

Řešení systému lineárních algebraických rovnic metodou Monte Carlo

Tomáš Hrabec

Bakalářská práce
2005



Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

Děkuji vedoucímu mé bakalářské práce, Ing. Bronislavu Chramcovovi, za vedení, rady a cenné připomínky při zpracování této bakalářské práce.

ABSTRAKT

Abstrakt česky

Cílem bakalářské práce je řešení systému lineárních algebraických rovnic metodou Monte Carlo. Je zpracována literární rešerše na toto téma. Dále byl navržen experiment pro řešení algebraických rovnic. Na základě navrženého experimentu byl poté vytvořen program v programovém prostředí Matlab pro řešení systému lineárních algebraických rovnic metodou Monte Carlo. Program také zahrnuje řešení pomocí numerických metod. Tento program je koncipován pro využití při výuce předmětu Simulace systémů. Na závěr je provedeno srovnání přesnosti výsledků podle metody Monte Carlo a přesnosti výsledků podle numerických metod.

Abstrakt ve světovém jazyce

The Aim of the bachelor thesis is solution of the system of linear algebraic equations by Monte Carlo's method. At first will be processed literary recherche on this theme. As next will designed experiment for solution of algebraic equations. And on the basis designed experiment will be create program in program's environment Matlab for solution of the systém of system of linear algebraic equations by Monte Carlo's Metod include solution by the help of numerical methods. This program will be drawn for utilization at education of object Simulation of systems. And the last will be comparison of accuary of results by Monte Carlo's method and by numerical methods.

OBSAH

OBSAH	4
ÚVOD.....	5
I. TEORETICKÁ ČÁST	6
1 METODA MONTE CARLO	7
1.1 METODA MONTE CARLO – JEJÍ VZNIK A PODSTATA.....	7
1.1.1 Odkud se vzal název Metoda Monte Carlo	8
1.2 NUMERICKÁ MATEMATIKA A METODA MONTE CARLO.....	10
1.3 ŘEŠENÍ SOUSTAV LINEÁRNÍCH ROVNIC POMOCÍ METODY MONTE CARLO.....	13
2 ŘEŠENÍ SOUSTAV LINÁRNÍCH ROVNIC POMOCÍ NUMERICKÝCH METOD	17
2.1 PROSTÁ ITERAČNÍ METODA	17
2.2 SEIDELOVA ITERAČNÍ METODA.....	17
2.3 JACOBIOVA METODA	18
2.4 GAUSSOVA-SEIDELOVA METODA	19
II. PRAKTICKÁ ČÁST.....	20
3 POPIS SIMULAČNÍHO MODELU.....	21
3.1 POPIS UŽIVATELSKÉHO PROSTŘEDÍ.....	21
3.2 UKÁZKA PROGRAMU PŘI ŘEŠENÍ SYSTÉMU LINEÁRNÍCH ROVNIC	22
3.3 POPIS ŘEŠENÍ PRO METODU MONTE CARLO	24
4 STATISTICKÉ VYHODNOCENÍ VÝSLEDKŮ.....	26
4.1 STATISTICKÉ VYHODNOCENÍ PRO SOUSTAVU 2. ŘÁDU	26
4.2 STATISTICKÉ VYHODNOCENÍ PRO SOUSTAVU 5. ŘÁDU	30
4.3 VYHODNOCENÍ.....	32
4.3.1 Vyhodnocení pro soustavu 2. řádu.....	32
4.3.2 Vyhodnocení pro soustavu 5.řádu.....	34
ZÁVĚR.....	36
SEZNAM POUŽITÉ LITERATURY.....	37
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	38
SEZNAM OBRÁZKŮ	39
SEZNAM TABULEK.....	40
SEZNAM PŘÍLOH.....	41

ÚVOD

Mým úkolem je navrhnout experiment pro řešení systému lineárních algebraických rovnic metodou Monte Carlo. Dále sestavit program v prostředí MATLAB, který realizuje navržený experiment. Program bude také obsahovat řešení pomocí numerických metod. Cílem je přizpůsobit tento program tak, aby se posluchačům předmětu Simulace systémů dostalo názorné ukázky, jakým způsobem jak která metoda pracuje. Jaký vliv má na řešení počet iterací a řád soustavy. Tato bakalářská práce by měla dále popsat výhody či nevýhody použití metody Monte Carlo.

V této bakalářské práci budu testovat dvě soustavy rozdílného řádu. A to soustavu 2. řádu a soustavu 5. řádu. Na těchto dvou soustavách budu praktikovat jak numerické metody tak metodu Monte Carlo. U obou soustav budu měnit počet iterací a pozorovat přesnost dosažení správných hodnot kořenů. Jako přesnou hodnotu budu považovat hodnoty vypočítané pomocí analytické metody. U obou soustav provedu statistické vyhodnocení. Toto statistické vyhodnocení bude spočívat v porovnání kvadratické odchylky chyby. To znamená jak moc je metoda Monte Carlo přesná či nepřesná při konkrétním počtu iterací. A na základě tohoto vyhodnocení provedu závěr.

I. TEORETICKÁ ČÁST

1 METODA MONTE CARLO

1.1 Metoda Monte Carlo – její vznik a podstata

V poslední době můžeme ve vývoji matematiky jakož i v jejích konkrétních metodách zaznamenat vznikání velmi užitečných oblastí, které svým názvem mohou u nezasvěcence vzbudit představu něčeho nejen neužitečného, ale dokonce i nepotřebného. Ale ve všech případech, které známe, je právě opak pravdou. To zjistíme, seznámíme-li se s problematikou, kterou řeší, nebo jsme-li svou činností přivedeni k úlohám, na jejichž řešení tradiční úlohy nestačí. Jmenujme alespoň některé : Matematická teorie informace, Matematická teorie her, Metoda Monte Carlo. Tyto tři uvedené užitečné matematické teorie a metody vznikly z konkrétních požadavků na řešení složitých problémů z široké palety oborů přírodovědných, technických, ekonomických a dalších. Při této příležitosti ale nesmíme zapomenout na jednu neobvyklou skutečnost spočívající ve faktu, že vznikly na základě požadavků na řešení problémů i samotné matematiky. Další zajímavý aspekt spočívá v tom, že nevychází ve své podstatě z teorie pravděpodobnosti a matematické statistiky, které patří v současné době k jedněm z nejrychleji se rozvíjejících matematických disciplín. Samy tyto výchozí matematicky velmi náročné disciplíny manipulují s pojmy, které jsou jak na své filozofické hladině, tak na hladině matematického definování, velmi složité. Jde především o pojmy náhody, pravděpodobnosti a statistiky. A prostřednictvím těchto pojmů, které jak v minulosti, tak ještě dodnes vyvolávají u mnohých nedůvěru, můžeme dnes řešit problémy tradičními metodami neřešitelné. A kromě toho, jak se pokusíme doložit, umožňují některé tradiční pojmy, postupy řešení a výsledky interpretovat způsobem naprosto neočekávaným. Ale jak praví známý slogan : „všechno již bylo někdy objeveno“, i v našich případech je nutné opět připomenout skutečnost, že základní ideje byly vysloveny historicky daleko dříve, než se ukázala možnost a nutnost jejich seriózních využití. Uvedená metoda je především mistrovskou ukázkou schopnosti matematicky modelovat a poté simulovat jevy, procesy či výpočty, cestami, které s řešením původně zadaného problému zdánlivě žádnou souvislost nemají. Kromě toho jej uvedené postupy donutí zamyslet se nad interpretací získaných výsledků, nad interpretací odlišující se od vžitých představ. Dále se potom ukáže, že uvedená simulace „netradičních souvislostí“ metodou Monte Carlo je schopna v matematické formě dospět k výsledku rychleji, „takřka bezpracně“, oproti tradičnímu postupu. Podstata uvedeného pojmu „netradiční souvislost“ spočívá v tom, že metodami

teorie pravděpodobnosti a matematické statistiky, které tvoří matematické východisko k pochycení zákonitostí hromadných náhodných jevů a procesů, jsme schopni řešit problémy a úlohy, nemající s náhodou vůbec nic společného. A ještě jednu, ne nepodstatnou poznámku. Metody a přístupy k řešení, které jsou předmětem této práce, nabyly svůj současný význam v souvislosti s prudkým rozvojem výpočetní techniky. Již z toho, co bylo byt' krátce řečeno, vyplývá, že jsou to teorie pravděpodobnosti a matematická statistika, které tvoří matematický základ metody Monte Carlo. Abychom tedy metodu Monte Carlo pochopili a mohli ji účelně využívat, musíme uvést některé základní pojmy a matematický aparát těchto dvou disciplín.

