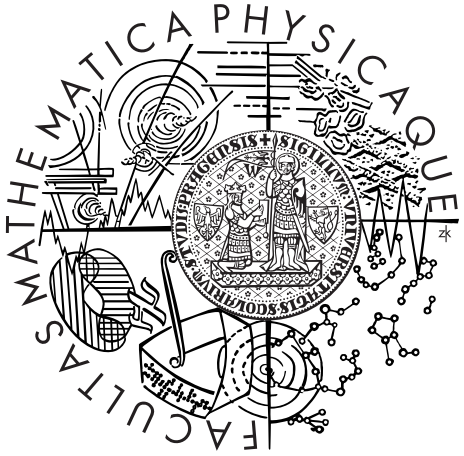
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Matyáš Novák

Problém vlastních čísel symetrických řídkých matic v souvislosti s výpočty elektronových stavů

Katedra aplikované matematiky

Vedoucí diplomové práce:	Prof. Ing. Miroslav Tůma, CSc.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství

Praha 2012

Tímto děkuji panu profesoru Tůmovi i doktoru Vackářovi za vedení práce, všechny jejich dobré rady a velkou trpělivost a pochopení a v neposlední řadě své ženě Agátě nejen za veškerou podporu při psaní práce.

Prohlašuji, že jsem tuto diplomovou 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: Problém vlastních čísel symetrických řídkých matic v souvislosti s výpočty elektronových stavů

Autor: Matyáš Novák

Katedra: Katedra aplikované matematiky

Vedoucí diplomové práce: prof., ing. Miroslav Tůma, CSc., Ústav informatiky Akademie Věd ČR, v. v. i.

Abstrakt: Ab-initio metody pro výpočty elektronových struktur tvoří jednu z důležitých oblastí materiálové fyziky. Úkolem této práce – v rámci řešení projektu zaměřeného na vývoj nové metody pro výpočty elektronových stavů v neperiodických strukturách, založené na teorii funkcionálu hustoty, pseudopotenciálech a metodě konečných prvků – bylo převést Kohn-Shamovy rovnice do tvaru vhodného k diskretizaci, navrhnout vhodnou metodu pro řešení zobecněného problému vlastních čísel, který touto diskretizací vznikne, a implementovat (či upravit existující) řešič pro jeho řešení.

Práce popisuje postup, kterým se z mnohočásticové Schrödingerovy rovnice získá generalizovaný problém vlastních čísel s aktualizací řádu k (rank-k-update) a věnuje se různým metodám pro jeho řešení. V rámci práce byl modifikován již existující řešič využívající blokovou Lanczosovu metodu pro výpočet vlastních čísel, integrován do frameworku Sfepy sloužícího k výpočtu metodou konečných prvků a vzniklý programový kód byl úspěšně otestován.

Klíčová slova: DFT, FEM, Lanczosova metoda, rank-n-update, vlastní čísla

Title: Eigenvalue problem of sparse symmetric matrix with connection to electronic state calculation

Author: Matyáš Novák

Department: Department of Applied Mathematics

Supervisor: prof., ing. Miroslav Tůma, CSc., Institute of Computer Science of the ASCR, v. v. i.

Ab-initio methods for calculating electronic structure represent an important field of material physics. The aim of this theses — within the project focused on developing the new method for calculating electronic states in non-periodic structures based on density functional theory, pseudopotentials, and finite elements methods — is to convert Kohn-Sham equations into the form suitable for discretisation, to suggest apropriate method for solving generalized eigenproblem resulting from this discretisation and to implement an eigenvalue solver (or to modify existing one).

The thesis describes a procedure for converting the many-particle Schrödinger equation into generalized rank-k-update eigenvalue problem and discusses various methods for its solution. Eigensolver Blzpack, which makes use of the block Lanczos method, has been modified, integrated into the Sfepy framework (a tool for the finite element method calculation) and resulting code has been successfully tested.

Keywords: DFT, FEM, Lanczos method, rank-n-update, eigenvalues

Obsah

1 Úvod **1**

1.1 Značení . . . 3
1.2 Definice . . . 4
1.2.1 Řídká matice . . . 4
1.2.2 B-ortogonalita . . . 5
1.2.3 Separovaný tvar operátoru . . . 5
1.2.4 Potenciál . . . 6
1.2.5 Rozklad vícedimenzionálních operátorů . . . 7

2 Řešení Schrödingerovy rovnice pomocí metody konečných prvků **9**

2.1 Metoda DFT pro řešení Schrödingerovy rovnice . . . 9
2.1.1 Shrödingerova rovnice . . . 10
2.1.2 Kohn-Shamovy rovnice . . . 10
2.1.3 Spin a obsazení stavů . . . 11
2.1.4 Operátory Kohn-Shamovy rovnice a jejich vlastnosti . . . 12
2.1.5 Konvergence metody . . . 13
2.1.6 Rozmazání (smearing) . . . 13
2.1.7 Algoritmus DFT cyklu . . . 13
2.2 Pseudopotenciály . . . 15
2.3 Diskretizace metodou konečných prvků . . . 18
2.3.1 Báze konečných prvků . . . 18
2.3.2 Slabá forma Kohn-Shamovy rovnice . . . 20
2.3.3 Diskretizace Kohn-Shamovy rovnice . . . 21
2.4 Struktura matice a řešení problému vlastních čísel . . . 23
2.4.1 Struktura operátorů Kohn-Shamovy rovnice . . . 24
2.4.2 Separabilní pseudopotenciál . . . 24
2.4.3 Problém vlastních čísel . . . 26

3 Řešení diskretizovaných úloh **28**

3.1 Dělení problémů vlastních čísel . . . 28
3.2 Obecné postupy užívané při řešení problému vlastních čísel . . . 29
3.2.1 Exaktní aritmetika a aritmetika s konečnou přesností . . . 29
3.2.2 Metody užívající rozkladů matic . . . 29
3.2.3 Rayleighův koeficient . . . 30
3.2.4 Konvergence iteračních metod . . . 31
3.2.5 Počítání dalších vlastních vektorů . . . 32
3.3 Metody založené na iteraci vektoru I – mocninné metody . . . 33
3.3.1 Mocninná metoda . . . 34

3.3.2 Bloková mocninná metoda (metoda iterace podprostoru) . 38
3.3.3 Metody užívající posunu a inverze (shift and invert) . . . 40
3.3.4 Mocninné metody a zobecněný problém vlastních čísel . . 42
3.4 Projekční metody I – krylovovské metody . . . 42
3.4.1 Arnoldiho metoda . . . 43
3.4.2 Reortogonalizace . . . 45
3.4.3 Arnoldiho metoda s částečnou ortogonalizací a restartovaná Arnoldiho metoda . . . 45
3.4.4 Implicitně restartovaná Arnoldiho metoda . . . 46
3.4.5 (Symetrická) Lanczosova metoda . . . 49
3.4.6 Nesymetrická Lanczosova metoda . . . 51
3.4.7 Krylovovské metody pro zobecněný problém vlastních čísel 53
3.5 Metody založené na iteraci vektoru II – minimalizační metody . . 55
3.5.1 Metoda největšího spádu . . . 56
3.5.2 Metoda největšího spádu pro zobecněný problém vlastních čísel . . . 59
3.5.3 Metoda sdružených gradientů . . . 60
3.6 Jacobi-Davidsonova metoda . . . 62
3.6.1 Davidsonova metoda . . . 62
3.6.2 Jacobi-Davidsonova metoda . . . 63
3.6.3 Techniky užívané variantami Jacobi-Davidsonova algoritmu 64
3.6.4 Jacobi-Davidsonova metoda s harmonickými Ritzovými čísly 66
3.6.5 B-ortogonální Jacobi-Davidsonova metoda pro zobecněný problém vlastních čísel . . . 66
3.6.6 QZ Jacobi-Davidsonova metoda pro zobecněný problém vlastních čísel . . . 67
3.6.7 Zhodnocení Jacobi-Davidsonovy metody . . . 71

4 Implementace **72**
4.1 Výběr řešiče . . . 72
4.2 Popis dalších části řešení . . . 75

5 Dosažené výsledky **78**
5.1 Modelový problém . . . 78
5.2 Analýza výkonu a přesnosti konvergence . . . 80
5.2.1 Paralelizace . . . 80
5.2.2 Přesnost řešení . . . 81
5.2.3 Vliv aktualizace řádu k na dobu výpočtu . . . 82
5.2.4 Velikost bloku . . . 82
5.2.5 Startovací vektory . . . 84
5.2.6 Závěr z provedené analýzy . . . 85

6 Závěr **87**

Literatura **88**

Seznam tabulek **95**

Seznam algoritmů **95**

7 Příloha 1 - dokumentace k řešiči zobecněného problému s aktualizací řádu k **i**

7.1 Požadavky i
7.2 Licence i
7.3 Instalace ii
7.4 Dokumentace exportovaných funkcí ve fortranu iii
7.5 Dokumentace třídy v Pythonu iv
7.6 Implementace vlastního řešiče vi

1. Úvod

Počítačové modelování materiálů, pro které byla v průběhu minulých desetiletí vyvinuta celá řada metod a přístupů, tvoří náplň jednoho z oborů, které se nyní intenzivně rozvíjejí — výpočetní materiálové fyziky (computational material science).

Mezi metodami pro počítačové modelování materiálů zaujímají zvlášť význačné postavení tzv. *ab-initio* metody, jinak též označované jako výpočty z prvních principů. Jejich charakteristickým rysem je nezávislost na jakýchkoliv empirických údajích ve vstupních datových souborech. Vstupními daty těchto výpočtů jsou pouze druhy a počty atomů, jimiž je materiál tvořen, jejich počáteční pozice a případné vnější vlivy působící na vzorek materiálu: tlak, teplota, elektrické nebo magnetické pole, elektrický náboj vzorku, akustické vlny apod. Na základě těchto vstupních dat je možné určit, jaké síly budou působit mezi atomy, jakým směrem budou měnit svoji polohu, jaká bude jejich rovnovážná poloha, jejich elektrické a magnetické momenty, jaká bude jejich celková energie a jak se tato energie bude měnit při změně vnějších podmínek – tj. jaké síly (ve zobecněném smyslu) vyvolá taková změna. Z těchto informací vyplývají mechanické, elektrické a magnetické vlastnosti materiálu, jeho struktura, případně stabilita různých strukturních modifikací za různých tlaků, teplot apod.

Je pochopitelné, že existuje velmi silná motivace pro to, mít v rukou co nejpřesnější a nejrychlejší nástroj pro *ab-initio* modelování materiálů. Takový nástroj otevírá cestu k rychlým experimentům na počítačovém modelu, v nichž lze zjistit, jaké bude mít ten či onen materiál zajímavé vlastnosti, aniž bychom museli (ve fázi těchto experimentů) takový materiál vyrobit – dokonce aniž bychom potřebovali vědět, zda a jak se dá takový materiál vyrobit. Lze si dobře představit i zavedení další úrovně těchto výpočtů, kdy by mohl být skenován prostor možných složení a strukturních konfigurací materiálů a cíleně vyhledáván materiál požadovaných vlastností. Je zřejmé, proč se často používá v této souvislosti pojem *„ab-initio material design“* – česky snad nejlépe *„počítačový návrh materiálů z prvních principů“* – přesto, že něco takového je zatím spíše hudbou budoucnosti (i když možná blízké).

Vlastnosti materiálů, ať už chemické, mechanické, elektrické nebo magnetické, jsou podmíněny stavem elektronů (více či méně) svázaných s jádry atomů, které tvořící daný materiál. Kolem výpočtů těchto stavů pomocí formalismu kvantové teorie se soustřeďuje prakticky veškeré úsilí ve výpočetní materiálové fyzice. Atomová jádra hrají pouze roli bodového náboje s danou hmotností, případně magnetickým momentem. Teoreticky je jasné, jakou úlohu je třeba řešit, abychom získali kompletní informaci o elektronových stavech: Diracovu rovnici pro mnohočásticovou vlnovou funkci popisující všechny elektrony v materiálu.

