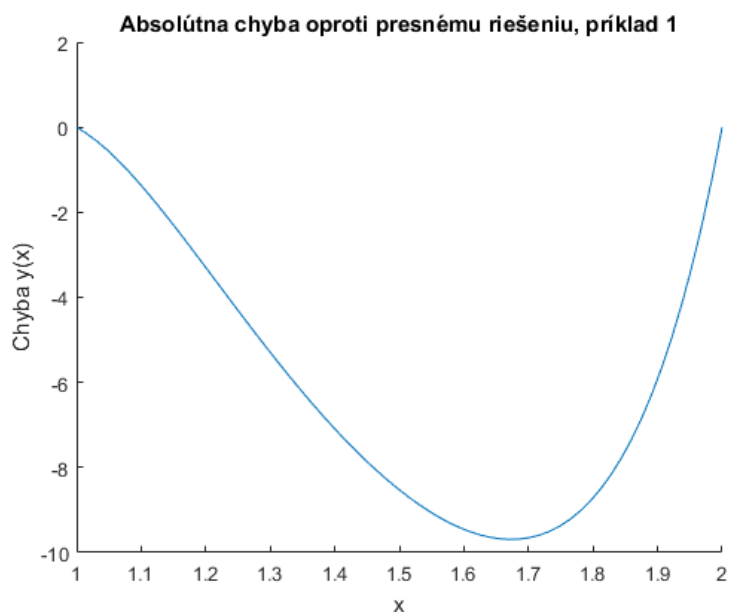
Výpis 2.1: Výpis pri počítaní príkladu 1

Diagonalne dominantna	
Iteracia kon. diff: 1, veľkosť matice:	3
Iteracia kon. diff: 2, veľkosť matice:	7
Iteracia kon. diff: 3, veľkosť matice:	15
Iteracia kon. diff: 4, veľkosť matice:	31
Iteracia kon. diff: 5, veľkosť matice:	63

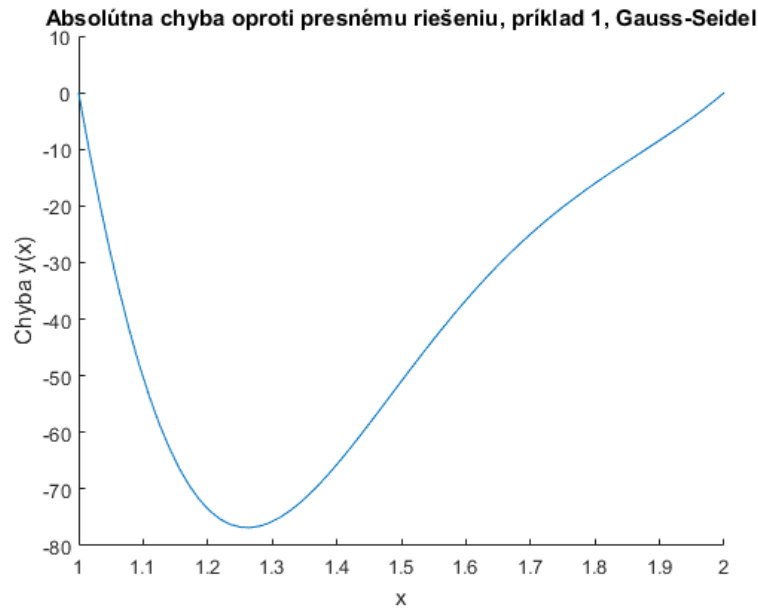
Prostredie programu po vypočítaní je možné vidieť na obrázku 2.2. Ako je spomenuté v kapitole 3, na výpočet sústavy bolo experimentované s dvoma metódami. Chyby pri každej metóde zobrazujú grafy na obr.2.3 a obr.2.4. Na dosiahnutie požadovanej presnosti sa musela vypočítať sústava s 63 rovnicami. Celý výpočet zabral menej ako 5 sekúnd.



Obr. 2.2: Príklad 1, výpočet, metóda 2



Obr. 2.3: Príklad 1, zobrazenie chyby, sústava riešená metódou 2 [9]



Obr. 2.4: Príklad 1, zobrazenie chyby, sústava riešená metódou 1, Gauss-Seidelova iteračná metóda

2.1.2 Príklad 2

V príklade 2 je použitá rovnaká rovnica (rovnica 2.1), avšak okrajové podmienky sú definované inak. Začiatočná podmienka je Sturmova a koncová je ponechaná z prechádzajúceho príkladu.

Príklad bol počítaný s rovnakými nastaveniami ako príklad 1 (obr. 2.2) až na zmenenú okrajovú podmienku.

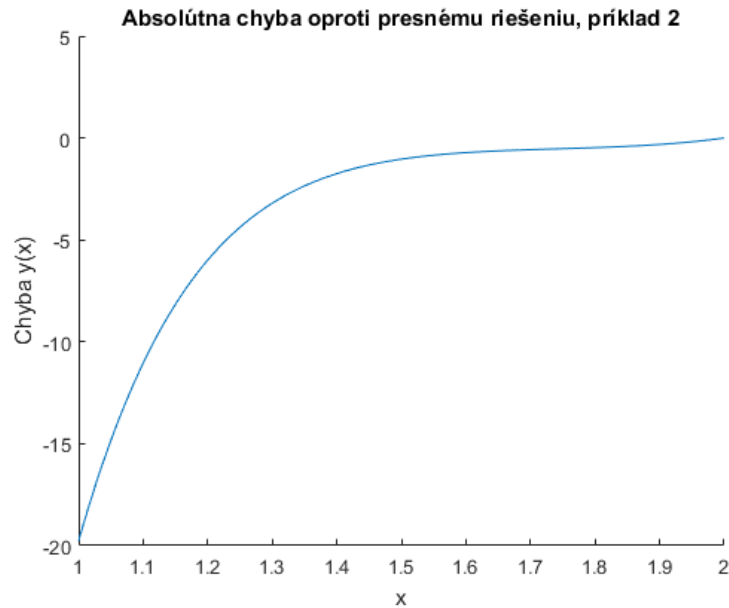
$$4.95y(1) + y'(1) = 375.3876433, \quad y(2) = 22174.87895 \quad (2.4)$$