1.1.1 Odkud se vzal název Metoda Monte Carlo

Vyslovíme-li slova Monte Carlo, u drtivé většiny lidí se vybaví představa hazardního hráčského doupěte v horším případě, resp. hazardních hráčů z vyšší společnosti v lepším případě. Pravdou je, že MONTE CARLO je název dnes po celém světě vyhlášeného městečka v Monackém knížectví, plného heren, ve kterých hlavním pojmem je ruleta jako nástroj výher na straně jedné a propadů na straně druhé. Metoda Monte Carlo má dnes již více jak padesátiletou historii. Byla formulována a současně i prakticky využita během druhé světové války předními vědeckými pracovníky Johnem von Neumannem a Stanislavem Ulamem ve Spojených státech amerických. Při výzkumu chování neutronů a možnosti jejich pronikání různými látkami se objevil důležitý problém, jak určit procento neutronů v určité spršce, která pronikne například nádrží vody určitých rozměrů. Tento problém nebylo možno řešit ani teoreticky ani prakticky, přestože si disponovalo takovými nezbytnými údaji jako: průměrná vzdálenost mezi dvěma srážkami neutronu s atomy uhlíku nebo kyslíku, množství energie, kterou ztrácí neutron během srážky, atd. Přes znalost těchto informací nebylo dlouho možné najít praktickou metodu pro předpověď úplného dalšího chování jednotlivého neutronu, jinak řečeno pro předpověď úplného dalšího chování jednotlivého neutronu, jinak řečeno pro předpověď celé další „historie jeho života“. Neumann a Ulam vyřešili problém geniálním nápadem využít k modelování pro předpovědi „historie života neutronu“ odedávna známou techniku kola rulety. Odtud také pochází název, který autoři metodě výstižně dali: METODA MONTE CARLO. Pomocí této metody bylo možné předpovědět trajektorii každého neutronu daného svazku. Simulace pohybu neutronů „putujících“ ve vodě a náhodně se srážejících s atomy vodíku a kyslíku vypadala následovně: je například známo, že náhodný jev – neutron při srážce

s atomem vodíku bude atomem pohlcen – nastane průměrně v jednom ze sta možných případů. Pro vytyčení trasy pohybu neutronu se roztočí kolo rulety rozdělené na sto dílků, z nichž pouze jeden je odlišně vyznačený a označuje „pohlcení neutronu atomem vodíku“. Jestliže se kolo rulety zastaví právě na tomto dílku, označuje se to „konec života“ neutronu. V opačné případě se zjistí pomocí jiného kola rulety směr a rychlost neutronu po srážce. Potom pomocí dalšího kola rulety se rovněž náhodně určí, jakou trajektorii proběhne neutron, než nastane další srážka buď s atomem vodíku nebo kyslíku, atd. Simulace „historie života“ neutronu se provádí tak dlouho, dokud buď není pohlcen, nebo dokud neztratí tolik energie, že jeho konečné vylétnutí z nádrže nás přestane zajímat. Přesto, že je takto celý problém formulován již ve značně zjednodušené formě, je celkem jasně zřetelný vysoký stupeň komplikovanosti celého problému. Když provedeme velký počet takových simulací, získáme dříve či později více méně přesnou informaci o procentu neutronů, kterým se podařilo uniknout z nádrže. Stupeň spolehlivosti a přesnosti této metody je určen matematicko-statistickými metodami teorie odhadu, o nichž dále pojednám. Již předem ale můžeme říci, že uvedené parametry hodnotící kvalitu výsledku budou odvislé od počtu provedených pokusů. Kdybychom chtěli skutečně simulovat cestu svazku, složeného např. ze sta tisíc neutronů, přičemž trajektorie každého neutronu by vedla přes stovky srážek s atomy vodíku a kyslíku, potom řešení uvedeného problému užitím kol rulety by bylo samozřejmě prakticky neproveditelné. Realizace by vyžadovala mnoho let, strávených pouze uváděním rulety v chod a zaznamenáváním náhodně získaných výsledků. Na druhé straně ale víme, že případná realizace takového experimentu by skutečně k hledaným výsledkům dospěla. A v tom spočívá genialita nápadu autorů metody. Shodou historické souběžnosti mohla být právě popsaná simulace již v té době reálně provedena samočinným počítačem majícím k dispozici vhodný soupis náhodných čísel pro každý z možných případů, které mohou potkat neutron na jeho trajektorii. V odvislosti od rychlosti současných počítačů je čas potřebný k získání spolehlivých a přesných výsledků velmi krátký. Postupem času se ukázalo, především díky teorie modelování, že podobným způsobem lze řešit složité problémy technické, ekonomické, problémy z oblasti činnosti telefonních centrál, řízení dopravy, hromadné obsluhy, kontroly stavu zásob a další. Jedno z největších překvapení se však zrodilo, když se ukázalo, že tímto způsobem lze řešit i některé problémy v samotné matematice, a to velmi efektivně, vezmeme-li v úvahu výkonnost současné výpočetní techniky. Z uvedeného rovněž vyplývá, že užitečnost a efektivnost metody Monte Carlo je podmíněna výkonností

výpočetní techniky, kterou máme k dispozici. Předpokládejme, že neutrony jsou jednotliví lidé, nádrž vody, ve které neutrony „putují“ je život oněch lidí. Atomy kyslíku a vodíku, se kterými se neutron sráží, reprezentují události: narození, uzavření manželství, rozvod, úmrtí a mnoho dalších. Pro zjednodušení se vyšlo z předpokladu, že základní jednotkou času žití člověka je jeden měsíc. Jednotlivé znaky pozorované na jednotkách výběru jako: věk, pohlaví, zaměstnání, počet dětí v rodině, atd. byly vybrány shodně s jejich rozdělením v celé populaci, k čemuž bylo nutné provést odpovídající statistické šetření. Zjištěné informace byly přeneseny na kola rulet a simulace se mohla realizovat. Tak vznikl první demografický model USA jako výsledek simulace metodu Monte Carlo. Zahrnoval období desíti let, počínaje dubnem roku 1950. Jeden z dalších závěrů, který je možný již nyní učinit, spočívá ve zjištění, vyplývajícím z využití modelu rulety, že metodu Monte Carlo je možné aplikovat všude tam, kde je řešení problémů závislé určitým způsobem na pravděpodobnostech, přičemž jde o řešení problémů, pro které je vypracování explicitního algoritmu řešení obtížné, ne-li vůbec nemožné.

1.2 Numerická matematika a metoda Monte Carlo

Stále větší nutnost aplikací nejrůznějších matematických metod v přírodovědných oborech, medicíně, technice, ekonomice a jinde nutí matematiku přicházet s prostředky, které v té které fázi vývoje zajistí největší upotřebitelnost. Zatímco z historického hlediska se donedávna matematika soustřeďovala především na řešení vlastních teoretických problémů, pak se v poslední době – zejména s objevením a rychlým rozšiřováním kvalitních počítačů – obrací pozornost stále více i na praktické využití těchto získaných výsledků. A je to právě výpočetní technika (numerická matematika), která reprezentuje spolu s programováním na počítačích mohutný prostředek pro praktické provádění numerických výpočtů. V hlavních rysech základní cesty současného vývoje numerické matematiky charakterizovat následujícími vzájemně souvisejícími skutečnostmi:

1. neodmyslitelné používání počítačů s vysokou a stále narůstající produktivitou;
2. realizace vyčíslovacích algoritmů s velkým počtem aritmetických operací;
3. rozpracování nových tříd algoritmů s ohledem k jejich možnosti realizace na počítačích.

Obvyklá tradiční cesta při řešení nějaké úlohy spočívá v tom, že se udá algoritmus, tj. posloupnost určitých operací, pomocí kterého se určí hledaná veličina θ buď přesně, nebo s předepsanou přesností. Označíme-li

$$\theta_1, \theta_2, \dots, \theta_n, \dots$$

odpovídající výsledky postupně nahromaděných početních procedur, potom požadujeme, aby

$$\lim_{n \rightarrow \infty} \theta_n = \theta$$

Přitom v případě konečného počtu operací se proces zastaví na nějakém kroku. V tomto případě je vyčíslovací proces přísně determinován: dva různí výpočtáři, odmyslíme-li chybu při výpočtu, přijdou při realizaci stejného algoritmu ke stejnému výsledku. Při výpočtové práci se matematika vždy střetávala s narůstající složitostí řešené problematiky a stále více se střetávala s narůstajícím řešením úloh, kdy zkonstruování určitého algoritmu je prakticky neproveditelné, resp. konec konců i sestrojený algoritmus je složitý a pro využití neefektivní. V takových případech se ukazuje jednoznačně výhodné zamyslet se nad možností obrátit pozornost k modelování matematické podmínky úlohy (jak tomu bylo v případě fyzikální problematiky) a využít k tomu pravděpodobnostních a matematicko-statistických postupů. Tato myšlenka se ukázala velmi plodná, a protože simulace metodou Monte Carlo ve fyzice měla za sebou řadu úspěšných řešení, byla obrácena pozornost právě na tuto metodu. Úspěch se zákonitě dostavil. Základní myšlenka metody Monte Carlo, jinak také metody statistických pokusů, v tomto případě spočívá na spojitosti a vztahu mezi pravděpodobnostními charakteristikami různých náhodných procesů (např. pravděpodobnostmi náhodných jevů, středními hodnotami náhodných veličin apod.) a veličinami, které jsou řešeními úloh z různých matematických oblastí (např. matematické analýzy). Ve smyslu této aplikace se tedy rozumí pod pojmem Monte Carlo všechny postupy numerického řešení matematických, fyzikálních a jiných problémů, realizované pomocí mnohokrát opakovaných náhodných pokusů. Odhady hledané veličiny se získávají statistickou cestou a mají tedy pravděpodobnostní charakter. Označíme v tom případě odhady

$$\theta_1, \theta_2, \dots, \theta_n, \dots$$

za hledanou hodnotu veličiny θ , které získáme statistickým zpracováním experimentálního materiálu, který jsme obdrželi jako výsledek nějakých mnohokrát

opakovaných náhodných pokusů. Požadujeme, aby v tomto případě veličina θ_n , kde n značí počet pokusů, která je tzv. veličinou náhodnou, při $n \rightarrow \infty$ konvergovala k hledané hodnotě θ v pravděpodobnosti. Tím rozumíme splnění vztahu, aby pro libovolně malé $\varepsilon > 0$ platilo