Prakticky je taková úloha velmi vzdálená našim reálným možnostem — přinejmenším pro cokoliv složitějšího, než je atom vodíku. Proto vznikla v průběhu minulých desetiletí řada metod, jak tuto potíž obejít. Metody lze rozdělit do kategorií podle různých kritérií:

a) dle způsobu, jakým aproximují mnohočásticovou vlnovou funkci
b) dle toho, jak se vypořádají se singularitami potenciálu v místě atomových jader a obrovskými rozdíly energií mezi vnitřními a valenčními elektronovými

stavy (cca 14 desítkových řádů) – v situaci, kdy právě jemné změny valenčních stavů poskytují nejcennější informace

c) podle toho, zda předpokládají translační symetrii (prostorovou periodicitu) materiálu, tedy zda aproximují materiál nekonečným krystalem

d) a dle typu báze, používané k popisu vlnových funkcí a relevantních fyzikálních veličin; především zda je tato báze obecná, tj. vhodná pro libovolný problém, nebo zda využívá nějakých fyzikálních předpokladů

Tato práce je součástí širšího projektu (navazujícího na granty GAAV IAA 100100637 a GAČR 101-09-1630), jehož cílem je zaplnit mezeru mezi dosavadními metodami pro *ab-initio* výpočty materiálových vlastností. Nově vyvíjená metoda je postavena (ve smyslu výše uvedených kritérií) na:

a) teorii funkcionálu hustoty (DFT — density functional theory), která převádí mnohočásticovou úlohu na jednoelektronové Kohn-Shamovy rovnice. Tento přístup je dnes ve fyzice pevných látek nejrozšířenější, neboť podstatně snižuje stupně volnosti a tím zjednodušuje výpočet

b) konceptu *ab-initio* selfkonsistentního (tzv. emphenvironment-reflecting) pseudopotenciálu, který umožňuje separovat vnitřní elektronové stavy spolu s atomovým jádrem a toto popsat jedním — byť netriviálním — linárním operátorem

c) absenci translační symetrie - neboť vývoji metod *nevyužívajících* předpoklad periodicity bylo dosud věnováno mnohem méně úsilí než metodám konstruovaným pro nekonečné krystaly, kterých už existuje značné množství

d) *metodě konečných prvků* (FEM — finite-element method), která je univerzální, nezatížená *ad-hoc* předpoklady, umožňuje bezproblémovou kontrolu konvergence a je dlouholetou praxí prověřená a neustále vyvíjená mnoha vývojáři průmyslových aplikací v oblasti komerční i v oblasti open-source

Cílem této diplomové práce je transformovat obecný pseudopotenciál do separabilního tvaru, použít ho při vytvoření matice Hamiltonova operátoru s aktualizací řádu k (rank-k-update) na základě Kohn-Shamových rovnic, vybrat vhodnou metodu pro nalezení vlastních vektorů a vlastních čísel Hamiltonova operátoru a tuto metodu prakticky implementovat.

Speciální pozornost pak bude věnována především metodám na řešení problému vlastních čísel, neboť v průběhu vývoje metody bylo zvažováno i použití jiných forem pseudopotenciálů, vedoucích k odlišnému tvaru Kohn-Shamových rovnic, po diskretizaci k jinému typu matice Hamiltonova operátoru a tedy i k jinému typu problému vlastních čísel. Výsledkem této práce je funkční kód využívající framework SfePy(Cimrman et al., 2011) pro implementaci FEM a upravený kód Blzpack (Marques, 1997) pro řešení problému vlastních čísel. Tento kód byl začleněn do programu na modelování elektronové struktury.

Pro řešení tohoto problému bylo třeba se seznámit se samotnými Kohn-Shamovými rovnicemi a jejich diskretizací (převedení analytických diferenciálních rovnic na soustavu lineárních rovnic pomocí metody konečných prvků) a taktéž zvolit jejich nejvhodnější formu pro následný výpočet a do této formy tyto rovnice převést. Toto popisuje kapitola 2.

Dále bylo třeba se seznámit s různými metodami řešení problému vlastních čísel, jejich výhodami a nevýhodami a jejich vhodností pro problémy vlastních čísel různých typů, což je popsáno v kapitole 3. Z těchto metod pak byla vybrána vhodná metoda k vyřešení problému vlastních čísel, vzniklého diskretizací Kohn-Shamových rovnic.

K zvolené metodě pak bylo třeba nalézt vhodný řešič a (neboť jak se ukáže, vhodný dostupný řešič není k dispozici) provést jeho nutné úpravy tak, aby byl schopen řešit vzniklý diskretizovaný problém – což je popsáno v kapitole 5.2.3. Základní výsledky získané tímto řešičem pak jsou uvedeny v poslední kapitole. Samotný řešič je elektronickou přílohou práce, stručná dokumentace k němu pak tištěnou přílohou práce.

1.1 Značení

Ve fyzikálních rovnicích je zvykem užívat trochu odlišné konvence než v numerické matematice. Jelikož je tento text na pomezí lineární algebry a fyziky s přesahy do informatiky, zavedeme níže popsané značení. Naší snahou při zavedení této konvence je, aby byl text srozumitelný jak lidem navyklým na notaci fyzikální, tak i pro čtenáře užívající notaci lineárně algebraickou. V některých případech jde tedy o nutný kompromis.

Budeme tedy používat:

- Velká dvojitá písmena latinky pro číselné obory ($\mathbb{N}, \mathbb{R}, \dots$).
- Písmeno $\mathcal{V}$ pro podmnožinu $\mathbb{R}^3$ – oblast prostoru, na které probíhá výpočet. Tato podmnožina musí být souvislým sjednocením konečného počtu konvexních množin.
- Řecké písmeno Γ pro hranice tohoto prostoru.
- Písmeno $\mathscr{H}$ označuje n-rozměrný separabilní (Formánek, 2004) Hilbertův prostor v kvadrátu integrovatelných[1] funkcí $\mathscr{H} = L^2(\mathcal{V})$, zpravidla prostor $L^2(\mathcal{V}) \otimes L^2(\mathcal{V}) \otimes L^2(\mathcal{V})$, kde $L^2(\mathcal{V})$ prostor reálných kvadraticky integrovatelných funkcí nad $\mathcal{V}$. [2]
- Velká písmena latinky odpovídají operátorům $(U, V, \dots)$ na $\mathscr{H}$, zároveň jimi budeme označovat matice vzniklé diskretizací těchto operátorů.
- Malá řecká písmena (ψ) odpovídají vlnovým funkcím, tedy prvkům $\mathscr{H}$. Budeme je taktéž užívat pro označení těch skalární veličin, pro které je užívání řeckých minuskulí zažité, především λ, θ a σ pro vlastní číslo respektive jeho odhad, ε pro energii[3], a π pro stejnojmenou konstantu.
- Řeckou kurzívou (φ) jsou značeny prvky báze konečných prvků.
- Tučnou latinkou $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ jsou značeny vektory z R^n či C^n.
- Malá písmena latinky (a,b,c) označují skalární veličiny.
- Operátory A^T, A^+, A^{-1} pro operaci transpozice, matici hermitovsky sdruženou a inverzi matice A.

[1] $\int_{\mathcal{V}} f(x)^2 dx < \infty$

[2] Zatímco $\mathcal{V}$ tedy značí *reálný prostor*, $\mathscr{H}$ označuje prostor všech možných stavů fyzikálního systému na tomto prostoru (v případě Kohn-Shamových rovnic tedy všech možných stavů dané částice).

[3] Tedy v podstatě, jak dále ukážeme, taktéž vlastní číslo.

– Horní index u matice značí dle kontextu mocninu či index iterace, index iterace může být pro zpřehlednění zvýrazněn závorkami: $A^{(j)}$.
  Horní index u vektoru značí index iterace, dolní index dle kontextu buďto index iterace, nebo složku vektoru.
– Lomené závorky $\langle \mathbf{x} | \mathbf{y} \rangle$ značí skalární součin vektorů $\mathbf{x}$ a $\mathbf{y}$.[4]
– $\langle \mathbf{x} | \mathbf{y} \rangle_T$ je bilineární forma nad vektory z R^n či C^n definovaná $\langle \mathbf{x} | \mathbf{y} \rangle_T = \mathbf{x}^T \mathbf{y}$.
– $\|\mathbf{x}\|_n$ je n-norma vektoru $\mathbf{x} \in \mathbb{C}^n$ definovaná

$$\sqrt[n]{\sum_{i=1}^{n} \mathbf{x}_n} \tag{1.1.1}$$

– $\|\mathbf{x}\|$ je eukleidovská (neboli 2-norma) vektoru $\mathbf{x}$.
– Symbol $\hbar$ označuje Planckovu konstantu.

1.2 Definice

Vzhledem k tomu, že v této práci budeme používat pojmy, které taktéž různé obory používají v různém smyslu a jejich význam tedy nemusí být čtenáři zřejmý, je vhodné si uvést některé definice

1.2.1 Řídká matice

Definice 1. *Řídká matice je taková matice, u které vede použití speciálních formátů neukládajících nulové prvky matice a k nim příslušných algoritmů ke zvýšení efektivity výpočtu. (Demmel, 1997)*

Užívají-li se vhodné formáty pro uložení řídké matice v paměti (např. „compressed sparse row“ či „compressed sparse column“ (Arbenz et al., 2006)), a vhodné algoritmy pro práci s ní, pak je výpočetní náročnost operací funkcí nikoli dimenze matice, ale počtu nenulových prvků matice.

Při práci s řídkými maticemi je však třeba užívat pouze ty operace, které řídkost matic neporušují – tzn. operace které nezvýší počet nenulových prvků matice. To možné operace velmi omezuje, neboť ani operace mocnění obecně nezachovává řídkost matice (např. je-li A matice, která má nenulový pouze první řádek a diagonálu, pak bude A^2 zcela zaplněná).

Pro popis řídkých matic zavedeme ještě následující termíny:

Definice 2. Struktura řídkosti *matice A je množina* $\operatorname{Pat} A$ *uspořádaných dvojic přirozených čísel, pro kterou platí* $[i, j] \in \operatorname{Pat}(a) \Leftrightarrow A_{ij} = 0$.

Pokud budeme mluvit o shodné struktuře řídkosti dvou matic, budeme tím rozumět shodnost prvků $\operatorname{Pat} A$, vzniklých nutně díky postupu konstrukce matice; „náhodně“ vzniklé nuly (tedy nuly vzniklé nikoli strukturální nutností, nýbrž např. z hlediska struktury výpočtu např. „náhodným“ odečtením dvou proti sobě působících sil) nebudeme brát v potaz.

Definice 3. Faktor zaplnění *matice f je podíl m/n^2, kde m je počet nenulových prvků a n je dimenze matice.*

[4] Tzn. v $\mathbb{R}^n$ je $\langle \mathbf{x} | \mathbf{y} \rangle = \mathbf{x}^+ \mathbf{y}$, v $\mathscr{H}$ nad $\mathcal{V}$ platí $\langle \psi | \phi \rangle = \int_{\mathcal{V}} \psi^+ \phi \, d\mathcal{V}$

1.2.2 B-ortogonalita

Definice 4. *Nechť B je lineární operátor nad vektorovým prostorem $\mathbb{V}$. Pak* $\mathrm{Ker}(B)$ *je lineární podprostor vektorů* $\mathrm{Ker}(B) = \{\mathbf{v} \in \mathbb{V} : B\mathbf{v} = \mathbf{0}\}$.

Definice 5. *Nechť B je lineární operátor s definičním oborem C^n. Pak* $\mathrm{Ker}^\perp(B)$ *je lineární podprostor vektorů* $\mathrm{Ker}^\perp(B) = \{\mathbf{v} \in \mathbb{V} : \mathbf{v} \perp \mathrm{Ker}(B)\}$.