Na obrázku 2.5 je zobrazený graf chyby. Je vidieť že porovnaním s obrázkom 2.3 je na začiatku intervalu väčšia, z dôvodu inak definovanej okrajovej podmienky. Chyba stále spadá do zadanej presnosti. Väčšia odchýlka v chybe môže byť spôsobená aj aproximáciou zadania okrajovej podmienky (bola vypočítaná s presného riešenia a výsledok zadaný s dostatočnou presnosťou (r.2.4)). V tomto príklade bolo už potrebné iterovať 7 krát a výsledný počet rovníc bol 257.

2.1.3 Príklad 3

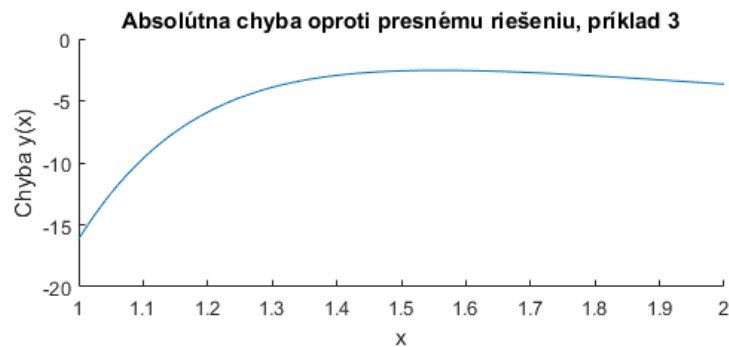
Príkladom 3 bude overený program aj pre Neumannovu okrajovú podmienku. Zase použijeme rovnakú rovnicu 2.1. Okrajové podmienky budú tentoraz nasledovné:

$$4.95y(1) + y'(1) = 375.3876433, \quad 0.1y'(2) = 10939.02632 \quad (2.5)$$



Obr. 2.5: Príklad 2, zobrazenie chyby, sústava riešená metódou 2 [9]

Program vypočítal riešenie pri ôsmej iterácii s počtom rovníc 513. Z dôvodu väčšieho počtu rovníc je aj riešenie diskretizované na osi x s menším krokom a to približne 0,002. Chyba výpočtu (obr.2.6) bola podobná ako v prechádzajúcom príklade.



Obr. 2.6: Príklad 3, zobrazenie chyby, sústava riešená metódou 2 [9]

2.1.4 Príklad 4

Príklad 4 je prevzatý z literatúry, konkrétne z [11]. Táto rovnica sa používa na počítanie prenosu energie alebo iných fyzikálnych veličín procesom vednia a difúzie. Pre 1D prípad je definovaná ako:

$$\epsilon y'' - y' = -1 \quad (2.6)$$

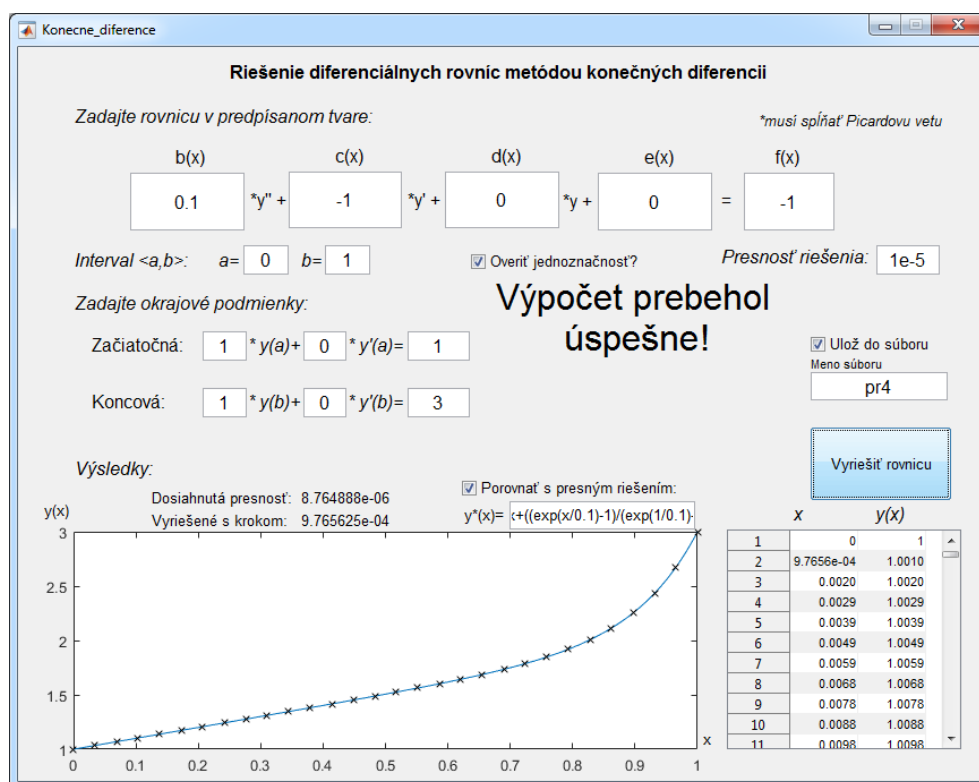
Pre tento príklad budú použité nasledovné okrajové podmienky:

$$y(0) = 1, \quad y(1) = 3 \quad (2.7)$$

Presné riešenie tejto rovnice je:

$$y(x) = 1 + x + \left(\frac{e^{x/\epsilon} - 1}{e^{1/\epsilon} - 1} \right) \quad (2.8)$$