$$\lim_{n \rightarrow \infty} P(|\theta_n - \theta| < \varepsilon) = 1$$

Výběr odhadu θ_n je podmíněn typem a zvláštnostmi specifikujícími řešenou úlohu. Hledaná hodnota θ v rámci uvedeného jevu, jako střední hodnota nějaké náhodné veličiny atd., prostě jako nějaký statistický parametr. Veličiny θ_n mají charakter statistických odhadů a souvisejí s hledanou hodnotou θ prostřednictvím pravděpodobnostních zákonitostí. Můžeme tedy říci, že při této interpretaci je výpočetní proces nedeterministický, jelikož je určován výsledky náhodných pokusů. Spojitost mezi některými náhodnými procesy a řešením odpovídajících diferenciálních rovnic je známa již od konce minulého století. Na vztah mezi procesem náhodného blouzení (např. částice v kapalině), upozornili (1928) ve svých pracích fyzikové Albert Einstein a Marian Smoluchovský. Klasický přístup byl charakterizován tím, že zkoumání náhodného procesu se pokládalo za vyčerpávající – samozřejmě v jistém smyslu, jestliže se určení charakteristik toho kterého procesu převedlo na řešení analytické úlohy. Přitom řešení analytické úlohy, nehledě na stupeň složitosti, tvořilo další nezávislou etapu zkoumání. Avšak teprve v roce 1949 bylo patrně poprvé poukázáno na okolnost nás především zajímavícím, že by uvedená klasická úloha mohla být obrácena. Místo řešení analytické úlohy je možné modelovat náhodný proces a využít statistických odhadů pravděpodobností, středních hodnot odpovídajících statistických charakteristik. Je tedy v rámci tohoto přístupu nutné vždy sestavit takový pravděpodobnostní model, jehož parametry reprezentují řešení formulované úlohy. Prakticky jsou zajímavé samozřejmě jen ty pravděpodobnostní modely, které připouštějí relativně jednoduchou realizaci a jsou proveditelné na soudobých počítačích, přičemž dovolují získat odhady neznámých parametrů (hodnoty výsledků úloh) s pokud co možno nejmenší chybou (měřenou např. rozptylem). I v tomto případě platí již shora uvedená poznámka, že efektivní využití metody Monte Carlo úzce souvisí s rozvojem výpočetní techniky. Ve prospěch metody Monte Carlo mluví i to, že není nutné znát vnitřní vztahy mezi výsledky experimentů a hledanými řešeními úloh a postačí pouze vyjasnit komplex podmínek, při jehož realizaci má místo vnější souvislost. Tato okolnost poskytuje možnost využít metodu Monte Carlo

pro řešení úloh nejrozmanitějšího typu, včetně úloh logického charakteru. Z matematických problémů, pro které je užití metody Monte Carlo již rozpracováno:

- a. výpočet určitých integrálů včetně násobných;
- b. řešení systémů lineárních rovnic;
- c. hledání kořenů rovnic, výpočet hodnoty funkce.

Další důležitá okolnost svědčící ve prospěch rozsáhlé aplikability metody Monte Carlo je v tom, že k řešení úloh naprosto různé podstaty je možné užít jednoho pravděpodobnostního modelu a naopak pro řešení jedné konkrétní úlohy je možné v rámci metody Monte Carlo vytvořit řadu různých simulačních schémat. Na závěr předchozích úvah podtrhneme následující skutečnost. Úspěch aplikace metody Monte Carlo závisí kromě jiného na úspěšném uspořádání náhodného pokusu v rámci simulace. To ale zase úzce souvisí se systémem náhodných čísel, pomocí nichž se výpočet provádí. Na jeho kvalitě závisí i kvalita úlohy. Je možné konstatovat, že úspěch celého výpočtu metodou Monte Carlo na počítači je určen v podstatě třemi základními faktory:

1. kvalitou generátoru náhodných čísel, resp. pseudonáhodných čísel;
2. výběrem racionálního algoritmu výpočtu;
3. kontrolou přesnosti získaného výsledku.

1.3 Řešení soustav lineárních rovnic pomocí metody Monte Carlo

Soustavu lineárních rovnic zapíšeme v maticovém tvaru

$$Ax = b, \tag{1}$$

Kde $A = [a_{ij}]$ pro $i, j = 1, 2, \dots, N$,

b – sloupcový vektor pravých stran.

Soustavu (1) upravme na tvar

$$x = a + Hx. \tag{2}$$

Transformaci soustavy (1) do tvaru (2), jenž je používán v iteračních postupech řešení soustavy lineárních rovnic, můžeme jednoduše provést tak, že matici A vyjádříme jako rozdíl

$$A = V - W,$$

Kde matici V volíme tak, abychom matici V^{-1} snadno našli. Položíme-li

$$H = V^{-1}W, \quad a = V^{-1}b,$$

dojdeme k tvaru

$$x = a + Hx.$$

Podobně jako u iteračních metod řešení soustav lineárních rovnic požadujeme, aby pro normu matice H platilo $\|H\| < 1$, kde norma matice H je např.

$$\|H\| = \max_i \left(\sum_j |h_{ij}| \right).$$

Pro začátek předpokládejme, že prvky matice H splňují podmínky:

$$1. \quad h_{ij} \geq 0 \quad \text{pro } i = 1, 2, \dots, N, \text{ a } j = 1, 2, \dots, N,$$

$$2. \quad \sum_{j=1}^N h_{ij} < 1 \quad \text{pro } i = 1, 2, \dots, N.$$

Pokud existují řešení soustavy (2), lze je napsat ve tvaru

$$x = [J - H]^{-1}a,$$

matici $[J - H]^{-1}$ můžeme za uvedených předpokladů vyjádřit ve formě řady

$$[J - H]^{-1} = J + H + H^2 + \dots = \sum_{k=0}^{\infty} H^k$$

takže $x = a + Ha + H^2a + H^3a + \dots$. Speciálně pro i -tou složku řešení platí

$$x_i = a_i + \sum_j h_{ij}a_j + \sum_k \sum_j h_{ik}h_{kj}a_j + \sum_m \sum_k \sum_j h_{im}h_{mk}h_{kj}a_j + \dots \quad (3)$$

Matici $[J-H]^{-1}$ můžeme za uvedených předpokladů považovat za fundamentální matici absorpčního Markovova řetězce. Doplňme matici H podmíněných pravděpodobností přechodu P tak, že přidáme jeden absorpční stav. Matice P má potom tvar

$$P = \left[\begin{array}{c|c} H & r \\ \hline 0 & 1 \end{array} \right], \quad (4)$$

Kde $0 = (0, 0, \dots, 0)$ a r je sloupcový vector se složkami

$$r_i = 1 - \sum_{j=1}^N h_{ij} \quad \text{pro } i = 1, 2, \dots, N.$$

Vytvořili jsme tedy Markovův řetězec s $N + 1$ stavy, ve kterém poslední stav je absorpční, neboť pravděpodobnost setrvání v tomto stavu je jednotková. Generujme nyní náhodné posloupnosti stavů Markovova řetězce s maticí podmíněných pravdě pravděpodobností přechodu (4) za předpokladu, že vychází z i -tého stavu pro $i = 1, 2, \dots, N$. Dostaneme posloupnosti stavů typu

$$\alpha_i = [i, j_1, j_2, \dots, j_k, N + 1],$$

kde $j_1, j_2, \dots, j_k \in \{1, 2, \dots, N\}$. Všechny posloupnosti α^i končí číslem absorpčního stavu. Posloupnostem α^i přiřadíme hodnoty náhodné veličiny $X(\alpha^i)$ předpisem

$$X(\alpha^i) = \frac{a_{jk}}{r_{jk}}. \quad (5)$$

Veličina X nabývá tudíž hodnot

$$\frac{a_1}{r_1}, \frac{a_2}{r_2}, \dots, \frac{a_N}{r_N}, \quad (6)$$

které závisí na čísle posledního stavu posloupnosti α^i před absorpcí. Pro střední hodnotu diskrétní náhodné veličiny $X(\alpha^i)$ platí

$$E[X(\alpha^i)] = \sum_{j=1}^N \frac{a_j}{r_j} p_j, \quad (7)$$

kde p_j jsou pravděpodobnosti, s nimiž velična X nabývá hodnot (6), tj. pravděpodobnosti, že v stavové posloupnosti v i -tém stavu, bude předposledním prvkem j -tý stav. Pravděpodobnosti p_j můžeme snadno určit následující úvahou. Pravděpodobnost, že systém přejde v k krocích ze stavu i do stavu j a v bezprostředně následujícím kroku do absorpčního stavu, je dána součinem

$$h_{ij}^k r_j \quad \text{pro } k = 1, 2, \dots, \quad (8)$$

kde h_{ij}^k je prvek matice H^k . Pravděpodobnosti p_j dostaneme sečtením součinů (8) přes všechna k , neboli

$$p_j = \sum_{k=0}^{\infty} h_{ij}^k r_j,$$

Dosazením do vzorce (7) dostaneme

$$E[X(\alpha^i)] = \sum_{j=1}^N \frac{a_j}{r_j} \sum_{k=0}^{\infty} h_{ij}^k r_j = \sum_{k=0}^{\infty} \sum_{j=1}^N h_{ij}^k a_j.$$

Srovnáme-li poslední výraz se vztahem (3), vidíme, že

$$x_i = E[X(\alpha^i)], \quad (9)$$

neboť

$$h_{ij}^0 = \delta_{ij}, \quad \text{kde } \delta_{ij} = 0 \text{ pro } i \neq j \text{ a}$$

$$\delta_{ij} = 1 \text{ pro } i = j$$

$$h_{ij}^1 = h_{ij}$$

$$h_{ij}^2 = \sum_k h_{ik} h_{kj},$$

$$h_{ij}^3 = \sum_m \sum_k h_{im} h_{mk} h_{kj}$$

atd.

Rovnost (9) přímo udává návod, jak přibližnou hodnotu i -té složky vektoru řešení soustavy lineárních rovnic (2) za uvedených předpokladů o matici H :

Generujeme náhodné posloupnosti stavů specifikovaného Markovova řetězce, které končí absorpčním stavem. Na základě dostatečně velkého počtu posloupností vycházejícího z i -tého stavu vypočítáme průměr hodnot veličiny X , přiřazených jednotlivým posloupnostem vztahem (5). Průměr je nestranným odhadem hodnoty $E[X(\alpha^i)]$, a tedy přibližnou hodnotou x_i . Pro odhad velikosti chyby je výhodné zároveň počítat výběrové rozptyly hodnot veličin $X(\alpha^i)$. Při stanovení intervalu spolehlivosti pro $E[X(\alpha^i)]$ můžeme použít postupů známých matematické statistiky.

2 ŘEŠENÍ SOUSTAV LINÁRNÍCH ROVNIC POMOCÍ NUMERICKÝCH METOD

V této práci budou využity k dosažení výsledků kromě metody Monte Carlo také numerické metody. A budou to tyto iterační metody. Prostá iterační metoda, Seidlova iterační metoda, Jacobiova metoda a Gauss-Seidelova. Iterační metody mají význam hlavně při řešení rozsáhlých soustav. Při řešení iteračními metody se vytváří posloupnosti postupných aproximací, které konvergují k přesnému řešení.