Definice 6. *Nechť $\mathbb{T}$ je těleso a $\mathbb{V}$ vektorový prostor. Pak* hermitovská forma[5] *je zobrazení $f(V \times V) \to \mathbb{T}$, pro které platí*

1) $f(\mathbf{x}+\mathbf{y}, \mathbf{v}+\mathbf{w}) = f(\mathbf{x},\mathbf{v}) + f(\mathbf{x},\mathbf{w}) + f(\mathbf{y},\mathbf{v}) + f(\mathbf{y},\mathbf{w})$

2) $f(a\mathbf{x}, b\mathbf{y}) = \overline{a}bf(\mathbf{x},\mathbf{y})$

3) $f(\mathbf{x},\mathbf{y}) = \overline{f(\mathbf{y},\mathbf{x})}$

Definice 7. *Nechť $\mathbb{V}$ je vektorový prostor se skalárním součinem $\langle\,|\,\rangle$. Pak* B-skalární součin *vektorů $\mathbf{x}, \mathbf{y} \in \mathbb{V}$ je hermitovská forma, definovaná*

$$\langle \mathbf{x} | \mathbf{y} \rangle_B = \langle \mathbf{x} | B\mathbf{y} \rangle$$

Je snadné se přesvědčit, že pro pozitivně definitní matici B takto definovaná hermitovská forma vyhovuje definici skalárního součinu. Pro semidefinitní matice toto neplatí ($\exists \mathbf{v} \neq \emptyset : \langle \mathbf{v} | \mathbf{v} \rangle = 0$), tato hermitovská forma je však skalárním součinem na $\mathrm{Ker}^\perp(B)$.

Definice 8. *Vektory $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$ jsou* B-ortogonální *(vzhledem k matici $B \in \mathbb{C}^{n\times n}$), je-li*

$$\langle \mathbf{x} | \mathbf{y} \rangle_B = 0$$

Báze V je B-ortonormální*, je-li*

$$\langle \mathbf{v}_i | \mathbf{v}_j \rangle_B = \delta_{ij}$$

Operátor B se do podprostoru generovaného B-ortonormální bází promítne jako identita:

$$VBV^+ = I \tag{1.2.1}$$

toho lze v projekčních algoritmech na řešení zobecněného problému vlastních čísel využít k převodu zobecněného problému na problém prostý.

1.2.3 Separovaný tvar operátoru

Máme-li hermitovský lineární operátor $B \in \mathbb{C}^{n\times n}$ s $\mathrm{Dim}(\mathrm{Ker}^\perp(B)) = m$, kde $n \ll m$, může být výhodné operátor převést do *separovaného tvaru*.

Definice 9. *Lineární operátor B je* separabilní*, pokud existují vektory $\mathbf{v}_i \in \{1, \ldots, m\}$ a čísla $d_i \in \mathbb{R}$ pro které platí:*

$$B = \sum_{i=1}^{m} \mathbf{v}_i d_i \mathbf{v}_i^+ \tag{1.2.2}$$

*Pravou stranu rovnice (*1.2.2*) budeme nazývat* separovaným tvarem *operátoru.*[6]

[5] Z definice je zřejmé, že pro reálná čísla tato definice splývá s definicí symetrické bilinerní formy.

[6] Někteří autoři ve fyzice pro toto vyjádření operátoru potenciálu užívají termín *separabilní potenciál*. My zde budeme užívat výše uvedené názvosloví, neboť termín *separabilní* ponecháme pro popis schopnosti operátoru být převeden do *separovaného tvaru*.

Povšimněme si, že rovnici (1.2.2) lze zapsat maticově ve tvaru VDV^+, kde V je matice $m \times n$ a D je diagonální matice $m \times m$ s prvky d_i na diagonále.

Věta 1. *Je-li $V \in \mathbb{C}^{m\times n}$ B-ortonormální báze obrazu hermitovského lineárního operátoru B, pak $B = BV(BV)^+$*

Důkaz. Na podprostoru generovaném V platí rovnost:

$$(BV(BV)^+)V = BV(V^+B^+V) = BV(V^+BV)^+ = BVI = BV \tag{1.2.3}$$

Bázi V lze doplnit do úplné báze vektory z kernelu B. Nechť $\mathbf{k} \in \mathrm{Ker}(B)$, pak využitím hermitovskosti B:

$$BV(BV)^+\mathbf{k} = BVV^+(B^+\mathbf{k}) = \emptyset \tag{1.2.4}$$

Z linearity operátoru B a platnosti pro úplnou bázi pak plyne platnost věty. □

Pokud je operátor B reálný indefinitní, nelze vytvořit B-ortonormální bázi. Lze však vytvořit B-ortogonální bázi a operátor rozložit na

$$B = (BV)D(BV)^+ \tag{1.2.5}$$

Při práci se separovaným tvarem operátoru je vhodné (např. kvůli numerické stabilitě) jeho vektory normovat tak, aby koeficienty $d_i \in \{0, -1, -1\}$. K tomu se hodí následující definice

Definice 10. *Je-li báze $\mathbf{v}_i$ B-ortogonální a platí-li, že $\langle \mathbf{v}_i | \mathbf{v}_i \rangle_B \in -1, 0, 1$, nazvěme ji* quasi-$B$-ortonormální *bází.*

Separovaný tvar je výhodné použít v případě operátoru, jenž má malou dimenzi vzhledem k dimenzi prostoru, na kterém je definován, a přitom je jeho matice hustá: separací se získá výpočetně i paměťově úspornější reprezentace operátoru.

Jednotlivé vektory z BV definují projekci do dané báze obrazu operátoru. Známe-li tedy podprostor, ze kterého pocházejí operandy operátoru, lze z rozkladu vypustit vektory z jím ortogonálních podprostorů, což dále ušetří jak paměť, tak i výpočetní čas.

1.2.4 Potenciál

Potenciálem je myšleno pole působící na vlnové funkce: tedy pole, popisující potenciální energii dané vlnové funkce. V našem případě půjde o pole generovaná atomovými jádry a jejich elektronovým obalem, působící na vlnové funkce elektronů: např. elektrostatické pole atomových jader, či Hartreeho potenciál, popisující elektrostatické působení valenčních elektronů, atd... Pro popis polí se v kvantové fyzice používá následující formalismus:

Definice 11. Potenciál *je lineární operátor na Hilbertově prostoru.*

Definice 12. *Operátor (popř. potenciál) $P : \mathscr{H} \to \mathscr{H}$, kde $\mathscr{H} = L^2(\mathcal{V} \to \mathbb{R})$ budeme nazývat* lokální, *existuje-li funkce f_P taková, že*

$$\forall \psi \in \mathscr{H}(\mathcal{V}) : \forall x \in \mathcal{V} : P\psi(x) = f_P(x, \nabla\psi(x), \nabla\nabla\psi(x), \ldots, \nabla^m\psi(x)) \tag{1.2.6}$$

Lokální operátor tedy závisí pouze na chování funkce v daném bodě (na její hodnotě a parciálních derivací maximálně m-tého řádu).

Z linearity lokálního potenciálu V plyne pro diskretizaci jeho triviální, ale důležitá vlastnost:

Věta 2 (O zachování nuly)**.** *Je-li V lokální potenciál, pak platí*

$$\forall y \in \mathcal{U}(x) : \psi(y) = 0 \Rightarrow V\psi(x) = 0 \tag{1.2.7}$$

kde $\mathcal{U}(x)$ je nekonečně malé okolí bodu x.

Důkaz. Z linearity V

$$V\mathbf{0}(x) = V(0\mathbf{0})(x) = 0V(\mathbf{0})(x) = 0 \tag{1.2.8}$$

Veškeré hodnoty derivace jak nulové funkce $\mathbf{0}$, tak i ψ jsou v okolí bodu x nula. Proto

$$V\psi(x) = f_P(x, \nabla\psi(x), \nabla\nabla\psi(x), \ldots) = \tag{1.2.9}$$
$$= f_P(x, \nabla\mathbf{0}(x), \nabla\nabla\mathbf{0}(x), \ldots) \ = V\mathbf{0}(x) = 0 \tag{1.2.10}$$

□

Definice 13. *Operátor $P : \mathscr{H} \to \mathscr{H}$ budeme nazývat* čistě lokální*, pokud existuje $f_P : \mathcal{V} \to \mathbb{C}$ taková, že*

$$\forall \psi \in \mathscr{H} : P\psi(x) = f_P(x, \psi(x)) \tag{1.2.11}$$

Čistě lokální potenciál P pak můžeme (díky linearitě operátoru potenciálu) napsat jako součin funkcí

$$(P\psi)(x) = p(x)\psi(x) \tag{1.2.12}$$

Funkci p nazvěme *potenciálovou funkcí* operátoru P.

1.2.5 Rozklad vícedimenzionálních operátorů

Působení některých vícedimenzionálních operátorů je v jednotlivých dimenzích na sobě nezávislé. Jak dále ukážeme, může být vhodné pracovat s každou dimenzí samostatně.

Definice 14. *Nechť $\mathscr{H}_A$ a $\mathscr{H}_B$ jsou Hilbertovy prostory s bázemi A a B, pak tenzorový součin $\mathscr{H}_A \otimes \mathscr{H}_B$ je Hilbertův prostor s bází $A \times B$. Jednotlivé prvky báze budeme značit $(\alpha_i \times \beta_j)$.*

Definice 15. *Nechť $\mathscr{H}_A$ a $\mathscr{H}_B$ jsou Hilbertovy prostory s bázemi α_i a β_i. Pak pro každé dva vektory $\alpha = \sum_i a_i\alpha_i$, $\beta = \sum_j b_j\beta_j$ je definován direktní součin*

$$\alpha \circledast \beta = \sum_i \sum_j a_i b_j (\alpha_i \times \beta_j) \tag{1.2.13}$$

Definice 16. *Direktním*[7] *součinem operátorů* $P_A : \mathscr{H}_A \to \mathscr{H}_A$ *a* $P_B : \mathscr{H}_B \to \mathscr{H}_B$ *je operátor* $P_A \circledast P_B$*, pro který platí*

$$(P_A \circledast P_B)(\alpha \circledast \beta) = (P_A\alpha) \circledast (P_B\beta) \tag{1.2.14}$$

Operátor, který lze napsat jako direktní součin dvou operátorů na ortogonálních podprostorech, nazvěme rozložitelný.

Věta 3. *Je-li potenciál* P_A *čistě lokální, pak pro potenciál* $P = P_A \circledast P_B$ *platí*

$$(P(\alpha \circledast \beta))(a, b) = p_a(a)\alpha(a) \circledast (P_B\beta)(b) \tag{1.2.15}$$

Důkaz. Tvrzení lze snadno prokázat z definice lokality a direktního součinu. □

[7] Zde se názvosloví liší, v některé literatuře je tento součin označován taktéž jako tenzorový.

2. Řešení Schrödingerovy rovnice pomocí metody konečných prvků

Jak již jsme nastínili v úvodu: jádrem výpočetní materiálové fyziky jsou výpočty elektronových stavů, z nichž vyplývají fyzikální vlastnosti daného materiálu. Obvyklou úlohou je vypočítat stabilní stav daného materiálu (molekuly, krystalu, ...), tedy určit takovou polohu atomů, pro kterou materiál nabývá minimální energie.

Nově vyvíjená metoda je postavena, jak již bylo výše zmíněno, na

a) teorii funkcionálu hustoty, neboli metodě DFT – density functional theory
b) konceptu *ab-initio* selfkonsistentního (tzv. environment-reflecting) pseudopotenciálu, který umožňuje atomové jádro spolu s vnitřními elektronovými stavy popsat jedním (byť netriviálním) operátorem
c) diskretizaci *metodou konečných prvků* (FEM — finite-element method): jedné z nejužívanějších a nejpokročilejších metod pro diskretizaci diferenciálních rovnic
d) obecnosti formulace bez předpokladu o symetrii (včetně translační symetrie)

2.1 Metoda DFT pro řešení Schrödingerovy rovnice

Stav systému n elektronů je popsán řešením n-částicové Diracovy rovnice, která vychází z (relativistické) kvantové teorie pole. Relativistické efekty se ovšem prakticky projeví pouze u elektronů s vysokou kinetickou energií, tzv. vnitřních elektronů. Oddělený výpočet vnitřních elektronových stavů při konstrukci pseudopotenciálu umožňuje „skrýt“ tyto relativistické efekty do operátoru pseudopotenciálu a pro výpočet vnějších – valenčních – elektronových stavů použít nerelativistickou limitu Diracovy rovnice: rovnici Schrödingerovu.