Na výpočet bude použitá konštanta $\epsilon = 0,1$. Presnosť je nastavená na 10^{-5} . Obrázok 2.7 zobrazuje grafické prostredie programu po vypočítaní príkladu. Výpočet trval asi 5 sekúnd. Z výpisu 2.2 je vidieť že bolo počítaných až 1023 rovníc. Výsledný krok je zobrazený nad grafom: $h = 9,77 * 10^{-4}$. Chyba výpočtu (rozdiel s presného riešenia a riešenia metódy konečných diferencií) je zobrazený na obr.2.8.

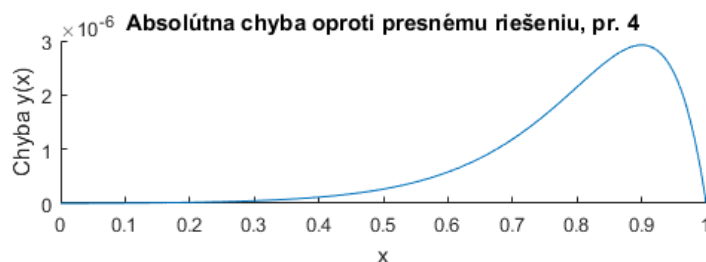


Obr. 2.7: Príklad 4, výpočet, metóda 2

Výpis 2.2: Výpis pri počítaní príkladu 4

```
Neostre diagonalne dominantna
Minimalne jedna rovnica ostrá, OK
Iteracia kon.diff: 1, veľkosť matice: 3
Iteracia kon.diff: 2, veľkosť matice: 7
Iteracia kon.diff: 3, veľkosť matice: 15
```

Iteracia kon. diff:	4, veľkosť matice:	31
Iteracia kon. diff:	5, veľkosť matice:	63
Iteracia kon. diff:	6, veľkosť matice:	127
Iteracia kon. diff:	7, veľkosť matice:	255
Iteracia kon. diff:	8, veľkosť matice:	511
Iteracia kon. diff:	9, veľkosť matice:	1023



Obr. 2.8: Príklad 4, zobrazenie chyby, sústava riešená metódou 2 [9]

Vo výpise 2.3 je vidieť ako vyzerá výstupný dátový súbor programu.

Výpis 2.3: Výstrižok z uložených dát príkladu 4, pr4.csv

```
x;y
0.0000000e+00;1.0000000e+00
9.7656250e-04;1.0009770e+00
1.9531250e-03;1.0019540e+00
...
9.9804688e-01;2.9787041e+00
9.9902344e-01;2.9893048e+00
1.0000000e+00;3.0000000e+00
```

2.1.5 Príklad 5

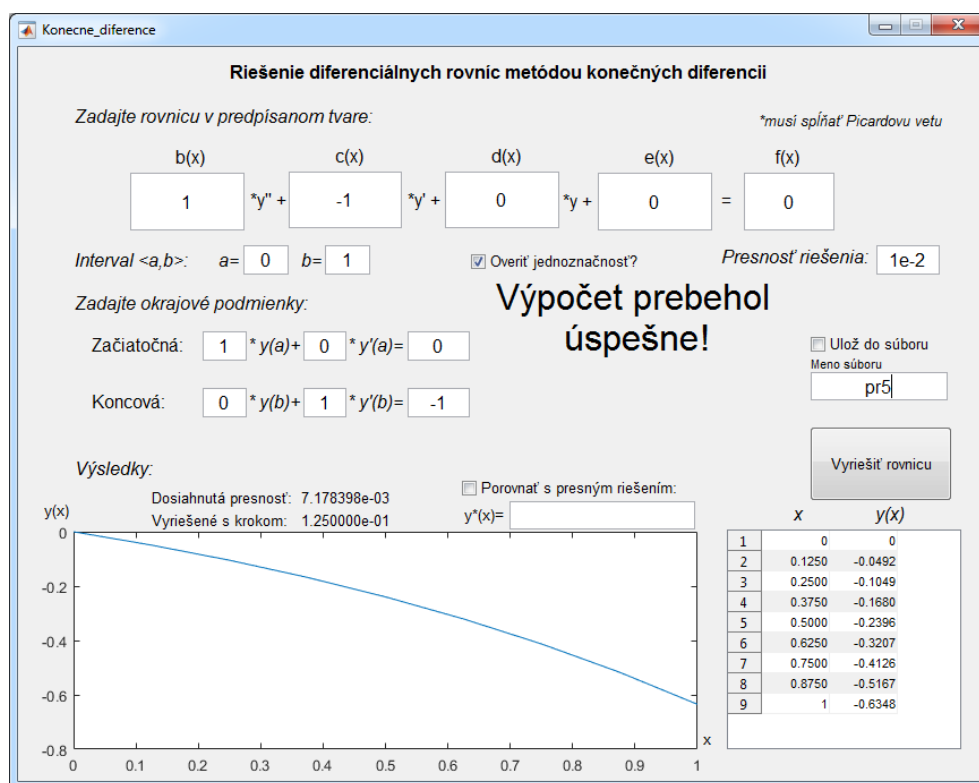
Príklad 5 je prevzatý z materiálov Stanford Univerzity [10]. Príklad je zadaný ako:

$$y'' - y' = 0, \quad (2.9)$$

s okrajovými podmienkami (Dirichletova a Neumannova):

$$y(0) = 0, \quad y'(1) = -1 \quad (2.10)$$

Tento príklad je riešený s presnosťou iba 10^{-2} . Už po dvoch iteráciách sa dosiahla presnosť $7,17 \cdot 10^{-3}$, takže výpočet sa zastavil s 8 rovnicami ako je vidieť na obr.2.9. Matica A bola správne vyhodnotená ako neostre diagonálne dominantná s minimálne jednou rovnicou splňujúcou vyššie uvedenú podmienku.



Obr. 2.9: Príklad 5, výpočet, metóda 2

3 ANALÝZA ZDROJOVÉHO KÓDU

V tejto kapitole budú rozobraté niektoré časti zdrojového kódu, ktoré si žiadajú dodatočné vysvetlenie. Väčšina kódu je komentovaná v nasledujúcej kapitole. V tejto kapitole tiež uvediem hodnoty niektorých premenných pre príklad 1.

3.1 Okrajové podmienky, 2 typy

Pre okrajové podmienky v kóde existujú dva prípady. Kód bol vytvorený iba pre Dirichletove podmienky a potom upravený aby podporoval aj všeobecné Sturmmove. Preto sú použité premenné `a_dirichlet` a `b_dirichlet`, ktoré rozlišujú tieto dva prípady. Ak je podmienka dirichletova, používa sa funkčná hodnota v premennej `ya` alebo `yb`. Inak sa používajú premenné s názvami `alfa`, `beta` a pre pravé strany $a_{RHS} = \gamma_1$ a $b_{RHS} = \gamma_2$. Zápis Sturmových podmienok zodpovedá rovnici 1.5.

Výpis 3.1: Kód určujúci podmienky, vo funkcii `kondiff_calc_v2`

```
a = podm_a(1);
a_alfa1 = podm_a(2);
a_alfa2 = podm_a(3);
if (a_alfa2 == 0)
    if (a_alfa1 == 0)
        msg='Zle_zadané_okraj_podm.';
        xres = 0;
        yres = 0;
        presnost_out = Inf;
        krok_out = Inf;
        return;
    end
    %podmienka nie je derivacia,
    %podmienka je dirichletova
    a_dirichlet = 1;
    ya=podm_a(4)/a_alfa1;
else
    a_dirichlet = 0;
    a_RHS = podm_a(4);
end
```

3.2 Koeficienty k

Koeficienty funkcie sú vyjadrené pomocou funkcie `coeffs` a uložené do premennej `c`. Koeficienty zodpovedajú koeficientom k_{-1}, k_0, k_1 z rovnice 1.10.

Výpis 3.2: Koeficienty (pre príklad 1), vo funkcii `kondiff_calc_v2`

```
[c ref] = coeffs(difeq,[y_ip y_i y_in]);  
-----výstup-----  
>>c= [ 2*h + 1/x, - 5*h^2 - 2/x, 1/x - 2*h, 2*x]  
>>ref=[ y_ip, y_i, y_in, 1]
```

3.3 Funkcia over jednoznačnosť

Jednoznačnosť riešenia úlohy sa overuje iba na 100 bodoch v rozsahu a do b , čo nie je ideálne riešenie, kvôli možným hodnotám medzi bodami tejto siete ktoré nemusia spĺňať podmienky neostrej dominantnosti. Preto je odporúčané si overiť jednoznačnosť úlohy pred použitím programu.

Výpis 3.3: Overovacia stiet, vo funkcii `over_jednoznacost`

```
x_span = linspace(a,b,n);  
%krok eval  
c_notH = subs(koeficienty,h,(b-a)/(n+1));  
diag_max = max(abs(eval(subs(c_notH(2),x_span))));  
sum_tri_max = max(abs(eval(subs(c_notH(1),x_span))+abs(subs(  
c_notH(3),x_span))));
```

3.4 Metódy riešenia sústavy rovníc

Pri tvorbe programu som porovnával dve metódy riešenia sústavy rovníc. Prvá (metóda 1) bola Gauss-Seidelova. V druhej metóde (metóda 2) je použitá funkcia `mldivide`. V dokumentácii k tejto funkcii [9] je uvedené že sa pre tridiagonálne matice používa solver, ktorý je optimalizovaný pre tento problém. Funkcia `mldivide` tiež funguje lepšie a rýchlejšie ako výpočet pomocou inverznaj matice ($Y = A^{-1} * F$), pretože inverzá matica je tu tvorená LU rozkladom (alebo LUL pre riedke matice) [8]. Pri metóde 2 je výpočet nastavený na 32 platných čísel a sleduje sa, či je tento predpoklad splnený pomocou čítania posledného varovania.