2.1 Prostá iterační metoda

Při prosté iterační metodě převádíme soustavu $Ax=b$ na ekvivalentní soustavu tvaru $x=Ux+v$ a vytvoříme posloupnost postupných aproximací podle vzorce

$$x^{(k+1)} = Ux^{(k)} + v \quad k=0,1,\dots$$

kde $x^{(0)}$ je libovolná počáteční aproximace. Pokud tato posloupnost konverguje, hovoříme o konvergentním iteračním procesu.

Nutná a postačující podmínka konvergence iteračního procesu: Iterační proces konverguje nezávisle na volbě počáteční aproximace $x^{(0)}$ tehdy, když spektrální poloměr matice

$$\rho(U) < 1$$

to znamená, pokud některá z norem matice U je menší než 1 - $\|U\| < 1$.

Pro praktické použití píšeme ve tvaru

$$x_i^{(k+1)} = v_i + \sum_{j=1}^n u_{ij} x_j^{(k)} \quad (10)$$

kde $i=1,\dots, n \quad k=0,1, 2,\dots$

2.2 Seidelova iterační metoda

Na rozdíl od prosté iterační metody při výpočtu i -tého vektoru $(k+1)$ aproximace dosazujeme do pravé strany už vypočítané souřadnice $x_1^{(k+1)}, \dots, x_{j-1}^{(k+1)}$.

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} u_{ij} x_j^{(k+1)} + \sum_{j=i}^n u_{ij} x_j^{(k)} + v_i \quad (11)$$

kde $i=1, \dots, n$ $k=0, 1, 2, \dots$

Nutná a postačující podmínka konvergence iteračního procesu je stejná jako u předchozí Prosté iterační metody.

2.3 Jacobiova metoda

Jak pro Jacobiovu tak pro Gauss-Seidelovu metodu se vychází ze soustavy $Ax=b$, přičemž všechny diagonální prvky matice A jsou nenulové (dosáhneme vhodné úpravou). Potom pro souřadnice vektoru řešení platí :

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

kde $i=1, \dots, n$

Soustava $Ax=b$ bude tedy upravené na ekvivalentní soustavu

$$x = U_2 x + v_2$$

kde

$$U_2 = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \dots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & \dots & \dots & \dots \\ \dots & \dots & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ -\frac{a_{n1}}{a_{nn}} & \dots & \dots & \dots & 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \dots \\ \frac{b_n}{a_{nn}} \end{pmatrix}$$

Pro $(k+1)$ aproximaci platí

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right) \quad (12)$$

Nutná a postačující podmínka pro konvergenci iteračního procesu: Je-li spektrální poloměr matice $\rho(U_2 < 1)$, potom metoda konverguje nezávisle na volbě x^0 .

Postačující podmínka: Pokud matice A soustavy je ostře diagonální dominantní, pak

Jacobiův iterační proces pro soustavu $x^{(k+1)} = U_2 x^{(k)} + v_2$ konverguje nezávisle na x^0 .

2.4 Gaussova-Seidelova metoda

Nutná a postačující podmínka pro konvergenci je stejná jako u Jacobiovy metody.

Při výpočtu $x_i^{(k+1)}$ hodnoty používáme již vypočítané hodnoty $x_1^{(k+1)} \dots x_{j-1}^{(k+1)}$ a výpočet postupných aproximací počítáme podle vzorce:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} - \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^k \right) \quad (13)$$

kde $i=1, \dots, n$.

II. PRAKTICKÁ ČÁST

3 POPIS SIMULAČNÍHO MODELU

3.1 Popis uživatelského prostředí

Po spuštění programu se uživateli zobrazí následující uživatelské prostředí:

The screenshot shows a window titled 'monte_carlo'. At the top, it asks 'Zadejte počet iterací :' with an input field. Below this is a purple instruction: 'Soustava je dána tvarem $Ax = b$. Prvky matice A zadávejte po řádcích, řádky oddělené středníkem. Prvky vektoru B zadávejte oddělené středníkem.' Below this are two input fields: 'Zadejte matici A :' and 'Zadejte vektor b :'. To the right of these fields is a button labeled 'Výpočet'. Below the input fields is a purple message: 'Byla zadána soustava ve tvaru :'. At the bottom, there is a section titled 'Výsledky :' with five rows, each containing a method name and an input field: 'Metode Monte Carlo', 'Prostá iterační metoda', 'Seidelova iterační metoda', 'Jacobiova metoda', and 'Gaussova-Seidelova metoda'. The last row is labeled 'Analyticky'.

Obr. 1. Uživatelské prostředí po spuštění

Jako první se zadává **počet iterací**. To znamená, kolik zpřesnění má daný simulační model provést pro metody Prostá iterační, Seidelova iterační, Jacobiova a Gauss-Seidelova. Pro metodu Monte Carlo tento údaj znamená kolik experimentů má daný systém provést. Čím větší počet iterací se zadá, tím delší bude výpočetní doba. Také hodně záleží na výkonnosti počítače, na kterém simulaci provádíme. Pod počtem iterací je **informativní věta**, která udává, jakým způsobem se má soustava zadávat. Soustava je ve tvaru $Ax=b$. Jako první se zadávají prvky matice A, zadávají se po řádcích, řádky oddělené středníkem. A prvky vektoru b se zadávají oddělené středníkem. Vedle oblasti pro zadávání soustavy je tlačítko **Výpočet**, kterým se spustí samotný program.

Další položkou je **pole**, ve kterém se vypíše daná soustava v řádkovém tvaru. Např. soustava zadaná tvarem matice A - „9 -2;-2 13“ a vektorem b – „7;11“ se do tohoto pole

vypíše jako „ $+9 x_1 - 2 x_2 = 7 \parallel -2 x_1 + 13 x_2 = 11$ “. Pod tímto polem se nachází **oblast výsledků**, ve které se vypisují výsledky jednotlivých metod.

3.2 Ukázka programu při řešení systému lineárních rovnic

Jako první se zadá údaj iterací. V druhém kroku se zadá matice A a vektor b ve správném tvaru. Hledání kořenů se spustí tlačítkem **Výpočet** a program vypočítá dle jednotlivých metod kořeny soustavy a vypíše je do příslušného pole. Viz Obr. 2.

monte_carlo

Zadejte počet iterací :

Soustava je dána tvarem $Ax = b$. Prvky matice A zadávejte po řádcích, řádky oddělené středníkem. Prvky vektoru B zadávejte oddělené středníkem.

Zadejte matici A :

Zadejte vektor b :

Výpočet

Byla zadána soustava ve tvaru :

$+10 x_1 + 3 x_2 - 2 x_3 - 1 x_4 + 1 x_5 = 0.1 \parallel +2 x_1 + 15 x_2 + 1 x_3 + 2 x_4 + 2 x_5 = 3.2 \parallel +2 x_1 - 5 x_2 + 13 x_3 + 3 x_4 + 1 x_5 = 5 \parallel +2 x_1 + 1 x_2 - 3 x_3 + 20 x_4 + 3 x_5 = 2 \parallel +1 x_1 + 2 x_2 - 3 x_3 + 1 x_4 + 17 x_5 = 5 \parallel$

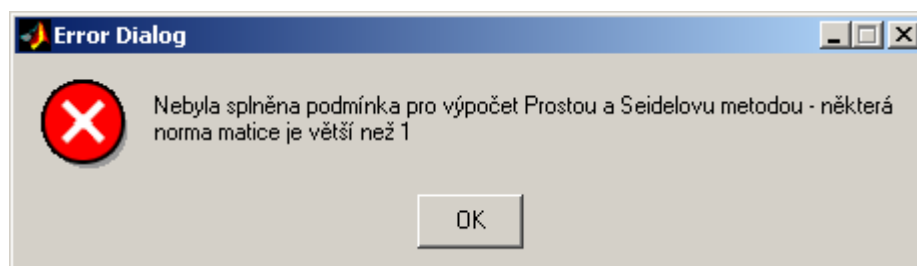
Výsledky :

Metode Monte Carlo	0.067925	0.21615	0.41786	0.11143	0.33131
Prostá iterační metoda	0.024046	0.12647	0.38097	0.097506	0.3393
Seidelova iterační metoda	0.024066	0.12648	0.38096	0.097516	0.33931
Jacobiova metoda	0.024064	0.12648	0.38095	0.097512	0.33931
Gaussova-Seidelova metoda	0.024066	0.12648	0.38096	0.097516	0.33931
Analyticky	0.024066	0.12648	0.38096	0.097516	0.33931

Obr. 2. Ukázka výpočtu kořenů

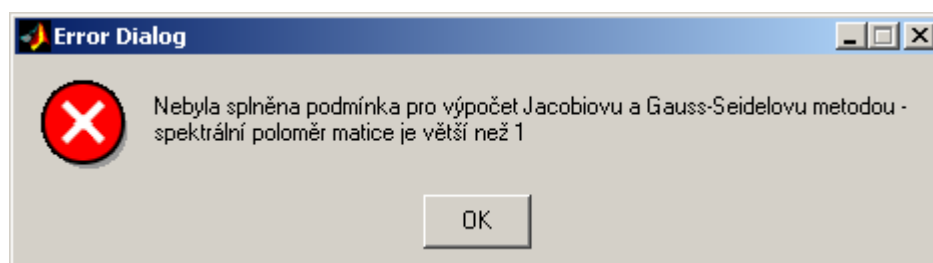
Při provádění programu je nejprve soustava upravena na tvar, který vyhovuje kritériím, při kterých je výpočet korektní. Těmito kritérii je buď řádková nebo sloupcová norma matice a spektrální poloměr matice. Program je ošetřen a pokud není splněna některá podmínka, uživatele informuje pomocí chybových hlášek.

