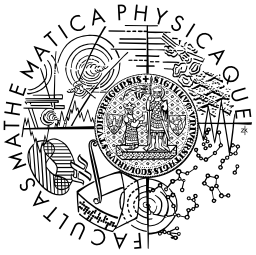

MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Ondřej Král

Intervalové lineární soustavy rovnic s lineárními závislostmi

Katedra aplikované matematiky

Vedoucí bakalářské práce: doc. Mgr. Milan Hladík, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2016

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Intervalové lineární soustavy rovnic s lineárními závislostmi

Autor: Ondřej Král

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: doc. Mgr. Milan Hladík, Ph.D., Katedra aplikované matematiky

Abstrakt: Tento text pojednává o hledání obálek řešení soustav intervalových lineárních rovnic s lineárně závislými parametry. Seznámíme se s pojmy intervalové aritmetiky a analýzy na ní postavené. Rozšíříme je na matice a lineární soustavy, kde představíme a tematicky rozdělíme nejnovější postupy, pomocí nichž se dá najít obálka jejich řešení. Většinu z nich naimplementujeme ve vývojovém prostředí MATLAB za pomoci intervalové knihovny INTLAB. Porovnáme jejich rychlost a přesnost na Toeplitzových, symetrických a náhodných maticích. Parametrům navrhneme vlastní úspornou datovou reprezentaci. Výsledky zanalyzujeme a vytvoříme jedinou funkci, která je zastřešuje a kterou bude moci uživatel používat, ať už se rozhodne pro rychlé, přesné, nebo paměťově úsporné výpočty.

Klíčová slova: lineární systémy, závislosti, obálka, MATLAB, INTLAB

Title: Interval linear system with linear dependencies

Author: Ondřej Král

Department: Department of Applied Mathematics

Supervisor: doc. Mgr. Milan Hladík, Ph.D., Department of Applied Mathematics

Abstract: The main problem discussed in this thesis is about finding an enclosure of the solution set of an interval linear system with linear dependencies. We get familiar with definitions from interval arithmetic and analysis. Then we extend them to matrices and linear systems, where we introduce several modern approaches to finding an enclosure and divide them thematically. Most of them are implemented in MATLAB using INTLAB library. We compare their precision and computational time on Toeplitz, symmetric and random matrices. For dependencies we design our memory saving representation. The results are interpreted and the final function, which can compute either fast, sharp or memory efficient, is build on individual methods.

Keywords: linear systems, dependencies, enclosure, MATLAB, INTLAB

Rád bych poděkoval svému vedoucímu, doc. Mgr. Milanovi Hladíkovi, Ph.D., za ochotu při konzultacích, za připomínky ke struktuře textu a celkově za všechen čas, který průběžně během tří let strávil tím, aby mě a ostatní zasvětil do matematiky, ze které vychází tato práce.

Obsah

Úvod

Výpočty s vektory, maticemi a řešení soustav, které jsou jimi zadané, nalezneme dnes v nějaké formě téměř v každé oblasti lidské činnosti. Výrobci procesorů a grafických karet dodávají se svým hardware knihovny, které umožňují na dané architektuře provádět tyto výpočty co nejrychleji. Vedle požadavku na rychlost je zde i požadavek na přesnost. Zatímco při teoretických výpočtech pracujeme se spojitými výrazy s libovolnou přesností, počítač je diskrétní a konečný stroj. V tomto případě se často mluví o zobrazení množiny reálných čísel do množiny float-point čísel, tedy těch, které se dají uložit v paměti počítače, ale pouze s omezenou přesností. Představíme-li si float-point množinu jako diskrétní body na reálné přímce, pokus o uložení reálného čísla, které se nachází mezi dvěma nejbližšími float-point body, vede k zaokrouhlení a k ztotožnění s jedním z nich. Také samotné operace s float-point čísly v počítači vedou ke vzniku chyb, které se ale dají kompenzovat tzv. hardwarovými error-free transformacemi (Manoukian a Constantinides, 2011), protože takto vzniklá chyba je reprezentovatelná float-point číslem. K odhadu tímto způsobem vzniklých chyb existují různé přístupy. V naší práci se seznámíme s jedním z nich, ale výsledek je ve finále stejný. Dostaneme určitý rozsah, v němž se nachází správná hodnota. Důležitá není nejistota uvnitř rozsahu, ale jistota mimo něj.

Intervalová analýza je oblast numerické matematiky, která místo počítání s konkrétními reálnými čísly používá jejich uzavřené intervaly. Cílem je modelovat určité nepřesnosti, které můžou nastat během měření, nebo během výpočtů. Například, hodnota gravitačního zrychlení závisí na poloze měření, ale škála hodnot, kterých může nabývat, se dá uzavřít do relativně úzkého intervalu, stejně tak i hmotnost izotopu uhlíku, nebo rozměry kovové součástky během tepelných změn.

Při implementaci numerických metod a při jejich běhu na počítači se potýkáme se zaokrouhlováním reálných čísel do prostoru reprezentovatelného počítačem. Pomocí intervalů ale můžeme reálné číslo uzavřít mezi dvěma nejbližšími, počítačem reprezentovatelnými hodnotami. Při práci s takovými hodnotami většinou chceme, aby byl výpočet tzv. verifikovaný, tedy abychom měli jistotu, že se přesný výsledek nachází v námi spočítaném intervalu. To je velice důležité u počítačem asistovaných důkazů, i když jsou stále pochybnosti o korektnosti takových postupů, hlavně s ohledem na spolehlivost elektroniky a teoreticky možnou systematickou chybu při opakovaném běhu programu.

Také bychom si měli uvědomit, že intervalové metody nejsou jen klasické metody rozšířené na intervaly a že často takto naivní přístup selhává a dochází k nepoužitelným výsledkům.

Cílem této práce je obohatit fakultní projekt, zabývající se intervalovými výpočty ve vývojovém prostředí MATLAB za použití knihovny INTLAB, o další nové metody a přístupy z oblasti lineárních systémů s lineárně závislými parametry a tím vyplnit určitou mezeru, která se v současnosti v MATLABu nachází.

1. Intervalová analýza

1.1 Intervalová aritmetika

Základem intervalové analýzy je interval a na něm definovaná aritmetika. Množinu všech reálných intervalů značíme jako $\mathbb{IR}$.

Definice 1. *Interval* $\boldsymbol{a}$ *je množina definována jako*

$$\boldsymbol{a} := [\underline{a},\bar{a}] = [a^c - a^\Delta, a^c + a^\Delta] = \{a; \underline{a} \leq a \leq \bar{a}\},$$
$$a^c := (\bar{a} + \underline{a})/2, \;\; a^\Delta := (\bar{a} - \underline{a})/2.$$

Následující definice pro základní operace s intervaly se dá popsat tak, že nehledě na konkrétní hodnoty, kterých operandy nabývají, chceme mít jistotu, že výsledek jejich operace náleží nějaké hodnotě z výsledného intervalu.

Definice 2. *Mějme intervaly* $\boldsymbol{a} \in \mathbb{IR}$ *a* $\boldsymbol{b} \in \mathbb{IR}$, $0 \notin \boldsymbol{b}$, *pak*

$$\boldsymbol{a} \circ \boldsymbol{b} := \{a \circ b; a \in \boldsymbol{a}, b \in \boldsymbol{b}\}.$$

Pro konkrétní operace snadno odvodíme přímý výpočet

$$\begin{aligned}
\boldsymbol{a} + \boldsymbol{b} &= [\underline{a} + \underline{b}, \bar{a} + \bar{b}],\\
\boldsymbol{a} - \boldsymbol{b} &= [\underline{a} - \bar{b}, \bar{a} - \underline{b}],\\
\boldsymbol{ab} &= [\min(\underline{a}\underline{b}, \bar{a}\bar{b}, \underline{a}\bar{b}, \bar{a}\underline{b}), \max(\underline{a}\underline{b}, \bar{a}\bar{b}, \underline{a}\bar{b}, \bar{a}\underline{b})],\\
\boldsymbol{a}/\boldsymbol{b} &= [\min(\underline{a}/\underline{b}, \bar{a}/\bar{b}, \underline{a}/\bar{b}, \bar{a}/\underline{b}), \max(\underline{a}/\underline{b}, \bar{a}/\bar{b}, \underline{a}/\bar{b}, \bar{a}/\underline{b})].
\end{aligned}$$

Následuje pár základních definic pro úplnost některých tvrzení v textu.

Definice 3. *Magnituda intervalu* $\boldsymbol{a}$

$$\mathrm{mag}(\boldsymbol{a}) := \max_{a \in \boldsymbol{a}}\{|a|\}.$$

Definice 4. *Absolutní hodnota intervalu* $\boldsymbol{a}$

$$|\boldsymbol{a}| := \{|a|; a \in \boldsymbol{a}\}.$$

Poslední zmínka k aritmetice, kterou budeme potřebovat, je o zákonu distributivity. Máme-li reálná čísla $a,b,c \in \mathbb{R}$, víme, že platí

$$c(a + b) = ca + cb.$$

Bohužel, při přímém výpočtu takového výrazu v intervalové aritmetice nedokážeme ve výrazu $\boldsymbol{ca} + \boldsymbol{cb}$ zachytit závislost hodnot $\boldsymbol{c}$ v obou členech a v podstatě počítáme s výrazem, kde jsou proměnné zcela nezávislé: $\boldsymbol{c}_1\boldsymbol{a} + \boldsymbol{c}_2\boldsymbol{b}$. Tím dochází ke zbytečnému přehodnocení intervalu. Platí ale jiný zákon, tzv. *subdistributivita*.

Tvrzení 1. *Mějme* $\boldsymbol{a},\boldsymbol{b},\boldsymbol{c} \in \mathbb{IR}$, *pak lze snadno ukázat, že*

$$\boldsymbol{c}(\boldsymbol{a} + \boldsymbol{b}) \subseteq \boldsymbol{ca} + \boldsymbol{cb}.$$

1.2 Obal a obálka

Často hodnoty některých funkcí na intervalech nejsou opět intervaly. Aby se nám lépe s těmito hodnotami počítalo, "obalíme"je intervalem a dále počítáme pouze s ním. Více rozměrným intervalům říkáme *boxy*. Obecně na tento problém narazíme kdykoliv, kdy potřebujeme počítat s množinou bodů, která není jednoduše popsatelná, nebo dokonce není ani spojitá.

Definice 5. *Obálka množiny* $\mathbf{\Sigma} \subseteq \mathbb{R}^n$ *je libovolný box* $\boldsymbol{y} \in \mathbb{IR}^n$*, pro který platí*

$$\mathbf{\Sigma} \subseteq \boldsymbol{y}.$$

Definice 6. *Intervalový obal množiny* $\mathbf{\Sigma} \in \mathbb{R}^n$ *je box* $\boldsymbol{z} \in \mathbb{IR}^n$ *takový, že*

$$\boldsymbol{z} = \bigcap_{\boldsymbol{y} \in \mathbb{IR}^n; \mathbf{\Sigma} \subseteq \boldsymbol{y}} \boldsymbol{y}.$$

Tedy obal je nejmenší možná obálka obsahující celé $\mathbf{\Sigma}$. Značíme $\square\mathbf{\Sigma}$. Metody pro přesný výpočet obalu obecné funkce jsou výpočetně těžké. V této práci se zabýváme pouze lineárními funkcemi ve formě matic a vektorů, kde je spousta NP-těžkých úloh (Rohn, 1994). Naším cílem je místo přesného obalu hledání co možná nejtěsnější obálky, kterou dovedeme spočítat v polynomiálním čase.

1.3 Intervalové matice

Intervalovou aritmetiku lze přirozeně rozšířit na matice. Zadefinujeme je stejně jako vektory a přidáme další definice potřebné k popsání našeho problému. V následující definicích bereme $\boldsymbol{A} \in \mathbb{IR}^{n \times n}$.

Definice 7. *Intervalová matice* $\boldsymbol{A}$ *je množina definována jako*

$$\boldsymbol{A} := [\underline{A}, \bar{A}] = \{A; \underline{A} \leq A \leq \bar{A}\}.$$

Definice 8. *Intervalová matice* $\boldsymbol{A}$ *je regulární* $\iff \forall A \in \boldsymbol{A} : A$ *je regulární.*

Definice 9. *Není-li intervalová matice* $\boldsymbol{A}$ *regulární, řekneme, že je iregulární.*

Definice 10. *Inverz regulární matice* $\boldsymbol{A}$ *se definuje následovně*

$$\boldsymbol{A}^{-1} := \{A^{-1}; \underline{A} \leq A \leq \bar{A}\}.$$

Taktéž výpočet $\square\boldsymbol{A}^{-1}$ je NP-těžký a počítáme místo toho s těsnou obálkou.

Definice 11. *O intervalové matici* $\boldsymbol{A}$ *řekneme, že je nezáporná, pokud*

$$\forall A \in \boldsymbol{A} : A \geq 0.$$

Definice 12. *Spektrální poloměr matice* A *značíme* $\rho(A)$ *a definujeme*

$$\rho(A) := \max\{|\lambda|; \lambda \text{ je vlastní číslo matice } A\}$$

Následující tvrzení jsou obecně známé důsledky z lineární algebry.

Tvrzení 2. *Je-li* $\boldsymbol{A} = [\underline{A},\bar{A}]$ *nezáporná a* $\rho(A)$ *spektrální poloměr matice* A*, pak*

$$\rho(\boldsymbol{A}) = [\rho(\underline{A}),\rho(\bar{A})]$$

Tvrzení 3. *Buď* $C \in \mathbb{R}^{n\times n}$ *a* $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ *, pak platí*

$$C\boldsymbol{A} \text{ je regulární} \Longrightarrow C \text{ je regulární a } \boldsymbol{A} \text{ je regulární.}$$

Jednu postačující podmínku pro regularitu intervalové matice si uvedeme i s důkazem, neboť jeho konstrukce poskytuje užitečný náhled do naší problematiky.

Věta 4. *(Beeck, 1975). Pokud* $\rho(|A^c|^{-1}A^{\Delta}) < 1 \implies \boldsymbol{A}$ *je regulární.*

Důkaz. Zadefinujme $\boldsymbol{M} := (A^c)^{-1}\boldsymbol{A}$. Tedy $\boldsymbol{M} = [I - |(A^c)^{-1}|A^{\Delta}, I + |(A^c)^{-1}|A^{\Delta}]$. Dostáváme $\forall M \in \boldsymbol{M} : |M - M^c| = |M - I| \leq |(A^c)^{-1}|A^{\Delta}$. Z teorie vlastních čísel pak plyne $\rho(M - I) \leq \rho(M^{\Delta}) < 1$. Odečtení I od M nám posune všechna vlastní čísla v komplexní rovině o 1 směrem k záporné části reálné osy. V kombinaci s nerovností $\rho(M - I) < 1$ vidíme, že M nemá nulové vlastní číslo a je tedy regulární. Výše uvedené platí pro všechna $M \in \boldsymbol{M}$. $\boldsymbol{M}$ je regulární a z tvrzení 3 je i $\boldsymbol{A}$ regulární.

□

Pokud matici $\boldsymbol{A}$ vynásobíme $C := (A^c)^{-1}$, dostaneme

$$\boldsymbol{M} := C\boldsymbol{A} = [I_n - |(A^c)^{-1}|A^{\Delta}, I_n + |(A^c)^{-1}|A^{\Delta}],$$

která nám spolu s tvrzením 3 pomůže rozhodnout regularitu. Test

$$\rho((|(A^c)^{-1}|A)^{\Delta}) < 1$$

nám nestačí, protože $(A^c)^{-1}$ nedokážeme spočítat přesně a místo toho počítáme s hodnotou $C \approx (A^c)^{-1}$. Střed $\boldsymbol{M}$ potom není I_n, ale jeho aproximace a pro jistotu použijeme silnější podmínku

$$\rho(\mathrm{mag}(I_n - C\boldsymbol{A})) < 1$$

která sice není silnější teoreticky, ale z pohledu omezené přesnosti ano.

Všechny zmíněné definice a tvrzení využijeme v další kapitole, kde přiblížíme problém, kterým se implementované funkce z této práce zabývají.

2. Lineární soustavy

2.1 Základní pojmy

Mějme $A \in \mathbb{R}^{n\times n}$ a x, $b \in \mathbb{R}^n$. *Soustavu lineárních rovnic*, nebo také *systém lineárních rovnic*, zapisujeme ve tvaru

$$Ax = b,$$

kde x je neznámým řešením. Pokud se začneme na hodnoty A a b dívat jako na intervaly, tedy zavedeme $\boldsymbol{A} \in \mathbb{IR}^{n\times n}$ a $\boldsymbol{b} \in \mathbb{IR}^n$, dostaneme nespočetně mnoho systémů:

$$Ax = b; A \in \boldsymbol{A}, b \in \boldsymbol{b},$$

zkráceně značených

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

Řešení těchto systémů zadefinujeme jako

$$\Sigma := \{x \in \mathbb{R}^n; \exists A \in \boldsymbol{A}\ \exists b \in \boldsymbol{b} : Ax = b\}.$$

V mnoha problémech nám místo komplikované reprezentace této množiny stačí nalézt interval $\boldsymbol{x} \in \mathbb{IR}^n$, pro který platí $\Sigma \subseteq \boldsymbol{x}$. Zajímají nás problémy, kdy je $\boldsymbol{A}$ regulární a řešení lze obalit konečným intervalem. Jelikož výpočet $\square\Sigma$ je NP-těžký, tak se spokojíme s těsnou obálku. Pomocí zkoumání vlastností Σ a z různých teoretických výsledků s tím spojených, dostáváme metody, které v některých speciálních případech lépe či hůře takové obálky naleznou.

2.2 Charakterizace řešení

V této práci je většina přímých metod pro výpočet obálky Σ založena na charakterizaci od Oettliho a Pragera.