Výpis 3.4: Metoda 2, vo funkcii `kondiff_calc_v2`

```
%metoda 2  
Y1 = eval(vpa(mldivide(A,F'),32));
```

Metóda 1 používa Gauss-Seidelovu iteračnú metódu. Nastavenie presnosti výpočtu musí byť rádovo rozdielne (väčšia presnosť ako požadovaný výsledok) aby sa dala správne vyhodnotiť presnosť algoritmu konečných diferencií. Experimentálne bolo zistené, že deliteľ 200 je optimálny pre nastavenie presnosti. Táto metóda vykazovala oveľa horšie výsledky (niekedy aj >100x dlhší čas výpočtu) pre sústavy s počtom rovníc viac ako 64. Vo všeobecnosti bola metóda 1 vždy pomalšia ako metóda 2 (aj pre malý počet rovníc). Z tohto dôvodu bolo rozhodnuté ponechať vo finálnej verzii metódu 2 pre výpočet sústavy lineárnych rovníc.

Príklad 1 v sekcii 2.1.1 bol riešený s oboma metódami.

Výpis 3.5: Metoda 2, vo funkcii kondiff_calc_v2

```
%metoda 1
max_it_SG = 3000;
[Y1, successSG] = GaussSeidel(A,F',presnost/200,...
    max_it_SG,start_ya,start_yb);
```

4 ZDROJOVÝ KÓD

Program začína stlačením tlačítka "Vyrieš rovnicu".

Výpis 4.1: Zdrojový kód spustený po stlačení tlačidla

```
function solve_Callback(hObject, eventdata, handles)
set(handles.busy, 'String', 'Čakajte▯prosím▯▯Výpočet▯prebieha');
guidata(hObject, handles);
drawnow();
%extrahovat data z GIU
inputa = strcat('( ', get(handles.inputa, 'String'), ')');
inputy = strcat('( ', get(handles.inputy, 'String'), ')');
inputyd = strcat('( ', get(handles.inputyd, 'String'), ')');
inputydd = strcat('( ', get(handles.inputydd, 'String'), ')');
inputRHS = strcat('( ', get(handles.inputRHS, 'String'), ')');
a = str2double(get(handles.a, 'String'));
b = str2double(get(handles.b, 'String'));
a_RHS = str2double(get(handles.ya, 'String'));
b_RHS = str2double(get(handles.yb, 'String'));
a_alfa1 = str2double(get(handles.a_alfa1, 'String'));
a_alfa2 = str2double(get(handles.a_alfa2, 'String'));
b_beta1 = str2double(get(handles.b_beta1, 'String'));
b_beta2 = str2double(get(handles.b_beta2, 'String'));
presnost = str2double(get(handles.presnost, 'String'));
over_jednoznacnost = (get(handles.jednoznacnost, 'Value'));
%% overit ci su vstupy platne
if a>=b
    msg = 'a▯je▯vačšie▯alebo▯rovné▯b,▯obráť▯podmienky';
    set(handles.busy, 'String', msg);
    guidata(hObject, handles);
    return;
end
if (strcmp(inputydd, '(0)'))
    msg = 'Druhá▯derivácia▯musí▯buť▯nenulová!';
    set(handles.busy, 'String', msg);
    guidata(hObject, handles);
    return;
end
if presnost==0
    msg = 'Presnosť▯musí▯byť▯>0!';
    set(handles.busy, 'String', msg);
    guidata(hObject, handles);
```

```

    return;
end
%% definovat substituce, rovnice
syms x y_ip y_i y_in h
%substitucie * h^2
subs_y = '(y_i*h^2)';
subs_yd = '(((y_ip-y_in)/2)*h)'; %'((y_ip-y_i)*h)';
subs_ydd = '(y_ip-2*y_i+y_in)';

%vytvorit lavu stranu rovnice
LHS = strcat('( ',inputa,')+( ',inputy,')*',subs_y,'+...
□□( ',inputyd,')*',subs_yd,'+( ',inputydd,')*',subs_ydd);
difeq = eval(LHS);
difeq = expand(difeq);

%vytvorit pravu stranu rovnice
RHS = eval(strcat(inputRHS,'*h^2'));

%% vypocet metody, predat data funkci
max_iter = 30; %obmedzenie itracii konecných diferencii

%stara verzia, dokaze iba derichletove podmienky
%[xres, yres, presnost_out, krok_out, msg] = kondiff_calc...
%      (difeq,RHS,a,b,a_RHS,b_RHS,presnost,max_iter);

%nova verzia, dokaze vseobecne sturmове podmienky
podm_a = [a a_alfa1 a_alfa2 a_RHS];
podm_b = [b b_beta1 b_beta2 b_RHS];
[xres, yres, presnost_out, krok_out, msg] = kondiff_calc_v2...
    (difeq,RHS,podm_a,podm_b,presnost,max_iter,over_jednoznacost);

%vypis s akym krokom bolo vyriesene, a s akou presnostou
set(handles.krok_out,'String',sprintf('%e',krok_out));
set(handles.presnost_out,'String',sprintf('%e',presnost_out));

%porovnanie s presnym riesenim, vykresli sa 30 bodov
axes(handles.plot);
if (get(handles.exact,'Value'))
    xpres = linspace(a,b,30);
    syms x;
    %presne riesenie napisane v GUI

```

```

fexact = eval(get(handles.Fexact, 'String'));
for t=1:30
    y_presne(t) = eval(subs(fexact, x, xpres(t)));
end
plot(xres, yres, xpres, y_presne, 'kx');
%debug, na vypis chyb.
save('aktualny_vypocet', 'xres', 'yres', ...
    'presnost_out', 'fexact');
else %ak nebolo zvolene porovnanie s presnym riesenim,
    %vykresli sa iba graf funkcie y
    plot(xres, yres);
end
%nastav status spravu
set(handles.busy, 'String', msg);
%nastav data do tabulky
set(handles.table, 'Data', [xres yres]);

%uloz ak bolo zvolene ulozenie tak uloz do suboru
if (get(handles.save, 'Value'))
    filename = strcat(get(handles.filename, 'String'), '.csv');
    save_data(xres, yres, filename);
end
guidata(hObject, handles);