Pokud není splněna řádková nebo sloupcová norma matice pro metodu Prostá iterační a Seidelova iterační, to znamená pokud je některá z norem větší než jedna, program na to upozorní chybovou hláškou:



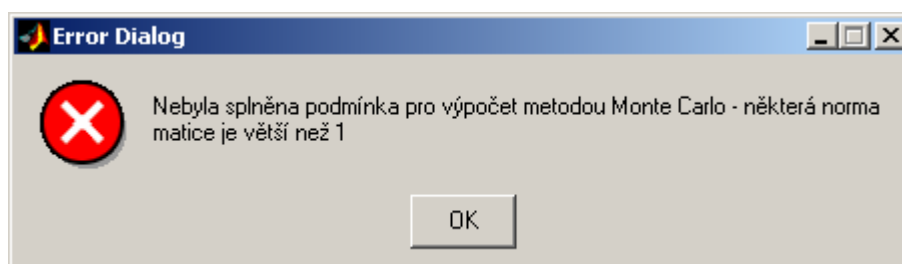
Obr. 3. Chybová hláška pro Prostou a Seidelovu metodu

Pokud není splněna podmínka pro Jacobiovu a Gauss-Seidelovu iterační metodu, což znamená že spektrální poloměr matice je větší než jedna, program uživatele opět upozorní chybovou hláškou:



Obr. 4. Chybová hláška pro Jacobiovu a Gauss-Seidelovu metodu

Podmínka pro metodu Monte Carlo je velikost kterékoliv z norem matice menší než jedna. Tato podmínka je stejná jako u metod Prostá iterační a Seidelova iterační. Při nesplnění podmínky program vypíše varovnou hlášku:



Obr. 5. Chybová hláška pro metodu Monte Carlo

Všechny výsledky se automaticky ukládají do souboru **Výsledky.txt**. Do tohoto souboru je vypsán řád soustavy a jednotlivé kořeny.

Užsledky pro soustavu 2. řádu

1. kořen

1	Prosta	1.0313	Seidelova	1.0313	Jacobiova	0.9924	Gauss-Seidel	0.9924	Monte Carlo	0.1750	Analytický	1.0000
2	Prosta	0.9961	Seidelova	0.9961	Jacobiova	0.9997	Gauss-Seidel	0.9997	Monte Carlo	0.2625	Analytický	1.0000
3	Prosta	1.0003	Seidelova	1.0003	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.3500	Analytický	1.0000
4	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.4375	Analytický	1.0000
5	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.6167	Analytický	1.0000
6	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.7042	Analytický	1.0000
7	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.8792	Analytický	1.0000
8	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.9667	Analytický	1.0000
9	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	1.0542	Analytický	1.0000
10	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	1.1417	Analytický	1.0000

2. kořen

1	Prosta	1.0000	Seidelova	1.0000	Jacobiova	0.9988	Gauss-Seidel	0.9988	Monte Carlo	0.1833	Analytický	1.0000
2	Prosta	0.9993	Seidelova	0.9993	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.2750	Analytický	1.0000
3	Prosta	1.0001	Seidelova	1.0001	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.3667	Analytický	1.0000
4	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.4583	Analytický	1.0000
5	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.5500	Analytický	1.0000
6	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.7333	Analytický	1.0000
7	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.8250	Analytický	1.0000
8	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	0.9167	Analytický	1.0000
9	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	1.0083	Analytický	1.0000
10	Prosta	1.0000	Seidelova	1.0000	Jacobiova	1.0000	Gauss-Seidel	1.0000	Monte Carlo	1.1000	Analytický	1.0000

Obr. 6. Výpis souboru Výsledky.txt

3.3 Popis řešení pro metodu Monte Carlo

V prvním kroku se musí soustava $Ax=b$ upravit na tvar odpovídající kritériím pro výpočet – normy matice, což je soustava $x=a+Hx$. Od diagonálních prvků se odečte jednička a tento zbytek se převede na levou stranu. V kroku druhém se celá soustava podělí levou stranou. Tímto získáme soustavu $x=a+Hx$, která splňuje, že norma matice nebude větší než jedna a také to znamená, že sumace prvků jednotlivých řádků nebude větší než jedna, což je taky jedna z podmínek pro výpočet metody Monte Carlo. V dalším kroku se zjistí tzv. „váhy“ soustavy, které jsou potřebné pro výpočet kořenů. „Váha“ soustavy nabývá dvou stavů: ± 1 . Zjištění vah soustavy provádí cyklus, který prochází upravenou matici Hx . Pokud je konkrétní prvek menší než nula, váha se nastaví na -1, pokud je větší než nula, váha se nastaví na +1. V dalším kroku se sestaví matice pravděpodobností, kde elementy jednotlivých řádků odpovídají pravděpodobnosti přechodů p_{ij} . V dalším kroku se musí určit přechody, dle kterých se budou hodnotit stavy pro generovaná čísla. Zjištěné náhodné číslo x potom implikuje přechody stavů. Pak následuje vlastní výpočet. Generuje se náhodné číslo, které je menší jedná, rovnoměrně rozložené na intervalu $\langle 0;1 \rangle$ pořízené generátorem pseudonáhodných čísel. Hned v prvním kroku je testováno, zda generované číslo již neskončilo „bloudění“, tzn. že překročilo stav S_{n+1} , jenž je nazýván „pohlující stěnou“. Odtud vyplývá, že proces „bloudění“ částice se zastavuje. Pokud ne, generuje se

další náhodné číslo, dokud není dosaženo stavu S_{n+1} . Odhad kořene je roven hodnotě střední hodnoty ξ_i .

$$x_i = E[\xi(T_i)] \sim \frac{1}{N} \sum_{k=1}^N X(T_i^k)$$

kde ξ_i je náhodná závislá na náhodných trajektoriích T_i .

4 STATISTICKÉ VYHODNOCENÍ VÝSLEDKŮ

Testování proběhlo pro 2 soustavy a to soustavu 2. řádu a soustavu 5. řádu. Soustavy byly testovány pro různý počet experimentů. Pokud byl například zadán počet iterací 20 znamená to, že 20krát musela částice překročit stav S_{n+1} . A pro každý z těchto experimentů bylo měření opakováno 20krát.

4.1 Statistické vyhodnocení pro soustavu 2. řádu

Danou soustavu jsem testoval pro 8 experimentů s počtem iterací - 2, 5, 10, 20, 50, 100, 500 a 800. Každý experiment byl 20krát opakován. A z těchto 20 pokusů jsem udělal průměrnou hodnotu a směrodatnou odchylku. Výsledky jsou přehledně zobrazeny v následujících tabulkách. Testovaná soustava byla zadána ve tvaru:

$$9x_1 - 2x_2 = 7$$

$$-2x_1 + 13x_2 = 11$$

Tab. 1. Výsledky pro počet iterací=2 a pro počet iterací=5 pro soustavu 2. řádu

Počet iterací = 2			Počet iterací = 5		
Číslo pokusu	Kořen x1	Kořen x2	Číslo pokusu	Kořen x1	Kořen x2
1	1,77080	1,81250	1	0,87500	1,27500
2	1,77080	1,37500	2	1,22500	0,91667
3	0,87500	1,37500	3	0,87500	1,81670
4	3,10420	1,37500	4	1,23330	1,45830
5	1,77080	1,83330	5	1,23330	1,28330
6	1,31250	1,37500	6	0,69167	1,45830
7	2,66670	1,37500	7	0,70000	1,45830
8	3,58330	0,91667	8	0,88333	1,10000
9	1,77080	1,37500	9	1,40830	1,10000
10	1,31250	2,72920	10	1,05000	1,10000
11	2,20830	0,93750	11	1,23330	1,64170
12	2,68750	2,27080	12	1,23330	0,91667
13	1,77080	0,91667	13	1,23330	0,56667
14	1,77080	1,37500	14	1,05830	1,10000
15	1,75000	0,93750	15	1,23330	1,27500
16	0,87500	1,37500	16	1,05000	0,91667
17	1,31250	1,37500	17	1,23330	0,73333
18	1,31250	1,37500	18	1,05000	1,10000
19	1,77080	1,81250	19	1,40830	1,10000
20	2,20830	1,35420	20	1,05830	1,45830
Průměr	1,88019	1,46354	průměr	1,10988	1,18875
směrodatná odchylka	0,6817279	0,44021307	směrodatná odchylka	0,201984564	0,2981329
analytické řešení	1	1	analytické řešení	1	1

Tab. 2. Výsledky pro počet iterací=10 a počet iterací=20 pro soustavu 2. řádu

Počet iterací = 10			Počet iterací = 20		
Číslo pokusu	Kořen x1	Kořen x2	Číslo pokusu	Kořen x1	Kořen x2
1	1,14580	1,00420	1	1,00830	1,09580
2	1,05420	1,27080	2	1,14170	0,91458
3	1,14580	0,73333	3	1,14380	1,36670
4	1,05420	1,18750	4	1,01040	1,05000
5	0,87083	1,18750	5	1,09790	1,00210
6	0,69583	1,27500	6	0,87500	0,96250
7	1,23750	1,10000	7	0,73958	1,45830
8	1,14170	1,10000	8	0,82917	1,05420
9	1,05830	1,00420	9	1,05420	1,18750
10	0,87917	0,91667	10	1,05420	1,00830
11	1,59580	1,00420	11	1,09580	1,18750
12	1,59580	1,27500	12	1,10000	0,96458
13	1,14170	0,83750	13	1,18750	1,18540
14	1,32080	1,54580	14	1,14380	1,09790
15	0,51667	1,27920	15	0,78542	1,00830
16	1,23750	1,36670	16	0,82917	1,14380
17	0,77917	0,55000	17	0,92083	1,05210
18	1,14170	0,82500	18	0,91875	1,23130
19	1,14580	1,27500	19	1,00830	0,82917
20	1,41250	1,27500	20	0,78333	0,96250
Průměr	1,11067	1,10063	průměr	0,96393	1,08813
směrodatná odchylka	0,264545483	0,235752	směrodatná odchylka	0,1370077	0,147589
analytické řešení	1	1	analytické řešení	1	1