Věta 5. *(Oettli a Prager, 1964). Množina řešení Σ se dá popsat nerovnicemi*

$$|A^c x - b^c| \le A^\Delta |x| + b^\Delta.$$

Důkaz. Mějme $x \in \Sigma$ takové, že $\exists A \in \boldsymbol{A}\ \exists b \in \boldsymbol{b} : Ax = b$.

$$|A^c x - b^c| = |(A^c - A)x + (Ax - b) + (b - b^c)| = |(A^c - A)x + 0 + (b - b^c)| \le$$
$$\le |(A^c - A)x| + |b - b^c| \le |A^c - A||x| + |b - b^c| \le A^\Delta |x| + b^\Delta.$$

Naopak, $x \in \mathbb{R}^n$ splňuje $|A^c x - b^c| \le A^\Delta |x| + b^\Delta$. Zadefinujeme $y \in [-1,1]^n$

$$y_i := \begin{cases} \dfrac{(A^c x - b^c)_i}{(A^\Delta |x| + b^\Delta)_i}, & \text{pokud je } (A^\Delta |x| + b^\Delta)_i > 0, \\ 1, & \text{jinak.} \end{cases}$$

Nyní můžeme psát

$$(A^c x - b^c)_i = y_i (A^\Delta |x| + b^\Delta)_i,$$

agregovaně

$$A^c x - b^c = \mathrm{diag}(y)(A^\Delta |x| + b^\Delta).$$

Buď $z := \mathrm{sgn}(x)$, pak $|x| = \mathrm{diag}(z)x$. Po dosazení

$$A^c x - b^c = \mathrm{diag}(y)A^\Delta \mathrm{diag}(z)x + \mathrm{diag}(y)b^\Delta,$$

$$(A^c - \mathrm{diag}(y)A^\Delta \mathrm{diag}(z))x = b^c + \mathrm{diag}(y)b^\Delta.$$

Snadno nahlédneme, že

$$A^c - \mathrm{diag}(y)A^\Delta \mathrm{diag}(z) \in \boldsymbol{A}, \;\; b^c + \mathrm{diag}(y)b^\Delta \in \boldsymbol{b},$$

takže $x \in \Sigma$.

□

Na základě důkazu věty 5 se dá odvodit tvrzení

Tvrzení 6. *(Beeck, 1973). Σ tvoří s průnikem každého ortantu konvexní polyedr.*

Tento fakt využívají některé algoritmy, které prochází jednotlivé ortanty prostoru a počítají zvlášť obal každé konvexní části. Tyto metody jsou ze své podstaty exponenciální, ale ne v každém případě. Více se o tomto tématu a dalších přístupech dozvíme v sekci 3.7.

2.3 Soustavy s parametry

Často nejsou koeficienty intervalové matice $\boldsymbol{A}$ zcela nezávislé. My se budeme zabývat případem, kdy jsou některé z nich lineárně závislé a pokusíme se této vlastnosti využít k zúžení obálky Σ.

Zavedeme si lineární systém s parametry, jejichž rozsah reprezentuje vektor $\boldsymbol{p} \in \mathbb{IR}^m$ a $A^k \in \mathbb{R}^{n\times n}$ koeficienty a pozici k-tého parametru. Píšeme

$$A(\boldsymbol{p})x = b(\boldsymbol{p}),$$

kde

$$A(\boldsymbol{p}) = \Sigma_{k=1}^m A^k \boldsymbol{p}_k, \;\; b(\boldsymbol{p}) = \Sigma_{k=1}^m b^k \boldsymbol{p}_k.$$

Definice pro množinu řešení takových soustav vypadá následovně

$$\Sigma := \{x; \exists p \in \boldsymbol{p} : A(p)x = b(p)\}.$$

Explicitní popis této množiny za speciálních podmínek ukázala Popova (2009). Je podobný větě 5 a my si ho v obecném případě uvedeme jako nutnou podmínku, jak popsal Hladík (2012).

Věta 7. *Mějme $x \in \mathbb{R}^n$ a $\boldsymbol{p} \in \mathbb{IR}^m$. Pokud je $x \in \Sigma$, pak je i řešením nerovnic*

$$|A(p^c)x - b(p^c)| \leq \Sigma_{k=1}^m p_k^\Delta |A^k x - b^k|.$$

Na základě věty 7 uvedeme v naší práci implementace Bauer-Skeeleho a Hans-Bliek-Rohnova odhadu zobecněného na lineární systémy s parametry a jejich vylepšení (Hladík, 2012). Více v sekci 3.2.

Posledním důležitým pojmem je intervalová *relaxace*. Jde o proces, kdy se zbavíme informace o lineárních závislostech jednotlivých parametrů. V podstatě jen vyčíslíme výraz

$$\Sigma_{k=1}^{m} A^k \boldsymbol{p}_k$$

z předchozí definice a označíme ho $\boldsymbol{A}$. Na tuto matici se díváme jako na klasickou intervalovou matici bez parametrů. Zrelaxování systému se nám může hodit, pokud má soustava hodně parametrů a jsou dostupné metody pro řešení intervalových rovnic bez parametrů. Pak můžeme nějakou jednodušší metodou vypočítat širší obálku, ale často rychleji.

3. Algoritmy

3.1 Přehled

Tato kapitola má posloužit jako základní přehled různých přístupů pro výpočet obalu nebo obálky množiny řešení intervalových systémů s lineárními závislostmi. V jednotlivých podkapitolách se pokusíme sdružit algoritmy založené na podobném principu. Dělení je spíše orientační a často se jednotlivé skupiny protínají. Zaměříme se hlavně na metody implementované v této práci. Pro úplnost uvedeme i pár dalších, kterým se ale už tolik věnovat nebudeme. Až na poslední sekci 3.7, ve které popíšeme základní myšlenky metod pro výpočet přesného obalu, se budeme zabývat rychlými algoritmy pro výpočet těsné obálky.

3.2 Přímé metody

Jako přímé metody označíme ty, které výslednou obálku řešení uvádějí v explicitním tvaru, kde ze vstupních dat dopočítáme hodnoty konkrétních výrazů a dosadíme. První ze dvou metod, které jsme implementovali a o které se budeme více zajímat, je Bauer-Skeeleho odhad (Bauer, 1966; Skeel, 1979), který je zobecněn na lineární systémy s parametry (Hladík, 2012).

Věta 8. *Za předpokladu, že $A(p^c)$ je regulární, označme*

$$\begin{aligned} M &:= \Sigma_{k=1}^{m} p_k^{\Delta} |A(p^c)^{-1} A^k|, \\ x^* &:= A(p^c)^{-1} b(p^c). \end{aligned}$$

Pokud $\rho(M) < 1$, pak

$$\begin{aligned} [x^* - (I \ - M)^{-1} \Sigma_{k=1}^{m} p_k^{\Delta} |A(p^c)^{-1}(A^k x^* - b^k)|, \\ x^* - (I \ - M)^{-1} \Sigma_{k=1}^{m} p_k^{\Delta} |A(p^c)^{-1}(A^k x^* - b^k)|] \end{aligned}$$

je obálka řešení Σ.

Druhou metodou je Hans-Bliek-Rohnův odhad, kde použijeme konkrétně Rohnovu formulaci (Rohn, 1993), zobecněnou na lineární systémy s parametry (Hladík, 2012).

Věta 9. *Za předpokladu, že $A(p^c)$ je regulární, označme*

$$\begin{aligned} M^* &:= (I \ - \Sigma_{k=1}^{m} p_k^{\Delta} |A(p^c)^{-1} A^k|)^{-1}, \\ x^* &:= A(p^c)^{-1} b(p^c), \\ x^0 &:= M^* |x^*| + \Sigma_{k=1}^{m} p_k^{\Delta} M^* |A(p^c)^{-1} b^k|. \end{aligned}$$

Pokud $\rho(M) < 1$, pak se řešení soustavy nachází v obálce definované

$$\begin{aligned} \bar{x}_i &:= \max\{\tilde{x}, v \ \tilde{x}\}, \\ \underline{x}_i &:= \min\{\underset{\sim}{x}, v \ \underset{\sim}{x}\}, \end{aligned}$$

kde

$$\begin{aligned}\tilde{x}_i &= x_i^0 + (x_i^* - |x_i^*|)m_{ii}^*,\\ \underset{\sim}{x}_i &= -x_i^0 + (x_i^* + |x_i^*|)m_{ii}^*,\\ v &= 1/(2m_{ii}^* - 1).\end{aligned}$$

Ačkoliv Hans-Bliek-Rohnův odhad je ve verzi bez parametrů přinejhorším stejně dobrý jako Bauer-Skeeleho (Rohn, 2010), u výsledků parametrické verze tuto jistotu nemáme (Hladík, 2012, Sekce 6).

Obálky $\boldsymbol{x}$ získané podle výše uvedených vět se dají ještě zlepšit (Hladík, 2012, Kapitola 4), budeme-li kontrolovat znaménka vektoru

$$\boldsymbol{a} := |A(p^c)^{-1}(A^k\boldsymbol{x} - b^k)|,$$

pak podle výsledku u jednotlivých komponent odstraníme absolutní hodnoty pravé strany trojúhelníkové nerovnosti

$$(|A(p^c)^{-1}(A^k\boldsymbol{x} - b^k)|)_i \le (|A(p^c)^{-1}A^k||\boldsymbol{x} - x^*| + |A(p^c)^{-1}(A^kx^* - b^k)|)_i.$$

Pro $\boldsymbol{a}_i \le 0$ dostaneme

$$(|A(p^c)^{-1}(A^k\boldsymbol{x} - b^k)|)_i \le -(A(p^c)^{-1}A^k(\boldsymbol{x} - x^*) - A(p^c)^{-1}(A^kx^* - b^k))_i$$

a pro $\boldsymbol{a}_i \ge 0$

$$(|A(p^c)^{-1}(A^k\boldsymbol{x} - b^k)|)_i \le (A(p^c)^{-1}A^k(\boldsymbol{x} - x^*) + A(p^c)^{-1}(A^kx^* - b^k))_i.$$

Pokud je nějaká komponenta $\boldsymbol{a}$ interval obsahující kladné i záporné hodnoty, tak nerovnost ponecháme nezměněnou. Nutno podotknout, že $\boldsymbol{x}$ může být výsledná obálka Bauer-Skeeleho, Hans-Bliek-Rohnova nebo také úplně jiného odhadu. Pokud se ale jedná o první dva případy, pak tato práce obsahuje implementace algoritmů (Hladík, 2012, Kapitola 4), které dávají přinejhorším stejně dobrou obálku jako nevylepšené verze. Pro ostatní případy nemůžeme s jistotou nic tvrdit, ale můžeme numerickým testováním vyzkoušet, jak si s tímto přístupem vedou.

3.3 Iterativní metody

Iterativní metody začínají s nějakým počátečním intervalem, který postupně upravují. Daly by se rozdělit na dvě skupiny, podle způsobu, jakým s intervaly pracují.

První skupina začíná s dostatečně širokým intervalem, který množinu řešení obsahuje. Nejčastěji bereme nadhodnocený odhad, který lze jednodušše spočítat z vlastností systému explicitním vzorcem. Poté každou iterací interval zúžíme. Musíme zajistit, že metoda konverguje a že v každém kroku nepřijdeme o žádné řešení. Takto dostaneme lokální obálku, která po několika iterací už nepůjde zlepšit.

Druhá skupina na to jde obráceně. Začne s libovolným bodem, který patří do množiny řešení, například x splňující $A(p^c)x = b(p^c)$. Poté v každé iteraci tento interval rozšíří pevně danou formulí a zkontroluje, jestli platí určité podmínky. Pokud ano, máme jistotu, že v takto rozšířeném intervalu se nachází celá množina řešení. Jde o takzvanou epsilon-inflaci.

V této práci implementujeme algoritmus ze druhé skupiny, který řeší soustavy bez parametrů. Použijeme ho v kombinaci s přístupem popsaným v sekci 3.4. Zde uvedeme trochu teorie, která je nezbytná k pochopení funkčnosti zmíněné metody.

Základem je Krawczykova metoda. Upravme $Ax = b$ následovně

$$\begin{aligned} x + Ax &= b + x, \\ x &= b + x - Ax, \\ x &= b + (I\ - A)x. \end{aligned}$$

Zavedeme Krawczykův operátor pro intervaly. Celý systém $\boldsymbol{A}\ x = \boldsymbol{b}$ přepodmíníme $C := (A^c)^{-1}$.

$$K(\boldsymbol{x}) := C\boldsymbol{b} + (I\ - C\boldsymbol{A})\boldsymbol{x}$$

Jedná se o jeden krok iterace, kdy $\boldsymbol{x}^k := \boldsymbol{x}$ a $\boldsymbol{x}^{k+1} := K(\boldsymbol{x})$. Dále využijeme následující tvrzení (Rump, 2010).

Tvrzení 10. *Mějme* $\boldsymbol{v}\ , \boldsymbol{x} \in \mathbb{IR}^n$ *a* $\boldsymbol{V}\ \in \mathbb{IR}^{n\times n}$*, pak*

$$\boldsymbol{v}\ + \boldsymbol{V}\ \boldsymbol{x} \subset \text{int}\boldsymbol{x} \Rightarrow \rho(\boldsymbol{V}) < 1,$$

kde int$\boldsymbol{x}$ je vnitřek intervalu $\boldsymbol{x}$.
Použijeme-li tvrzení 10 na Krawczykův operátor, dostaneme

$$C\boldsymbol{b} + (I\ - C\boldsymbol{A})\boldsymbol{x} \subset \text{int}\boldsymbol{x} \Rightarrow \rho(I - C\boldsymbol{A}) < 1.$$

Zde si povšimněme, že důsledkem je regularita $\boldsymbol{A}$, jak jsme zmínili na konci první kapitoly 1.3. V kombinaci se známou větou o pevném bodě spojité funkce, v našem případě funkce $x = K(x)$, máme vše potřebné. Nastane-li v nějakém kroku iterace $\boldsymbol{x}^{k+1} \subset \text{int}\ \boldsymbol{x}^k$, tak víme, že matice $\boldsymbol{A}$ je regulární a Krawczykův operátor je spojitá funkce, tedy pro každé pevné $A \in \boldsymbol{A}, b \in \boldsymbol{b}\ \exists x \in \boldsymbol{x} : Ax = b$. Navíc je množina řešení spojitá a pevné body jsou jen vnitřní body $\boldsymbol{x}$, takže $\boldsymbol{x}$ obsahuje všechna řešení.

Na tomto základě staví Rumpův algoritmus (Rump, 2010, Algoritmus 10.7), který navíc obsahuje zmíněnou epsilon-inflaci, kdy se v každém kroku $\boldsymbol{x}$ rozšíří podle následujícího výrazu.

$$\begin{aligned} C &:= (A^c)^{-1}, \\ x_s &:= Cb^c, \\ \boldsymbol{z}\ &:= C(\boldsymbol{b} - \boldsymbol{A}\ x_s), \\ \boldsymbol{R} &:= I\ - C\boldsymbol{A}, \\ \boldsymbol{y} &:= \boldsymbol{x}^k[0.9, 1.1] + [-e, e], \\ \boldsymbol{x}^{k+1} &:= \boldsymbol{z}\ + \boldsymbol{R}\boldsymbol{y}, \end{aligned}$$

kde e je nějaké malé číslo a testujeme $\boldsymbol{z}\ + \boldsymbol{R}\boldsymbol{y} \subset \text{int}\boldsymbol{y}$. Právě tuto verzi v naší práci používáme.

3.4 Residuální forma

Residuální forma není ani tak metoda, jako spíše přístup, jakým se díváme na řešení intervalové soustavy. Zavedeme-li si

$$x^* := A(p^c)^{-1}b(p^c),$$
$$x := x^* + y,$$

pak můžeme systém přetransformovat následovně

$$A(\boldsymbol{p})y = b(\boldsymbol{p}) - A(\boldsymbol{p})x^*,$$

kde neznámou je y. Výsledný obal $\boldsymbol{x}$ původního řešení dostaneme jako $x^* + \boldsymbol{y}$. Tento přístup dává smysl, pokud bereme v potaz závislosti parametrů, které na pravé straně vytkneme (viz. subdstributivita 1.1)

$$(\Sigma_{k=1}^m \boldsymbol{p}_k A^k)y = \Sigma_{k=1}^m \boldsymbol{p}_k(b^k - A^k x^*).$$

Tuto soustavu zrelaxujeme na obyčejnou intervalovou soustavu bez parametrů, která má užší intervaly u parametrů než kdybychom relaxovali přímo. Vyřešíme jí libovolnou parametrickou či neparametrickou metodou.

V této práci k řešení residuálního tvaru používáme již zmíněný Rumpův algoritmus 3.3 a přímou metodu popsanou Skalnou (Skalna, 2006). K podobnému výsledku došel i Hladík (Hladík, 2012) a jde o přímý výpočet obálky, ke které za určitých podmínek konverguje Krawczykův operátor.

Věta 11. *Buď $C \in \mathbb{R}^{n\times n}$ a $x^0 \in \mathbb{R}$. Označme $D := \mathrm{mag}(I - \Sigma_{k=1}^m (RA^k)\boldsymbol{p}_k)$. Pokud $\rho(D) < 1$, pak*

$$\Sigma \subseteq x^0 + (I\ - D)^{-1}\mathrm{mag}(\Sigma_{k=1}^m R(b^k - A^k x^0)\boldsymbol{p}_k)[-1, 1].$$

Nejlepších výsledků bývá dosaženo pro $R := A(p^c)^{-1}$ a $x^0 := A(p^c)^{-1}b(p^c)$

3.5 Využívaní monotonie

Obal všech hodnot, kterých může funkce f na intervalu $\boldsymbol{x}$ nabývat, značíme $f(\boldsymbol{x})$. Monotonie se dá využít k rychlému výpočtu obalu. Stačí nám například zjistit, že je $f(x)$ na $\boldsymbol{x}$ rostoucí a pak její obal nalezneme jako $f(\boldsymbol{x}) = [f(\underline{x}), f(\bar{x})]$.

Stejně tak využijeme této vlastnosti u jednotlivých parametrů lineárního systému. Zjistíme-li, že i-tá složka množiny řešení je monotonní vůči několika parametrům, můžeme horní a dolní mez dopočítat pomocí soustav, které mají některé parametry degenerované na bodové intervaly. Uvedeme zde obecný postup, kde se každý krok může lišit v závislosti na implementaci.

