Semestrální práce SN2

1 Nalezení vlastních čísel - metoda bisekce

1.1 Popis metody

Metoda bisekce se řadí mezi metody výpočtu vlastních čísel symetrické matice A. Přesněji řečeno, algoritmus počítá vlastní čísla matice HB, která představuje třídiagonální Hessenbergovu matici. Ta vznikla transformací matice A pomocí Householderových reflexí a je vůči ní podobná. Tento postup je možno provést, neboť platí, že každé dvě podobné matice mají stejná vlastní čísla.

V dalším kroku jsou identifikovány prvky hlavní diagonály a_i a prvky naddiagonály (resp. poddiagonály) b_i. Pomocí nich je rekurentním zápisem definována posloupnost polynomů $p_r(x)$, které pro symetrickou třídiagonální ireducibilní matici A mají *Sturmovu vlastnost*. Ta říká, že vlastní čísla matice A_{r-1} jsou ostře oddělena vlastními čísly matice A_r. Konečně, zavedeme-li $V(\lambda)$ jako počet znaménkových změn v posloupnosti $p_0(\lambda), p_1(\lambda), ...p_n(\lambda)$ a využijeme-li při určování $k-$tého největšího vlastního čísla λ_k poznatek *Geršgorinovy věty*, že $\lambda_k \in < a, b >$, můžeme pro danou symetrickou matici A při implementaci cyklů půlení intervalů nalézt všechna její vlastní čísla.

1.2 Některé další metody výpočtu

Pro určení vlastních čísel matice byla v minulosti vyvinuta řada algoritmů. Některé metody výpočtu si zde nyní stručně uvedeme.

Mocninná metoda umožňuje výpočet největšího vlastního čísla dané matice A. Jedná se o nejjednodušší z metod a její konvergence je lineární. Kromě základního algoritmu byla z důvodu hrozby podtečení a přetečení vyvinuta také jeho normalizovaná verze.

Konkurentem metody bisekce při řešení úloh se symetrickou maticí, případně s maticí v třídiagonálním tvaru, je *metoda Rayleighových podílů*. Aproximaci vlastního čísla hledáme pomocí řešení přeurčené soustavy rovnic metodou nejmenších čtverců. Konvergence metody je kvadratická, pro symetrické matice až kubická. Její nevýhodou je opakovaná faktorizace matice při řešení přeurčené soustavy; je-li však matice třídiagonální, je tento problém díky nenáročné faktorizaci překonán.

Jacobiho metoda je vhodná pro implementaci na paralelních počítačích. Spočívá ve využití Jacobiho matic rovinné rotace pro nulování mimodiagonálních prvků matice A. Metoda se snadno programuje a dokáže spočítat hledaná vlastní čísla s velkou přesností.

Nejrychlejším algoritmem pro výpočet všech vlastních čísel a vektorů symetrické matice je v současnosti třídiagonalizace následovaná *metodou rozděl a panuj (DaC)*. Princip spočívá v rozdělení velkého (rodičovského) problému na několik dílčích (dceřinných). Zpětně se pak vhodnou kombinací řešení dceřinných problémů dá sestavit řešení původního rodičovského problému. V algoritmu se využívá blokových matic a jejich spektrálních rozkladů. Úpravami se pak přejde na problém určení kořenů sekulárních rovnic, které představují vlastní čísla matice A.

Mezi další metody pro určení vlastních čísel matice se řadí například metoda inverzní iterace, metoda redukce, metoda simultánní iterace, QR metoda či metoda iterací v podprostorech s projekcí a její modifikace. Jejich popis je k nalezení například v textu *Vybrané statě z numerických metod* od pana docenta Libora Čermáka.

1.3 Vypracovaný program

V prostředí MatLab byl vytvořen program, jehož algoritmus realizuje výpočet vlastních čísel symetrické matice pomocí metody bisekce. Při tvorbě algoritmu byly použity poznámky a stručné úryvky kódu, které jsou k nalezení v textu *Vybrané statě z numerických metod*. Funkčnost programu byla mimo jiné ověřena na několika testovacích maticích a kód byl doplněn o potřebné korekce nutné pro výpočet vlastních čísel některých specifických matic (jednotková, nulová). V následující části textu se budu výhradně věnovat úloze ze zadání semestrální práce.