Teorie funkcionálu hustoty (DFT — density functional theory), kterou představili Hohenberg, Kohn a Sham v roce 1965 (Hohenberg – Kohn, 1964; Kohn – Sham, 1965) ukazuje, že

– vlastnosti mnohoelektronového systému v základním stavu jsou jednoznačně determinovány jako funkcionály prostorově závislé hustoty elektronového náboje
– mnohočásticovou Schrödingerovu rovnici pro takový systém lze řešit jako systém jednoelektronových tzv. Kohn-Shamových rovnic.

DFT je v současné době nejrozšířenějším rámcem pro řešení mnohočásticové Schrödingerovy rovnice v materiálové fyzice a částečně i v kvantové chemii. Následující kapitola se pokusí ve velmi zjednodušené formě představit závěry plynoucí z teorie funkcionálu hustoty a zavést základní veličiny, které budou používány v dalších kapitolách.

2.1.1 Shrödingerova rovnice

Stav systému n-elektronů je ve stacionární Schrödingerově rovnici v tzv. x-reprezentaci (Formánek, 2004) popsán vlnovou funkcí

$$\psi(\mathbf{r}_1, \cdots, \mathbf{r}_n) : (\mathbb{R}^3)^n \to \mathbb{C} \tag{2.1.1}$$

jejíž kvadrát udává pravděpodobnost výskytu elektronu na daném místě. Při (s časem) neměnných podmínkách musí být tato funkce řešením *stacionární Schrödingerovy rovnice*

$$H\psi = \epsilon\psi \tag{2.1.2}$$

kde H je operátor energie vyjádřený jako projektor do podprostoru fyzikálně možných stavů systému o dané energii ϵ. Jeho vlastní vektory tedy odpovídají přípustným stavům systému a jeho vlastní čísla ϵ energii daného stavu. Tato rovnice není (kvůli komplikovanosti operátoru H) pro složitější systémy řešitelná. Numerické řešení této rovnice je pro přílišný počet stupňů volnosti taktéž příliš obtížné.

Dalo by se říci, že DFT v jistém smyslu pohlíží na elektrony nikoli jako na jednotlivé (nerozlišitelné) částice, nýbrž jako na „elektronový plyn“ popsaný elektronovou hustotou $\eta : \mathcal{V} \to \mathbb{R}$

$$\eta = \langle \psi | \psi \rangle = \int \dots \int \psi(\mathbf{r}_1, \dots, \mathbf{r}_n)\psi(\mathbf{r}_1, \dots, \mathbf{r}_n)^+ \, d\mathbf{r}_1 \, d\mathbf{r}_2 \dots d\mathbf{r}_n \tag{2.1.3}$$

s celkovou energií, tak zvanou *totální energií*, získanou jako funkcionál[1] elektronové hustoty součtem potenciální energie E_p a kinetické energie E_k.

$$E(\eta) = E_\mathrm{p}(\eta) + E_\mathrm{k}(\eta) \tag{2.1.4}$$

2.1.2 Kohn-Shamovy rovnice

Z teorie funkcionálu hustoty plyne, že stacionární Shrödingerovu rovnici lze pro systém n elektronů v základním[2] stavu zapsat jako soustavu jednoelektronových Kohn-Shamových rovnic pro $i \in \{1, \dots, n\}$:

$$H\psi_i = \left(-\frac{1}{2}\nabla^2 + V_\mathrm{h} + V_\mathrm{xc} + V_\mathrm{ion} \right) \psi_i = \epsilon_i \psi_i \tag{2.1.5}$$

Jejím řešením jsou vlastní vektory vyjadřující přípustné stavy elektronů v daném systému.

$$\psi_i : \mathbb{R}^3 \to \mathbb{C} \tag{2.1.6}$$

a jim příslušná vlastní čísla ϵ_i, která určují energii elektronu v daném stavu.

Funkci ψ_i můžeme napsat v goniometrickém tvaru

$$\psi_i(\mathbf{x}) = \psi_i^a(\mathbf{x})(\cos(\psi_i^f(\mathbf{x})) + i\sin(\psi_i^f(\mathbf{x}))) \tag{2.1.7}$$

kdy *amplituda* $\psi_i^a(\mathbf{x})$ je reálná funkce $\psi_i^a : \mathbb{R}^3 \to \mathbb{R}$ a *fáze* $\psi_i^f(\mathbf{x})$ je funkce do prostoru úhlů $\psi_i^f : \mathbb{R}^3 \to \langle 0, 360)$.

[1] Obecně zobrazení funkce na číslo, zde $(\mathcal{V} \to \mathbb{R}) \to \mathbb{R}$

[2] Neexcitovaném stavu, tzn. stavu, kdy mají elektrony nejnižší možnou energii.

Měřitelné fyzikální vlastnosti (výskyt elektronu, jeho energie etc.) závisí pouze amplitudě vlnové funkce, nikoli na její fázi. Je-li stav systému podmíněn pouze fyzikálně (tzn. postavením jader atomů, vnějšími silami apod.) bez předpokladů o symetrii implikujících okrajové podmínky[3], není fáze vlnové funkce jednoznačně určena

Přestože je tedy Schrödingerova rovnice komplexní, není v našem případě zapotřebí s komplexní složkou vlnových funkcí pracovat. Proto budeme ve zbytku této kapitoly pracovat s reálnými vlnovými funkcemi a výsledné diskretizované matice budou reálné. Budeme-li tedy dále mluvit o vlnové funkci ψ, ve skutečnosti tím budeme myslet její amplitudovou složku ψ^a.

2.1.3 Spin a obsazení stavů

Pauliho princip stanovuje, že dva fermiony (a tedy i elektrony) nemohou sdílet stejný stav. Stav elektronu však není určen pouze vlnovou funkcí ψ, nýbrž i tzv. spinem. Tato veličina, která vymizela při přechodu od Diracovy rovnice k rovnici Schrödingerově, může pro elektron nabývat pouze dvou hodnot: $-1/2$ a $+1/2$. Proto každý vypočtený stav z Schrödingerovy rovnice může být obsazen dvěma elektrony.

Jelikož se elektrony v základním (neexcitovaném) stavu snaží minimalizovat svoji energii, obsadí stavy v pořadí od nejnižších energií. Prvních n stavů (s přihlédnutím ke spinu: každý spočtený vlastní vektor určuje dva stavy lišící se pouze spinem) tedy tvoří neexcitovaný stav celého systému.

$$|\psi\rangle = |\psi_1\rangle|\psi_2\rangle \dots |\psi_n\rangle \tag{2.1.8}$$

Nábojovou hustotu tvoří suma kvadrátů obsazených stavů

$$\eta = \sum_{i=1}^{n} |\psi_i|^2 \tag{2.1.9}$$

a totální energii lze vyjádřit jako funkcionál vlnových funkcí

$$E_{\mathrm{k}}(\eta) = \sum_{i=1}^{n} \left\langle \psi_i \middle| -\frac{1}{2}\nabla^2\psi_i \right\rangle \tag{2.1.10}$$

$$E_{\mathrm{p}}(\eta) = \int_{\mathcal{V}} V_{\mathrm{pot}}(r)\,\eta(r)\;\; dr \tag{2.1.11}$$

kde $V_{\mathrm{pot}} : \mathcal{V} \to \mathbb{R}$ je funkcionál potenciální energie.

Odvození Kohn-Shamovy rovnice (2.1.5) a její přesnější formulaci lze nalézt např. v (Dreizler – Gross, 1990; Martin, 2004), výtah s ohledem na aplikaci DFT v naší metodě v (Čertík, 2008), zde pouze zmíníme význam jednotlivých operátorů.

[3] V případě periodických struktur (např. nekonečný krystal) se zpravidla počítá jedna buňka symetrie. Symetrie se pak vynucuje okrajovými podmínkami, kdy každá strana buňky musí „navazovat“ na stranu protilehlou. V tomto případě musí navazovat nejen amplituda, ale i fáze. Periodické materiály se tedy (v takovémto modelu) musí počítat v oboru komplexních čísel.

2.1.4 Operátory Kohn-Shamovy rovnice a jejich vlastnosti

Laplaceův operátor $-\frac{1}{2}\nabla^2$ odpovídá kinetické energii částice. Z jeho definice lze snadno ukázat, že je to hermitovský lokální potenciál.

Hartreeho potenciál je potenciál elektrostatické interakce mezi elektrony.

Nechť E_h je funkcionál potenciální energie dané elektrostatickými silami, tj.

$$E_\mathrm{h}(\eta) = \frac{1}{2}\int_{\mathcal{V}}\int_{\mathcal{V}} \frac{\eta(r)\eta(r')}{|r-r'|}\,dr\,dr'. \tag{2.1.12}$$

Pak lze k ní příslušný potenciál vyjádřit jako derivaci této energie podle nábojové hustoty

$$V_\mathrm{h}(r) = \frac{\delta E_h}{\delta\eta(r)} = \frac{1}{2}\int_{\mathcal{V}} \frac{\eta(r')}{|r-r'|}\,dr' \tag{2.1.13}$$

či jako řešení Poissonovy rovnice

$$-\frac{1}{4\pi}\nabla^2 V_\mathrm{h} = \eta \tag{2.1.14}$$

Tento potenciál je rovněž lokální a hermitovský.

Výměnný a korelační potenciál Přechod od mnohočásticové vlnové funkce k jednoelektronovým vlnovým funkcím vyžaduje zavedení členů popisujících vzájemné ovlivňování částic. Zavádíme výše zmíněný Hartreeho potenciál pro elektrostatické síly a *výměnný a korelační potenciál* pro všechna ostatní působení částic (např. Pauliho vylučovací princip).

Přesný tvar V_xc vyjádřený jako funkce elektronové hustoty není znám, jsou známy pouze některé jeho vlastnosti a požadavky, které musí splňovat. Používají se různé aproximace, jichž je více typů. V této práci je užíván potenciál, který je výsledkem tzv. *Local density aproximation*, přesnější rozbor lze nalézt v (Čertík, 2008).

Potenciál atomových jader vyjádřený funkcí $V_\mathrm{ion} : \mathbb{R}^3 \to \mathbb{R}$ je superpozicí elektrostatických potenciálů M atomových jader se středy v R_j a náboji Z_j

$$V_\mathrm{ion}(\mathbf{x}) = \sum_{j=1}^{M} V_\mathrm{ion}^{[}(j)](\mathbf{x}-\mathbf{R}_j) \tag{2.1.15}$$

kdy elektrostatický potenciál atomového jádra je sféricky symetrický coloumbický potenciál

$$V_{\mathrm{ion}(j)}(\mathbf{x}) = \frac{Z_j}{|\mathbf{x}|} \tag{2.1.16}$$

2.1.5 Konvergence metody

Některé složky operátoru H jsou závislé na elektronové hustotě. Ta je na počátku výpočtu neznámá, jako startovací hodnota se zpravidla užívá superpozice nábojových hustot atomů nebo i konstatntní nábojová hustota. V každém kroku výpočtu se pak vypočte potřebný počet nejnižších (dle energie) možných stavů elektronů a z těchto stavů se spočte nová elektronová hustota. Výpočet skončí v okamžiku, kdy elektronová hustota zkonverguje, tedy, kdy

$$|\eta_{i+1} - \eta_i| < e \tag{2.1.17}$$

Tento postup se nazývá *(selfkonvergentní) DFT cyklus.*

2.1.6 Rozmazání (smearing)

V průběhu DFT cyklu se stává, že energie dvou sousedních elektronových stavů se mění tak, že tyto dva stavy mění pořadí. Pokud jde o stavy na hranici „obsazenosti“, tj. tyto dva stavy se střídají v roli obsazeného a neobsazeného stavu, dochází k problémům s konvergencí.