Základem přístupu je určení monotonie x_i vůči všem parametrům $\boldsymbol{p}_k$. To uděláme pomocí znaménka parciální derivace $\dfrac{\partial x_i}{\partial \boldsymbol{p}_k}$, pro jehož výpočet si odvodíme

následující vzorec, zderivujeme-li soustavu podle $\boldsymbol{p}_k$.

$$\begin{aligned}\frac{\partial A(\boldsymbol{p})}{\partial \boldsymbol{p}_k}x + A(\boldsymbol{p})\frac{\partial x}{\partial \boldsymbol{p}_k} &= \frac{\partial b(\boldsymbol{p})}{\partial \boldsymbol{p}_k},\\ A^k x + A(\boldsymbol{p})\frac{\partial x}{\partial \boldsymbol{p}_k} &= b^k,\\ A(\boldsymbol{p})\frac{\partial x}{\partial \boldsymbol{p}_k} &= b^k - A^k x.\end{aligned}$$

Hodnotu x musíme pochopitelně nahradit předem spočítanou obálkou $\boldsymbol{x}^*$. Někdo navrhuje rychlou, ale také někdy pro naše účely přehodnocující intervalovou Gaussovu eliminaci (Rohn, 2004), jiní zase přesnější Rumpův iterativní algoritmus, který bere v potaz i závislosti mezi koeficienty (Popova, 2004).

Dále, samotný výpočet parciálních derivací lze udělat výpočtem méně těsné obálky $\boldsymbol{B}$ pro množinu $\{A(p)^{-1}|p \in \boldsymbol{p}\}$ a vyčíslením výrazu

$$\boldsymbol{B}^{-1}(b^k - A^k\boldsymbol{x}^*),$$

nebo vyřešením soustavy

$$A(\boldsymbol{p})\frac{\partial x}{\partial \boldsymbol{p}_k} = b^k - A^k\boldsymbol{x}^*$$

nějakou z metod, které jsme uvedli v této kapitole.

Označme $\boldsymbol{d} := \dfrac{\partial x}{\partial \boldsymbol{p}_k}$. Na základě znaménka zadefinujeme nový vektor parametrů $\boldsymbol{p}^i$ pro každé x_i následovně

$$\boldsymbol{p}^i_k = \begin{cases}[\underline{p}_k, \bar{p}_k], & \text{pokud } \underline{d}_i \geq 0,\\ [\bar{p}_k, \underline{p}_k], & \text{pokud } \bar{d}_i \leq 0,\\ [p^c_k, p^c_k], & \text{pokud } \boldsymbol{d}_i = [0,0].\end{cases}$$

Pokud pro každé $\boldsymbol{p}^i$ nastane jeden z těchto případů, můžeme vypočítat přesný obal z $2n$ bodových systémů

$$(\square\Sigma)_i = [(A(\underline{p}^i)^{-1}b(\underline{p}^i))_i, (A(\bar{p}^i)^{-1}b(\bar{p}^i))_i].$$

Všimněme si, že hodnota parciální derivace je interval, tedy může nastat případ, kdy znaménko nepůjde určit. To se dá řešit více způsoby. Můžeme nechat $\boldsymbol{p}^i_k = \boldsymbol{p}_k$, aplikovat přesnější metodu, nebo konkrétní parametr rozdělit na dva intervaly a zjistit, jak se chová vůči x_i na každém zvlášť.

Skalna navrhuje pro každé x_i sestavit novou soustavu rovnic, ve které odstraníme parametry, jejichž znaménka se nám podařilo určit, a proces opakovat, dokud dochází k určování u dalších parametrů (Skalna, 2007).

3.6 Dělení parametrů

Dělení parametrů je úloha typu rozděl a panuj. Vybereme parametr $\boldsymbol{p}_k$, nějakým způsobem ho rozdělíme v bodě c na dva intervaly

$\boldsymbol{p}_{k_1} := [\underline{p}_k, c]$ a $\boldsymbol{p}_{k_2} := [c, \bar{p}_k]$, zavedeme dva nové vektory parametrů $\boldsymbol{u}$ a $\boldsymbol{v}$, které dostaneme z $\boldsymbol{p}$ nahrazením $\boldsymbol{p}_k$ za $\boldsymbol{p}_{k_1}$ a $\boldsymbol{p}_{k_2}$. S nimi vyřešíme pomocí nějaké z již zmíněných metod systémy

$$A(\boldsymbol{u})x_1 = b(\boldsymbol{u}),$$
$$A(\boldsymbol{v})x_2 = b(\boldsymbol{v}),$$

a nakonec najdeme společný obal obou řešení, $\square \cup_{i=1}^{2} \boldsymbol{x}_i$.

Kritéria, podle kterých můžeme vybírat parametry a místo, kde je rozdělit, bývají různá. My se tímto problémem nebudeme zabývat do hloubky, uvádíme ho zde pro úplnost. V naší implementaci používáme pro testovaní metodu, která postupně dělí největší intervaly v polovině do předem stanovené hloubky rekurze. Za zmínku stojí i přístup, kdy se nedělí prostor parametrů, ale prostor, ve kterém se nachází soustava samotná.

3.7 Obalové metody

Obalové metody je označení pro algoritmy, které počítají obal řešení soustavy. Jak už víme, jedná se obecně o NP-těžký problém, ale existují různé případy a heuristiky, kdy se doba běhu velmi sníží.

Asi nejznámější je Janssonův algoritmus pro neparametrické soustavy. K jeho popisu potřebujeme jednu doposud nepopsanou vlastnost množiny řešení lineárních systémů.

Tvrzení 12. *Pokud $\Sigma \neq \emptyset$ pak může nastat právě jedna z těchto možností:*

- *Σ je souvislá a omezená a to právě v případě, že $A(\boldsymbol{p})$ je regulární.*
- *Σ se skládá z jedné či více oddělených množin, které jsou všechny neomezené.*

Janssonův Algoritmus:

1. Nalezni počáteční obálku řešení, například pomocí nějaké už popsané metody.
2. Vyber ortant, ve kterém se může nacházet řešení a vypočítej obal řešení redukovaného na tento ortant. Pokud je výsledkem prázdná množina, zvolíme jiný ortant z obálky.
3. Zvol sousední ortant aktuálního, který je v počáteční obálce a aktuální řešení se ho dotýká. Jdi na krok 2.
4. Vypočítej výsledný obal ze sjednocení obalů jednotlivých ortantů.

Poznámka. Jelikož je množina řešení v libovolném ortantu konvexní, jedná se v kroku 2 o polynomiální výpočet. Krok 3 funguje právě díky tvrzení 12. Pokud je množina souvislá, vybereme takto pouze ortanty, které řešení obsahují a ne všech 2^n. Pokud souvislá není, bude aktuální množina neomezená, což zjistíme při výpočtu některého obalu. Výsledek nedostaneme, zato otestujeme regularitu matice.

Další metody zde neuvedeme, ani žádné neimplementujeme.

4. Analýza

4.1 Popis problému

Implementujeme metody a algoritmy pro výpočet obálky řešení intervalových lineárních systémů s lineárně závislými parametry pro případné použití ve fakultní knihovně MATLABu, která se zabývá intervalovými výpočty.

4.2 MATLAB/INTLAB

K implementaci algoritmů z kapitoly 3 jsme použili známé vývojové prostředí MATLAB spolu s intervalovou knihovnou INTLAB

`http://www.ti3.tu-harburg.de/rump/intlab/`,

která dodává implementaci intervalové aritmetiky a základních intervalových metod spolu s knihovnou VERSOFT

`http://uivtx.cs.cas.cz/~rohn/matlab/`,

která obsahuje různé verifikované výpočty, pro nás hlavně důležitý verspectrad(), pomocí nehož odhadujeme spektrální poloměr matice lineárního systému. Prostředí MATLAB nabízí důležitou funkci setround(), s níž můžeme nastavit směr zaokrouhlování float-point výpočtů na procesoru. Díky standardu IEEE jsou režimy zaokrouhlení dolů, zaokrouhlení k nejbližšímu a zaokrouhlení nahoru implementovány na většině procesorů. Celý balíček INTLAB staví na této funkci následovně. Nastaví zaokrouhlování dolů pomocí setround(-1), provede aritmetickou operaci, nastaví zaokrouhlování nahoru pomocí setround(1), provede tu samou operaci a vrátí nastavení zaokrouhlování na původní hodnotu. Tím jsme získali horní a dolní odhad výsledku operace, ze kterého vytvoříme objekt interval.

MATLAB není jediné prostředí podporující intervalové výpočty. Zvolili jsme ho kvůli rozšířenosti, rozsáhlosti a preferencím skupiny na naší fakultě, zabývající se podobnou problematikou.

4.3 Rigorózní výsledek

Všechny naše výpočty jsou rigorózní, tedy máme jistotu, že skutečný výsledek se nachází v námi spočítaných intervalech. Toho jsme dosáhli pomocí VERSOFTu a nahrazením float-point aritmetiky za intervalovou z INTLABu. Je to vlastnost velmi důležitá v oblasti matematických důkazů prováděných počítačem a obecně při výpočtech, kde jistota hraje velkou roli.

4.4 Datová reprezentace

První otázkou bylo, jak reprezentovat intervalové systémy. Zvolili jsme dva způsoby. První, přímočarý, kdy je každá matice A^k reprezentována jako dvourozměrné pole a celkově je systém uložen ve dvou polích o rozměrech $n \times n \times m$ a $n \times m$.

Takto vytvořené reprezentace jsou rychlé, ale často řídké a zbytečně zabírají paměť. Druhá reprezentace využívá MATLAB objektu *cell*, který se chová podobně jako pole, ale umožňuje v každé buňce alokovat matici jiného rozměru. Místo celé matice A^k uložíme do jednotlivých buněk trojice (x,y,v), které kódují polohu a hodnotu koeficientu u jednotlivých výskytů parametrů. Při běhu programu se pak z takového kompaktního zápisu vytvoří už klasická matice A^k, která je reprezentována dvourozměrným polem. Tomu samozřejmě odpovídá určitá časová zátěž, kterou porovnáme v testovací části. Mezi oběma reprezentacemi lze jednoduše přepínat pomocí globální proměnné *dataModel*. To by mohlo v budoucnu způsobit potíže, zvláště pokud někdo pojmenuje svoji globální proměnnou stejně a oba kódy poběží ve stejném pracovním prostředí MATLABu. Naopak to ale volání jednotlivých funkcí zpřehlednilo a uživatel se kdykoliv může ujistit s jakou datovou reprezentací pracuje.

Další problém pramení z povahy rigorózních výpočtů. Máme-li nějaký problém, který chceme vyřešit pomocí počítače, musíme si uvědomit, že při jeho převedení do paměti se dopustíme prvních zaokrouhlovacích chyb. Tedy, pokud náš problém popisují nějaká reálná čísla a chceme získat rigorózní výsledek, musíme hned na začátku správně obalit tato reálná čísla intervaly, které je obsahují. Z našeho pohledu je to problém závislý zcela na uživateli, naše implementace pracuje jak s reálnými, tak s intervalovými koeficienty u parametrů, i když z pohledu teoretického je intervalové zadání to správné pro ověřený výsledek.

4.5 Metody

Metody, které jsme vybrali jako součást implementace této práce, jsou dle našeho názoru a kontextu příbuzných prací jedny z nejefektivnějších pro řešení daného problému. Jako základní metody bereme Hladíkovu Bauer-Skeeleho a Hans-Bliek-Rohnovu generalizaci, jejich vylepšení, Rohnovu a Skalné přímou metodu použitou na soustavu v residuálním tvaru. Jako pokročilejší využíváme monotonii a dělení parametrů, které jsou časově náročné, ale dokážeme s jejich pomocí dosáhnout libovolné přesnosti. Myšlenkou je prozkoumat vlastnosti základních metod, určit, které použít při požadavku na přesný výpočet, které při požadavku na rychlý výpočet a které vhodně zkombinovat s pokročilejšími.

Při dělení parametrů dělíme ten s nejširším intervalem. Metoda je implementovaná tak, aby probíhala iterativně a nikoliv rekurzivně, jak by se mohlo zdát z teoretického popisu. Nejprve si rozdělíme prostor parametrů. Po rozdělení prostoru parametrů už běží klasický výpočet pro každý subprostor a výsledek se průběžně aktualizuje jako sjednocení jednotlivých mezivýsledků. Více funkci popíšeme v poslední kapitole.

Využívání monotonie je nejvíce flexibilní část. V postupu, který jsme uvedli v sekci 3.5 dochází k řešení několika lineárních soustav. Nejdříve spočítáme počáteční obálku. Tu bychom měli spočítat nejpřesnější metodou, protože jde jen o jeden výpočet a může nám v krajních případech hodně pomoci. Poté pro každý parametr spočteme monotonii jednotlivých složek $\boldsymbol{x}$ jako m dalších lineárních systémů, kde m je počet parametrů. Zde nám jde jen o znaménko první derivace a

měli bychom použít metodu v závislosti na počtu parametrů a povaze problému. Pro méně parametrů nemusí být přesná a pomalá metoda nevyhovující. Na druhou stranu, občas není potřeba počítat velmi přesně, zvlášť když jsou vyšetřované funkce takové, že jejich první derivace je daleko od nuly. Na základě počtu určených derivací (některé výsledky můžou být intervaly obsahující jak záporné, tak kladné hodnoty a tedy znaménko neurčité) se můžeme rozhodnout jak pokračovat dále. V nejlepším případě určíme všechna znaménka a výpočtem $2n$ systémů pro horní a dolní mez každé složky $\boldsymbol{x}$ získáme až na zaokrouhlování přesný obal. V naší implementaci určíme počáteční obal pomocí metody Skalne pro reziduální tvar. V části pro určení znamének derivací zrelaxujeme parametrické systémy na obyčejné a použijeme rychlou metodu verifylss(). Nakonec konkrétní meze jednotlivých složek $\boldsymbol{x}$ dopočteme zase pomocí metody Skalne. Počet určených znamének nebereme v potaz. Jedná se o základní implementaci.

4.6 Související práce

Intervalová aritmetika je formou knihovny nebo balíčku implementována v několika prostředích a jazycích. Mathematica, C++ (ibex-lib, Boost Interval Arithmetic Library), Python (pyIbex) a další. Stejný problém jako řešíme my, řeší i Popova (2004) v balíčku IntervalComputations'LinearSystems' pro Mathematica, který dostupný skrze projekt webComputing na adrese

`http://cose.math.bas.bg/webComputing/`.

Vstup se zadává skrze webový formulář nebo soubor obsahující dávku příkazů. Na výběr je méně metod než nabízí podpůrný balíček. Nemůžeme objektivně srovnat výsledky s naší prací, neboť výpočetní výkon takové platformy často převyšuje výkon osobního počítače, na kterém probíhají testy z následující kapitoly.

Dále je tu samostatný software, který řeší obecné problémy programování s omezujícími podmínkami, ve kterém sadou rovnic můžeme modelovat naše lineární systémy. Konkrétně RSolver

`http://rsolver.sourceforge.net/`

a Realpaver

`http://pagesperso.lina.univ-nantes.fr/info/perso/permanents/granvil/realpaver/`,

založené na branch & bound algoritmech. Dostupné jsou pouze na některých Unixových systémech a používají vlastní syntax pro popis omezujících podmínek. Stejně jako u webComputing je přímé srovnání problematické, protože obojí využívá odlišných prostředí a přístupů.

Naše funkce jsou jednoduše integrovatelné do prostředí MATLAB a do projektů v něm napsaných. Také implementujeme rozdílné metody a snažíme se uživateli ulehčit zadávání vstupu pro některé problémy, například ve tvaru Toeplizových matic, jak ukážeme dále v této práci. Jeden z našich cílů není soupeřit s těmito projekty, ale spíše vyplnit mezeru, která se v balíčcích pro vývojové prostředí MATLAB nachází.

5. Testování

5.1 Metodika

Testování proběhlo na sestavě s Intel Core i5-4460 (3.20GHz), 8GB RAM, Windows 10, MATLAB 2015b, INTALAB 9, VERSOFT 10. Nejdříve jsme uniformně náhodně vygenerovali střed parametrů p^c a vytvořili kolem něj intervaly o konstantní šířce 2δ, tedy výsledný vektor $[p^c - \delta, p^c + \delta]$. Abychom v implementaci nemuseli složitě rozlišovat parametrické hodnoty od neparametrických, první parametr zpravidla necháváme degenerovaný s hodnotou [1,1] a v odpovídající matici A^1 se tyto neparametrické hodnoty nacházejí. Všechny námi testované typy matic obsahují pouze parametrické hodnoty, ale i tak prvního parametru využíváme, když na diagonálu A^1 dosadíme konstantu závislou na dimenzi systému, abychom snížili počet neregulárních matic. Matice A^k, které reprezentují koeficienty u parametrů $\boldsymbol{p}_k$, obsahují *double* hodnoty, rozmístěné podle typu testované matice. Vektor b je uniformně náhodná konstanta (parametr nulové šířky nebo také degenerovaný interval) z prostoru $[-10, 10]^n$. Metody porovnáváme časově jako průměr doby běhu 10ti opakovaných výpočtů změřených metodami tic a toc, dále také podle průměrné šířky řešení definované

$$\frac{\Sigma_{i=1}^{n}(\bar{x}_i - \underline{x}_i)}{n}.$$

Základní metody, které testujeme, si pro přehlednost v tabulkách označíme aliasy.

BS	Bauer-Skeeleho generalizace.
BSref	Vylepšení Bauer-Skeeleho generalizace.
HBR	Hans-Bliek-Rohnova generalizace.
HBRref	Vylepšení Hans-Bliek-Rohnovy generalizace.
Rres	Rumpův iterativní algoritmus, reziduální tvar.
Sres	Skalné přímá metoda, reziduální tvar.
Mono	Algoritmus využívající monotonie.
SubdivN	Dělení prostoru parametrů podle N aktuálně nejširších parametrů.

Výsledky posledního přístupu, SubdivN, jsme se rozhodli uvést stranou od ostatních, neboť jde vlastně o volání některých už implementovaných metod na podproblémy.

5.2 Toeplitzovy matice

Toeplitzovy matice mají stejné hodnoty na „uhlopříčkách“, například

$$\begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}.$$

Je to první druh matic, na který se zaměříme. Hodnoty p^c generujeme uniformně v rozmezí $[-10, 10]$, testované šířky intervalů jsou $\{0.05, 0.1, 0.5,1\}$ a dimenze $\{5, 10, 15, 20, 25, 50, 100\}$. Celá tabulka výsledků je dostupná v příloze, zde si ukážeme jen data pro jednu konkrétní dimenzi, v ostatních dimenzích se metody chovají podobně. Pro přehlednost výsledky relativizujeme k výsledkům metody BS pro poloměr 0.05.