Zadaná matice A je tvaru

$$A = \begin{pmatrix} 3,75 & -0,25 & -1,25 & 2,75 \\ -0,25 & 2,75 & 2,75 & 1,25 \\ -1,25 & 2,75 & 4,75 & -0,25 \\ 2,75 & 1,25 & -0,25 & 7,75 \end{pmatrix}$$

, požadovaná přesnost výpočtu je 10^{-6}. Vzorové volání programu v prostředí MatLab:

```
[vlastnicisla]=eigvalOndruch;
```

, přičemž nalezená vlastní čísla se pro uživatele uloží do vektoru `vlastnicisla`. Vlastní skript pro MatLab opatřený detailním komentářem je k nalezení v zaslané příloze.

Poznámka: převod matice A na horní Hessenbergovu matici je ve skriptu prováděn pomocí funkce `hess`, která je již obsažena v MatLabu. Je-li to ovšem v rámci vypracování semestrální práce vyžadováno, lze převod provést

také pomocí skriptu `hessen`, který byl vytvořen v rámci jednoho z úvodních cvičení v předmětu SN2 a je rovněž obsažen v příloze. Řádek, který funkci `hessen` spouští, je v přiloženém kódu skriptu `eigvalOndruch` v poznámkovém režimu jako komentář.

1.4 Získané výsledky

Po spuštění programu v okně MatLabu se vypíší hodnoty `eigMatlab`, které představují vlastní čísla vypočítaná pomocí funkce `eig(A)`, a hodnoty `eigval`, tedy vlastní čísla vypočítaná pomocí naprogramované metody bisekce. Pro dané zadání jsou to konkrétně tyto hodnoty:

```
eigMatlab =
0.630252993894419
2.006188321565837
7.020968597378200
9.342590087161543

eigval =
0.630252850646194
2.006188282113691
7.020968871030584
9.342589583077212
```

Lze si povšimnout, že výsledky dosažené dvěma různými metodami se od sebe liší zcela nepatrně. Dá se usoudit, že výpočet metodou bisekce proběhl při dodržení zadané tolerance 10^{-6}.

Jelikož při testování programu nebylo objeveno zadání s maticí druhého a vyššího řádu, pro které by program při výpočtu selhal, lze považovat vypracovaný skript za funkční a připravený k praktickému použití, byť pro výpočty vlastních čísel matic vyšších řádů připadá v úvahu optimalizace kódu k dosažení nižší časové a paměťové náročnosti.

Je-li to nutné, mohu protokol doplnit o některé další dosažené dílčí výsldky výpočtu, případně jiné doplňující komentáře. V takovém přpadě mě prosím kontaktujte.

2 AB3-AM3-PECLE

2.1 Popis metody

Metoda *AB3-AM3-PECLE* se řadí mezi tzv. metody prediktor-korektor, které spadají do skupiny lineárních mnohokrokových metod pro řešení počátečních problémů. Zkratka *AB3* označuje prediktor, kterým je *Adams-Bashforthova metoda* třetího řádu, *AM3* je zkratkou pro korektor, kterým zde je *Adams-Moultonova metoda* třetího řádu.

Algoritmus metody bývá často označován pěticí písmen *PECLE*, kdy určení hodnoty y_{n+1} proběhne v pěti krocích. V úvodním kroku P se uskuteční předpověď (predikce) aproximace y^*_{n+1} pomocí AB metody a následně se v kroku E pro tuto předpověď vyhodnotí výraz $f_{n+1} = f(t_{n+1}, y^*_{n+1})$. V kroku C proběhne korekce předpovědi a s použitím výsledku předešlého kroku E je vyhodnocen výraz lokální extrapolace y^{**}_{n+1}. Ve čtvrtém kroku L provedeme výpočet *Milneova odhadu* lokální chyby est_n s pomocí dříve vypočítaných hodnot a nakonec položíme $y_{n+1} = y^{**}_{n+1} + est_n$. V závěrečném kroku E si předchystáme hodnotu f_{n+1} pro krok P v novém opakování cyklu.

Metoda *AB3-AM3-PECLE* je čtvrtého řádu, její oblast stability je větší než u prediktoru *AB3*, ale menší než u korektoru *AM3*. Modifikace obecné metody prediktor-korektor *ABk-AMk-PECLE* dále umožňují řízení délky kroku a řádu metody, těmito úpravami se však v rámci vypracovaného úkolu nebudeme podrobněji zabývat.