Proto se zavádí tzv. *rozmazání*: místo pevného obsazení n stavů se určí rozmezí energií $(\epsilon_{\min}, \epsilon_{\max})$: stavy s energií nižší než $\epsilon_{\min}$ se považují za obsazené, stavy s energií vyšší než $\epsilon_{\max}$ za prázdné. Stavy s energií mezi těmito hodnotami se pak obsadí pouze částečně. Pro tyto účely se zavádí spojitá váhová funkce

$$\text{weight}(k) : \mathbb{N} \to \mathbb{R} : \text{weight}(k) \begin{cases} = m & \epsilon_{\min} >= \epsilon_k \\ \in (0, m) & \epsilon_{\min} < \epsilon_k < \epsilon_{\max} \\ = 0 & \epsilon_k >= \epsilon_{\max} \end{cases} \tag{2.1.18}$$

kde m je maximální počet elektronů, které mohou sdílet daný stav: tzn. ve výpočtu nezahrnujícím spin $m = 2$. Střed[4] rozmezí $(\epsilon_{\min}, \epsilon_{\max})$ se nazývá *Fermiho mez*. Pokud by nebylo použito rozmazání, byly by všechny stavy s energií nižší než Fermiho mez obsazeny a všechny stavy s energií vyšší prázdné. Vhodná šířka rozmezí i tvar váhové funkce v tomto rozmezí závisí na počítaném problému. Smearing může odrážet i fyzikální realitu nenulové elektronové teploty: pak by měl vycházet z Fermi-Diracova (či v aproximaci Gaussova) rozdělení. Exaktnější rozbor smearingu a posupy, kterak stanovit jednotlivé parametry pro rozmazávání, lze nalézt např. v (Marzari, 1996).

Užije-li se rozmazání, spočte se elektronová hustota pomocí

$$\eta_{i+1} = \sum_{i=1}^{k} \text{weight}(k) |\psi_j^i|^2 \tag{2.1.19}$$

2.1.7 Algoritmus DFT cyklu

V každé iteraci DFT cyklu se nejprve spočtou z aktuální nábojové hodnoty nový tvar operátorů Kohn-Shamových rovnic; ta se následně vyřeší. Kvůli rozmazávání je třeba v každém kroku cyklu počítat více elektronových stavů, než kolik je v systému elektronů (respektive (při bezspinovém výpočtu) než polovina

[4] Přesněji bod zlomu skokové funkce mající shodný integrál s váhovou funkcí.

elektronů), nutný minimální počet elektronů je třeba. Po výpočtu elektronových stavů se určí nová Fermiho mez, rozmazání a nábojová hustota. Algoritmus cyklu tedy vypadá takto:

Algoritmus 2.1.1: DFT cyklus

1: Polož počáteční hustotu η_0
2: **repeat** $i = 1, 2 \ldots$
3: Vyřeš $-\frac{1}{4\pi}\nabla^2 V_{\mathrm{h}} = \eta_i$
4: Spočti V_{xc}
5: Vyřeš k řešení ψ_j^i Kohn-Shamovy rovnice (2.1.5)
6: Urči Fermiho mez a rozmazání
7: Spočti novou nábojovou hustotu dle rovnice (2.1.19)
8: **until** $\eta_{i+1} \neq \eta_i$

Užité operátory se během výpočtu mění, nejde tedy o pravou Newton-Raphsonovu metodu – proto není konvergence zaručena. Praxe však ukazuje, že při použití vhodného mixování výpočet zpravidla zkonverguje. Názory na nejvhodnější mixování se liší, při našich experimentech se ukázalo jako nejvýhodnější Broydenovo mixování (Čertík, 2008), v případě složitějších výpočtů pak mixování Andersonovo.

2.2 Pseudopotenciály

Kohn-Shamovu rovnici je třeba vyřešit pro n elektronů - tedy je třeba získat n nejnižších vlastních čísel Kohn-Shamovy rovnice (2.1.5). Jelikož jsou matice vzniklé diskretizací této rovnice řídké (jak ukážeme v kapitole 2.4), je velmi výhodné pro řešení vlastních čísel užít iterační algoritmy, které tuto řídkost umí velmi dobře využít a zvládají i matice větších dimenzí. Tyto řešiče však nejsou schopny najednou získat všechna vlastní čísla matice: výpočetní i paměťová náročnost těchto algoritmů je přímo úměrná počtu vyžadovaných vlastních čísel. Proto je vhodné počet požadovaných vlastních čísel omezit, což je jeden z důvodů (i když zdaleka ne jediný), pro použití *pseudopotenciálů*.

Elektrony lze rozdělit na dva typy: *vnitřní* a *valenční*. Do chemických vazeb prakticky vstupují pouze valenční elektrony, tedy elektrony z nejvyšších (dle energie) stavů. Těchto elektronů je oproti vnitřním menšina (s výjimkou lehkých atomů).

Vnitřní elektrony jsou na daleko nižších energetických hladinách a jejich vlnové funkce jsou lokalizované. Proto prakticky neinteragují (s výjimkou elektrostatické síly) s elektrony ostatních atomů. Naopak valenční elektrony se účastní chemických vazeb, jsou odpovědné za síly mezi atomy a jejich prostorové rozložení se v důsledku vazeb k ostatním atomům v okolí podstatně mění.

Kdybychom chtěli v Kohn-Shamových rovnicích počítat se všemi elektrony, narazili bychom při diskretizaci na několik vážných problémů:

– Vnitřní elektrony podstatně zvyšují počet stupňů volnosti.

– Energie vnitřních elektronů je v porovnání s valenčními velmi vysoká. Sledujeme-li změny celkové energie při interakci valenčních energií, musíme porovnávat drobné změny velkých čísel. To vyžaduje velmi přesný výpočet.

– Energie vnitřních elektronů vyžaduje v případě těžších atomů relativistický výpočet, protože relativistické efekty jsou nezanedbatelné. Relativistický výpočet by vyžadoval diskretizaci čtyřkomponentové Diracovy rovnice, což se v praxi ukazuje jako příliš složitý problém.

– Elektrostatické potenciály V_{ion} mají v jádře atomu singularitu, což vede k numerickým problémům.

– Tvar potenciálů v blízkosti jader je velmi strmý, k zachycení potenciálu je třeba velmi jemná síť.

Tyto těžkosti lze odstranit užitím *pseudopotenciálů*, kdy se ke každému atomovému jádru přiřadí i jemu příslušné vnitřní elektrony a na vzniklé „superjádro“ se pohlíží jako na jeden objekt reprezentovaný jedním lineárním operátorem. V Kohn-Shamových rovnicích (2.1.5) se tedy operátory V_{ion}^n nahradí operátorem $V_{\text{ion}}^{`n}$.

$$H\psi_i = \left(-\frac{1}{2}\nabla^2 + V_{\text{h}} + V_{\text{xc}} + \sum V_{\text{ion}}^{`n}\right)\psi_i = \epsilon_i\psi_i \tag{2.2.1}$$

Výroba pseudopotenciálu

Pseudopotenciál $V_{\text{ion}}^{`n}$ je konstruován tak, aby jeho účinek na valenční elektrony byl ekvivalentní (s požadovanou přesností, zpravidla do prvního řádu v závislosti na energii) účinku atomového jádra a vnitřních elektronů. Ve vzdálenější oblasti od atomového jádra (za hraniční vzdáleností R_c od jádra), kam vnitřní elektrony zasahují pouze elektrostatickými silami je shodný s původním tzv. „*all-electron*“ potenciálem.

Jelikož v blízkosti jádra jsou interakce elektronů s jádrem řádově větší než jakákoli jiná interakce, je chování vnitřních elektronů prakticky determinováno potenciálem jádra, který je sféricky symetrický; jejich nábojová hustota je taktéž sféricky symetrická.

Jádro s vnitřními elektrony lze tedy popsat pomocí sféricky symetrického pseudopotenciálu.

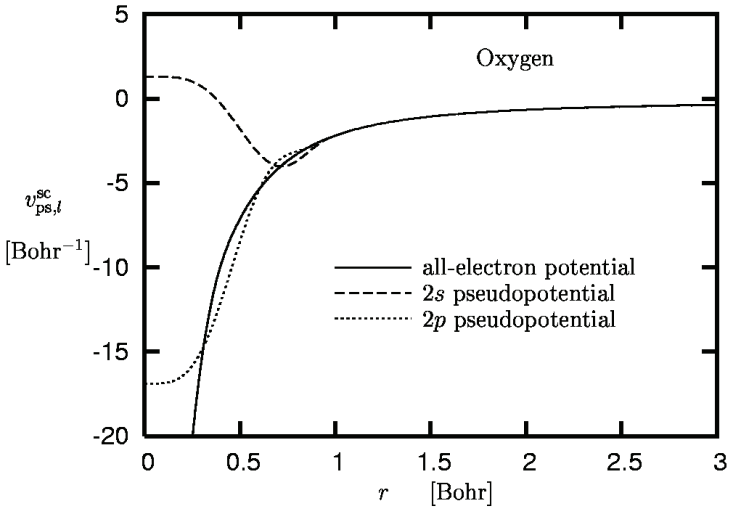
Uvnitř poloměru R_c je možné tvar pseudopotenciálu při dodržení určitých podmínek volit. Lze ho tedy uzpůsobit tak, aby byl vhodný pro numerické výpočty: tedy odstranit singularitu v místě jádra a odstranit strmé gradienty.

Takto vytvořený pseudopotenciál však není „univerzální“, daní za vhodnost k numerickým výpočtům je zaprvé to, že takto vytvořený pseudopotenciál dostatečně přesně aproximuje chování all-electron potenciálu pouze pro určité konfigurace a jim příslušné energie valenčních stavů (o funkčnosti pseudopotenciálu ve více či méně odlišných okolních prostředích a elektronových konfiguracích atomu se mluví jako o *transferabilitě*), zadruhé je takový operátor nutně nelokální.

Exaktnější formulaci pseudopotenciálů a základní postupy při jejich konstrukci lze nalézt např v (Pickett, 1989; Kleinman – Bylander, 1982).

Pseudopotenciál $V_{\text{ion}}^{`n}$ je tedy nelokální, lze však rozdělit na dalekodosahový lokální člen V_{loc},[5] jehož asymptotický tvar charakterizuje elektrostatické působení

[5]Dále, nebude-li třeba, nebudeme pro jednoduchost uvádět index n.



Obrázek 2.1: Příklad pseudopotenciálů pro kyslík

jádra a vnitřních elektronů,[6] a krátkodosahové nelokální složky V_{ps}.

$$V_{\text{ion}}^{`} = V_{\text{loc}} + V_{\text{ps}} \tag{2.2.2}$$

L-dependent pseudopotenciál

Pro vhodně zkonstruovaný pseudopotenciál, užije-li se sférická báze $Y_{l,m}$ složená ze *sférických harmonických funkcí* (Formánek, 2004), lze nelokální část l-dependent (pseudo)potenciálu rozložit na sumu direktních součinů čistě *lokální radiální* a nelokální úhlové složky:

$$V_{\text{ps}} = \sum_{l=0}^{\infty} V_{\text{ps}}^{l} = \sum_{l=0}^{\infty} \left(V_l \circledast \sum_{m=-l}^{l} Y_{l,m}(Y_{l,m})^{+} \right) \tag{2.2.3}$$

Operátor $\sum_{m=-l}^{l} Y_{l,m} Y_{l,m}^{+}$ působí na Hilbertově prostoru úhlů $L^2(\langle 0, 360\rangle \times \langle 0, 180\rangle)$ a je nelokální.

V_{l} je *čistě lokální* operátor definovaný na „jednorozměrném“ radiálním Hilbertově prostoru $L^2(\mathbb{R})$. Je různý pro elektrony s různým úhlovým momentem, vyjádřeným orbitálním kvantovým číslem l, proto se nazývá *l-dependent*.

Většina pseudopotenciálů, používaných při výpočtu elektronové struktury je převeditelných do l-dependent tvaru. Tyto potenciály se pak liší konstrukcí radiální *potenciálové funkce* V_{l} definující *čistě lokální* potenciál V_{l}. Nyní při výpočtech užíváme pseudopotenciály vypočtené dle (vac).

Jelikož vnitřní elektrony atomu jsou striktně lokalizované v blízkosti jeho jádra, jejich působení je prostorově omezené. V_{ps} a tedy i jeho radiální složky V_l jsou tedy krátkodosahové. Díky sférické symetrii V_{ps} [7] se středem symetrie v jádře atomu jsou sféricky symetrické i jeho l-složky V_{ps}^{l} (nikoli však už $V_{\text{l}} \circledast Y_{l,m}$).