Tab. 3. Výsledky pro počet iterací=50 a počet iterací=100 pro soustavu 2. řádu

Počet iterací = 50			Počet iterací = 100		
Číslo pokusu	Kořen x1	Kořen x2	Číslo pokusu	Kořen x1	Kořen x2
1	0,90917	1,15170	1	0,86375	0,97125
2	1,03670	0,98917	2	1,09130	1,06130
3	1,16250	1,02330	3	1,00080	1,01500
4	0,90917	1,11420	4	0,96333	0,99833
5	1,05330	1,00580	5	0,96375	1,06000
6	0,92667	1,07920	6	0,87375	0,95292
7	0,89417	1,13330	7	1,04540	0,98875
8	0,98250	0,95417	8	0,93000	0,98792
9	0,94667	0,97167	9	0,93625	1,06080
10	0,90917	0,97250	10	0,95542	0,98875
11	0,73167	1,06080	11	0,99292	1,14130
12	1,03750	1,09580	12	1,03540	0,99917
13	1,00080	0,95167	13	1,01790	1,00670
14	0,89167	1,15170	14	0,99125	0,96958
15	0,98250	1,00670	15	1,07250	1,06080
16	0,85500	1,02330	16	1,04500	1,06960
17	1,09000	1,02580	17	0,97375	1,01420
18	1,00000	1,15000	18	0,96417	1,11460
19	0,91000	1,04330	19	1,03620	1,07000
20	1,01920	1,02670	20	1,00130	1,17630
Průměr	0,95193	1,04654	průměr	0,98969	1,03536
směrodatná odchylka	0,0921594	0,0654582 7	směrodatná odchylka	0,05820435 5	0,058526 3
analytické řešení	1	1	analytické řešení	1	1

Tab. 4. Výsledky pro počet iterací=500 a počet iterací=800 pro soustavu 2. řádu

Počet iterací = 500			Počet iterací = 800		
Číslo pokusu	Kořen x1	Kořen x2	Číslo pokusu	Kořen x1	Kořen x2
1	1,00460	0,98075	1	0,94625	1,03670
2	0,96583	1,02110	2	0,94219	1,03770
3	0,95575	0,99558	3	0,98052	1,03810
4	0,97508	1,02280	4	0,99833	0,98875
5	1,00260	1,05060	5	0,97094	1,01090
6	0,98733	1,03710	6	0,98516	1,00570
7	0,95900	1,00510	7	0,97266	1,05270
8	0,94083	1,03320	8	0,96224	1,00060
9	0,97383	1,02440	9	0,95656	1,00950
10	0,95650	1,02460	10	0,97010	1,01820
11	0,96267	1,05180	11	0,99063	1,04170
12	1,00310	1,02020	12	0,96776	1,01260
13	0,96283	1,00250	13	0,96714	1,01390
14	0,95208	1,00940	14	1,00740	1,06910
15	0,96992	1,03150	15	0,96000	0,99245
16	0,96258	0,98692	16	0,93828	1,01460
17	1,01190	1,01440	17	1,01520	1,01390
18	0,99100	1,06280	18	0,96229	1,06200
19	0,96075	1,02430	19	1,02450	1,03240
20	0,95858	1,01170	20	0,97885	1,00910
Průměr	0,97222	1,02054	průměr	0,97686	1,02303
směrodatná odchylka	0,019753171	0,020509	směrodatná odchylka	0,02275267	0,021673
analytické řešení	1	1	analytické řešení	1	1

4.2 Statistické vyhodnocení pro soustavu 5. řádu

Soustavu 5. řádu jsem testoval pro 8 různých počtů iterací a to pro počet iterací = 2, 5, 10, 20, 50, 100, 150 a 200. Tato soustava nebyla testována pro takový počet iterací jako předchozí soustava 2. řádu, neboť výpočetní doba byla neúměrně dlouhá. Při takovéto simulaci hraje významnou roli výpočetní výkon počítače, na kterém simulaci provádíme. Pro ukázkou jsou zde 3 tabulky. Další jsou přiloženy na CD.

Testovaná soustava byla zadána ve tvaru:

$$10x_1 + 3x_2 - 2x_3 - x_4 + x_5 = 0,1$$

$$2x_1 + 15x_2 + x_3 + 2x_4 + 2x_5 = 3,2$$

$$2x_1 - 5x_2 + 13x_3 + 3x_4 + x_5 = 5$$

$$2x_1 + x_2 - 3x_3 + 20x_4 + 3x_5 = 2$$

$$x_1 + 2x_2 - 3x_3 + x_4 + 17x_5 = 5$$

Tab. 5. Výsledky pro počet iterací=2 pro soustavu 5. řádu

Počet iterací = 2					
Číslo pokusu	kořen x1	kořen x2	kořen x3	kořen x4	kořen x5
1	0,1197	0,1866	0,4637	0,0415	0,5158
2	0,0870	0,3894	0,6861	0,0016	0,0313
3	0,5314	0,2923	0,4257	0,0594	0,6239
4	0,1304	0,2286	0,2711	0,4744	0,5208
5	0,1917	0,2286	0,9476	0,1104	0,6189
6	0,1609	0,1811	0,4637	0,4279	0,3747
7	0,0360	0,1284	0,9001	0,0016	0,3651
8	0,3807	0,0880	0,4688	0,1703	0,3439
9	0,5880	0,2812	0,4636	0,6443	0,4688
10	0,0779	0,0783	0,4262	0,1546	0,4688
11	0,1254	0,0304	0,7807	0,2150	0,5679
12	0,0177	0,2869	0,4161	0,2519	0,7853
13	0,3575	0,1866	0,6139	0,2094	0,3484
14	0,2852	0,3429	0,7752	0,5113	0,4688
15	0,2336	0,3429	0,0992	0,3607	0,2755
16	0,1254	0,1866	0,1007	0,4800	0,3545
17	0,0688	0,6128	0,5411	0,1850	0,5724
18	0,0415	0,3429	0,4233	0,0396	0,0100
19	0,0161	0,3429	0,4111	0,0016	0,5735
20	0,0743	0,2746	0,6771	0,0980	0,5103
průměr	0,18246	0,25158	0,5178	0,22192	0,43992
směrodatná odchylka	0,16184	0,12781	0,22344	0,19096	0,18306
	9	5	1	3	1
analytické řešení	0,02406	0,12648	0,38096	0,09751	0,33931

	6			6	
--	---	--	--	---	--

Tab. 6. Výsledky pro počet iterací=20 pro soustavu 5. řádu

Počet iterací = 20					
Číslo pokusu	kořen x1	kořen x2	kořen x3	kořen x4	kořen x5
1	0,0090	0,1974	0,3648	0,0344	0,2925
2	0,0841	0,1514	0,3793	0,0312	0,3914
3	0,0325	0,3231	0,4632	0,1000	0,2463
4	0,0353	0,3414	0,3760	0,2729	0,2438
5	0,0020	0,1993	0,3869	0,0900	0,3344
6	0,0525	0,1935	0,4474	0,1314	0,3037
7	0,0739	0,1577	0,3725	0,1069	0,2845
8	0,0698	0,2436	0,5459	0,1174	0,3136
9	0,0094	0,1884	0,3625	0,1541	0,2361
10	0,0290	0,3139	0,5027	0,0319	0,3297
11	0,0458	0,2458	0,2656	0,0244	0,2849
12	0,0117	0,1613	0,3149	0,1228	0,2825
13	0,1163	0,1316	0,3662	0,1663	0,3229
14	0,0582	0,1857	0,4660	0,1263	0,4005
15	0,0137	0,2008	0,3375	0,0307	0,2719
16	0,0382	0,1776	0,3828	0,2057	0,3682
17	0,0915	0,1275	0,4446	0,1530	0,3753
18	0,0348	0,1972	0,4561	0,1399	0,2551
19	0,0310	0,1679	0,4082	0,1151	0,4152
20	0,0345	0,2657	0,3759	0,0852	0,3857
průměr	0,04366	0,20854	0,4010	0,11199	0,31691
směrodatná odchylka	0,02990 1	0,06019 7	0,06457 8	0,06212 8	0,05487 3
analytické řešení	0,02406 6	0,12648	0,38096	0,09751 6	0,33931

Tab. 7. Výsledky pro počet iterací=200 pro soustavu 5. řádu

Počet iterací = 200					
Číslo pokusu	kořen				
	kořen x1	kořen x2	x3	kořen x4	kořen x5
1	0,0164	0,1467	0,3796	0,0921	0,3119
2	0,0120	0,1729	0,3857	0,0790	0,2749
3	0,0365	0,1478	0,3919	0,0764	0,3047
4	0,0604	0,1458	0,3799	0,0813	0,2948
5	0,0055	0,1678	0,3788	0,0749	0,2645
6	0,0166	0,1125	0,4274	0,0901	0,3278
7	0,0099	0,1214	0,3851	0,0604	0,2549
8	0,0119	0,1147	0,3716	0,0885	0,2745
9	0,0251	0,1367	0,3914	0,0695	0,2806
10	0,0033	0,1035	0,4054	0,0318	0,2844
11	0,0502	0,1720	0,4003	0,0900	0,2843
12	0,0206	0,1258	0,3775	0,0808	0,3157
13	0,0448	0,1248	0,4155	0,0955	0,3234
14	0,0229	0,1489	0,3837	0,0880	0,2977
15	0,0236	0,1378	0,4283	0,1012	0,2964
16	0,0075	0,1278	0,3989	0,1098	0,3258
17	0,0227	0,1325	0,4257	0,0742	0,2975
18	0,0015	0,1578	0,3878	0,0639	0,2778
19	0,0483	0,1698	0,3761	0,0945	0,3115
20	0,0380	0,1789	0,3477	0,0918	0,2862
průměr	0,02389	0,14230	0,3919	0,08167	0,29447
směrodatná odchylka	0,01665 4	0,02172 5	0,02005	0,01659 8	0,02012 5
analytické řešení	0,02406 6	0,12648	0,38096	0,09751 6	0,33931

4.3 Vyhodnocení

Přemýšlel jsem, jakým způsobem mám porovnávat výsledky pro jednotlivé soustavy. Nakonec jsem nasimulované hodnoty vyhodnotil pomocí kvadratické odchylky chyb.

$$\sum_{i=1}^n (x_i - y_i)^2$$

kde x_i je vypočítaná hodnota konkrétního kořenu, a y_i je analytické řešení.