2.2 Některé další metody výpočtu

Nejzákladnějšími numerickými metodami pro řešení počátečních problémů jsou *Eulerovy metody*. Lze se setkat s explicitní Eulerovou metodou *EE* a implicitní Eulerovou metodou *IE*, dále také s lichoběžníkovou metodou. *EE* se vyznačuje malou oblastí absolutní stability, oproti tomu *IE* má oblast absolutní stability obrovskou. Její nevýhoda však může spočívat v nutnosti řešit obecně nelineární rovnice pro výpočet y_{n+1}.

Explicitní metody Runge-Kutta (RK) využívají ve svých formulích konstanty, které se zapisují do *Butcherovy tabulky*. Rozlišujeme řád a stupeň konkrétní RK metody, v praxi se běžně používá například metoda *Runge-Kutta-Bogacki-Shampine*, *Runge-Kutta-Fehlberg* nebo *Runge-Kutta-Dormand-Prince*. Řada těchto metod je také implementována v prostředí MatLab.

Pro řešení tuhých problémů je vhodné použít například *metody zpětného derivování* (zkratka *BDF*). Ty se vyznačují neomezenou oblastí absolutní stability. Metody spočívají v použití derivace interpolačního polynomu $P'_k(t_{n+1})$, která nahradí derivaci $y'_k(t_{n+1})$.

2.3 Vypracovaný program

V prostředí MatLab byl vytvořen program, jehož algoritmus realizuje řešení zadané soustavy diferenciálních rovnic pomocí metody *AB3-AM3-PECLE*, přičemž startovací hodnoty y_1 a y_2 jsou spočítány *Heunovou metodou*. Při tvorbě algoritmu byly použity poznámky a stručné úryvky kódu, které jsou k nalezení v textu *Vybrané statě z numerických metod*. Funkčnost programu byla mimo jiné ověřena na několika testovacích soustavách rovnic. V následující části textu se budu výhradně věnovat úloze ze zadání semestrální práce.

Vstupy ze zadání včetně soustavy rovnic a počátečních podmínek jsou uvedeny níže:

$$y_1' = -y_1 \cdot \sin\frac{1.4+t}{3} - y_2, \qquad y_1(0) = 2,$$

$$y_2' = \frac{1.6 \cdot t}{1+t^3} \cdot y_2 + 3 \cdot \cos(t), \qquad y_2(0) = 9.$$

Řešte na intervalu $\langle 0, 4\rangle$, pro dělení $N = 1000$.

Vzorové volání programu v prostředí MatLab:

```
[t,y]=pecleOndruch;
```

, přičemž vytvořený vektor časové složky se uloží do proměnné `t`, do pole `y` se uživateli uloží hledané funkční hodnoty. Vlastní skript pro MatLab opatřený detailním komentářem je k nalezení v zaslané příloze.