Pro *l-dependent* pseudopotenciál se rovněž užívá označení *semilokální*, neboť jej lze rozložit do direktního součinu lokální a nelokální složky. Tím se liší od

[6] Tedy defakto elektrostatické působení patřičného iontu.

[7] Nebo z opačného pohledu díky sférické symetrii $\sum_{m=-1}^{l} Y_{l,m}$.

separabilních, plně nelokálních pseudopotenciálů, které jsou nelokální i v radiální komponentě. Tuto jejich radiální komponentu pak tvoří rozvoj do podprostorů odpovídajících projektorům generovaným pomocí radiální báze.

2.3 Diskretizace metodou konečných prvků

Metodu konečných prvků (*finite element method*, dále *FEM*) byla zvolena pro následující výhody:

obecnost Některé metody nutí vlnové funkce do předem připravených „tvarů“: nemají obecnou bázi, nýbrž bázi tvořenou rozkladem aproximace řešení (např. atomových funkcí apod.). Pokud je tento odhad špatný, tyto metody buď nekonvergují, nebo konvergují ke špatnému řešení. FEM má obecnou bázi, a tudíž tímto neduhem netrpí. Proto je tato metoda aplikovatelná na libovolný problém.

prověřenost FEM je prověřená metoda s teoreticky prověřenou konvergencí. Zároveň je to progresivní metoda na jejímž vývoji se dále pracuje (nelineární báze, adaptabilita), což zaručuje další možnosti k vylepšení konvergence a rychlosti.

škálovatelnost Volbou sítě je možné řídit náročnost výpočtu i přesnost v jednotlivých bodech prostoru, v případě větších dimenzí lze metodu snadno paralelizovat pomocí vhodného řešiče.

dostupnost Je na výběr několik kvalitních kódů vhodných pro implementaci.

2.3.1 Báze konečných prvků

FEM spočívá v aproximaci Hilbertova prostoru $\mathscr{H}$ nad prostorem $\mathcal{V}$ vektorovým prostorem konečných prvků Ω. Tento prostor je zkonstruován následujícím způsobem:[8]

Definice 17. *Nechť* $\mathbf{M}$ *je množina bodů v* R^n *Konvexní obal množiny* $\mathrm{conv}(\mathbf{M}) \subset \mathcal{V}$ *je množina splňující*

1) $\mathbf{M} \subset \mathrm{conv}(\mathbf{M})$

2) $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n \in \mathrm{conv}(\mathbf{M}) \Rightarrow \mathbf{x} + (\mathbf{y} - \mathbf{x}) * i \in \mathrm{conv}(\mathbf{M}), \quad kde\ i \in \langle 0, 1 \rangle.$

Definice 18. Mnohostěn $\mathrm{conv}(\mathbf{M})$ *na prostoru* $\mathcal{V}$ *je konvexní obal konečné množiny* $\mathbf{M}$. *Prvky* $\mathbf{v}_i$ *množiny* $\mathbf{M}$ *se nazývají* vrcholy mnohostěnu.

Definice 19. *Mnohostěn* $\mathcal{M}$ *je* regulární, *pokud:*

- $\dim \mathcal{M} = \dim V$
- *Neobsahuje zbytečný vrchol:* $\forall \mathbf{v}_i \in \mathbf{M} : \mathrm{conv}(\mathbf{M}) \neq \mathrm{conv}(\mathbf{M} \setminus \mathbf{v}_i)$

Definice 20. Síť[9] $\{\mathcal{M}_m\}$ *pokrývající prostor* $\mathcal{V}$ *je množina* m *regulárních mnohostěnů* $\{\mathcal{M}_i : i \in 1, dots, m\}$ *s celkem* n *vrcholy* $\mathbf{v}_j$, *které (Frey – George, 2000)*

1) Pokrývají celý prostor: $\bigcup_{i=1}^{n} \mathcal{M}_i = \mathcal{V}$

[8] Jde o poněkud zjednodušenou definici.

[9] Velmi často se síť označuje anglickým slovem *mesh*.

2) Mají společné vrcholy: $\forall i, j : \mathbf{v}_j \in \mathcal{M}_i \Rightarrow \mathbf{v}_j$ *je vrchol* $\mathcal{M}_i$

3) Průnik vnitřků mnohostěnů je prázdný:

$$\forall i, j \in \{1 \dots m\}: Int(\mathcal{M}_i) \cap Int(\mathcal{M}_j) = \emptyset$$

V trojdimenzionálním prostoru se nejčastěji užívá síť složená z čtyřstěnů či krychlí. O množství elementů sítě se někdy mluví jako o *jemnosti* či *hrubosti* sítě.[10].

Definice 21. Společná hranice *mnohostěnů* $\mathcal{M}_i$ *a* $\mathcal{M}_j$ *je průnik* $\mathcal{M}_i \bigcap \mathcal{M}_j$. *Mnohostěny jsou* sousední*, pokud mají neprázdnou společnou hranici. Vrcholy* $\mathbf{v}_i$ *a* $\mathbf{v}_j$ *jsou sousední, pokud* $\exists \mathcal{M}_k \in \{\mathcal{M}_m\} : \mathbf{v}_i \in \mathcal{M}_k \ \ \& \ \ \mathbf{v}_j \in \mathcal{M}_k$.

Lze ukázat, že mnohostěny jsou *sousední*, pokud mají alespoň jeden společný vrchol.

Konečný prvek

Definice 22. Konečný prvek *je trojice* $(\mathcal{M}, \Pi, \Sigma)$ *kde (Braess, 2007):*

- $\mathcal{M}$ *je regulární mnohostěn.*
- Π *je lineární podprostor prostoru spojitých funkcí nad* $\mathcal{M}$ *s konečnou bází* φ_i *dimenze s.*
- Σ *je množina s funkcionálů* Σ_i *nad* Π*, jednoznačně identifikující prvky* Π.

V běžně užívaných implementacích konečných prvků je Π prostorem polynomů stupně maximálně k jednoznačně určených svými hodnotam (popř. pro konečné prvky užívané k řešení diferenciálních rovnic vyšších řádů i svými derivacemi) v pevně daných bodech. Funkcionál Σ_i je pak definován jako hodnota či některá parciální derivace v daném bodě (dále budeme pro jednoduchost toto o funkcionálech Σ_i předpokládat).

Podle maximálního stupně k pak mluvíme o *lineárních*, *kvadratických* či *kubických* (konečných) prvcích. V případě lineárních prvků se zpravidla vyčíslují hodnoty ve vrcholech mnohostěnu a bází konečných prvků jsou tzv. *kloboukové* funkce, které mají vždy právě v jednom vrcholu hodnotu jedna, zatímco v ostatních jsou nulové.

Nebude-li hrozit záměna, budeme končený prvek identifikovat pomocí jemu příslušného mnohostěnu, tzn. vrchol konečného prvku označuje vrchol mnohostěnu příslušnému k danému prvku atd...

Prostor konečných prvků a jeho báze

Následující tvrzení budou podána poněkud zjednodušeně, exaktnější definice lze nalézt v (Ciarlet – Lions, 1991). Je-li dána síť $\{\mathcal{M}_m\}$, pak lze nad každým mnohostěnem vybudovat konečný prvek $(\mathcal{M}_i, \Pi_i, \Sigma_i)$. Jsou-li dva konečné prvky sousední, pak je možné „ztotožnit“ funkcionály, které se vyhodnocují na společné hranici prvků. Tím získáme množinu funkcionálů Σ_j definující prvky prostoru konečných prvků Ω.

U lineárních konečných elementů nyní používaných naším programem je prostor Ω_l po částech lineárních funkcí určených svými hodnotami ve vrcholech sítě a Σ_j je množina funkcionálů přiřazujících prvkům Ω_l hodnotu v j-tém vrcholu sítě.

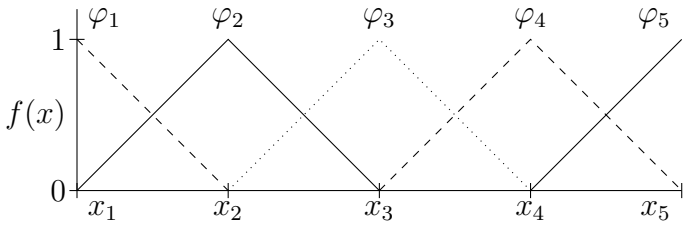
[10]Teorii konečných prvků lze formulovat i tak, že povoluje i „zakřivené“ elementy sítě (Ciarlet – Lions, 1991)

Definice 23. *Báze konečných prvků pro danou síť a nad ní vybudovaný prostor konečných prvků Ω se sadou funkcionálů Σ_j je množina funkcí φ_i, pro kterou platí:*

$$\forall i, j : \Sigma_j(\varphi_i) = \delta_{i,j} \tag{2.3.1}$$

Prostor konečných prvků Ω je pak lineárním obalem těchto bázových funkcí.

Obrázek 2.2: Příklad jednodimenzionální lineární báze konečných prvků φ_i



Příslušné funkcionály Π_i jsou definované jako

$$\Pi_i(f) = f(x_i)$$

Síť se sestává z vrcholů: x_i a mnohostěnů $\{i \in \{1,2,3,4\} : \{x_i, x_{i+1}\}\}$.

Bázi prostoru konečných prvků Ω_l tvoří tzv. kloboukové funkce: pro každý vrchol sítě existuje právě jedna bázová funkce, která v daném vrcholu má hodnotu 1 a lineárně klesá k sousedním vrcholům sítě, na zbytku prostoru je nulová (viz obrázek 2.2).

Definice 24. *Bázové funkce φ_i, φ_j jsou* sousední*, pokud existuje $\mathbf{x} \in \mathcal{V} : \varphi_i(\mathbf{x}) \neq 0 \ \& \ \varphi_j(\mathbf{x}) \neq 0$.*

Je zřejmé, že dvě bázové funkce jsou sousední právě tehdy, pokud jsou definované funkcionály Σ_i a Σ_j vyhodnocovanými v bodech $\mathbf{x}_i$ a $\mathbf{x}_j$, pokud existuje mnohostěn $\mathcal{M}_k \in \{\mathcal{M}_m\} : \mathbf{x}_i \in \mathcal{M}_i \ \& \ \mathbf{x}_j \in \mathcal{M}_k$.

2.3.2 Slabá forma Kohn-Shamovy rovnice

Zavedeme-li potenciál $V = V_\mathrm{h} + V_\mathrm{xc} + \sum V_\mathrm{ion}^{`}$ je možno Kohn-Shamovu rovnici (2.1.5) zapsat jako

$$\left(-\frac{1}{2}\nabla^2 + V\right)\psi = \epsilon\psi \tag{2.3.2}$$

přičemž $\psi \in L^2(\mathcal{V})$.

Přenásobením *testovací funkcí* $\varphi \in L^2(\mathcal{V})$ a zintegrováním přes počítaný prostor převedeme rovnici do tzv. *slabého* či *integrálního* tvaru.

$$\int \left(-\frac{1}{2}\nabla^2\psi\varphi + V\psi\varphi\right)\, d\mathcal{V} = \int \epsilon\psi\varphi \, d\mathcal{V} \tag{2.3.3}$$

Slabý tvar rovnice je ekvivalentní se silným, tzn. ψ je řešením (2.3.2) $\Leftrightarrow$ $\forall\varphi \in L^2(\mathcal{V})$ platí rovnice (2.3.3).