```

Výpis 4.2: Funkcia kondiff_calc_v2

```

function [xres, yres, presnost_out, krok_out, msg]...
    = kondiff_calc_v2(difeq, RHS, podm_a, podm_b, presnost, ...
        max_iter, over_jednoznacost)
%% logika na podmienky, dirichlet alebo obecne
a = podm_a(1);
a_alfa1 = podm_a(2);
a_alfa2 = podm_a(3);
if (a_alfa2 == 0)
    if (a_alfa1 == 0)
        msg='Zle_zadané_okraj._podm.';
        xres = 0;
        yres = 0;
        presnost_out = Inf;
        krok_out = Inf;
        return;
    end
end

```

```

    %podmienka nie je derivacia ,
    %podmienka je dirichletova
    a_dirichlet = 1;
    ya=podm_a(4)/a_alfa1;
else
    a_dirichlet = 0;
    a_RHS = podm_a(4);
end
b = podm_b(1);
b_beta1 = podm_b(2);
b_beta2 = podm_b(3);
if (b_beta2 == 0)
    if (b_beta1 == 0)
        msg='Zle zadanie okraj. podm.';
        xres = 0;
        yres = 0;
        presnost_out = Inf;
        krok_out = Inf;
        return;
    end
    b_dirichlet = 1;
    yb=podm_b(4)/b_beta1;
else
    b_dirichlet = 0;
    b_RHS = podm_b(4);
end
%% krok site , krok vzdy mensi ako 1
if ((b-a)>=2)
    %ak je rozsah vacsi ako 2 tak krok 0.5
    krok_start = 0.5;
else
    %ak je rozsah maly <2, krok stvrtina rozsahu
    krok_start = (b-a)/4;
end
%% extrahovanie koeficientov pre maticu
syms x y_ip y_i y_in h
%extrahovanie funkcie koeficientov pri
%yi+1. yi, yi-1 do vektrov c
[c ref] = coeffs(difeq,[y_ip y_i y_in]);
%c(1) je koeficient pri y_i+1
%c(2) je koeficient pri y_i

```

```

%c(3) je koeficient pri y_{i-1}
%koeficienty su funkcie h a x

%% jednoznacost?
if (over_jednoznacost)
    %overenie jednoznacnosti riessenia. Ci matica bude
    %diagonalne dominantna. overenie je iba na kontrolu
    %uzivatlskeho zadania. Uzivatel si MUSI overit
    %jednoznacnost riesenia. Tato funkcia nezarucuje
    %100% istotu ze sustava je riesitelna ak vyhodnoti
    %ze je riesitelna!
    if (over_jednoznacnost(c,a,b,100)==0)
        xres = 0;
        yres = 0;
        presnost_out = Inf;
        krok_out = Inf;
        msg = 'Nedá sa rozhodnúť jednoznačnosť riesenia!';
        return;
    end
end
%% samotny algoritmus na metodu.
krok_actual = krok_start;
for q=1:max_iter
    %subituce kroku
    c_actual = subs(c,h,krok_actual);%k-1,k1,k+1
    c_actual = eval(c_actual);
    RHS_actual = subs(RHS,h,krok_actual);%prava strana
    RHS_actual = eval(RHS_actual);
    %vyrvorenie site
    xh = a+krok_actual : krok_actual : b-krok_actual;
    n = length(xh); %pocet rovníc
    %vutvor tridiag maticu A
    diagn(1:n-1)= eval(subs(c_actual(3),x,xh(2:n)));
    diagp(1:n-1)= eval(subs(c_actual(1),x,xh(1:n-1)));
    diaghl(1:n)= eval(subs(c_actual(2),x,xh));
    A = diag(diagn,-1) + diag(diaghl) + diag(diagp,1);
    %prava strana, tiež dosadit sit
    F(1:n) = eval(subs(RHS_actual,x,xh));
    if (a_dirichlet == 1) %zaciatocna podm dirichletova
        F(1) = F(1) - eval(subs(c_actual(3),x,xh(1)))*ya;
    else %nie dirichletova

```



```

%pridat stlpec do matice A ako prvý, hodnota y(a)
%sa bude počítat v sustave
A_a = zeros(n,1); %stlpec
A_a(1) = eval(subs(c_actual(3),x,xh(1)));
A = [A_a A]; %pridanie do matice
%ROVNICA (1.16)
%pridat dalsiu rovniciu do sustavy na zaciatok
%a_RHS=gamma1,a_alfa1,a_alfa2 su k dispozicii
%rovnica (a_alfa1 - 3*a_alfa2/(2*h))*y_0 +
%(4*a_alfa2/(2*h))*y_1 + (-a_alfa2/(2*h))*y_2= a_RHS
a_LHS = zeros(1,n+1);
a_LHS(1) = (a_alfa1 - 3*a_alfa2/(2*krok_actual));
a_LHS(2) = (4*a_alfa2/(2*krok_actual));
a_LHS(3) = (-a_alfa2/(2*krok_actual));

%rovnica nastavena, teraz maticova uprava
%aby bola A tridiagonalna po pripojeni a_LHS
%na prvý riadok. cize vynulovat prvok a_LHS(3)
factor_to_tridiag = -a_LHS(3)/A(1,3);
%prvý riadok A sa vynasobi faktorom a pricita k a_LHS
a_LHS = a_LHS + (factor_to_tridiag.*A(1,:));
%to iste na pravej strane
a_RHS_actual = a_RHS + (factor_to_tridiag*F(1));

%rovnica je teraz pripravena na spojenie s A
A = [a_LHS;A];
F = [a_RHS_actual,F];

%inkrementuj n lebo som pridal novu rovniciu,
%riesenie bude aj pre zaciatocny bod y(a)
xh = [a xh];
n=n+1;
end

if (b_dirichlet == 1) %podobne ako so zaciatocnou
    F(n) = F(n) - eval(subs(c_actual(1),x,xh(n)))*yb;
else
    %pridat stlpec do matice A, pre koncový bod yb
    A_b = zeros(n,1);
    A_b(n) = eval(subs(c_actual(1),x,xh(n)));
    A = [A A_b];