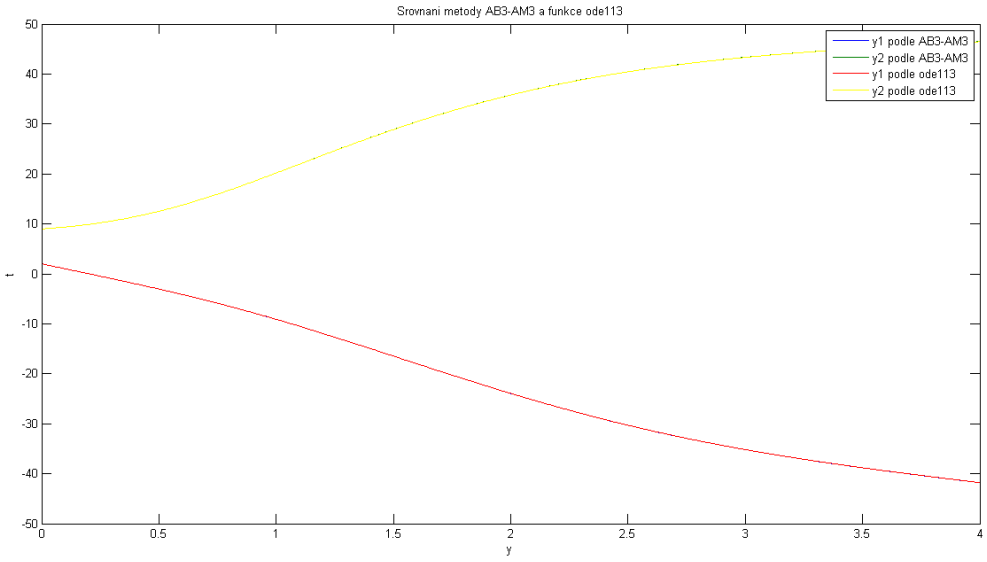
2.4 Získané výsledky

Po spuštění programu v okně MatLabu se v novém okně zobrazí graf řešení na zadaném intervalu, a to jak pro metodu AB3-AM3-PECLE, tak i pro příkaz `ode113`. Graf řešení pro vypracované zadání je včetně popisu a legendy zobrazen na obrázku 1. Lze si povšimnout, že rozdíly funkčních hodnot získaných dvěma numerickými metodami jsou velmi malé.

Číselně byly vypočítané funkční hodnoty srovnány v čase $t = 4$. V okně MatLabu se po spuštění skriptu zobrazí příslušné výsledky:

```
y1 v čase t=4 podle AB3-AM3-PECLE
-41.802246624378071

y1 v čase t=4 podle ode113
-41.793399774809927
```



Obrázek 1: Graf řešení

```
y2 v čase t=4 podle AB3-AM3-PECLE
 46.486696348450927

y2 v čase t=4 podle ode113
 46.474518810073100
```

Lze si povšimnout, že výsledky dosažené dvěma různými metodami se od sebe mírně liší. Příkaz Matlabu `ode113` ve skutečnosti používá metody *ABk-AM(k+1)-PECE* pro $k = 1, 2, ...12$, přičemž tak mění jak řád metody, tak i délku kroku, a je tedy oproti naprogramované metodě *AB3-AM3-PECLE* kvalitnější. Vzniklé odchylky v získaných funkčních hodnotách by tedy šlo eliminovat právě touto modifikací vytvořeného skriptu.

Vypracovaný skript lze považovat za funkční a připravený k praktickému použití, byť pro řešení složitějších a rozsáhlejších soustav připadá v úvahu optimalizace kódu k dosažení nižší časové a paměťové náročnosti.

Je-li to nutné, mohu protokol doplnit o některé další dosažené dílčí výsldky výpočtu, případně jiné doplňující komentáře. V takovém případě mě prosím kontaktujte.

3 Metoda konečných prvků

3.1 Popis metody

Metoda konečných prvků (MKP) se řadí mezi základní metody pro numerické řešení lineárních obyčejných diferenciálních rovnic druhého řádu, které jsou pro zadání okrajových úloh typické. Přestože se přednosti MKP dají plně ocenit zejména u úloh ve dvou a třech prostorových proměnných, lze metodu použít také pro řešení jednorozměrných úloh. Při diskretizaci okrajové úlohy se vychází ze slabé formulace okrajového problému

$$\begin{aligned} -[p(x)u']' + q(x)u &= f(x), \qquad x \in (0,\ell) \\ u(0) &= g_0, \\ -p(\ell)u'(\ell) &= \alpha_\ell u(\ell) - \beta_\ell. \end{aligned}$$

Řešení $u(x)$ slabé formulace daného okrajového problému se nazývá *slabé řešení*, u nějž je zaručena jak jeho existence, tak i jednoznačnost (narozdíl od *klasického řešení*). Na intervalu $< 0, \ell >$ zvolíme dělení, určíme délku největšího dílku dělení h a s pomocí *bázových funkcí* w_i zkonstruujeme po částech lineární funkci $U(x)$, která je přibližným řešením našeho problému a prochází body $[x_i, \Delta_i]$. S použitím *elementárních matic tuhosti* $\mathbf{K}^i$ a *elementárních vektorů zatížení* $\mathbf{F}^i$, které se určí podle specifických zápisů obsahujících členy $p(x), q(x), f(x)$ z rovnice zadaného okrajového problému, konstanty α_i, β_i z okrajových podmínek a délku dělení h, se vytvoří *globální matice tuhosti* $\mathbf{K}$ a *globální vektor zatížení* $\mathbf{F}$. Zohledníme-li dále známou vazbu Dirichletova typu s použitím *eliminačního postupu*, získáme soustavu lineárních rovnic

$$\mathbf{K\Delta} = \mathbf{F},$$

kdy matice $\mathbf{K}$ je třídiagonální. Vyřešením této soustavy obdržíme hodnoty $\Delta_i = U_i$, které představují hledané hodnoty funkce $U(x)$ v bodech x_i, poslední zbývající hodnotu (x_0 nebo x_N) získáme právě z Dirichletovy podmínky. V dalším postupu lze dále s pomocí korekce $\delta(x)$ určit tvar slabého řešení $u(x)$.