Použitím první Greenovy identity

$$(\nabla^2\psi)\varphi = (\nabla\psi)(\nabla\varphi) - \nabla((\nabla\psi)\varphi) \tag{2.3.4}$$

rozvineme první člen rovnice (2.3.3)

$$\int \left(-\frac{1}{2}(\nabla\psi)(\nabla\varphi) + (V\psi)\varphi\right)\, d\mathcal{V} - \int \frac{1}{2}\nabla((\nabla\psi)\varphi)\, d\mathcal{V} = \int \epsilon\psi\varphi\, d\mathcal{V} \tag{2.3.5}$$

aplikujeme Gaussovu větu

$$\int \left(-\frac{1}{2}(\nabla\psi)(\nabla\varphi) + (V\psi)\varphi\right)\, d\mathcal{V} - \int \frac{1}{2}(\nabla\psi)\varphi\, r\, d\mathcal{S} = \int \epsilon\psi\varphi\, d\mathcal{V} \tag{2.3.6}$$

a užitím vztahu $(\nabla\psi)r = \dfrac{d\psi}{dr}$ získáme konečný tvar rovnice

$$\int \left(-\frac{1}{2}(\nabla\psi)(\nabla\varphi) + (V\psi)\varphi\right)\, d\mathcal{V} - \int \frac{1}{2}\frac{d\psi}{dr}\, d\mathcal{S} = \int \epsilon\psi\varphi\, d\mathcal{V} \tag{2.3.7}$$

kde $\mathcal{S}$ označuje hranici $\mathcal{V}$.

2.3.3 Diskretizace Kohn-Shamovy rovnice

Diskretizace spočívá v projekci rovnice do prostoru Ω a tedy v aproximaci elektronové vlnové funkce ψ lineární kombinací báze konečných prvků.

$$\psi = \sum x_j\varphi_j \tag{2.3.8}$$

Hilbertův prostor a operátory na něm definované jsou lineární; tato linearita se zachová i po projekci do prostoru Ω. Proto není třeba řešení počítat pro libovolnou testovací funkci, nýbrž stačí zaručit platnost pro všechny prvky báze konečných prvků φ_i. Je tedy třeba vyřešit následující soustavu i rovnic pro neznámé

$$\sum_j \left(\int -\frac{1}{2}(\nabla\varphi_j)(\nabla\varphi_i)d\mathcal{V}\right) x_j + \sum_j \left(\int V\varphi_j\varphi_i\, d\mathcal{V}\right) x_j - \sum_j \left(\int \frac{1}{2}\frac{d\varphi_j}{dr}\, d\mathcal{S}\right) x_j = \sum_j \left(\int \epsilon\varphi_j\varphi_i\, d\mathcal{V}\right) x_j \tag{2.3.9}$$

Tuto rovnici lze napsat i v jednodušším maticovém tvaru

$$K\mathbf{x} + V\mathbf{x} - F\mathbf{x} = \epsilon M\mathbf{x} \tag{2.3.10}$$

kde jednotlivé operátory jsou dány následujícími maticemi

$$K_{ij} = \int -\frac{1}{2}(\nabla\varphi_j)(\nabla\varphi_i)\, d\mathcal{V} \tag{2.3.11}$$

$$V_{ij} = \int (V\varphi_j)\varphi_i\, d\mathcal{V} \tag{2.3.12}$$

$$M_{ij} = \int \varphi_j\varphi_i\, d\mathcal{V} \tag{2.3.13}$$

$$F_{ij} = \int \frac{1}{2}\frac{d\varphi_j}{dr}\, d\mathcal{S} \tag{2.3.14}$$

Jelikož elektrony jsou zpravidla lokalizované kolem jader a při vytvoření dostatečné oblasti jsou tedy jejich vlnové funkce na hranici nulové, je nulový i plošný integrál

$$\int \frac{1}{2} \frac{d\varphi_j}{dr} \, d\mathcal{S} \tag{2.3.15}$$

a matici F_{ij} lze při volbě vhodné oblasti z dalšího výpočtu vynechat. Aplikací metody FEM na Kohn-Shamovy rovnice jsme tedy problém převedli na zobecněný problém vlastních čísel.

Numerická integrace

Převedením problému do slabé formy a projekcí do báze konečných prvků jsme se nezbavili nutnosti integrace, pouze ji omezili na dostatečně malý prostor, neboť stačí spočíst integrály přes jednotlivé prvky báze. Vzhledem k lokalitě a linearitě báze lze tyto integrály počítat numericky.

Numerická integrace funkce f se provádí Gaussovou integrací: zvolí se integrační body $\mathbf{x}_n$ a jejich váhy w_n tak, aby součet vah integračních bodů v každém elementu byl roven číslu s. Pro numerickou integraci na elementu $\mathcal{M}_j$ pak platí následující vztah:

$$\int_{\mathcal{M}_j} f(\mathbf{x}) \, d\mathbf{x} \approx \left(\sum_{i=1}^{n} w_n f(\mathbf{x}_i) \right) J(\mathcal{M}_j) \tag{2.3.16}$$

kde $J(\mathcal{M}_j)$ je objem mnohostěnu $\mathcal{M}_j$, popsaný jako jakobián transformace mezi jednotkovým referenčním mnohostěnem o objemu s a elementem $\mathcal{M}_j$. Při vhodné volbě bodů je numerická integrace přesná pro polynomy stupně $2n - 1$. Přesnější vyjádření lze nalézt v (Dhatt – Touzot, 1984).

Přesnost aproximace metodou konečných prvků

Přesnost metody konečných prvků závisí na několika parametrech. Bližší rozbor je příliš vzdálen od tématu této práce, a tak se omezíme pouze na stručné vyjmenování nejdůležitějších parametrů.

Přesnost numerické integrace Přesnost lze zvyšovat volbou většího množství integračních bodů (či volbou některé pokročilejší metody). Vylepšování numerické integrace sice nenabízí tak velké možnosti pro zlepšení přesnosti jako následující metody, numerická integrace však zabírá (oproti ostatním částem výpočtu) minimální čas, a tak se její prodloužení prakticky neprojeví na době výpočtu.

Volba sítě Zjemnění sítě vede k lepší aproximaci funkcí a k zvýšení přesnosti. Nejlepší výsledky lze získat pomocí *adaptabilních* sítí, kdy se zjemňuje pouze ta část sítě, kde je aproximace základní sítí příliš hrubá. Zjemění sítě vede k podstatnému nárůstu přesnosti, zároveň však i k podstatnému nárůstu dimenze výsledného problému vlastních čísel a tedy i zvýšení časových i paměťových nároků na výpočet.

Volba báze konečných prvků S růstem dimenze báze konečného prvku podstatně roste přesnost výpočtu (Braess, 2007). Daní za to je nejen nárůst dimenze báze konečných prvků, ale i zvýšení faktoru zaplnění výsledných řídkých matic. Jak ukazují výsledky příbuzných metod (Pask et al., 2009)[11], zisk ze zvýšení stupně polynomů daleko převyšuje zmíněné nevýhody. Tato možnost nabízí největší možnosti pro zpřesnění výpočtu, proto v současné době probíhá implementace kvadratických prvků do naší metody.

2.4 Struktura matice a řešení problému vlastních čísel

U běžně počítaných problémů je dimenze matic z rovnice (2.4.1) příliš velká na to, aby ji bylo možno v rozumném čase spočíst pomocí algoritmů pro husté matice. Proto je třeba užít algoritmů pro řídké matice a tedy zajistit, aby všechny matice figurující ve výpočtu byly řídké.

Nyní tedy budeme analyzovat operátory v rovnici (2.3.9), přičemž naší snahou bude buďto ukázat, že jsou lokální a tedy jimi generovaná matice je řídká, nebo je převést do jiného výpočetně úsporného tvaru.

Věta 4. *Jsou-li operátory O, P lineární lokální, matice $A = \int (O\varphi_i)(P\varphi_j)^+ d\mathcal{V}$ reprezentující operátor v bázi konečných prvků bude mít prvek a_{ij} nenulový, pouze pokud bázové funkce φ_i a φ_j jsou sousední.*

Důkaz. Z věty *o zachování nuly* plyne, že $(O\varphi_i)(P\varphi_j) \neq 0$ pouze v bodě, v jehož nekonečně malém okolí jsou bázové funkce nenulové.

Pokud φ_i a φ_j nejsou sousední, nemají bázové funkce $O\varphi_i$ a $P\varphi_j$ společný nenulový bod, nanejvýš se tedy mohou „dotýkat“. Všude mimo body dotyku je alespoň jedna bázová funkce na okolí bodu nulová a tvrzení tak okamžitě plyne z věty 2.4. Množina bodů dotyku má míru nula a tedy hodnotu integrálu neovlivní. □

Za poznámku stojí, že numerická integrace tuto vlastnost nepokazí, pokud jsou bázové funkce na svých „hranicích“ nulové,[12] či pokud se integrační body nevolí z hraničních bodů bázových funkcí.

Při diskretizaci diferenciálních matic je velmi vhodné udržet všechny operátory lokální, neboť okamžitým důsledkem věty 4 je, že matice lokálního operátoru v metodě FEM je *řídká* (v případě diskretizace jednorozměrného prostoru dokonce tridiagonální). Počet nenulových prvků řídké matice vzniklé z aplikace FEM na lokální potenciály roste s počtem konečných prvků, tedy s dimenzí matice, zatímco v případě nelokálního potenciálu vzniká obecná matice, kde počet jejích nenulových prvků roste s kvadrátem její dimenze.

[11] Tato v citaci zmíněná metoda (cílená na periodické struktury) zavádí také zajímavé rozšíření konečných prvků, kdy k polynomiální bázi konečných prvků přidá funkce vytvořené z atomových funkcí, které přesněji aproximují potenciál atomového jádra, čímž vznikne neortogonální báze konečných prvků.

[12] To klasická báze konečných prvků splňuje, ovšem některé speciální (Pask et al., 2009) báze používané právě pro výpočet elektronové struktury nikoli.

2.4.1 Struktura operátorů Kohn-Shamovy rovnice

Matice K a M v rovnici jsou zjevně symetrické a řídké, neboť jak operátor identity, tak i derivace jsou lokální. Jak jsme ukázali v kapitole 2.1.1 v rovnici (2.1.4), operátor V se skládá ze součtu několika operátorů, z toho ∇, V_{h}, V_{xc} a V_{loc} jsou taktéž symetrické lokální, jejich matice tedy budou taktéž řídké. Za zmínku stojí, že jelikož se všechny tyto operátory aplikují ve skalárním součinu bázových funkcí, jejich matice budou mít stejnou *strukturu řídkosti*.

Nejsložitější je případ operátoru potenciálu V_{ps}. Ten je nelokální, ale krátkodosahový, tzn. působí jen na funkce nenulové v blízkém okolí atomového jádra. Dá se předpokládat, že pro velké počítané systémy bude spíše růst počet počítaných atomových jader než počet buněk v dosahu nelokální části ionizačního potenciálu jednoho jádra (jinými slovy že s přibývajícím počtem jader nebude třeba tolik zahušťovat síť), a nárůst počtu nenulových prvků v matici potenciálu V_{ps} bude tedy pouze lineární. Přesto už výpočetní náročnost jednoho atomového jádra je příliš velká a je tedy třeba hledat výpočetně úspornější vyjádření pseudopotenciálu.

Jak jsme ukázali v kapitole 2.2, úhlová složka l-dependent pseudopotenciálu je v *separovaném* tvaru (viz definice 1.2.2 na straně 5). Nabízí se proto možnost do tohoto tvaru převést i radiální složku pomocí B-ortogonality (připomeňme si její definici v kapitole 1.2.2 na straně 5).