Dimenze:	100	průměrná šířka řešení		
Poloměr:	0.05	0.1	0.5	1
BS	1.00000	2.03074	10.59155	22.89549
BSref	0.99769	2.02133	10.35620	21.95581
HBR	1.44822	2.93890	15.38232	33.04861
HBRref	1.00557	2.04630	11.11162	24.88838
Rres	1.00077	2.03343	10.60538	23.20038
Sres	1.00000	2.03074	10.59155	22.89549
Mono	0.99251	2.00058	9.82920	19.80327

Nejpřesnější metodou je samozřejmě Mono, ale dále uvidíme, že i nejpomalejší. BS a Sres v rámci přesnosti testování dosahovali stejných výsledků. Druhá nejpřesnější metoda je BSref. Vynecháme-li Mono, HBRref a BSref, může nás zajímat, nejde-li dosáhnout přesnějšího výpočtu kombinací dvou a více metod, tedy vypočítat dvě obálky a najít jejich průnik. Zjistili jsme, že na našich datech nelze. Stejně přesné BS a Sres jsou v inkluzi s obálkami vypočítanými HBR a Rres.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	0.3720	0.3704	0.3661	0.3658
BSref	32.4652	32.4440	31.6614	32.0354
HBR	0.5565	0.5595	0.5530	0.5544
HBRref	26.9426	26.9209	26.0113	26.5539
Rres	0.3542	0.3567	0.3545	0.3552
Sres	0.3541	0.3549	0.3551	0.3541
Mono	84.0543	84.2100	84.3037	84.2985

Pozn: Výsledky jsou v sekundách.

Časy BSref, HBRref a Mono jsou daleko za ostatními. Kandidátem na nejrychlejší metodu je Sres, který je většinou rychlejší než podobný BS a méně přesný Rres.

Poslední odstavec k Toeplitzovým maticím patří algoritmu dělícímu prostor parametrů. Z testování jsme vypozorovali, že jak roste velikost soustavy, přestává být výsledek SubdivN kompetitivní s ostatními přístupy, neboť je zapotřebí velké množství času k získání srovnatelných výsledků. Avšak pro menší systémy nabízí zajímavou alternativu. Pro dimenzi 5 dává SubdivN lepší průměrné šířky řešení než BSref a rychleji. Zde je nutné podotknout, že BSref dostane na vstupu obálku spočítanou BS a čas jejího výpočtu by se měl pro přesnější porovnání přičíst

k časům BSref. Stále ale není tak rychlý jako zbylé metody nebo tak přesný jako pomalé Mono. Pro dimenzi 10 SubdivN dává lepší výsledky pouze pro širší intervaly. Ve vyšších dimenzích se BSref stává lepší metodou. Následující výsledky jsou zrelativizovány k výsledku BSref pro dimenzi 10 a poloměr 0.05.

Dimenze:	10	průměrná šířka řešení		
Poloměr:	0.05	0.1	0.5	1
BSref	1.00000	2.26894	12.34885	25.56072
Subdiv1	1.00113	2.27368	12.50124	26.14576
Subdiv2	1.00057	2.27209	12.41986	25.82682
Subdiv3	1.00008	2.27078	12.35080	25.53077
Subdiv4	0.99968	2.26909	12.29270	25.28045

Dimenze:	10	průměrná čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS + BSref	0.3324	0.3343	0.3361	0.3282
Subdiv1	0.0570	0.0572	0.0571	0.570
Subdiv2	0.1148	0.1151	0.1151	0.1149
Subdiv3	0.2324	0.2329	0.2324	0.2324
Subdiv4	0.4706	0.4720	0.4717	0.4697

Pozn: Výsledky jsou v sekundách.

Je tedy vidět, že využití SubdivN klesá s dimenzí systému a naopak roste se šířkou intervalů. Naskytuje se tedy otázka, zda-li by se dala najít nějaká numericky získaná heuristika pro rozhodnutí, kterou z metod (SubdivN nebo BSref) použít na konkrétně zadaný systém.

5.3 Symetrické matice

Buď a_{ij} hodnota na i-tém řádku a j-tém sloupci matice A. Pro symetrické matice platí

$$a_{ij} = a_{ji}.$$

Jak si jednotlivé algoritmy vedly, co se průměrné šířky řešení týče, se shoduje s výsledky na Toeplitzových maticích. Pro přehlednost relativizujeme výsledky vzhledem k výsledku metody BS pro poloměr 0.05.

Dimenze:	100	průměrná šířka řešení		
Poloměr:	0.05	0.1	0.5	1
BS	1.00000	2.05268	11.07360	24.06775
BSref	0.99912	2.04887	10.97425	23.68218
HBR	1.09541	2.24890	12.14164	26.33567
HBRref	1.00059	2.05487	11.14325	24.40767
Rres	1.00073	2.05531	11.08794	24.39040
Sres	1.00000	2.05268	11.07360	24.06775
Mono	0.99254	2.02224	10.29090	21.06936

Pořadí metod v rychlosti výpočtu se od Toeplitzových matic o něco změnilo. Pro dimenze 5, 10, 15, 20 a 25 zůstává nejčastěji na prvním místě Sres, následována Rres a BS. Ale pro dimenze 50 a 100 získává náskok ne tak přesný HBR.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	9.0803	9.0802	9.0723	9.0744
BSref	818.5669	817.7769	793.4381	799.5879
HBR	8.0467	8.0369	8.0056	8.0022
HBRref	672.2198	672.1887	642.3842	648.6375
Rres	8.7047	8.7077	8.7056	8.7119
Sres	8.7066	8.7085	8.7041	8.7028
Mono	2079.2256	2077.6726	2078.8601	2081.4506

Pozn: Výsledky jsou v sekundách.

Výsledky SubdivN uvádíme pouze v příloze. Jako konkurenční metoda se v tomto případě neprojevila pro žádné systémy. Pro menší bychom ji mohli přesností a časem zařadit mezi BSref a Mono. Je to zřejmě kvůli vyššímu počtu parametrů. Tedy i z pohledu počtu parametrů by se snad dala určit heuristika, kdy SubdivN použít a kdy ne.

5.4 Náhodné matice

Náhodné matice jsme generovali pomocí následujícího schématu. Přidali jsme další faktor, který řídíme, a to počet parametrů, který je nastaven na tři hodnoty $[n, 2n, 3n]$, kde n je dimenze systému. Stejně jako předtím dochází k vytváření velkého počtu singulárních matic, takže k diagonále přičítáme tentokrtát $100n$. Když se matice vytváří, rozhodne se, kolik výskytů budou mít jednotlivé parametry (uniformě náhodné celé číslo z intervalu $[1, 10]$), jejich pozice (uniformě náhodné celočíselné dvojice (x,y) z rozmezí dimenze matice), velikosti koeficientů (unioformě náhodné double číslo z intervalu $[-10,10]$). Zbytek se chová stejně jako u symetrických a Toeplitzových maticí.
Na těchto maticích jsme si potvrdili předchozí výsledky. Metody si zachovávají stejné pořadí v přesnosti a s menší změnou, kterou vysvětlíme níže, i v rychlosti. Nyní můžeme navrhnout, které funkce se budou volat se kterým požadavkem na přesnost, protože z našeho pohledu uživatel zadává různé náhodné matice a u nich už nyní víme, jak si s našimi metodami vedou. Konkrétní výsledky přikládáme do tabulek na konci.

Dimenze:	100	průměrný čas výpočtu	
Parametrů:	100	200	300
BS	0.1859	0.3641	0.5442
BSref	16.1231	32.0575	48.0589
HBR	0.4037	0.5569	0.7121
HBRref	13.5338	26.7009	39.8922
Rres	0.1788	0.3504	0.5232
Sres	0.1786	0.3513	0.5244
Mono	42.6646	83.6026	124.9061

Pozn: Výsledky jsou v sekundách, pro poloměr parametrů 0.05.

Jak se dalo čekat, zpomalení je přibližně lineární. Zajímavější je koeficient lineárního členu. Vidíme, že HBR zpomaluje nejméně, stejně tak HBRref pro pomalejší metody. Trend Sres vypadá o málo strmější než Rres, ale na to nemáme dost pozorování. Výsledky potvrzují naše závěry z předchozích sekcí s tím, že pro systémy s velkým množstvím parametrů se metody Rres, HBR a HBRref můžou ukázat jako zajímavé alternativy, kvůli časové náročnosti, i když zaostávají v přesnosti.

5.5 Porovnání datových reprezentací

5.5.1 Toeplitzovy matice

První ukážeme doby běhů algoritmů pro Toeplitzovy matice, kdy jsou data reprezentovaná polem.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	0.3720	0.3704	0.3661	0.3658
BSref	32.4652	32.4440	31.6614	32.0354
HBR	0.5565	0.5595	0.5530	0.5544
HBRref	26.9426	26.9209	26.0113	26.5539
Rres	0.3542	0.3567	0.3545	0.3552
Sres	0.3541	0.3549	0.3551	0.3541
Mono	84.0543	84.2100	84.3037	84.2985

Pozn: Reprezentace polem. Výsledky jsou v sekundách.

Výsledky jsou stejné jako v části 5.2, neboť rychlost algoritmů porovnáváme na datech v poli. Další tabulka ukazuje časy na totožných datech, ale reprezentovaných buňkovým formátem.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	0.3724	0.3720	0.3683	0.3687
BSref	32.5682	32.4917	31.699	32.1125
HBR	0.5607	0.5586	0.5545	0.5533
HBRref	26.9318	26.8833	26.0201	26.5694
Rres	0.3564	0.3563	0.3563	0.3568
Sres	0.3562	0.3560	0.3559	0.3562
Mono	84.6241	84.6726	84.7760	84.792

Pozn: Reprezentace buňkou. Výsledky jsou v sekundách.

Předpokládali jsme, že dojde k výraznému zpomalení, ale výsledky si jsou velmi podobné. Zřejmě samotná dekomprese dat v buňce na celé pole tvoří jen malou část z celkové doby, po kterou výpočet běží. Zároveň je ale buňkový formát mnohem kompaktnější. Samozřejmě velkou roli hraje počet různých parametrů a počet výskytů konkrétního parametru. Čím více tato čísla rostou, tím se teoreticky bude dekomprese prodlužovat. Pro Toeplitzovy matice však můžeme doporučit používat buňkový formát, zvlášť jedná-li se o velké soustavy. Přikládáme tabulku s paměťovou náročností, měřenou MATLAB funkcí whos(), konkrétně datovou položkou *bytes*, kterou vrací. Hodnoty odpovídají velikosti matice $\boldsymbol{A}$ s *double* koeficienty v pracovním prostředí MATLAB.

dimenze	buňka	pole
5	1.84 kb	1.95 kb
10	4.69 kb	15.63 kb
15	8.71 kb	52.73 kb
20	13.91 kb	125.00 kb
25	20.27 kb	224.14 kb
50	69.69 kb	1.91 Mb
100	256.41 kb	15.26 Mb

5.5.2 Symetrické matice

U symetrických matic oproti Toeplitzovým vzrostl počet parametrů ze $2n-1$ na $n^2/2$ a tím se už rozpoznatelně zpomalil výpočet používající buňkový formát. Pro porovnání uvádíme časy pro symetrické matice ještě jednou. Jde o reprezentaci polem.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	9.0803	9.0802	9.0723	9.0744
BSref	818.5669	817.7769	793.4381	799.5879
HBR	8.0467	8.0369	8.0056	8.0022
HBRref	672.2198	672.1887	642.3842	648.6375
Rres	8.7047	8.7077	8.7056	8.7119
Sres	8.7066	8.7085	8.7041	8.7028
Mono	2079.2256	2077.6726	2078.8601	2081.4506

Pozn: Reprezentace polem. Výsledky jsou v sekundách.

Reprezentace buňkou už viditelně zpomaluje, ale dáme-li výsledky do poměru, tak například pro metodu Mono jde o nárůst na 101.2%, což je stále velmi málo.

Dimenze:	100	průměrný čas výpočtu		
Poloměr:	0.05	0.1	0.5	1
BS	9.1843	9.1950	9.1902	9.2052
BSref	828.2750	828.7785	805.1888	811.5076
HBR	8.1098	8.1094	8.1177	8.1209
HBRref	684.6867	685.6571	657.5808	663.8190
Rres	8.7841	8.8013	8.8058	8.8081
Sres	8.7845	8.7996	8.8078	8.7992
Mono	2098.4348	2100.6190	2102.9985	2106.8858

Pozn: Reprezentace buňkou. Výsledky jsou v sekundách.

Nakonec i paměťové porovnání reprezentace matice $\boldsymbol{A}$.

dimenze	buňka	pole
5	2.49 kb	3.13 kb
10	8.63 kb	43.75 kb
15	18.66 kb	212.70 kb
20	32.61 kb	659.38 kb
25	50.46 kb	1.55 Mb
50	198.31 kb	24.34 Mb
100	786.98 kb	385.36 Mb

5.5.3 Náhodné matice

Výsledky pro náhodné matice nalezne čtenář v přiložených tabulkách na konci práce. Výsledky se shodují s trendem pozorovaným u předchozích maticí. Nebudeme zde uvádět další tabulky.

5.6 Závěr

Na základě těchto pozorování jsme se rozhodli postupovat takto: pro defaultní výpočet používat nejrychlejší metodu Sres, pro přesnější výpočet BSref a pro nejpřesnější Mono. Metody BS a Sres ze zdají být numericky ekvivalentní a možná

by mohlo bližší prozkoumání odhalit i teoretickou ekvivalenci. Co se týče metody SubdivN, dává dobré výsledky pro menší systémy nebo sirší intervaly. Abychom ji ale mohli používat úspěšně, museli bychom analyzovat vstupní data a poté se rozhodnout pro její nasazení. Dále upozorňujeme na časové výsledky HBR a HBRref, které s počtem parametrů získávají časový náskok nad ostatními. Nakonec doporučujeme používat naší buňkovou reprezentaci. Pro Toeplitzovy i symetrické matice jsou znatelně méně paměťově náročné a k časovému zpomalení téměř nedochází. Samozřejmě zpomaluje úměrně počtu parametrů.

6. Uživatelská dokumentace

6.1 MATLAB/INTLAB

Pro zadání vstupu naší funkci potřebujeme pouze pár pojmů z INTLABu a MATLABu. Zde je jednoduchý přehled základních příkazů.

Pro práci s polem si ukážeme pár ilustrativních příkladů.

```
>> A= zeros(3,3)
A=
     0     0     0
     0     0     0
     0     0     0
>> A(1,1)
ans =
     0
>> A(:,1)
ans =
     0
     0
     0
```

Další datová položka, kterou používáme, je buňka.

```
>> C{1} = zeros(3,3);
>> C{1}
ans =
     0     0     0
     0     0     0
     0     0     0
```

Chová se jako pole, ale každá položka buňky může být zcela odlišný objekt.

Funkce z INTLABu pro tvorbu intervalů.

```
>> a = infsup(-3,3) % Zadáme interval ve formě infima a suprema.
intval a =
[   -3.0000,    3.0000]
>> a = midrad(0,3) % Zadáme interval ve formě středu a poloměru.
intval a =
[   -3.0000,    3.0000]
>> a = intval('0.1')
intval a =
[    0.0999,    0.1001]
```

Poslední funkce převede hodnotu 0.1 na interval. Hodnotu 0.1 nelze v počítači reprezentovat přesně, a tak ji obalí intervalem malého poloměru. Takto by se měli zadávat konkrétní hodnoty, aby nedošlo k nesprávnému zaokrouhlení a tím k rozbití rigorozity výpočtu.

6.2 Volání funkce ilspenc

Naše hlavní funkce se jmenuje ilspenc(). Zastřešuje všechny implementované metody a můžeme ji zavolat s různými požadavky na přesnost a rychlost. Podle toho se vybere konkrétní metoda, kterou se soustava vyřeší. Pokud během výpočtu dojde k chybě, většinou se na konzoli vypíšou bližší informace a výpočet skončí. V tomto případě funkce vrátí interval, který projde testem isnan(), což je funkce z knihovny INTLAB. Tedy vrátí něco jako NaN, ale pro intervaly.

Syntax: ilspenc(A, b, p, option)

Návratová hodnota: *vektor intervalů* (obálka řešení)

Parametry:

A	Matice koeficientů u parametrů systému k vyřešení. **Datový typ:** viz. další sekce.
b	Pravá strana systému k vyřešení. **Datový typ:** viz. další sekce.
p	Vektor parametrů. **Datový typ:** *vektor intervalů.*
option (volitelný)	**Hodnoty:** 'DEFAULT' - Základní algoritmus (defaultní hodnota). 'TIGHT' - Přesnější metoda. 'TIGHTEST' - Nejpřesnější metoda. 'FASTEST' - Nejrychlejší metoda. **Datový typ:** *řetězec.*

6.3 Formát vstupních dat

Základem přepínání datové reprezentace je globální proměnná *dataModel*. Tu můžeme před výpočtem incializovat dvěmi hodnotami.

```
>> global dataModel = '3D';
>> global dataModel = 'cell';
```

Nenastavím-li při volání naší funkce proměnnou *dataModel*, program předpokládá, že objekty A a b jsou reprezentovány polem. Tedy, matici A^k, kterou jsme popsali v teoretické části, očekává na pozici A(:,:,k) a vektor b^k na pozici b(:,k). Jak už bývá zvykem, první index čísluje řádky, druhý sloupce. Znak ':' znamená všechny řádky/sloupce, tudíž A(:,:,k) vrátí celou matici (dvourozměrné pole).

Zadáme-li *dataModel* = 'cell', očekává naše funkce objekty A a b ve formátu buněk. Zde zadefinujeme formát, který je nutno dodržet pro správný běh programu.

První buňka, tedy A{1} a b{1}, je vyhrazena pro řídící data.

```
>> A{1} = [m; n; p;]
>> b{1} = [m; p;]
```

Hodnota m udává počet řádků systému, n počet sloupců. Poslední hodnota p funguje jako příznak, který zatím může nabývat dvou hodnot. Číslo 0, pokud jsou jednotlivé koeficienty datového typu *double*, číslo 1 pro typ *interval*. Do budoucna můžou mít vlastní příznak i různé typy matic, jako jsou například symetrické nebo Toeplitzovy matice.

Další buňky kódují matici A^k a vektor b^k následujícím způsobem.

```
>> A{k+1} = [x1 x2 ... ; y1 y2 ... ; v1 v2 ... ;]
>> b{k+1} = [x1 x2 ... ; v1 v2 ... ;]
```

Pro matici A^k, trojice hodnot (x1, y1, v1) určuje pozici, x1 číslo řádku, y1 číslo sloupce, a hodnotu v1 na této pozici. Podobně dvojice (x1, v1), ale pro vektor b^k.