3.2 Některé další metody výpočtu

Pro numerické řešení jednorozměrných okrajových úloh se hojně používá *diferenční metoda*, někdy také zvaná jako *metoda sítí*. Metoda spočívá v nahrazení derivace *diferenčním podílem*, což vede na lineární soustavu rovnic tvaru

$$\mathbf{KU} = \mathbf{F},$$

kterou řešíme pro neznámé hodnoty v uzlových bodech $U_1, U_2, ...U_N$. Matice $\mathbf{K}$ je pozitivně definitní a její tvar je dán členy rovnice $p(x), q(x)$ a délkou dělení h, vektor $\mathbf{F}$ navíc zahrnuje také člen $f(x)$ pravé strany rovnice. Modifikací lze řešit také rovnici s konvekčním členem, která je tvaru

$$-[p(x)u']' + r(x)u' + q(x)u = f(x), \qquad x \in (0, \ell).$$

Jinou metodou pro řešení okrajových úloh je *metoda konečných objemů* (MKO). Její princip spočívá v integraci modelové rovnice přes tzv. *buňku*, kdy dostáváme *bilanční rovnici*. Aproximací derivace pomocí centrální diference, použitím obdélníkové formule pro spočtení integrálů v rovnici a po zahrnutí okrajových podmínek přejdeme po zanedbání chyby na soustavu lineárních rovnic, jejímž výstupem jsou hledaná přibližná řešení U_i. Tato soustava má stejný tvar jako v případě diferenční metody.

3.3 Vypracovaný program

V prostředí MatLab byl vytvořen program, jehož algoritmus realizuje řešení zadaného okrajového problému pomocí metody konečných prvků. Při tvorbě algoritmu byly použity poznámky a stručné úryvky kódu, které jsou k nalezení v textu *Vybrané statě z numerických metod*. Funkčnost programu byla mimo jiné ověřena na několika zadáních rovnic a okrajových podmínek. V následující části textu se budu výhradně věnovat úloze ze zadání semestrální práce.

Vstupy ze zadání zahrnující rovnici, okrajové podmínky a krok dělení jsou uvedeny níže:

Metodou konečných prvků s krokem $h = 0.1$ vyřešte okrajový problém:

$$-\left[(7 \cdot x^2 + 1) \cdot y'\right]' + \frac{5.5}{x+1} \cdot y = \cos(x^2),$$

$$y(0) = 12, \qquad y'(3) = -6.3 \cdot y(3) + 3.46\,.$$

Vzorové volání programu v prostředí MatLab:

```
[x,delta]=mkpOndruch;
```

, přičemž vytvořený vektor uzlových bodů se uloží do proměnné `x`, do pole `delta` se uživateli uloží hledané funkční hodnoty v uzlových bodech.

Vlastní skript pro MatLab opatřený detailním komentářem je k nalezení v příloze.

3.4 Získané výsledky

3.4.1 Úkol a)

Srovnáním přiděleného zadání s obecným tvarem modelové úlohy okrajového problému byly určeny tvary vystupujících členů:

$$p(x) = 7x^2 + 1$$
$$q(x) = \frac{5,5}{x+1}$$
$$f(x) = \cos(x^2)$$
$$g_0 = 12$$
$$\alpha_\ell = 403,2$$
$$\beta_\ell = 221,44,$$

přičemž hodnoty α_ℓ, β_ℓ byly po úpravě odvozeny na papíře.

3.4.2 Úkol b)