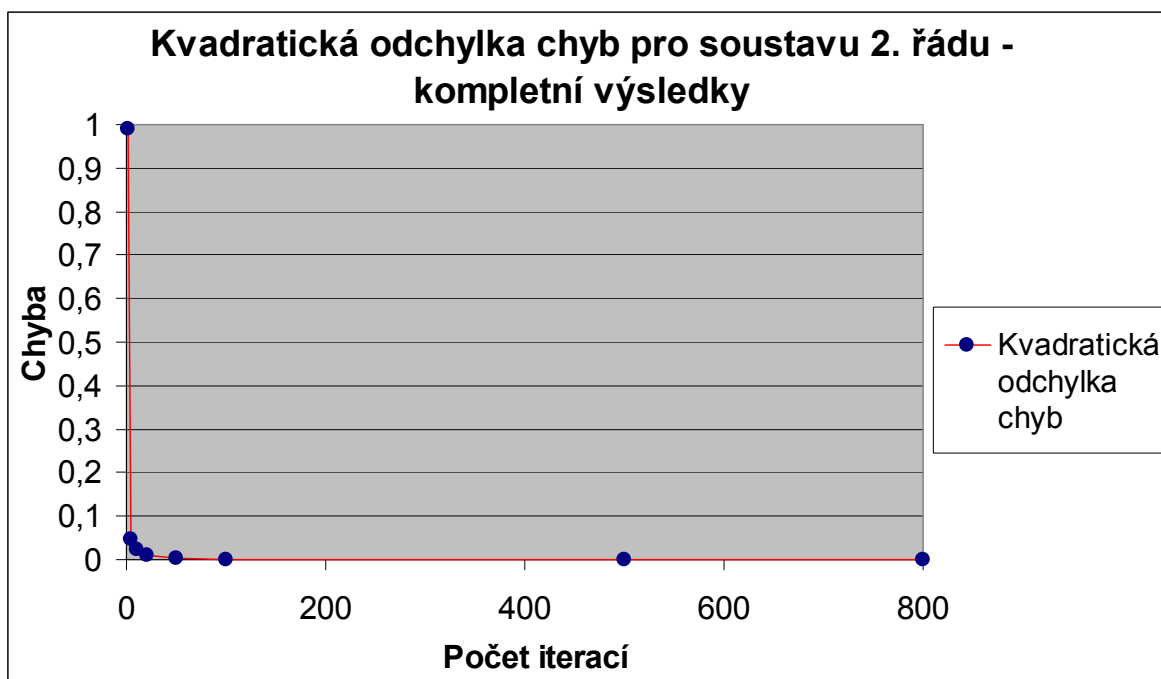
4.3.1 Vyhodnocení pro soustavu 2. řádu

Pro soustavu 2. řádu vyšly následující hodnoty:

Tab. 8. Kvadratická odchylka chyb
pro soustavu 2. řádu

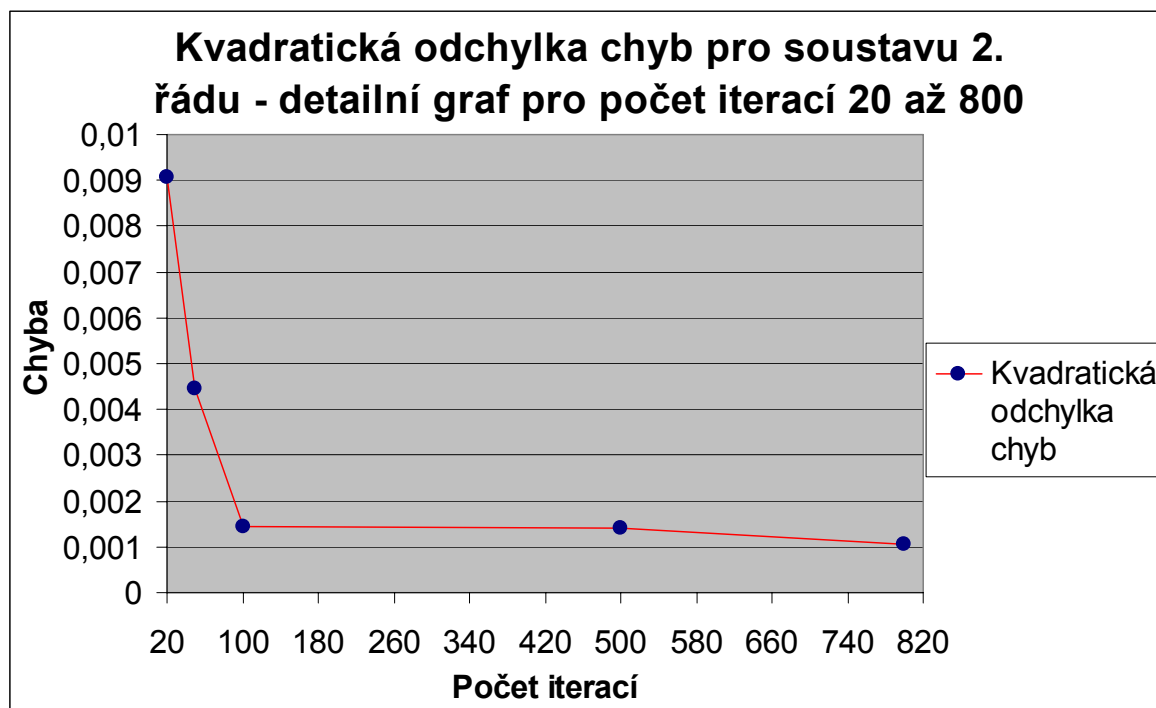
Počet iterací	Kvadratická odchylka chyb
2	0,9896
5	0,0477
10	0,02237
20	0,009067
50	0,00447
100	0,001438
500	0,001417
800	0,001066

Pro lepší čitelnost jsem hodnoty rozdělil do dvou grafů - 1.část : Kompletní graf



Obr. 7. Kvadratická odchylka chyb pro soustavu 2. řádu (kompletní výsledky)

2.část: Detailní graf pro počet iterací 20 až 800



Obr. 8. Kvadratická odchylka chyb pro soustavu 2. řádu (detailní graf pro počet iterací 20 až 800)

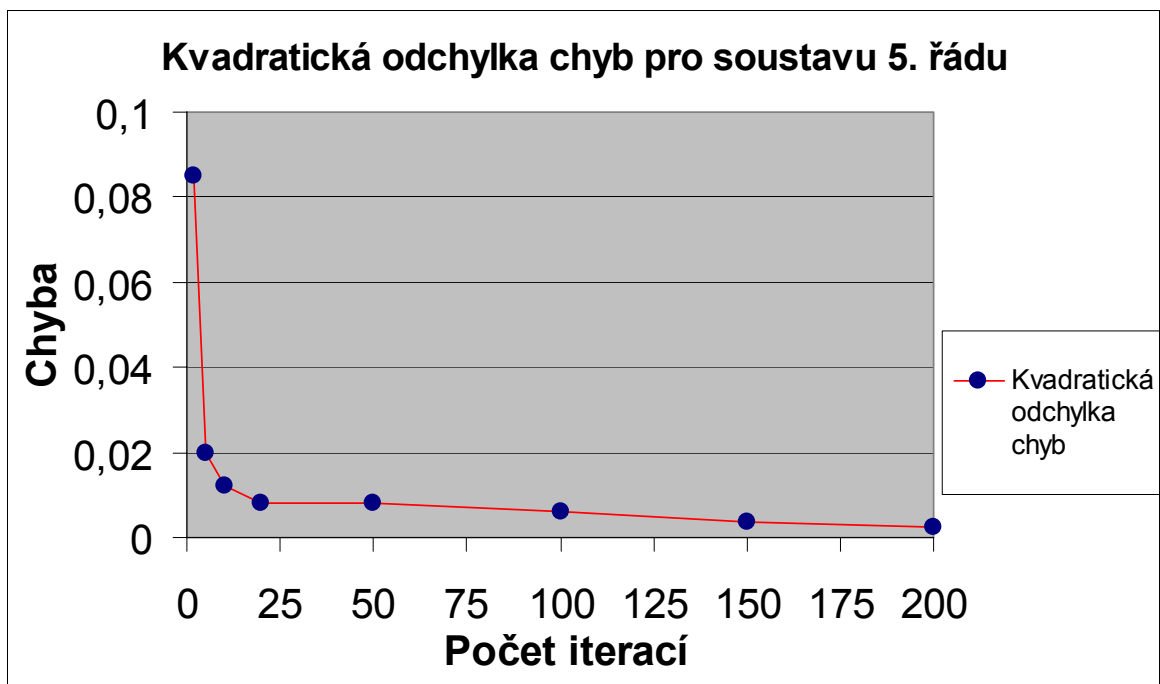
4.3.2 Vyhodnocení pro soustavu 5.řádu

Pro soustavu 5. řádu vyšly následující hodnoty:

Tab. 9. Kvadratická odchylka chyb
pro soustavu 5. řádu

Počet iterací	Kvadratická odchylka chyb
2	0,08506
5	0,01986
10	0,01221
20	0,00823
50	0,00808
100	0,006189
150	0,003777
200	0,002631

Graf pro soustavu 5. řádu:



Obr. 9. Kvadratická odchylka chyb pro soustavu 5. řádu

ZÁVĚR

Na základě experimentu jsem vytvořil program pro řešení systému lineárních algebraických rovnic metodou Monte Carlo. Tento program řeší soustavu rovnic pomocí metody Monte Carlo, dále pomocí metod Prosté iterační, Seidelovy iterační, Jacobiovy a Gauss-Seidelovy a jako poslední hledá program analytické řešení. Jednotlivé mezivýsledky jsou ukládány do souboru, takže uživatel má veškerá data k dispozici, aby si je mohl prohlédnout a vidět rozdíly mezi jednotlivými metodami během výpočtu. Tato bakalářská práce je také literární rešerší pro dané téma.