```

```

%rovnicu  $(b\_beta2/(2*h))*y\_inn + (-4*b\_beta2/(2*h))$ 
%* $y\_in + (b\_beta1 + 3*b\_beta2/(2*h))*y\_i = b\_RHS$ 
b_LHS = zeros(1,n+1);
b_LHS(n+1) = (b_beta1 + 3*b_beta2/(2*krok_actual));
b_LHS(n) = (-4*b_beta2/(2*krok_actual));
b_LHS(n-1) = (b_beta2/(2*krok_actual));

%rovnicu nastavena, teraz maticova uprava aby bola
%A tridiagonalna po pripojeni b_LHS na posledny riadok.
factor_to_tridiag = -b_LHS(n-1)/A(n,n-1);
%posledny riadok A sa vynasobi faktorom a pricita k
b_LHS
b_LHS = b_LHS + (factor_to_tridiag.*A(n,:));
%to iste na pravej strane
b_RHS_actual = b_RHS + (factor_to_tridiag*F(n));
%rovnicu je teraz pripravena na spojenie s A
A = [A;b_LHS];
F = [F, b_RHS_actual];
%riesenie bude aj pre koncovy bod:  $y(b)$ 
xh = [xh b];
n=n+1;
end

%debug, vypis cislo iteracie a velkost matice
fprintf(1, 'Iteracia_kon.diff:%2d, velkost_matice: %5d\n', q, n
);

%DVE METODY RIESENIA SUSTAVY
metoda = 2;
if metoda ==1
    %#1 Gauss-Seidelova metoda
    if (a_dirichlet==1)&&(b_dirichlet==1)
        start_ya = ya; start_yb = yb;
    else
        start_ya = 0; start_yb = 0;
    end
    %disp('Cas Gauss-Sidel:'); tic
    max_it_SG = 3000;
    [Y1, successSG] = GaussSeidel(A,F', presnost/100, ...
        max_it_SG, start_ya, start_yb);
    %toc

```

```

if (successSG ==0)
    %popisane nizsie
    if (b_dirichlet == 0)b = [];yb = [];end
    if (a_dirichlet == 0)a = [];ya = [];end
    xres = [a xh b]';
    yres = [ya Y1' yb]';
    presnost_out = max(abs(abs(Y0) - abs(Y1(2:2:n))));
    krok_out = krok_actual;
    msg = ...
        sprintf('Gauss-Seidel_nekonverguje_po_%d_iter.!',
            ...
            ,max_it_SG);
    return;
end
else
    %#2 Matlab metoda
    %disp('Cas matlab /: ');tic;
    Y1 = eval(vpa(mldivide(A,F'),32));
    %toc;
    [msg,msgid]=lastwarn;
    if strcmp(msgid,'MATLAB:nearlySingularMatrix')
        xres = [a xh b]';
        yres = [ya Y1' yb]';
        if q>1
            presnost_out = max(abs(abs(Y0) - abs(Y1(2:2:n))
                ));
        else
            presnost_out = presnost;
        end
        krok_out = krok_actual;
        msg = ...
            sprintf('Vypocet_nedosiadol_presnost!');
        warning('Vypocet_skoncil');%na vymazanie chyby
        return;
    end
end
if (q==max_iter)
    msg = sprintf('Presnost_nedosiadnutá_po_%d_iteráciách!',q);
    presnost_out = max(abs(abs(Y0) - abs(Y1(2:2:n))));
    krok_out = krok_actual;
    if (b_dirichlet == 0)

```

```

        b = [];
        yb = [];
    end
    if (a_dirichlet == 0)
        a = [];
        ya = [];
    end
    xres = [a xh b]';
    yres = [ya Y1' yb]';
    return;
end

if (q>1) %vzdy sa musia vykonat aspon 2 iteracie ,
    %aby bolo ako vyhodnotit presnost; vyhodnotenie
    %Y0 je minuly vypocet, Y1 je aktualny
    %porovnanie v uzlovych bodoch, absolutna hodnota.
    if (a_dirichlet == 1)
        presnost_out = max(abs(abs(Y0) - abs(Y1(2:2:n))));
    else
        presnost_out = max(abs(abs(Y0) - abs(Y1(1:2:n))));
    end
    if (presnost > presnost_out)
        krok_out = krok_actual;
        if (b_dirichlet == 0)
            %bod b a riesenie je vo vektore xh a Y1,
            %takze sa nepridava z okrajovych podmienok
            b = [];
            yb = [];
        end
        if (a_dirichlet == 0)
            %bod a a riesenie je vo vektore xh a Y1,
            %takze sa nepridava z okrajovych podmienok
            a = [];
            ya = [];
        end
        xres = [a xh b]';
        yres = [ya Y1' yb]';
        msg = 'Vypocet prebehol uspesne!';
        return;
    end %presnost
end %q>1