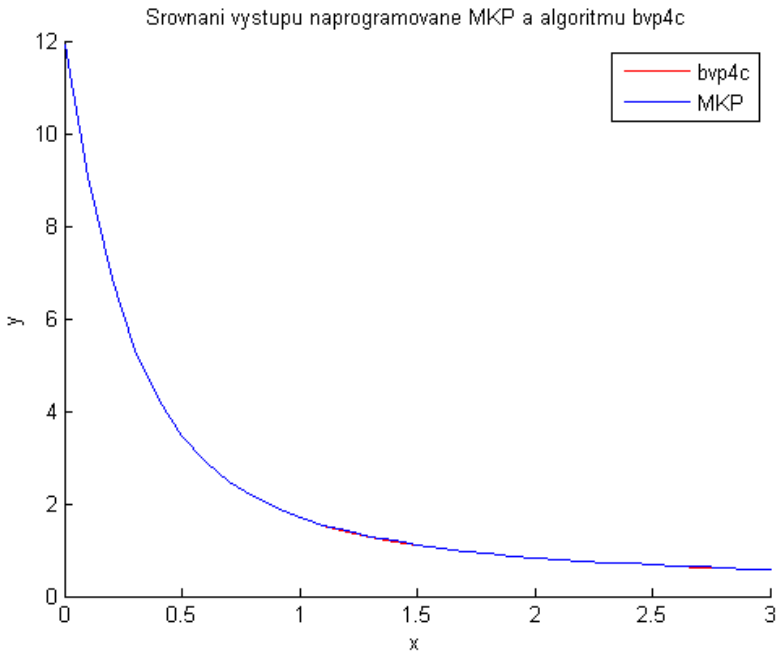
Pro přidělené zadání, pro nějž je Dirichletova podmínka předepsána v počátečním bodě $x = 0$, byl v rámci eliminačního postupu v soustavě $\mathbf{KU} = \mathbf{F}$ vynechán její první řádek a první sloupec. Ze zadané podmínky tedy dostáváme $\Delta_0 = g_0 = 12$ a z řešení soustavy rovnic pak získáme hodnoty $\Delta_1, ...\Delta_N$, kde $N = \frac{(b-a)}{h} = \frac{(3-0)}{0,1} = 30$.

3.4.3 Úkol c)

Po spuštění programu v okně MatLabu se v novém okně zobrazí graf řešení na zadaném intervalu, a to jak pro metodu konečných prvků, tak i pro příkaz `bvp4c`. Graf řešení pro vypracované zadání je včetně popisu a legendy zobrazen na obrázku 2.

Číselně byly vypočítané funkční hodnoty jak pro naprogramovanou metodu MKP, tak pro skript `bvp4c`. V okně MatLabu se po spuštění skriptu zobrazí příslušné výsledky:

```
delta =
12.000000000000000
```



Obrázek 2: Graf řešení

```
9.068264365997905
6.874201697056807
5.319728158237370
4.232532435937489
3.461858643064394
2.901810599476975
2.483434330935295
2.162529252182736
1.910475816488530
1.708361923717443
1.543390606121309
1.406692391667738
1.291975588273761
1.194676349109845
1.111410917809699
1.039614355478807
0.977297287290036
0.922879631232747
0.875076374909345
```

```
0.832819970750450
0.795209443409840
0.761479324712967
0.730982945293523
0.703185045244785
0.677658655991340
0.654081272525247
0.632225965127751
0.611944626335811
0.593143082873036
0.575750967142285

bvp4cRes =
12.000000000000000
9.071850140898119
6.875064203182691
5.316117853345485
4.225631779475138
3.453184526719275
2.892415025801927
2.473928479369777
2.153247251889988
1.901599504026780
1.699994270310596
1.535594758066909
1.399510854608507
1.285438430265510
1.188803410233211
1.106210856337455
1.035082273910981
0.973412171461393
0.919602596974917
0.872350560740678
0.830572718718291
0.793357095259200
0.759934494025048
0.729663532145152
0.702023631470726
0.676610418945379
0.653128391386806
0.631376862970008
```

```
0.611227392008685
0.592593954156175
0.575400478143664
```

Dále se pro uživatele vypíše také maximální absolutní odchylka mezi řešeními získanými dvěma postupy, kdy pro dané zadání vyšlo

```
res =
0.003585774900214
```

Jak hodnota maximální absolutní odchylky, tak i vizuální srovnání obou vykreslených grafů řešení napovídají, že získané hodnoty se od sebe liší nepatrně. Vyšší přesnosti by dále mohlo být dosaženo při použití jemnějšího dělení; v takovém případě je však nutno počítat s vyššími nároky na paměť počítače a čas výpočtu.

Vypracovaný skript lze považovat za funkční a připravený k praktickému použití, byť pro řešení složitějších úloh připadá v úvahu optimalizace kódu k dosažení nižší časové a paměťové náročnosti.

Je-li to nutné, mohu protokol doplnit o další teoretické poznatky z oblasti metody konečných prvků, dodat některé další dosažené dílčí výsldky výpočtu, případně jiné doplňující komentáře. V takovém případě mě prosím kontaktujte.