6.4 Volání ilspenctoeplitz

Pro uživatelsky přívětivější zadávání dat jsme vytvořili funkci, která vytvoří potřebnou datovou reprezentaci za uživatele, ale pouze pro Toeplitzovy matice.

Syntax: ilspenctoeplitz(p, b)

Návratová hodnota: [A, b, p] (*reprezentace polem / buňkou*)

Parametry:

p	Vektor parametrů. **Datový typ:** *pole intervalů.*
b	Pravá strana systému k vyřešení. **Datový typ:** *pole intervalů / pole double hodnot.*

Pořadí parametrů udává, kde se v Toeplitzově matici nacházejí. Máme-li vektor parametrů zadaný jako

$$\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5, \boldsymbol{p}_6, \boldsymbol{p}_7, \boldsymbol{p}_8, \boldsymbol{p}_9),$$

pak chápeme, že jejich pozice vypadají takto

$$\begin{pmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_3 & \boldsymbol{p}_4 & \boldsymbol{p}_5 \\ \boldsymbol{p}_6 & \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_3 & \boldsymbol{p}_4 \\ \boldsymbol{p}_7 & \boldsymbol{p}_6 & \boldsymbol{p}_1 & \boldsymbol{p}_2 & \boldsymbol{p}_3 \\ \boldsymbol{p}_8 & \boldsymbol{p}_7 & \boldsymbol{p}_6 & \boldsymbol{p}_1 & \boldsymbol{p}_2 \\ \boldsymbol{p}_9 & \boldsymbol{p}_8 & \boldsymbol{p}_7 & \boldsymbol{p}_6 & \boldsymbol{p}_1 \end{pmatrix}.$$

Funkce vrací soustavu v reprezentaci polem nebo buňkou, podle nastavení globální proměnné *dataModel* (řetězec '3D' nebo 'cell'). Vektor $\boldsymbol{p}$ se pozmění. Na začátek se přidá hodnota [1,1], tedy degenerovaný interval, který reprezentuje konstanty. Pravá strana (vektor b), může být jak pole intervalů, tak pole double hodnot. V každém případě ji interně chápeme jako onen první parametr nulové šířky.

7. Programátorská dokumentace

7.1 Úvodní komentář

Programová část této práce není nijak rozsáhlá. Jelikož nejde o žádnou pokročilou aplikaci, vývojové prostředí MATLAB ani pro takový typ projektů není vhodnou volbou, skládá se náš kód ze samostatných funkcích, bez složitější struktury a objektů. V následujících sekcích vás zběžně seznámíme s rolí jednotlivých funkcí a implementačními problémy, které jsme museli řešit. Nebudeme rozepisovat a komentovat jednotlivé řádky kódu, potřebný komentář už zdrojový kód obsahuje.

7.2 MATLAB/INTLAB/VERSOFT

Z knihovny INTLAB jsme použili objekt *interval* a na něm implementovanou intervalovou aritmetiku. Dále funkci **verifylss**(), která vypočítá rigorózní obálku intervalového systému bez závislostí a funkci **mid**(), vracející střed intervalu, intervalové matice či intervalového vektoru. Poslední externí funkcí je **verspecrad**() z knihovny VERSOFT (knihovna pro verifikované výpočty stavící na INTLABu), počítající verifikovaný spektrální radius.

7.3 Funkce

Pokud neuvedeme jinak, návratová hodnota všech dále zmíněných funkcí je intervalový vektor, který je rigorózně spočítanou obálkou množiny řešení. Volací konvenci představujeme pouze v sekci patřící dané funkci. Některé argumenty se opakují, například A značí matici A intervalového systému, b pravou stranu, p vektor parametrů. Dále je už nebudeme vysvětlovat. Pokud funkci zmiňujeme v jiné sekci, než ta do které patří, dovolujeme si zápis zkrátit a uvádíme pouze název funkce. Názvy funkcí se drží dohodnuté konvence v rámci zmiňované fakultní knihovny.

7.3.1 ilspenc

Hlavní funkce, jejíž formát volání jsme probírali v kapitole 6. Kontrolujeme zde správný počet parametrů podle proměnné *nargin*, správné nastavení globální proměnné *dataModel*, regularitu vstupní matice pomocí **ilspencisregular**() a v neposlední řadě se pomocí *switch* na parametru *option* určuje, jakou metodu funkce zavolá:
'DEFAULT' - **ilspencresidual**() s parametrem 'SKALNA'.
'TIGHTEST' - **ilspencmono**().
'TIGHT' - **ilspencbauerskeel**() a na výsledek použije **ilspencbsref**().
'FAST' a 'DEFAULT' zatím volají stejnou metodu.

7.3.2 ilspenc(hbr/hbrref/bauerskeel/bsref)

Argument x značí už nějakou předem spočítanou obálku.
ilspenchbr(A, b, p) - generalizace Hans-Bliek-Rohnova odhadu.
ilspenchbrref(A, b, p, x) - vylepšení HBR odhadu.
ilspencbauerskeel(A, b, p) - generalizace Bauer-Skeeleho odhadu.
ilspencbsref(A, b, p, x) - vylepšení BS odhadu.

Více funkce nebudeme z programátorského pohledu rozebírat, jde o přímé implementace metod z kapitoly 3.

7.3.3 ilspenc(residual/skalna/rump/residualform)

Další funkce je **ilspencresidual**(A, b, p, option). Spočítá se residuální tvar pomocí [Ares, bres] = **ilspenresidualform**(A, b, p) a poté se podle hodnoty *option* vybere, kterou metodou se vyřeší:
'RUMP' - **ilspencrump**(Ares, bres, 15), která residuální tvar vyřeší Rumpovým iterativním algoritmem, číslo 15 udává počet iterací a je navrženo samotným autorem.
'SKALNA' - **ilspencskalna**(Ares, bres), která používá přímou metodu Skalne.
'SIMPLE' - **verifylss**(Ares, bres), residuální tvar je vlastně obyčejná intervalová soustava bez parametrů.

7.3.4 ilspenciterate

Funkce **ilspenciterate**(A, b, iterations, option) předá část parametrů konstruktoru třídy **divisor**(p, iterations, option). Třída **divisor** se stará o rozdělení prostoru parametrů p na $2^{iterations}$ subprostorů, které postupně můžeme enumerovat voláním funkce **divisor**.**NextPar**(), která vrací další subprostor ve formě upraveného vektoru parametrů. Argument *option* udává pravidlo, podle kterého se vybírá další parametr, který rozdělí v polovině. 'MAX' znamená výběr nejširšího, 'RND' slouží pro uniformě náhodný výběr. Na takto rozdělené subprostory se volá **ilspencresidual**() s argumentem 'SKALNA'. Zatím není zahrnuta v návrhu **ilspenc**().

Třída **divisor** je navržena tak, aby pomocí vnitřní funkce **findMax**(option) nalezla pole indexů, na kterých se nacházejí intervaly k rozpůlení podle daného pravidla. Dále obsahuje privátní proměnnou *state*, udávající aktuální subprostor. Jelikož jednotlivé intervaly pouze půlíme, lze *state* reprezentovat jako pole jedniček a nul, kde 0 znamená spodní polovina a 1 horní polovina intervalu. Díváme-li se na *state* jako na binární zápis čísla, další stav dostaneme jako

$$(state + 1) \mod 2$$

zavoláním funkce **divisor**.**UpdateState**(). V kombinaci s předem spočítaným polem indexů k rozdělení, jsme schopni iterativně enumerovat všechny subprostory.

7.3.5 ilspencmono

Návrh funkce **ilspencmono**(A, b, p, option) jsme popsali v sekci 4.5. Počáteční obálka se spočítá pomocí **ilspencresidual**(A, b, p, 'SKALNA'), jednotlivá znaménka monotonie pak funkcí **verifylss**(). Aktuální implementace funguje pouze s *option* hodnotou 'NOIMPROVE', která označuje přístup, kdy na základě počtu určených znamének nic nepodnikáme. Ponechali jsme zde prostor pro rozšíření o jiné přístupy, například rozdělení intervalů, které mají neurčitá znaménka, na menší nebo rekurzivní dopočítání znamének v systému, kde už jsme některá znaménka určili a můžeme tedy takové intervaly zdegenerovat na bodové.

Při výpočtu derivací se řeší soustava, která má jako pravou stranu intervalový vektor b, jak jsme popisovali v teoretické části. Abychom mohli takto explicitně zadaný výraz předat našim funkcím, musíme použít **ilspencmakeb**(b), kde dojde ke zjištění aktuální datové reprezentace z globální položky *dataModel* a převedení b do této reprezentace. Funkce pak vrací upravený intervalový vektor b, který už je ve správném tvaru a kompatibilní s dalšími funkcemi pro výpočet obálky řešení. Konkrétně v implementaci **ilspencmono**() ale funkci nepoužíváme, neboť **verifylss**() umí pracovat pouze s intervalovým vektorem jako pravou stranou a na aktuální datovou reprezentaci nebere ohled.

Odhady jednotlivých složek řešení $\boldsymbol{x}$ počítáme zvlášť pomocí oddělené funkce **ilspencmonogetbound**(A, b, p, d, i, option), kde d je intervalový vektor, ve kterém se nacházejí derivace i-té složky $\boldsymbol{x}$ pro všechny parametry. V této funkci probíhá část, kdy se na základě znamének derivací odvodí pro $\boldsymbol{x}_i$ nový vektor parametrů a vyřeší se s ním celý systém. Funkce vrací jediný interval, nový odhad pro $\boldsymbol{x}_i$. *Option* musí nabývat jedné z hodnot:
'SHARP' - pro výpočet pomocí **ilspencresidual**() s argumentem 'SKALNA'.
'FAST' - pro relaxaci **ilspencrelax**() a následný výpočet s **verifylss**().
Jedná se o funkci, kterou uživatel nepotřebuje vidět a v naší implementaci se volá s hodnotu 'SHARP'.

7.3.6 Pomocné funkce

ilspencisregular(A, p) vrací hodnotu 1, pokud je matice regulární, 0 pokud test neuspěl. Jedná se o parametrickou implementaci tvrzení 2 a věty 4.

ilspencradius(vector) vrací interval obsahující radius intervalového vektoru.

Volání [A, b, p] = **ilspenctoeplitz**(p, b) bylo už popsáno v kapitole 6. Vektor parametrů p vrací s přidaným prvním parametrem $p(1) = intval('1')$ kvůli budoucí kompatibilitě se systémy obsahujícími neparametrizované hodnoty a otestuje jestli sedí počet parametrů (podle dimenze b se určí rozměr A a ten určuje počet parametrů jednoznačně). V závislosti na hodnotě globální proměnné *dataModel* je návratová hodnota A a b objekt v dané reprezentaci. Pro vytvoření potřebných matic v reprezentaci polem využíváme MATLAB funkce **toeplitz**(). Pro reprezentaci buňkou se indexy dopočítají mechanicky.

[A, b] = **ilspencrelax**(A, b, p) zrelaxuje parametrický systém na normální.

Další funkce, které uvedeme, mají za úkol přidat mezi datovou reprezentaci a běh zbylých metod vrstvu, zjednodušující implementaci jiných datových reprezentací. Pokud bude potřeba výpočet urychlit, tyto metody se dají rychle přeimplementovat, aby pracovali přímo s jednou konkrétní reprezentací.

[m,n] = **ilspencmatrixdim**(A) vrací rozměry matice (řádky, sloupce).

ilspencbdim(b) vrací rozměr pravé strany systému. Při správně zadaném vstupu se tato hodnota rovná n z předchozí funkce.

ilspencmatrixcenter(A, p) spočítá $A(p^c)$ a **ilspencbcenter**(b, p) spočítá $b(p^c)$.

ilspencgetak(A, k) vrací A^k a **ilspencgetbk**(b, k) vrací b^k. Pokud je reprezentace pole, vrací přímo obsah daného indexu, pokud je to buňka, vytvoří se klasické pole na základě kompaktních informací, které buňka obsahuje. Zde vzniká časový rozdíl mezi oběma reprezentacemi, neboť dochází k intenzivní alokaci nové paměti a jejímu vyplnění. V tomto bodě také hraje roli parametr z datové reprezentace, který jsme zmiňovali v sekci 6.3. Pokud je 0, alokuje se matice typu *double*, pokud 1, tak typu *interval*. Další komplikaci přináší náš kompaktní zápis [x; y; v;], kdy musí být všechny tři hodnoty stejného typu. U *double* hodnot žádný problém není, MATLAB čísla typu 1.0000 konvertuje na *integer* a dá se s nimi indexovat. Bohužel to neplatí pro intervaly typu [1.0000,1.000] a musíme indexovat například infimem.

Zde připomeňme, že MATLAB používá tzv. lazy-copying. Místo toho, aby se celá reprezentace matice A kopírovala jako lokální proměnná některé z výše uvedených funkcí, předává se pouze reference na původní místo v paměti. Pouze při zápisu do této proměnné se zkopíruje, ale to se v aktuální implementaci neděje, a tak při jejich intenzivním volání nedochází v tomto ohledu k žádnému zpomalení.

Závěr

Implementovali jsme nejnovější přístupy pro řešení soustav intervalových lineárních rovnic s lineárně zavislými parametry v prostředí MATLAB. Na základě teoretické analýzy jsme navrhli konkrétní způsob implementace a podle výsledků numerické analýzy jsme vytvořili jedinou funkci, která se bude uživatelem volat a zastřešuje vybranou podmnožinu všech metod. Správným umístěním intervalové aritmetiky jsme dosáhli rigorózního výpočtu. V nově navržené a úsporné datové reprezentaci lze jednoduše vytvářet Toeplitzovy matice a ostatní algoritmy s ní bez problému pracují díky vrstvě funkcí sloužících jako interface. Její rychlost se pro Toeplitzovy matice ukázala srovnatelnou s rychlostí reprezentace polem. Jak roste počet parametrů v systému, vznikají měřitelné rozdíly, které jsou ale stále v rozumných mezích pro například symetrické matice a dochází k velmi vysoké úspoře paměti. Nastínili jsme některé možnosti, jak naší funkci upravit či vylepšit, hlavně v případě testovaní monotonie, kdy lze zvyšovat různými přístupy k neurčeným znaménkům přesnost výpočtu. Dále i možné použití dělení prostoru parametrů pro menší soustavy nebo široké intervaly a generalizaci Hans-Bliek-Rohnova algoritmu pro systémy s velkým počtem parametrů.

Práce přináší hned několik výsledků. Hlavním je samotná implementace v jazyce MATLABu. Jelikož se jedná o velmi specifický problém, momentálně nevíme o alternativě v tomto prostředí. V jiných to jsou již zmiňované branch & bound algoritmy řešící obecnější problémy, běžící na platfořme UNIX, a aplikace používající technologii Wolfram webMathematica, jejíž zdrojový kód je bohužel uzavřený a nelze si vybrat mezi více přístupy či metodami. Z toho důvodu ani nemůžeme objektivně rychlost a přesnost výpočtů porovnat.
Dále jsme navrhli velmi úspornou reprezentaci pomocí buňek, která je sice pomalejší než přímá reprezentace polem, ale tento časový rozdíl roste velmi pomalu a pro například Toeplitzovy matice je nerozpoznatelný.
Také o jiném numerickém porovnání jednotlivých přístupů v době psaní práce nevíme. Díky němu jsme zjistili, které metody si vedou prokazatelně lépe ve všech ohledech a které zaostávají. Získaná data nám umožnila nastínit, jak případně pokračovat dále a potenciál některých metod pro konkrétní problémy.
V neposlední řadě může teoretická část textu sloužit jako stručný úvod do dané problematiky, neboť je psána s důrazem na jednoduchost a vymezením se na pouze potřebná fakta a definice.

Seznam použité literatury

Bauer, F. L. (1966). Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *ZAMM*, **46**(7), 409–421.

Beeck, H. (1973). Charakterisierung der Lösungsmenge von Intervallgleichungssystemen. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, **53**(12), T181–T182.

Beeck, H. (1975). Zur Problematik der Hüllenbestimmung von Intervallgleichungssystemen. In *Interval Mathematics*, pages 150–159. Springer.

Hladík, M. (2012). Enclosures for the solution set of parametric interval linear systems. *Int. J. Appl. Math. Comput. Sci.*, **22**(3), 561–574. doi: 10.2478/v10006-012-0043-4.

Manoukian, M. V. a Constantinides, G. A. (2011). Accurate floatingloating point arithmetic through hardware error-free transformations. In *ARC*, pages 94–101. Springer.

Oettli, W. a Prager, W. (1964). Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numerische Mathematik*, **6**(1), 405–409.

Popova, E. D. (2009). Explicit characterization of a class of parametric solution sets. *Comptes rendus de l'Académie bulgare des Sciences*, **62**(10), 1207–1216.

Popova, E. D., E. D. (2004). Parametric interval linear solver. *Numerical Algorithms*, **37**(1-4), 345–356.

Rohn, J. (1994). NP-hardness results for linear algebraic problems with interval data. In *Topics in validated computations*. North-Holland.

Rohn, J. (1993). Cheap and tight bounds: The recent result by E. Hansen can be made more efficient. *Interval Computations*, **4**(13-21), 2.

Rohn, J. (2004). A method for handling dependent data in interval linear systems. Technical Report 911, Institute of Computer Science, Academy of Sciences of the Czech Republic.

Rohn, J. (2010). An improvement of the Bauer-Skeel bounds. *Technichal Report No*, **1065**, 1–9.

Rump, S. M. (2010). Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, **19**, 287–449.

Skalna, I. (2006). A method for outer interval solution of systems of linear equations depending linearly on interval parameters. *Reliable computing*, **12** (2), 107–120.

Skalna, I. (2007). On checking the monotonicity of parametric interval solution of linear structural systems. In *Parallel Processing and Applied Mathematics*, pages 1400–1409. Springer.

Skeel, R. D. (1979). Scaling for numerical stability in Gaussian elimination. *Journal of the ACM*, **26**(3), 494–526.