```

```

%polovicny krok a opakuj vypocet
krok_actual = krok_actual/2;
Y0 = Y1; %aktualny vypocet kopia do minuleho
end %iteracie koecnych diferencii
end %function

```

Výpis 4.3: Funkcia GaussSeidel

```

function [Y, success] = GaussSeidel(A,b,presnost,max_it,ya,yb)
n = length(b);
%tu bude vysledok, ako zaciatočna
%hodnota je priamka prepojujúca okrajové podm
%ak nie sú dirichletové, vstup je 0,0
Y = linspace(ya,yb,n)';
chyba = 1e10*ones(n,1);

%% Ay=f riešim y
it = 0; %pocítadlo iterácii
while max(chyba) > presnost %opakuj pokiaľ nebude dosť presne
    it = it + 1; %na zastavenie nekonečnej slučky
    if (it > max_it)
        success = 0;
        return;
    end
    Xold = Y; %uložiť hodnoty aby sa dala porovnať chyba
    for i = 1:n %všetky rovnice
        j = 1:n; %pocítat cez všetky stĺpce
        j(i) = []; %okrem stĺpca i
        Xtemp = Y; %kopia
        Xtemp(i) = []; %nepočíta sa s xi
        Y(i) = (b(i) - sum(A(i,j) * Xtemp)) / A(i,i);
        %v ďalšej iterácii použije výsledky z
        %predchádzajúcej pre už vypočítané Y
    end
    chyba = abs(abs(Y) - abs(Xold));
end
success = 1; %konvergovalo s menej ako max_it iteráciami
end

```

Výpis 4.4: Funkcia save_data

```

function save_data(xres,yres,file_name)
    fileID = fopen(file_name,'w');

```

```
fprintf(fileID , 'x;y\n');  
fprintf(fileID , '%.7e;%.7e\r\n' ,[xres yres] );  
fclose(fileID);  
%type(file_name)  
end
```

ZÁVER

V tomto projekte bol vytvorený funkčný program na počítanie okrajovej úlohy metódou konečných difereencií, ktorý funguje do druhého rádu v 1D prípade. Zadaný príklad na vypočítanie má vplyv na rýchlosť výpočtu. Pre niektoré príklady, prostredie Matlab ukázalo chybu out of memory už pri počítaní približne 8000 rovníc. Tento fakt samozrejme závisí aj na parametroch počítača a architektúre procesora.

Projekt by mohol byť vylepšený o mnoho funkcií, ako napríklad optimalizácie overenie jednoznačnosti, pridanie možnosti riešenia rovníc až do 3 rádu (aproximácie v kapitole 2.3.2 v publikácii [11]), prípadne riešenie parciálnych rovníc, a tak ďalej.

LITERATÚRA

- [1] Jaromír Baštinec, Michal Novák : Moderní numerické metody, FEKT VUT Brno
- [2] Irena Růžičková, Rudolf Hlavička : Numerické metody, FSI VUT Brno
- [3] MÍKA, Stanislav, Petr PŘIKRYL a Marek BRANDNER. *Speciální numerické metody: numerické metody řešení okrajových úloh pro diferenciální rovnice* [online]. Plzeň: Vydavatelský servis, 2006 [cit. 2017-05-01]. Texty z aplikované matematiky. ISBN 80-868-4313-0. Dostupné z: <<http://home.zcu.cz/~mika/SNM2/SNM2.pdf>>
- [4] Germund Dahlquist, Ake Björck : Numerical Mathematics and Scientific Computation, Linköping, Sweden, 2007
- [5] Babuška, I. – Práger, M. – Vitásek. E.: Numerical Processes in Differential Equations. Praha, SNTL; London, Interscience 1966
- [6] Vitásek, E.: Numerické metody. Praha, SNTL 1987.
- [7] GUTERMUTH Denise. *Picard's Existence and Uniqueness Theorem* [online]. 2011 [cit. 2017-05-02]. Dostupné z: <<https://embedded.eecs.berkeley.edu/eecsx44/lectures/Spring2013/Picard.pdf>>
- [8] Inv, Matrix inverse. *MathWorks* [online]. [cit. 2017-05-02]. Dostupné z: <<https://www.mathworks.com/help/matlab/ref/inv.html>>
- [9] Mldivide, Solve systems of linear equations $Ax = B$ for x . *MathWorks* [online]. [cit. 2017-05-02]. Dostupné z: <<https://www.mathworks.com/help/matlab/ref/mldivide.html>>
- [10] *Numerical solution of boundary value problems* [online]. In: . Stanford, USA, 2002 [cit. 2017-05-02]. Dostupné z: <https://web.stanford.edu/~fringer/teaching/numerical_methods_02/handouts/lecture9.pdf>
- [11] ZHILIN, Li, Qiao ZHONGHUA a Tang TAO. *Numerical Solutions of Partial Differential Equations - An Introduction to Finite Difference and Finite Element Methods* [online]. 2011 [cit. 2017-05-02]. Dostupné z: <http://www4.ncsu.edu/~zhilin/TEACHING/MA587/fd_fem.pdf>
- [12] Gauss Seidel Method. *MathWorks* [online]. [cit. 2017-05-02]. Dostupné z: <<https://www.mathworks.com/matlabcentral/fileexchange/32051-gauss-seidel-method>>