Ještě než zhodnotím metodu Monte Carlo, musím podotknout některé veličiny ovlivňující průběh výpočtu dané metody. A to je generování pseudonáhodných čísel, neboť záleží na algoritmu výpočtu náhodných čísel. A tou další veličinou je výpočetní výkon počítače, na kterém simulace probíhá.

A teď k vlastnímu závěru, což je vyhodnocení metody Monte Carlo. U první soustavy, což je soustava 2. řádu, je při malém počtu iterací vidět velká chyba. Při zvětšení počtu iterací chyba klesá. Zatímco u druhé soustavy, což je soustava 5. řádu, je vidět i při malém počtu iterací malá chyba. Z toho vyplývá, že metoda Monte Carlo je výhodná pro soustavy rovnic s více proměnnými a s co největší hodnotou iterací. Pro soustavy s malým počtem neznámých – např. již zmiňovaná soustava 2. řádu je velká chyba při malém počtu iterací. Při simulaci soustavy 5. řádu jsem narazil na problém s délkou doby výpočtu. Při počtu iterací = 200 začíná být doba potřebná k výpočtu značně dlouhá. Zatímco u soustavy 2. řádu délka doby výpočtu při počtu iterací = 900 je zanedbatelná. To vidím jako nevýhodu metody Monte Carlo. Jako hlavní výhodu vidím i při malém počtu iterací přesnost výsledků pro soustavy s velkými počty proměnnými.

Doufám, že tento program bude svou názorností problému dobrou učební pomůckou pro předmět Simulace systémů.

SEZNAM POUŽITÉ LITERATURY

- [1] FABIÁN, F., KLUIBER, Z.: Metoda Monte Carlo a možnosti jejího uplatnění. Prospektum s.r.o., Praha, 1998.
- [2] HUŠEK, R., LAUBER, J.: Simulační modely. SNTL-Nakladatelství technické literatury, Praha, 1987.
- [3] CYHELSKÝ, L., KAHOUNOVÁ, J., HINDLS, R.: Elementární statistická analýza, Management Press, Praha, 1999.
- [4] VICHER, M.: Numerická matematika. Elektronická příručka.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MATLAB - je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálu, návrhy řídicích a komunikačních systémů.

GUIDE – modul programového prostředí Matlab

SEZNAM OBRÁZKŮ

<i>Obr. 1. Uživatelské prostředí po spuštění</i>	21
<i>Obr. 2. Ukázka výpočtu kořenů</i>	22
<i>Obr. 3. Chybová hláška pro Prostou a Seidelovu metodu</i>	23
<i>Obr. 4. Chybová hláška pro Jacobiovu a Gauss-Seidelovu metodu</i>	23
<i>Obr. 5. Chybová hláška pro metodu Monte Carlo</i>	23
<i>Obr. 6. Výpis souboru Výsledky.txt</i>	24
<i>Obr. 7. Kvadratická odchylka chyb pro soustavu 2. řádu (kompletní výsledky)</i>	33
<i>Obr. 8. Kvadratická odchylka chyb pro soustavu 2. řádu (detailní graf pro počet iterací 20 až 800)</i>	34
<i>Obr. 9. Kvadratická odchylka chyb pro soustavu 5. řádu</i>	35

SEZNAM TABULEK

<i>Tab. 1. Výsledky pro počet iterací=2 a pro počet iterací=5 pro soustavu 2. řádu</i>	<i>26</i>
<i>Tab. 2. Výsledky pro počet iterací=10 a počet iterací=20 pro soustavu 2. řádu.....</i>	<i>27</i>
<i>Tab. 3. Výsledky pro počet iterací=50 a počet iterací=100 pro soustavu 2. řádu.....</i>	<i>28</i>
<i>Tab. 4. Výsledky pro počet iterací=500 a počet iterací=800 pro soustavu 2. řádu.....</i>	<i>29</i>
<i>Tab. 5. Výsledky pro počet iterací=2 pro soustavu 5. řádu</i>	<i>30</i>
<i>Tab. 6. Výsledky pro počet iterací=20 pro soustavu 5. řádu</i>	<i>31</i>
<i>Tab. 7. Výsledky pro počet iterací=200 pro soustavu 5. řádu</i>	<i>32</i>
<i>Tab. 8. Kvadratická odchylka chyb pro soustavu 2. řádu</i>	<i>33</i>
<i>Tab. 9. Kvadratická odchylka chyb pro soustavu 5. řádu</i>	<i>34</i>

SEZNAM PŘÍLOH

- P I Část zdrojového kódu metody Monte Carlo.
- P II Zdrojový kód Prosté iterační metody.
- P III Zdrojový kód Seidelovy iterační metody.
- P IV Zdrojový kód Jacobiovy metody
- P V Zdrojový kód Gauss-Seidelovy metody

PŘÍLOHA P I: ČÁST ZDROJOVÉHO KÓDU METODY MONTE CARLO.

```
1 %Vlastní výpočet
2 for i=1:1:n
3
4 soucet=0;
5 m=0;
6
7 e(i)=0;
8 ee=0;
9
10 start=i;
11 stav=i;
12
13 for p=1:1:po+1
14 msoucet=1;
15 gen_cislo=rand(1,1);
16 soucet=beta(start);
17
18 while(gen_cislo<T([stav],[n+1]))
19
20     for j=1:1:n
21         if((gen_cislo>T([stav],[j])) && (gen_cislo<T([stav],[j+1])))
22             m=stav;
23             stav=j;
24         end
25     end
26     msoucet=msoucet*v([m],[stav]);
27     soucet=soucet+msoucet*beta(stav);
28     gen_cislo=rand(1,1);
29
30 end
31 e(i)=e(i)+soucet;
32 vysl(p,i)=e(i)/po;
33
34 end
35
36 end
37 vysledek=e/po;
```

PŘÍLOHA P II: ZDROJOVÝ KÓD PROSTÉ ITERAČNÍ METODY.

```
1 %Výpočet Prosté iterační metody
2 k=1;
3 i=1;
4 temp=0;
5 for j=1:1:n
6     w([1],[j])=0;
7 end
8
9 while(k<(po+1))
10     for i=1:1:n
11         for j=1:1:n
12             temp=temp+U([i],[j])*w([k],[j]);
13         end
14         w([k+1],[i])=temp+v(i);
15         w(k,i)=w([k+1],[i]);
16         temp=0;
17         if(i==n)
18             k=k+1;
19         end
20     end
21 end
22
23 for i=1:1:n
24     Prosta(i)=w([po+1],[i]);
25 end
26 Prosta;
27
28 handles.prosta=strcat(num2str(Prosta));
29 set(handles.edit5,'String',handles.prosta);
```

PŘÍLOHA P III: ZDROJOVÝ KÓD SEIDELOVY METODY

```
1 %Výpočet Seidelovy iterační metody
2 k=1;
3 i=1;
4 first=0;
5 second=0;
6
7 for j=1:1:n
8     s([1],[j])=0;
9 end
10
11 while(k<(po+1))
12     for i=1:1:n
13         for j=1:1:(i-1)
14             first=first+U([i],[j])*s([k+1],[j]);
15         end
16
17         for j=i:1:n
18             second=second+U([i],[j])*s([k],[j]);
19         end
20         s([k+1],[i])=first+second+v(i);
21         s(k,i)=s([k+1],[i]);
22         first=0;
23         second=0;
24         if(i==n)
25             k=k+1;
26         end
27     end
28 end
29
30 for i=1:1:n
31     Seidel(i)=s([po+1],[i]);
32 end
33 Seidel;
34
35 handles.seidel=strcat(num2str(Seidel));
36 set(handles.edit6,'String',handles.seidel);
```

PŘÍLOHA P IV: ZDROJOVÝ KÓD JACOBIOVY METODY

```

1 %Výpočet Jacobiovy metody
2 k=1;
3 i=1;
4 temp=0;
5
6 for j=1:1:n
7     y([1],[j])=0;
8 end
9
10 while(k<(po+1))
11     for i=1:1:n
12         for j=1:1:n
13             if (j~=i)
14                 temp=temp+C([i],[j])*y([k],[j]);
15             end
16         end
17         y([k+1],[i])=(1/C([i],[i]))*(B(i)-temp);
18         y(k,i)=y([k+1],[i]);
19         temp=0;
20         if(i==n)
21             k=k+1;
22         end
23     end
24 end
25
26 for i=1:1:n
27     Jacobi(i)=y([po+1],[i]);
28 end
29 Jacobi;
30
31 handles.jacobi=strcat(num2str(Jacobi));
32 set(handles.edit7,'String',handles.jacobi);

```

PŘÍLOHA P V: ZDROJOVÝ KÓD GAUSS-SEIDELOVY METODY

```

1 %Výpočet Gaussovy-Seidelovy metody

```

```

2 k=1;
3 i=1;
4 first=0;
5 second=0;
6
7 for j=1:1:n
8     z([1],[j])=0;
9 end
10
11 while(k<(po+1))
12     for i=1:1:n
13         for j=1:1:(i-1)
14             first=first+C([i],[j])*z([k+1],[j]);
15         end
16
17         for j=(i+1):1:n
18             second=second+C([i],[j])*z([k],[j]);
19         end
20         z([k+1],[i])=(1/C([i],[i]))*(B(i)-first-second);
21         z(k,i)=z([k+1],[i]);
22         first=0;
23         second=0;
24         if(i==n)
25             k=k+1;
26         end
27     end
28 end
29
30 for i=1:1:n
31     GaussSeidel(i)=z([po+1],[i]);
32 end
33 GaussSeidel;
34
35 handles.gaussseidel=strcat(num2str(GaussSeidel));
36 set(handles.edit8,'String',handles.gaussseidel);

```