Tabulky výpočtů

Rozhodli jsme se naměřené hodnoty přidat v tištěné podobě k textu práce, aby byly pro čtenáře jednoduše dostupné a pomohli mu se v našich průběžných výsledcích orientovat. Význam jednotlivých sloupců je zřejmý. Uvádíme průměrou šířku řešení, průměrné časy pro obě reprezentace, vlastnosti systému a čísla metod, které se shodují s čísly v kapitole 5. Nejdříve uvedeme tabulky pro Toeplitzovy a symetrické matice. Následuje samostatná tabulka pro metodu SubdivN, která místo čísla metody obsahuje sloupec s počtem dělení vektoru parametrů. Na závěr přikládáme tabulku pro náhodné matice, která je delší než ostatní, protože jako další vlastnost systému zde uvádíme počet parametrů. Jelikož u posledních dvou testů, tedy u SubdivN a náhodných matic, již nechceme porovnávat časovou náročnost obou reprezentací, probíhaly testy pouze na reprezentaci polem a uvedené popisy sloupců tomu odpovídají. Pro lepší čitelnost jsou průměrné šířky řešení zrelativizovány k výsledku metody 7 pro dimenzi 100 a pro odpovídající poloměr intervalu parametrů.

Tabulky výpočtů - Toeplitzovy matice

Poloměr: 0.05				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
23.73113	0.01604	0.01656	5	1
23.70054	0.08352	0.08256	5	2
28.94077	0.02772	0.02886	5	3
23.99226	0.08387	0.08494	5	4
23.74661	0.01807	0.01720	5	5
23.73113	0.01784	0.01552	5	6
23.56330	0.22318	0.22837	5	7
10.14595	0.02864	0.02894	10	1
10.12776	0.30372	0.30102	10	2
13.21854	0.05062	0.05047	10	3
10.17344	0.28618	0.28361	10	4
10.15254	0.02855	0.02889	10	5
10.14595	0.02801	0.02828	10	6
10.07433	0.75135	0.76556	10	7
7.21603	0.04110	0.04173	15	1
7.20112	0.66837	0.65862	15	2
9.89024	0.07395	0.07367	15	3
7.24874	0.61503	0.59898	15	4
7.22106	0.04219	0.04330	15	5
7.21603	0.04077	0.04167	15	6
7.16260	1.61692	1.64623	15	7
4.96942	0.05455	0.05460	20	1
4.95993	1.17883	1.17034	20	2
6.72900	0.09725	0.09787	20	3
4.98761	1.04652	1.04041	20	4
4.97271	0.05571	0.05716	20	5
4.96942	0.05335	0.05456	20	6
4.93399	2.81700	2.86481	20	7
4.05420	0.06890	0.06851	25	1
4.04588	1.83380	1.81555	25	2
5.55149	0.12138	0.12141	25	3
4.07627	1.61648	1.60497	25	4
4.05710	0.06959	0.07152	25	5
4.05420	0.06673	0.06799	25	6
4.02400	4.34970	4.41616	25	7
2.00929	0.15035	0.15248	50	1
2.00465	7.34934	7.37121	50	2
2.86624	0.25305	0.25603	50	3
2.01587	6.31474	6.31735	50	4
2.01065	0.14805	0.15181	50	5
2.00929	0.14715	0.15022	50	6
1.99419	18.19115	18.52612	50	7
1.00755	0.37196	0.37243	100	1
1.00523	32.46518	32.56824	100	2
1.45916	0.55653	0.56073	100	3
1.01316	26.94259	26.93182	100	4
1.00832	0.35419	0.35639	100	5
1.00755	0.35405	0.35616	100	6
1.00000	84.05430	84.62409	100	7

Poloměr: 0.10				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
22.45674	0.01546	0.01570	5	1
22.39086	0.07980	0.07993	5	2
26.90742	0.02704	0.02730	5	3
22.58878	0.08228	0.08219	5	4
22.48478	0.01546	0.01572	5	5
22.45674	0.01521	0.01513	5	6
22.11831	0.21919	0.22137	5	7
11.43993	0.02875	0.02896	10	1
11.40027	0.30553	0.30093	10	2
14.71286	0.05094	0.05051	10	3
11.48247	0.28622	0.28056	10	4
11.45395	0.02875	0.02897	10	5
11.43993	0.02814	0.02844	10	6
11.27274	0.75024	0.76849	10	7
7.01719	0.04081	0.04150	15	1
6.98953	0.66125	0.65837	15	2
9.53251	0.07356	0.07381	15	3
7.06886	0.60407	0.60008	15	4
7.02583	0.04209	0.04328	15	5
7.01719	0.04073	0.04173	15	6
6.91472	1.61715	1.64207	15	7
5.37252	0.05423	0.05483	20	1
5.35043	1.17234	1.16546	20	2
7.39768	0.09721	0.09819	20	3
5.40593	1.04553	1.04289	20	4
5.37914	0.05613	0.05701	20	5
5.37252	0.05335	0.05463	20	6
5.29377	2.81790	2.87270	20	7
4.09872	0.06859	0.06843	25	1
4.08038	1.83248	1.82448	25	2
5.76347	0.12104	0.12238	25	3
4.11841	1.61096	1.60800	25	4
4.10381	0.06913	0.07124	25	5
4.09872	0.06621	0.06789	25	6
4.03697	4.34612	4.42399	25	7
2.00787	0.15036	0.15295	50	1
1.99923	7.37811	7.36707	50	2
2.82493	0.25371	0.25605	50	3
2.01700	6.33280	6.31225	50	4
2.01037	0.14819	0.15122	50	5
2.00787	0.14732	0.14967	50	6
1.97849	18.22289	18.55153	50	7
1.01508	0.37036	0.37205	100	1
1.01037	32.44400	32.49170	100	2
1.46903	0.55946	0.55861	100	3
1.02286	26.92094	26.88333	100	4
1.01642	0.35669	0.35631	100	5
1.01508	0.35487	0.35604	100	6
1.00000	84.20996	84.67261	100	7

Poloměr: 0.50				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
21.32662	0.01571	0.01564	5	1
21.02093	0.08023	0.07938	5	2
26.03882	0.02716	0.02707	5	3
21.24521	0.08202	0.08135	5	4
21.37456	0.01567	0.01571	5	5
21.32662	0.01495	0.01511	5	6
19.89686	0.21854	0.22101	5	7
12.85356	0.02902	0.02876	10	1
12.62860	0.30712	0.30000	10	2
16.51193	0.05143	0.05037	10	3
12.94525	0.28715	0.27879	10	4
12.88519	0.02896	0.02879	10	5
12.85356	0.02836	0.02818	10	6
11.92412	0.75469	0.76768	10	7
8.08936	0.04117	0.04155	15	1
7.92764	0.66270	0.65338	15	2
10.90995	0.07341	0.07318	15	3
8.24925	0.59911	0.59195	15	4
8.10000	0.04137	0.04269	15	5
8.08936	0.04023	0.04119	15	6
7.50265	1.61683	1.64189	15	7
5.35142	0.05412	0.05491	20	1
5.24816	1.16751	1.15378	20	2
7.18588	0.09727	0.09714	20	3
5.38213	1.03520	1.02857	20	4
5.35900	0.05508	0.05685	20	5
5.35142	0.05294	0.05398	20	6
4.96791	2.82258	2.86950	20	7
4.44772	0.06848	0.06840	25	1
4.35046	1.83019	1.81464	25	2
6.16745	0.12155	0.12175	25	3
4.52477	1.60669	1.60289	25	4
4.45335	0.06837	0.06964	25	5
4.44772	0.06718	0.06756	25	6
4.12117	4.33944	4.42233	25	7
2.16229	0.15074	0.15286	50	1
2.11548	7.30134	7.29506	50	2
3.07232	0.25386	0.25549	50	3
2.23647	6.27671	6.26078	50	4
2.16518	0.14710	0.15024	50	5
2.16229	0.14711	0.15029	50	6
2.00639	18.21651	18.56118	50	7
1.07756	0.36614	0.36835	100	1
1.05362	31.66142	31.69919	100	2
1.56496	0.55295	0.55449	100	3
1.13047	26.01132	26.02010	100	4
1.07897	0.35450	0.35635	100	5
1.07756	0.35512	0.35588	100	6
1.00000	84.30367	84.77599	100	7

Poloměr: 1.00				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
32.98204	0.01533	0.01577	5	1
31.85601	0.07894	0.08009	5	2
39.12053	0.02674	0.02732	5	3
32.58532	0.08151	0.08184	5	4
33.43057	0.01604	0.01657	5	5
32.98204	0.01475	0.01524	5	6
28.11464	0.22042	0.22406	5	7
13.45039	0.02834	0.02893	10	1
12.97428	0.29988	0.30111	10	2
17.58061	0.05004	0.05065	10	3
13.98671	0.28155	0.28151	10	4
13.62962	0.02899	0.02963	10	5
13.45039	0.02769	0.02835	10	6
11.62991	0.75319	0.76807	10	7
8.19423	0.04090	0.04141	15	1
7.87082	0.66953	0.65821	15	2
11.06690	0.07357	0.07427	15	3
8.43859	0.60809	0.60459	15	4
8.30380	0.04300	0.04367	15	5
8.19423	0.04024	0.04105	15	6
7.06300	1.62248	1.65286	15	7
6.00255	0.05473	0.05469	20	1
5.76504	1.18559	1.17190	20	2
8.28277	0.09738	0.09794	20	3
6.27498	1.05953	1.05331	20	4
6.08226	0.05593	0.05749	20	5
6.00255	0.05296	0.05392	20	6
5.19445	2.81594	2.86347	20	7
4.88027	0.06884	0.06862	25	1
4.68784	1.84111	1.82506	25	2
6.75191	0.12112	0.12164	25	3
5.08325	1.62453	1.61819	25	4
4.94475	0.06822	0.07095	25	5
4.88027	0.06671	0.06790	25	6
4.22512	4.35573	4.42849	25	7
2.35574	0.15018	0.15247	50	1
2.26014	7.38031	7.38014	50	2
3.32348	0.25367	0.25513	50	3
2.49609	6.35716	6.35959	50	4
2.38699	0.14743	0.15094	50	5
2.35574	0.14666	0.14993	50	6
2.03878	18.27425	18.55073	50	7
1.15615	0.36576	0.36868	100	1
1.10870	32.03539	32.11252	100	2
1.66885	0.55436	0.55328	100	3
1.25678	26.55387	26.56941	100	4
1.17154	0.35523	0.35678	100	5
1.15615	0.35413	0.35618	100	6
1.00000	84.29846	84.79201	100	7

Tabulky výpočtů - symetrické matice

Poloměr: 0.05				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
18.30842	0.02409	0.02857	5	1
18.29353	0.12944	0.13495	5	2
19.73891	0.03456	0.04281	5	3
18.35309	0.12492	0.13009	5	4
18.32080	0.02398	0.02887	5	5
18.30842	0.02343	0.03543	5	6
18.17308	0.34208	0.36150	5	7
10.92997	0.07501	0.07718	10	1
10.91877	0.83782	0.84883	10	2
12.03081	0.09092	0.09366	10	3
10.93100	0.74160	0.75350	10	4
10.93779	0.07469	0.07755	10	5
10.92997	0.07419	0.07697	10	6
10.84609	2.03437	2.10113	10	7
7.08934	0.15968	0.16292	15	1
7.08256	2.67578	2.68440	15	2
7.79861	0.17664	0.18100	15	3
7.09170	2.31989	2.33754	15	4
7.09435	0.15873	0.16295	15	5
7.08934	0.15835	0.16271	15	6
7.03612	6.31334	6.47092	15	7
4.84903	0.27807	0.28289	20	1
4.84432	6.19674	6.19434	20	2
5.32213	0.29255	0.29741	20	3
4.85095	5.31045	5.33888	20	4
4.85243	0.27661	0.28496	20	5
4.84903	0.27604	0.28245	20	6
4.81262	14.42663	14.68939	20	7
4.12133	0.43079	0.43878	25	1
4.11720	11.91493	11.97407	25	2
4.52116	0.43750	0.44810	25	3
4.12325	10.18223	10.26259	25	4
4.12413	0.42811	0.44172	25	5
4.12133	0.42768	0.43972	25	6
4.09037	27.59793	28.25864	25	7
1.90992	1.86021	1.89800	50	1
1.90815	94.00776	94.54608	50	2
2.09199	1.75394	1.79211	50	3
1.91066	79.27094	80.16107	50	4
1.91125	1.82802	1.87178	50	5
1.90992	1.82734	1.87369	50	6
1.89577	227.40286	232.06909	50	7
1.00752	9.08031	9.18427	100	1
1.00663	818.56692	828.27502	100	2
1.10364	8.04673	8.10975	100	3
1.00811	672.21983	684.68667	100	4
1.00826	8.70472	8.78412	100	5
1.00752	8.70656	8.78453	100	6
1.00000	2079.22558	2098.43480	100	7

Poloměr: 0.10				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
19.52041	0.02306	0.02373	5	1
19.47952	0.12505	0.12677	5	2
21.53097	0.03344	0.03491	5	3
19.62988	0.12191	0.12475	5	4
19.54595	0.02318	0.02396	5	5
19.52041	0.02248	0.02341	5	6
19.21288	0.33701	0.34422	5	7
9.62366	0.07542	0.07672	10	1
9.60478	0.83836	0.84658	10	2
10.58111	0.09086	0.09346	10	3
9.62475	0.74002	0.75284	10	4
9.63582	0.07454	0.07702	10	5
9.62366	0.07433	0.07661	10	6
9.47931	2.02940	2.09235	10	7
6.90767	0.16000	0.16303	15	1
6.89457	2.68092	2.69628	15	2
7.56078	0.17696	0.18099	15	3
6.90861	2.31913	2.33871	15	4
6.91643	0.15940	0.16405	15	5
6.90767	0.15876	0.16340	15	6
6.80376	6.31572	6.45931	15	7
5.42301	0.27805	0.28327	20	1
5.41252	6.18677	6.22134	20	2
5.95384	0.29109	0.29904	20	3
5.42699	5.29945	5.36145	20	4
5.42981	0.27753	0.28622	20	5
5.42301	0.27633	0.28376	20	6
5.34190	14.44371	14.77653	20	7
4.08437	0.43316	0.44014	25	1
4.07641	11.93996	12.00202	25	2
4.48784	0.43785	0.44968	25	3
4.08741	10.16922	10.29954	25	4
4.08958	0.42905	0.44221	25	5
4.08437	0.42837	0.44140	25	6
4.02323	27.64751	28.33354	25	7
2.11447	1.86291	1.89064	50	1
2.11042	94.12898	93.82253	50	2
2.32055	1.75434	1.79249	50	3
2.11664	79.38860	79.72959	50	4
2.11715	1.82801	1.87182	50	5
2.11447	1.82939	1.87041	50	6
2.08300	227.23330	231.69068	50	7
1.01505	9.08021	9.19498	100	1
1.01317	817.77686	828.77849	100	2
1.11208	8.03692	8.10936	100	3
1.01614	672.18870	685.65706	100	4
1.01635	8.70769	8.80132	100	5
1.01505	8.70846	8.79961	100	6
1.00000	2077.67265	2100.61895	100	7

Poloměr: 0.50				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
24.15085	0.02309	0.02381	5	1
23.95892	0.12432	0.12603	5	2
26.17780	0.03363	0.03469	5	3
23.80352	0.11981	0.12244	5	4
24.18061	0.02302	0.02385	5	5
24.15085	0.02243	0.02323	5	6
22.41426	0.33480	0.34504	5	7
13.04881	0.07495	0.07693	10	1
12.93527	0.83564	0.84436	10	2
14.24680	0.09069	0.09360	10	3
13.08786	0.73517	0.74850	10	4
13.06269	0.07467	0.07722	10	5
13.04881	0.07413	0.07677	10	6
12.09176	2.03427	2.10028	10	7
7.31070	0.16056	0.16341	15	1
7.24787	2.66906	2.66495	15	2
7.97478	0.17702	0.18058	15	3
7.32808	2.29885	2.31355	15	4
7.31965	0.15863	0.16440	15	5
7.31070	0.15899	0.16372	15	6
6.79054	6.33983	6.47254	15	7
5.18010	0.27925	0.28190	20	1
5.13248	6.16457	6.16962	20	2
5.67239	0.29140	0.29931	20	3
5.19275	5.24386	5.30668	20	4
5.18658	0.27708	0.28361	20	5
5.18010	0.27643	0.28141	20	6
4.80952	14.41788	14.77760	20	7
4.47867	0.43168	0.43852	25	1
4.43857	11.85025	11.90942	25	2
4.91934	0.43802	0.44701	25	3
4.50075	10.06820	10.19966	25	4
4.48427	0.42585	0.43811	25	5
4.47867	0.42838	0.43795	25	6
4.15988	27.63397	28.26765	25	7
2.18597	1.86376	1.89339	50	1
2.16648	93.23621	92.90556	50	2
2.39363	1.75140	1.79314	50	3
2.19774	78.10420	78.62475	50	4
2.18883	1.82451	1.87337	50	5
2.18597	1.82697	1.86968	50	6
2.03260	227.37186	232.20410	50	7
1.07606	9.07227	9.19022	100	1
1.06640	793.43812	805.18880	100	2
1.17984	8.00556	8.11769	100	3
1.08283	642.38421	657.58083	100	4
1.07745	8.70557	8.80580	100	5
1.07606	8.70407	8.80780	100	6
1.00000	2078.86006	2102.99850	100	7

Poloměr: 1.00				
Šířka řešení	Čas - pole	Čas - buňka	Dimenze	Metoda
23.86686	0.02304	0.02380	5	1
23.50336	0.12567	0.12773	5	2
25.47664	0.03391	0.03475	5	3
23.45604	0.12108	0.12389	5	4
24.19076	0.02391	0.02477	5	5
23.86686	0.02250	0.02333	5	6
20.71941	0.33428	0.34699	5	7
10.15783	0.07484	0.07711	10	1
9.98533	0.84348	0.85003	10	2
11.04500	0.09039	0.09355	10	3
10.14609	0.73859	0.75359	10	4
10.29431	0.07532	0.07835	10	5
10.15783	0.07402	0.07688	10	6
8.85489	2.04084	2.10141	10	7
7.26997	0.15959	0.16180	15	1
7.15319	2.68604	2.68224	15	2
7.86905	0.17705	0.18052	15	3
7.28617	2.31509	2.33125	15	4
7.36742	0.15762	0.16403	15	5
7.26997	0.15800	0.16153	15	6
6.36049	6.32503	6.46530	15	7
5.60330	0.27838	0.28223	20	1
5.51370	6.19909	6.22627	20	2
6.10462	0.29152	0.29981	20	3
5.64464	5.28640	5.35303	20	4
5.67860	0.27706	0.28423	20	5
5.60330	0.27645	0.28178	20	6
4.88726	14.41539	14.78402	20	7
4.59142	0.43145	0.44099	25	1
4.51660	11.95145	12.01612	25	2
5.02678	0.43749	0.44947	25	3
4.64035	10.15777	10.30160	25	4
4.65334	0.42502	0.44070	25	5
4.59142	0.42691	0.43987	25	6
4.00324	27.65541	28.37636	25	7
2.25122	1.86468	1.89607	50	1
2.21482	94.00876	93.73500	50	2
2.46062	1.75202	1.79365	50	3
2.27899	78.90933	79.45359	50	4
2.28145	1.82633	1.86973	50	5
2.25122	1.82449	1.87113	50	6
1.97002	227.70369	232.75860	50	7
1.14231	9.07436	9.20523	100	1
1.12401	799.58791	811.50763	100	2
1.24995	8.00223	8.12088	100	3
1.15844	648.63751	663.81897	100	4
1.15762	8.71193	8.80806	100	5
1.14231	8.70285	8.79917	100	6
1.00000	2081.45064	2106.88580	100	7

Tabulky výpočtů - metoda SubdivN

Poloměr: 0.05					
Šířka řešení (Toeplitz.)	Čas (Toeplitz.)	Šířka řešení (symetrická)	Čas (symetrická)	Dimenze	Počet dělení
23.53181	0.04976	18.16696	0.04750	5	1
23.50971	0.06481	18.16096	0.09544	5	2
23.49471	0.13098	18.15628	0.19345	5	3
23.48395	0.26645	18.15086	0.39345	5	4
23.47300	0.54452	18.14486	0.79236	5	5
23.46492	1.10943	18.13550	1.62723	5	6
23.45416	2.27104	18.12833	3.27032	5	7
10.06708	0.05703	10.84767	0.15022	10	1
10.06150	0.11475	10.84680	0.30069	10	2
10.05651	0.23239	10.84592	0.60406	10	3
10.05247	0.47064	10.84519	1.21837	10	4
10.04939	0.95424	10.84416	2.45140	10	5
10.04709	1.93048	10.84328	4.92521	10	6
10.04459	3.90398	10.84241	9.89737	10	7
7.16145	0.08461	7.03614	0.32249	15	1
7.15837	0.16869	7.03600	0.64463	15	2
7.15568	0.34098	7.03570	1.28945	15	3
7.15337	0.68855	7.03556	2.58231	15	4
7.15164	1.38664	7.03526	5.17414	15	5
7.15011	2.78486	7.03497	10.42899	15	6
7.14876	5.59688	7.03483	20.86143	15	7
4.93235	0.11141	4.81270	0.56072	20	1
4.93100	0.22354	4.81255	1.11837	20	2
4.92966	0.44969	4.81255	2.24142	20	3
4.92831	0.90734	4.81241	4.48844	20	4
4.92716	1.81334	4.81226	8.96923	20	5
4.92639	3.64305	4.81226	18.05883	20	6
4.92562	7.31040	4.81212	36.15179	20	7
4.02422	0.13806	4.09043	0.86310	25	1
4.02326	0.27845	4.09043	1.73074	25	2
4.02229	0.55876	4.09043	3.48024	25	3
4.02153	1.12348	4.09028	6.94051	25	4
4.02076	2.24732	4.09028	13.96266	25	5
4.01999	4.50685	4.09028	27.95811	25	6
4.01941	9.06653	4.09014	55.82109	25	7
1.99462	0.29986	1.89567	3.70077	50	1
1.99443	0.60118	1.89567	7.40223	50	2
1.99404	1.20656	1.89567	14.81099	50	3
1.99385	2.41799	1.89567	29.61990	50	4
1.99366	4.85194	1.89567	59.24740	50	5
1.99327	9.72149	1.89567	118.54568	50	6
1.99308	19.48657	1.89567	237.09205	50	7
1.00038	0.71699	1.00000	17.59965	100	1
1.00019	1.43604	1.00000	35.19598	100	2
1.00019	2.87333	1.00000	70.44519	100	3
1.00019	5.75432	1.00000	140.72727	100	4
1.00000	11.51769	1.00000	281.55195	100	5
1.00000	23.05001	1.00000	563.18948	100	6
1.00000	46.13503	1.00000	1126.11468	100	7

Poloměr: 0.10					
Šířka řešení (Toeplitz.)	Čas (Toeplitz.)	Šířka řešení (symetrická)	Čas (symetrická)	Dimenze	Počet dělení
22.07830	0.03141	19.21806	0.04693	5	1
22.04469	0.06367	19.20530	0.09486	5	2
22.01913	0.12983	19.19347	0.19084	5	3
21.98902	0.26544	19.17772	0.38711	5	4
21.95891	0.54184	19.16075	0.78704	5	5
21.93827	1.10634	19.14165	1.59232	5	6
21.92019	2.24892	19.12204	3.24631	5	7
11.26292	0.05723	9.47947	0.15132	10	1
11.25507	0.11507	9.47804	0.30327	10	2
11.24853	0.23289	9.47662	0.60770	10	3
11.24020	0.47201	9.47512	1.21711	10	4
11.23461	0.95367	9.47377	2.44619	10	5
11.22571	1.92907	9.47234	4.92494	10	6
11.21852	3.90910	9.47077	9.93519	10	7
6.91176	0.08536	6.80482	0.32307	15	1
6.90759	0.16907	6.80432	0.64718	15	2
6.90381	0.34068	6.80382	1.29498	15	3
6.90049	0.68504	6.80339	2.59328	15	4
6.89718	1.38587	6.80297	5.19560	15	5
6.89462	2.77628	6.80240	10.38099	15	6
6.89178	5.65943	6.80190	20.88798	15	7
5.29294	0.11099	5.34239	0.55766	20	1
5.29076	0.22106	5.34217	1.11642	20	2
5.28868	0.44849	5.34196	2.23288	20	3
5.28707	0.90342	5.34175	4.46661	20	4
5.28565	1.81870	5.34160	8.96822	20	5
5.28375	3.64625	5.34139	18.06193	20	6
5.28176	7.34310	5.34110	36.06667	20	7
4.03853	0.13883	4.02374	0.86689	25	1
4.03683	0.27856	4.02367	1.73412	25	2
4.03513	0.56055	4.02352	3.47412	25	3
4.03371	1.12607	4.02345	6.93110	25	4
4.03285	2.26089	4.02338	13.88057	25	5
4.03162	4.53366	4.02324	27.87067	25	6
4.03039	9.09348	4.02317	55.87972	25	7
1.97898	0.30135	2.08312	3.70199	50	1
1.97851	0.60277	2.08312	7.39659	50	2
1.97803	1.20934	2.08312	14.79371	50	3
1.97766	2.42377	2.08305	29.62094	50	4
1.97737	4.86094	2.08305	59.21221	50	5
1.97709	9.73802	2.08305	118.45818	50	6
1.97680	19.52350	2.08305	236.95926	50	7
1.00066	0.71602	1.00000	17.71207	100	1
1.00057	1.43330	1.00000	35.33830	100	2
1.00047	2.86924	1.00000	70.68169	100	3
1.00028	5.74396	1.00000	141.41894	100	4
1.00019	11.49836	1.00000	282.37686	100	5
1.00009	23.02583	1.00000	566.88022	100	6
1.00000	46.10458	1.00000	1132.93273	100	7

Poloměr: 0.50					
Šířka řešení (Toeplitz.)	Čas (Toeplitz.)	Šířka řešení (symetrická)	Čas (symetrická)	Dimenze	Počet dělení
19.62961	0.03127	22.37400	0.04694	5	1
19.43693	0.06293	22.30266	0.09459	5	2
19.29580	0.12836	22.23103	0.19195	5	3
19.19753	0.26220	22.15316	0.38962	5	4
19.13998	0.53534	22.05941	0.79223	5	5
19.01832	1.09384	21.95333	1.60661	5	6
18.93396	2.23247	21.85967	3.25123	5	7
11.91983	0.05705	12.11698	0.15141	10	1
11.84223	0.11509	12.10852	0.30441	10	2
11.77639	0.23242	12.09809	0.61159	10	3
11.72099	0.47167	12.08679	1.22616	10	4
11.67441	0.95321	12.07781	2.45214	10	5
11.63530	1.92860	12.06705	4.91030	10	6
11.60373	3.89511	12.05817	9.90949	10	7
7.51155	0.08407	6.79158	0.32177	15	1
7.47744	0.16738	6.78884	0.64555	15	2
7.44673	0.33970	6.78656	1.29392	15	3
7.41914	0.68656	6.78426	2.59609	15	4
7.39388	1.37910	6.78196	5.18663	15	5
7.37159	2.76977	6.77984	10.41969	15	6
7.35173	5.60277	6.77761	20.85911	15	7
4.97172	0.11081	4.81318	0.55865	20	1
4.95475	0.22372	4.81226	1.12372	20	2
4.93893	0.45115	4.81137	2.24956	20	3
4.92431	0.90260	4.81032	4.49032	20	4
4.91092	1.82183	4.80938	8.99601	20	5
4.89851	3.66104	4.80843	18.05760	20	6
4.88719	7.33533	4.80743	36.07978	20	7
4.13730	0.13734	4.16187	0.87319	25	1
4.12527	0.27900	4.16134	1.76916	25	2
4.11379	0.56220	4.16077	3.49307	25	3
4.10308	1.12927	4.16022	6.94874	25	4
4.09303	2.26439	4.15971	13.87344	25	5
4.08356	4.54785	4.15916	27.83675	25	6
4.07476	9.09871	4.15858	55.84031	25	7
2.01389	0.30078	2.03150	3.69437	50	1
2.01135	0.60222	2.03142	7.41258	50	2
2.00867	1.20835	2.03136	14.81792	50	3
2.00580	2.42268	2.03131	29.61921	50	4
2.00303	4.85621	2.03124	59.26965	50	5
2.00033	9.73694	2.03117	118.53012	50	6
1.99772	19.49756	2.03111	237.08552	50	7
1.00425	0.71610	1.00004	17.83718	100	1
1.00366	1.43461	1.00004	35.56187	100	2
1.00292	2.87170	1.00003	70.83855	100	3
1.00217	5.74672	1.00003	141.12519	100	4
1.00144	11.50510	1.00001	285.08497	100	5
1.00071	23.02966	1.00000	564.96350	100	6
1.00000	46.09320	1.00000	1134.06238	100	7

Poloměr: 1.00					
Šířka řešení (Toeplitz.)	Čas (Toeplitz.)	Šířka řešení (symetrická)	Čas (symetrická)	Dimenze	Počet dělení
27.95535	0.03142	20.72680	0.04650	5	1
27.24337	0.06330	20.58117	0.09424	5	2
26.74561	0.12891	20.47050	0.19104	5	3
26.40585	0.26202	20.25120	0.38932	5	4
26.23740	0.53602	20.04841	0.79148	5	5
25.83826	1.09511	19.89925	1.59796	5	6
25.54908	2.22861	19.73748	3.23085	5	7
11.57638	0.05701	8.87885	0.14991	10	1
11.43516	0.11491	8.86586	0.30178	10	2
11.30409	0.23238	8.85100	0.60579	10	3
11.19325	0.46967	8.83770	1.21918	10	4
11.10121	0.95166	8.82317	2.45664	10	5
11.02101	1.92511	8.81128	4.90967	10	6
10.95434	3.88862	8.79947	9.89713	10	7
7.09251	0.08422	6.36047	0.31861	15	1
7.02581	0.16712	6.35614	0.64229	15	2
6.96417	0.34133	6.35221	1.28909	15	3
6.90921	0.68780	6.34773	2.59023	15	4
6.86056	1.38410	6.34289	5.20776	15	5
6.81716	2.78979	6.33994	10.42985	15	6
6.77872	5.61515	6.33515	20.82349	15	7
5.20017	0.11030	4.90390	0.56227	20	1
5.16280	0.22209	4.90198	1.12422	20	2
5.12847	0.44952	4.89996	2.25040	20	3
5.09682	0.90448	4.89790	4.50612	20	4
5.06761	1.81452	4.89566	8.98900	20	5
5.04111	3.64651	4.89342	17.98188	20	6
5.01704	7.33226	4.89138	36.12761	20	7
4.23869	0.13837	4.01898	0.87343	25	1
4.21696	0.27901	4.01799	1.75051	25	2
4.19340	0.56006	4.01692	3.50607	25	3
4.17101	1.12476	4.01601	6.98972	25	4
4.15024	2.25775	4.01493	13.94027	25	5
4.13087	4.53956	4.01370	27.93166	25	6
4.11281	9.10220	4.01267	55.87249	25	7
2.05046	0.29985	1.97089	3.69690	50	1
2.04663	0.60140	1.97075	7.41534	50	2
2.04134	1.20816	1.97061	14.81300	50	3
2.03569	2.41930	1.97047	29.67106	50	4
2.02983	4.84998	1.97035	59.32627	50	5
2.02414	9.72812	1.97023	118.59904	50	6
2.01860	19.50083	1.97008	237.22083	50	7
1.00770	0.71668	1.00010	17.62019	100	1
1.00657	1.43474	1.00009	35.23676	100	2
1.00537	2.87293	1.00007	70.46560	100	3
1.00434	5.75204	1.00005	141.02305	100	4
1.00293	11.51245	1.00003	281.89069	100	5
1.00151	23.04820	1.00002	563.86677	100	6
1.00000	46.12425	1.00000	1127.68535	100	7

Tabulky výpočtů - náhodné matice

Poloměr: 0.05				
Šířka řešení	Čas	Dimenze	Parametrů	Metoda
72.51163	0.01071	5	5	1
72.48837	0.04974	5	5	2
88.37209	0.02374	5	5	3
80.86047	0.05676	5	5	4
72.53488	0.01075	5	5	5
72.51163	0.00993	5	5	6
72.32558	0.14423	5	5	7
215.58140	0.01729	5	10	1
215.27907	0.08815	5	10	2
281.37209	0.02899	5	10	3
229.13953	0.09030	5	10	4
215.69767	0.01725	5	10	5
215.58140	0.01682	5	10	6
214.27907	0.24327	5	10	7
296.41860	0.02415	5	15	1
295.44186	0.12911	5	15	2
426.44186	0.03520	5	15	3
315.65116	0.12599	5	15	4
296.67442	0.02437	5	15	5
296.41860	0.02364	5	15	6
293.41860	0.34661	5	15	7
29.81395	0.01703	10	10	1
29.81395	0.16659	10	10	2
35.27907	0.04142	10	10	3
32.95349	0.17030	10	10	4
29.81395	0.01725	10	10	5
29.81395	0.01669	10	10	6
29.76744	0.44997	10	10	7
49.25581	0.02998	10	20	1
49.23256	0.31519	10	20	2
61.41860	0.05239	10	20	3
53.74419	0.29718	10	20	4
49.27907	0.03028	10	20	5
49.25581	0.02957	10	20	6
49.11628	0.80408	10	20	7
94.51163	0.04329	10	30	1
94.39535	0.46569	10	30	2
122.13953	0.06429	10	30	3
99.44186	0.42568	10	30	4
94.55814	0.04325	10	30	5
94.51163	0.04264	10	30	6
94.00000	1.15940	10	30	7
13.53488	0.02349	15	15	1
13.53488	0.35806	15	15	2
15.16279	0.05979	15	15	3
14.58140	0.34476	15	15	4
13.53488	0.02353	15	15	5
13.53488	0.02300	15	15	6
13.51163	0.90152	15	15	7
27.04651	0.04202	15	30	1
27.04651	0.67114	15	30	2
31.13953	0.07473	15	30	3
28.48837	0.61204	15	30	4
27.04651	0.04206	15	30	5

27.04651	0.04159	15	30	6
27.00000	1.64547	15	30	7
43.58140	0.06171	15	45	1
43.55814	1.00030	15	45	2
52.69767	0.09219	15	45	3
45.79070	0.89268	15	45	4
43.60465	0.06206	15	45	5
43.58140	0.06110	15	45	6
43.44186	2.42068	15	45	7
8.04651	0.03011	20	20	1
8.04651	0.61240	20	20	2
8.79070	0.07672	20	20	3
8.60465	0.57252	20	20	4
8.04651	0.03017	20	20	5
8.04651	0.02954	20	20	6
8.04651	1.52468	20	20	7
15.25581	0.05571	20	40	1
15.25581	1.18573	20	40	2
17.04651	0.09905	20	40	3
16.00000	1.06326	20	40	4
15.25581	0.05595	20	40	5
15.25581	0.05497	20	40	6
15.23256	2.86835	20	40	7
23.65116	0.08123	20	60	1
23.65116	1.76198	20	60	2
27.06977	0.12138	20	60	3
24.60465	1.55409	20	60	4
23.67442	0.08207	20	60	5
23.65116	0.08049	20	60	6
23.60465	4.20098	20	60	7
4.86047	0.03636	25	25	1
4.86047	0.93539	25	25	2
5.18605	0.09375	25	25	3
5.09302	0.85831	25	25	4
4.86047	0.03665	25	25	5
4.86047	0.03584	25	25	6
4.86047	2.29798	25	25	7
10.41860	0.06870	25	50	1
10.41860	1.82917	25	50	2
11.37209	0.12217	25	50	3
10.83721	1.62142	25	50	4
10.41860	0.06857	25	50	5
10.41860	0.06798	25	50	6
10.39535	4.38122	25	50	7
15.69767	0.10117	25	75	1
15.69767	2.72061	25	75	2
17.62791	0.15033	25	75	3
16.30233	2.38432	25	75	4
15.69767	0.10108	25	75	5
15.69767	0.10036	25	75	6
15.65116	6.45888	25	75	7
1.27907	0.07762	50	50	1
1.27907	3.64838	50	50	2
1.32558	0.19105	50	50	3
1.32558	3.22283	50	50	4
1.27907	0.07601	50	50	5
1.27907	0.07580	50	50	6

1.27907	9.38101	50	50	7
2.67442	0.15102	50	100	1
2.67442	7.24045	50	100	2
2.76744	0.25610	50	100	3
2.74419	6.26238	50	100	4
2.67442	0.14780	50	100	5
2.67442	0.14768	50	100	6
2.67442	18.23753	50	100	7
4.00000	0.22324	50	150	1
4.00000	10.91636	50	150	2
4.20930	0.32049	50	150	3
4.09302	9.37304	50	150	4
4.00000	0.22003	50	150	5
4.00000	0.22007	50	150	6
4.00000	27.14781	50	150	7
0.32558	0.18587	100	100	1
0.32558	16.12315	100	100	2
0.32558	0.40372	100	100	3
0.32558	13.53382	100	100	4
0.32558	0.17883	100	100	5
0.32558	0.17864	100	100	6
0.32558	42.66455	100	100	7
0.62791	0.36408	100	200	1
0.62791	32.05753	100	200	2
0.65116	0.55691	100	200	3
0.65116	26.70087	100	200	4
0.62791	0.35037	100	200	5
0.62791	0.35126	100	200	6
0.62791	83.60255	100	200	7
1.00000	0.54424	100	300	1
1.00000	48.05889	100	300	2
1.02326	0.71210	100	300	3
1.02326	39.89216	100	300	4
1.00000	0.52320	100	300	5
1.00000	0.52438	100	300	6
1.00000	124.90613	100	300	7

Poloměr: 0.10				
Šířka řešení	Čas	Dimenze	Parametrů	Metoda
109.79268	0.01089	5	5	1
109.65854	0.04960	5	5	2
139.52439	0.02358	5	5	3
121.08537	0.05648	5	5	4
109.85366	0.01084	5	5	5
109.79268	0.01019	5	5	6
109.08537	0.14443	5	5	7
228.34146	0.01729	5	10	1
227.64634	0.08866	5	10	2
308.08537	0.02932	5	10	3
242.54878	0.09049	5	10	4
228.57317	0.01731	5	10	5
228.34146	0.01691	5	10	6
225.59756	0.24441	5	10	7
354.26829	0.02353	5	15	1
352.15854	0.12523	5	15	2
501.54878	0.03440	5	15	3
376.91463	0.12263	5	15	4
354.79268	0.02355	5	15	5
354.26829	0.02318	5	15	6
347.57317	0.34043	5	15	7
31.47561	0.01712	10	10	1
31.47561	0.16740	10	10	2
38.03659	0.04198	10	10	3
35.15854	0.17018	10	10	4
31.48780	0.01721	10	10	5
31.47561	0.01651	10	10	6
31.37805	0.44910	10	10	7
68.32927	0.03016	10	20	1
68.25610	0.31782	10	20	2
82.76829	0.05266	10	20	3
72.08537	0.29775	10	20	4
68.36585	0.03033	10	20	5
68.32927	0.02953	10	20	6
67.86585	0.80663	10	20	7
96.78049	0.04285	10	30	1
96.59756	0.46520	10	30	2
120.63415	0.06423	10	30	3
101.39024	0.42532	10	30	4
96.86585	0.04325	10	30	5
96.78049	0.04234	10	30	6
95.86585	1.16030	10	30	7
15.31707	0.02307	15	15	1
15.31707	0.34796	15	15	2
17.40244	0.05819	15	15	3
16.45122	0.33861	15	15	4
15.31707	0.02307	15	15	5
15.31707	0.02254	15	15	6
15.28049	0.89251	15	15	7
31.65854	0.04221	15	30	1
31.64634	0.67010	15	30	2
36.85366	0.07455	15	30	3
33.20732	0.61082	15	30	4
31.67073	0.04217	15	30	5

31.65854	0.04174	15	30	6
31.52439	1.63993	15	30	7
48.45122	0.06085	15	45	1
48.40244	0.99396	15	45	2
58.18293	0.09122	15	45	3
50.32927	0.88836	15	45	4
48.47561	0.06110	15	45	5
48.45122	0.06062	15	45	6
48.12195	2.41352	15	45	7
7.96341	0.02991	20	20	1
7.96341	0.60878	20	20	2
8.84146	0.07626	20	20	3
8.59756	0.57021	20	20	4
7.97561	0.03011	20	20	5
7.96341	0.02935	20	20	6
7.95122	1.51295	20	20	7
15.32927	0.05545	20	40	1
15.32927	1.17484	20	40	2
17.20732	0.09800	20	40	3
16.24390	1.05367	20	40	4
15.34146	0.05600	20	40	5
15.32927	0.05493	20	40	6
15.29268	2.84241	20	40	7
26.01220	0.08185	20	60	1
26.00000	1.76470	20	60	2
29.73171	0.12192	20	60	3
27.09756	1.55618	20	60	4
26.02439	0.08188	20	60	5
26.01220	0.08085	20	60	6
25.89024	4.21235	20	60	7
5.31707	0.03653	25	25	1
5.31707	0.93683	25	25	2
5.74390	0.09395	25	25	3
5.64634	0.85973	25	25	4
5.31707	0.03656	25	25	5
5.31707	0.03733	25	25	6
5.31707	2.29623	25	25	7
10.67073	0.06880	25	50	1
10.67073	1.82967	25	50	2
11.65854	0.12206	25	50	3
11.13415	1.62299	25	50	4
10.67073	0.06887	25	50	5
10.67073	0.06830	25	50	6
10.64634	4.38167	25	50	7
16.36585	0.10094	25	75	1
16.35366	2.72172	25	75	2
18.18293	0.15044	25	75	3
16.92683	2.37877	25	75	4
16.36585	0.10089	25	75	5
16.36585	0.09971	25	75	6
16.30488	6.45856	25	75	7
1.34146	0.07786	50	50	1
1.34146	3.66664	50	50	2
1.40244	0.19156	50	50	3
1.39024	3.23430	50	50	4
1.34146	0.07599	50	50	5
1.34146	0.07585	50	50	6

1.34146	9.39740	50	50	7
2.75610	0.15069	50	100	1
2.75610	7.23944	50	100	2
2.86585	0.25545	50	100	3
2.82927	6.25334	50	100	4
2.75610	0.14767	50	100	5
2.75610	0.14777	50	100	6
2.74390	18.20747	50	100	7
4.23171	0.22232	50	150	1
4.21951	10.90905	50	150	2
4.45122	0.31966	50	150	3
4.32927	9.37046	50	150	4
4.23171	0.22003	50	150	5
4.23171	0.21932	50	150	6
4.21951	27.11584	50	150	7
0.32927	0.18557	100	100	1
0.32927	16.12549	100	100	2
0.34146	0.40461	100	100	3
0.34146	13.55066	100	100	4
0.32927	0.17865	100	100	5
0.32927	0.17825	100	100	6
0.32927	42.67942	100	100	7
0.65854	0.36437	100	200	1
0.65854	32.05548	100	200	2
0.67073	0.55669	100	200	3
0.67073	26.69068	100	200	4
0.65854	0.35115	100	200	5
0.65854	0.35109	100	200	6
0.65854	83.63514	100	200	7
1.01220	0.54329	100	300	1
1.01220	48.04813	100	300	2
1.02439	0.71021	100	300	3
1.02439	39.89163	100	300	4
1.01220	0.52335	100	300	5
1.01220	0.52412	100	300	6
1.00000	124.95214	100	300	7

Poloměr: 0.50				
Šířka řešení	Čas	Dimenze	Parametrů	Metoda
94.99517	0.01068	5	5	1
94.67391	0.05000	5	5	2
128.12077	0.02360	5	5	3
112.43237	0.05687	5	5	4
95.18841	0.01094	5	5	5
94.99517	0.01010	5	5	6
92.55072	0.14492	5	5	7
224.14010	0.01719	5	10	1
220.97585	0.08827	5	10	2
307.86473	0.02916	5	10	3
242.33092	0.09025	5	10	4
225.20773	0.01783	5	10	5
224.14010	0.01666	5	10	6
210.98309	0.24481	5	10	7
380.23913	0.02362	5	15	1
370.92029	0.12639	5	15	2
513.07246	0.03477	5	15	3
407.69324	0.12254	5	15	4
383.54831	0.02444	5	15	5
380.23913	0.02296	5	15	6
347.71256	0.34211	5	15	7
34.07246	0.01708	10	10	1
34.01932	0.16619	10	10	2
40.83333	0.04165	10	10	3
37.47585	0.16863	10	10	4
34.12077	0.01711	10	10	5
34.07246	0.01685	10	10	6
33.46860	0.44799	10	10	7
68.81884	0.03058	10	20	1
68.47101	0.31408	10	20	2
85.20773	0.05287	10	20	3
73.71981	0.29432	10	20	4
68.95894	0.03011	10	20	5
68.81884	0.02941	10	20	6
66.72947	0.80391	10	20	7
104.21981	0.04287	10	30	1
103.05556	0.46062	10	30	2
133.16908	0.06396	10	30	3
109.10145	0.42405	10	30	4
104.45169	0.04309	10	30	5
104.21981	0.04229	10	30	6
98.88889	1.16374	10	30	7
13.85024	0.02316	15	15	1
13.84300	0.34575	15	15	2
15.69324	0.05813	15	15	3
15.12077	0.33247	15	15	4
13.86232	0.02322	15	15	5
13.85024	0.02265	15	15	6
13.71739	0.88963	15	15	7
27.30193	0.04241	15	30	1
27.24155	0.66655	15	30	2
31.46135	0.07476	15	30	3
28.83092	0.60796	15	30	4
27.34541	0.04295	15	30	5

27.30193	0.04197	15	30	6
26.73913	1.65757	15	30	7
41.87440	0.06168	15	45	1
41.68841	0.99576	15	45	2
49.91546	0.09281	15	45	3
44.00000	0.88418	15	45	4
41.96135	0.06187	15	45	5
41.87440	0.06123	15	45	6
40.55556	2.42113	15	45	7
7.56763	0.02981	20	20	1
7.56763	0.60515	20	20	2
8.27536	0.07646	20	20	3
8.06039	0.56452	20	20	4
7.57246	0.03018	20	20	5
7.56763	0.02940	20	20	6
7.51449	1.51952	20	20	7
17.27053	0.05566	20	40	1
17.24879	1.17855	20	40	2
19.53140	0.09902	20	40	3
18.27778	1.04963	20	40	4
17.29227	0.05577	20	40	5
17.27053	0.05504	20	40	6
17.00483	2.86378	20	40	7
26.67874	0.08172	20	60	1
26.60628	1.75463	20	60	2
30.55314	0.12156	20	60	3
27.89372	1.53735	20	60	4
26.72464	0.08150	20	60	5
26.67874	0.08116	20	60	6
26.06280	4.20875	20	60	7
4.87923	0.03639	25	25	1
4.87923	0.92294	25	25	2
5.26087	0.09377	25	25	3
5.18116	0.84662	25	25	4
4.88164	0.03643	25	25	5
4.87923	0.03576	25	25	6
4.85266	2.29673	25	25	7
10.71981	0.06846	25	50	1
10.71256	1.81016	25	50	2
11.63043	0.12235	25	50	3
11.13285	1.59810	25	50	4
10.73188	0.06874	25	50	5
10.71981	0.06837	25	50	6
10.58937	4.38333	25	50	7
17.29952	0.10061	25	75	1
17.27053	2.69031	25	75	2
19.47101	0.15034	25	75	3
17.99034	2.34911	25	75	4
17.32609	0.10093	25	75	5
17.29952	0.09982	25	75	6
16.97585	6.46349	25	75	7
1.38164	0.07769	50	50	1
1.38164	3.62811	50	50	2
1.42271	0.19255	50	50	3
1.41787	3.17381	50	50	4
1.38164	0.07588	50	50	5
1.38164	0.07578	50	50	6

1.37923	9.39869	50	50	7
2.64734	0.15044	50	100	1
2.64734	7.14886	50	100	2
2.75604	0.25571	50	100	3
2.71739	6.13265	50	100	4
2.64734	0.14750	50	100	5
2.64734	0.14725	50	100	6
2.63043	18.26775	50	100	7
4.22947	0.22256	50	150	1
4.22705	10.77004	50	150	2
4.45411	0.31993	50	150	3
4.33575	9.18452	50	150	4
4.23188	0.21984	50	150	5
4.22947	0.21996	50	150	6
4.19082	27.16681	50	150	7
0.33816	0.18505	100	100	1
0.33816	15.56198	100	100	2
0.34541	0.40351	100	100	3
0.34541	12.86502	100	100	4
0.33816	0.17844	100	100	5
0.33816	0.17811	100	100	6
0.33816	42.71712	100	100	7
0.67874	0.36393	100	200	1
0.67874	30.97858	100	200	2
0.69324	0.55688	100	200	3
0.68841	25.37471	100	200	4
0.67874	0.35184	100	200	5
0.67874	0.35087	100	200	6
0.67633	83.73315	100	200	7
1.00483	0.54282	100	300	1
1.00483	46.43239	100	300	2
1.02657	0.71071	100	300	3
1.01932	37.91863	100	300	4
1.00483	0.52357	100	300	5
1.00483	0.52408	100	300	6
1.00000	125.03649	100	300	7

Poloměr: 1.00				
Šířka řešení	Čas	Dimenze	Parametrů	Metoda
97.45714	0.01080	5	5	1
96.51071	0.04987	5	5	2
127.05476	0.02350	5	5	3
114.15000	0.05644	5	5	4
97.95833	0.01121	5	5	5
97.45714	0.01001	5	5	6
92.30833	0.14567	5	5	7
188.40000	0.01726	5	10	1
182.00119	0.08886	5	10	2
247.88333	0.02937	5	10	3
200.05476	0.09078	5	10	4
190.58214	0.01820	5	10	5
188.40000	0.01684	5	10	6
167.30119	0.24539	5	10	7
425.13095	0.02356	5	15	1
403.60238	0.12727	5	15	2
545.11667	0.03480	5	15	3
446.19048	0.12435	5	15	4
430.95238	0.02451	5	15	5
425.13095	0.02326	5	15	6
352.68095	0.34151	5	15	7
32.39286	0.01709	10	10	1
32.31071	0.16568	10	10	2
39.15238	0.04161	10	10	3
36.11429	0.16762	10	10	4
32.46071	0.01725	10	10	5
32.39286	0.01661	10	10	6
31.46786	0.44927	10	10	7
64.94167	0.03002	10	20	1
64.13690	0.31462	10	20	2
81.40119	0.05268	10	20	3
69.25595	0.29556	10	20	4
65.30714	0.03042	10	20	5
64.94167	0.02950	10	20	6
60.66548	0.80308	10	20	7
97.95833	0.04277	10	30	1
96.08095	0.46418	10	30	2
122.05238	0.06390	10	30	3
101.36310	0.42698	10	30	4
98.90000	0.04391	10	30	5
97.95833	0.04259	10	30	6
89.11667	1.16869	10	30	7
13.80476	0.02333	15	15	1
13.79167	0.34688	15	15	2
15.85476	0.05824	15	15	3
15.18333	0.33316	15	15	4
13.82738	0.02339	15	15	5
13.80476	0.02269	15	15	6
13.53214	0.89369	15	15	7
28.15357	0.04266	15	30	1
28.02500	0.67461	15	30	2
32.84643	0.07544	15	30	3
29.87143	0.61289	15	30	4
28.22500	0.04270	15	30	5

28.15357	0.04212	15	30	6
27.00714	1.66521	15	30	7
49.06071	0.06081	15	45	1
48.60833	0.98982	15	45	2
57.68214	0.09120	15	45	3
51.01786	0.88078	15	45	4
49.27500	0.06132	15	45	5
49.06071	0.06041	15	45	6
46.02500	2.40203	15	45	7
7.60357	0.02961	20	20	1
7.60119	0.60282	20	20	2
8.21071	0.07610	20	20	3
8.03333	0.56217	20	20	4
7.61310	0.03024	20	20	5
7.60357	0.02914	20	20	6
7.49881	1.51475	20	20	7
16.21905	0.05591	20	40	1
16.18690	1.17908	20	40	2
17.92262	0.09910	20	40	3
16.93571	1.05058	20	40	4
16.25119	0.05581	20	40	5
16.21905	0.05531	20	40	6
15.76548	2.86418	20	40	7
26.47500	0.08214	20	60	1
26.32500	1.76026	20	60	2
30.58095	0.12209	20	60	3
27.63214	1.54135	20	60	4
26.53452	0.08141	20	60	5
26.47500	0.08108	20	60	6
25.24048	4.20608	20	60	7
5.18095	0.03625	25	25	1
5.17976	0.92186	25	25	2
5.66429	0.09349	25	25	3
5.56429	0.84895	25	25	4
5.18571	0.03655	25	25	5
5.18095	0.03571	25	25	6
5.12143	2.29805	25	25	7
10.04405	0.06825	25	50	1
10.02857	1.81202	25	50	2
11.00952	0.12186	25	50	3
10.48333	1.59906	25	50	4
10.06310	0.06913	25	50	5
10.04405	0.06765	25	50	6
9.80357	4.38567	25	50	7
16.79881	0.10094	25	75	1
16.75000	2.69861	25	75	2
18.88929	0.14989	25	75	3
17.61429	2.34832	25	75	4
16.83571	0.10060	25	75	5
16.79881	0.10033	25	75	6
16.20476	6.46095	25	75	7
1.23690	0.07734	50	50	1
1.23690	3.61394	50	50	2
1.28690	0.19140	50	50	3
1.27976	3.16708	50	50	4
1.23810	0.07589	50	50	5
1.23690	0.07538	50	50	6

1.23095	9.38879	50	50	7
2.79167	0.14961	50	100	1
2.79048	7.17552	50	100	2
2.90952	0.25619	50	100	3
2.86786	6.17260	50	100	4
2.79405	0.14726	50	100	5
2.79167	0.14753	50	100	6
2.75952	18.28862	50	100	7
4.10714	0.22212	50	150	1
4.10476	10.80017	50	150	2
4.31190	0.32033	50	150	3
4.19762	9.20403	50	150	4
4.11310	0.21975	50	150	5
4.10714	0.22012	50	150	6
4.03810	27.18785	50	150	7
0.33095	0.18514	100	100	1
0.33095	15.56971	100	100	2
0.33810	0.40479	100	100	3
0.33690	12.88013	100	100	4
0.33095	0.17836	100	100	5
0.33095	0.17798	100	100	6
0.32976	42.74400	100	100	7
0.66071	0.36416	100	200	1
0.66071	31.00495	100	200	2
0.67381	0.55706	100	200	3
0.67143	25.40002	100	200	4
0.66071	0.35105	100	200	5
0.66071	0.35104	100	200	6
0.65714	83.89235	100	200	7
1.00833	0.54354	100	300	1
1.00833	46.45477	100	300	2
1.03214	0.70862	100	300	3
1.02381	37.93020	100	300	4
1.00833	0.52406	100	300	5
1.00833	0.52482	100	300	6
1.00000	125.10540	100	300	7

Seznam použitých zkratek

HBR: Hans-Bliek-Rohn
BS: Bauer-Skeel