2.4.2 Separabilní pseudopotenciál

Je-li α_i^l je quasi-V_{l}-ortonormální[13] báze jednorozměrného radiálního Hilbertova prostoru, pak

$$V_{\mathrm{l}} = \sum_i (V_{\mathrm{l}}\alpha_i^l) d_i^l (V_{\mathrm{l}}\alpha_i^l)^+ \tag{2.4.1}$$

a tedy

$$\langle \psi | \, V_{\mathrm{ps}} \psi' \rangle = \tag{2.4.2}$$

$$\sum_{l=0}^{\infty} \left\langle \psi \middle| \left(V_l \circledast \sum_{m=-l}^{l} Y_{l,m} Y_{l,m}^+ \right) \psi' \right\rangle = \tag{2.4.3}$$

$$\left\langle \psi \middle| \sum_{l=0}^{\infty} \left(\sum_{i=0}^{\infty} V_{\mathrm{l}}\alpha_i^l d_i^l (V_{\mathrm{l}}\alpha_i^l)^+ \circledast \sum_{m=-l}^{l} Y_{l,m} (Y_{l,m})^+ \right) \psi' \right\rangle = \tag{2.4.4}$$

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{i=0}^{\infty} d_i^l \langle \psi | \, \left(V_{\mathrm{l}}\alpha_i^l (V_{\mathrm{l}}\alpha_i^l)^+ \circledast Y_{l,m} Y_{l,m}^+ \right) \psi' \rangle = \tag{2.4.5}$$

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{i=0}^{\infty} d_i^l \langle \psi | \, \left(V_{\mathrm{l}}\alpha_i^l \circledast Y_{l,m} \right) \left((V_{\mathrm{l}}\alpha_i^l)^+ \circledast Y_{l,m}^+ \right) \psi' \rangle = \tag{2.4.6}$$

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{i=0}^{\infty} d_i^l \left\langle \left(V_{\mathrm{l}}\alpha_i^l \circledast Y_{l,m} \right)^+ \psi \middle| \, \left(V_{\mathrm{l}}\alpha_i^l \circledast Y_{l,m} \right)^+ \psi' \right\rangle \tag{2.4.7}$$

[13] Jelikož počítáme v $\mathbb{R}$ a pseudopotenciály jsou obecně indefinitní, nelze zde použít „pravá“ V_{l}-ortonormalita.

Tím získáme *separabilní pseudopotenciál* reprezentovaný vektory $\vartheta_{i,l,m}$

$$\vartheta_{i,l,m} = V_{\mathrm{l}}\alpha_i^l \circledast Y_{l,m} \tag{2.4.8}$$

a konstantami d_i^l ve tvaru

$$V_{\mathrm{ps}} = \sum_{i,l,m} d_i^l(\vartheta_{i,l,m})(\vartheta_{i,l,m})^+ \tag{2.4.9}$$

Diskuse transformace souřadnic aneb analytický pohled

Za bližší diskusi stojí převod mezi kartézskými a sférickými souřadnicemi, který jsme provedli v rovnici (2.4.1). Vzhledem k separabilitě operátorů je tento rozklad legální: v daném rozměru je patřičná složka konstantní a jde tedy vytknout vně integrálu. Tento přechod mezi souřadnicemi však nelze udělat zcela mechanicky.

Známým faktem je, že při integraci, přecházíme-li z jedné soustavy do druhé, musíme integrovaný výraz vynásobit *integračním faktorem*. V případě sférických souřadnic je tento integrační faktor $r^2 sin\theta$. Skalární součin vlnové funkce je definován jako integrál v kartézských souřadnicích. Provedli-li jsme tedy rozklad 2.4.8, jde o převedení integrace do sférických souřadnic.

Při výpočtu konečněprvkových elementů matice integrujeme numericky, a tedy v kartézských souřadnicích – potud je vše v pořádku. Po vektorech α_i^l však požadujeme quasi-V_{l}-ortonormalitu s takovou normou, aby v kartézských souřadnicích se standardní normou bylo $\alpha_i^l \circledast Y_{l,m}$ quasi-V_{l}-ortonormální. V analytickém vyjádření tedy požadujeme, aby

$$\iiint \alpha_i^l(x,y,z) * Y_{l,m}(x,y,z)\, dx\, dy\, dz = 1 \tag{2.4.10}$$

a tedy

$$\iiint \alpha_i^l(r) * Y_{l,m}(\theta,\sigma) r^2 sin\theta\, dr\, d\theta\, d\varphi = 1 \tag{2.4.11}$$

$$\int \alpha_i^l(r) r^2\, dr \iint Y_{l,m}(\theta,\sigma) sin\theta\, d\theta\, d\varphi = 1 \tag{2.4.12}$$

Proto pro normování v radiálním prostoru musíme užít normu:

$$|\alpha| = \int \alpha(r)\alpha(r) r^2\, dr \tag{2.4.13}$$

Stejně tak sférické harmonické funkce $Y_{l,m}$ musí být normovány v součinu se $sin\theta$. Zde stojí za poznámku, že se tato norma pro sférické harmonické funkce považuje za standardní a pokud není explicitně udáno jinak, jsou funkce $Y_{l,m}$ normovány právě takto.[14]

Omezení projekční báze separabilních pseudopotenciálů

Suma v rovnici (2.4.9) je sice nekonečná, nyní však ukážeme, že ji lze omezit na konečně mnoho prvků. Separovaný popisuje, jak působí operátor na jednotlivé

[14] V kvantové fyzice, v některých jiných oborech se používají jiné zvyklosti

prvky báze. Pokud víme, že budeme při výpočtu elektronové struktury tento operátor aplikovat pouze na některé prvky báze, lze rozvoj operátoru omezit pouze na tyto prvky. Tento „omezený“ rozvoj pak na zvolené prvky báze působí stejně jako původní (nerozvinutý) operátor, zatímco „vyřazené“ prvky báze zobrazuje na nulový vektor.

Nelokální složky pseudopotenciálu jsou krátkodosahové. Působí tedy pouze v těsné blízkosti atomového jádra, kde se nalézají pouze valenční elektrony daného atomu — nikoli elektrony sousedních atomů. V těchto místech se vlnové funkce valenčních elektronů příliš neliší od vlnových funkcí elektronů izolovaného atomu $\psi_{l,n}$. Tyto vlnové funkce $\psi_{l,n}$ tvoří úplnou ortonormální bázi prostoru.

Proto je rozumné vzít jako projekční bázi α_i^l jako podmnožinu $\psi_{l,n}$. Vzhledem k tomu, že elektrony obsazují vždy pouze stavy s nejnižší energií, a vzhledem k ortonormalitě $\psi_{l,n}$ stačí do báze zahrnout několik málo prvních prvků této báze.

Maximální l je třeba volit vždy dle počítaného problému. Praxe ukazuje, že často postačují první dva až tři prvky rozvoje, pouze výjimečně u těžkých jader je nutné zahrnout i čtvrtý člen. Při vhodné volbě radiální báze stačí v drtivé většině případů zahrnout stavy pro jednu až tři energie (přičemž čím větší l, tím více roste u vyšších stavů energie, a tedy tím méně jich je třeba zahrnout). Celkový počet členů rozvoje se tedy pohybuje kolem deseti prvků.

2.4.3 Problém vlastních čísel

Definice 25. *Korekce matice $A^{n\times n}$ zapsaná ve tvaru $A + VV^+$, kde V je matice tvaru $n \times k$, se nazývá* aktualizace řádu k *(*rank-k-update*).*

Díky postupům popsaným v předchozích kapitolách lze rovnici 2.4.1 zapsat jako součet řídkých matic se stejnou strukturou řídkosti a aktualizace řádu k

$$\left(K + V_{\mathrm{h}} + V_{\mathrm{xc}} + V_{\mathrm{loc}} + \sum_{j=0}^{M} \sum_{l=0}^{\max[l]_{(j)}} \sum_{m=-l}^{l} \sum_{i=0}^{\max[i]_{(j,l)}} \binom{\vartheta_{j,i}}{l,m} \binom{d_{j,i}}{l} \left(\binom{\vartheta_{j,i}}{l,m} \right)^{+} \right) \mathbf{x} = \varepsilon M \mathbf{x} \tag{2.4.14}$$

a tedy v maticovém zápisu

$$\left(K_V + UDU^+ \right) \mathbf{x} = \lambda M \mathbf{x} \tag{2.4.15}$$

Jelikož potenciály ϑ_{nilm} jsou krátkodosahové, budou mít nenulové hodnoty pouze v konečných elementech, které spadají do „sféry vlivu“ daného atomového jádra. Toho lze využít pro optimalizaci a pracovat s vektory jako s řídkými pomocí dvojic (index, hodnota). Vzhledem k tomu, že sada vektorů od jednoho atomového jádra má stejnou strukturu řídkosti, lze tento formát ještě zefektivnit pomocí struktury (index, [hodnoty]). Při vhodném seřazení prvků konečné sítě (dle náležení ke konkrétnímu jádru) by šlo formát uložení těchto vektorů ještě zefektivnit, přínos by však pravděpodobně nebyl tak velký, aby ospravedlnil daleko větší náročnost na generování sítě konečných prvků.

Jelikož takovéto optimalizace mají význam až pro větší počet atomových jader, zatím nebyly provedeny a v programu používáme standardní „hustý“ fromát vektorů pro *aktualizace*.

Reálná čísla a reálné harmonické funkce

Tato metoda výpočtu elektronové struktury není cílena na řešení periodických struktur (pro které je vhodné např. užití metody planewaves, použité např. v programu WIEN2k[15]), pro výpočet není třeba komplexních čísel. Proto vlnové funkce užité při výpočtech budou reálné.

Pro úhlovou bázi separabilních pseudopotenciálů je tedy vhodné místo komplexních sférických harmonik $Y_{l,m}$ užít jejich reálné protějšky, *reálné sférické harmonické funkce*, aby zůstala *aktualizace* taktéž reálná.

$$Y^R_{l,m} = \begin{cases} Y_{l,0} & \text{if } m = 0 \\ \dfrac{1}{\sqrt{2}}\left(Y_{l,m} + (-1)^m Y_{l,-m}\right) = \sqrt{2} N_l^m P_l^m(\cos\theta)\cos m\varphi & \text{if } m > 0 \\ \dfrac{1}{i\sqrt{2}}\left(Y_{l,-m} - (-1)^m Y_{l,m}\right) = \sqrt{2} N_l^m P_l^{-m}(\cos\theta)\sin m\varphi & \text{if } m < 0. \end{cases} \tag{2.4.16}$$

V definici reálných sférických harmonických funkcí označuje P_l^m asociované Legenderovy polynomy a N_l^m jim příslušné normalizační funkce (Formánek, 2004).

Pro reálné vlnové funkce a tedy i separabilní pseudopotenciály U platí identita $U^+ = U^T$, což s sebou přináší na některých místech zjednodušení výpočtu.

[15] http://www.wien2k.at/ [2011]

3. Řešení diskretizovaných úloh

Aplikací metody konečných prvků (popř. jiné diskretizační metody) na diferenciální rovnici zpravidla vzniká *problém vlastních čísel*

$$A\mathbf{x} = \lambda B\mathbf{x}, \quad \text{kde } \mathbf{x} \in \mathbb{C},\ A, B \in \mathbb{C}^{n\times n},\ \mathbf{x} \neq \mathbf{0}$$

Konkrétní formulace problému pak může mít různé speciální vlastnosti (např. hermitovskost matic), které řešení problému usnaďnují.

V této části práce se budeme věnovat v praxi užívaným metodám pro řešení problému vlastních čísel. Nebudeme se přitom omezovat pouze na metody vhodné pro řešení našeho problému vyjádřeného rovnicí (2.4.15) a to ze dvou důvodů:

- Při vývoji naší metody pro výpočet elektronové struktury nebylo z počátku jasné, jakou formulaci rovnice a především psudopotenciálů zvolíme, různé formulace přitom vedou k různým typům problémů vlastních čísel. Jednou z klíčových otázek pro výběr konečné formulace byla přitom právě snadná řešitelnost vzniklého problému vlastních čísel. Výhody námi vybrané formulace jsou tak ilustrovány mj. právě i uvedením metod, které nevyužívají speciálních vlastností matic v rovnici (2.4.15).
- Problém vlastních čísel je velmi často řešeným problémem v různých fyzikálních metodách. Jedním z cílů této práce je udělat stručný ucelený přehled metod s jejich základním popisem, sloužící k orientaci v různých typech problémů vlastních čísel existují i metodách k jejich řešení, a k popisu, kterak lze využít různé vlastnosti problému vlastních čísel k jeho efektivnímu řešení výběrem vhodné metody. Předpokládá se využití jednak v případě spolupráce na dalších problémech, jednak při dalším vývoji stávající metody (např. pro výpočet periodických struktur, či využití dalšího řešiče v případech, kdy jde s výhodou využít některé jeho vlastnosti apod.)

Druhému z důvodů je trochu přizpůsobena i tato kapitola: jejím cílem není podat ucelený kompletní výklad matematické teorie sloužící k výpočtům vlastních čísel, protože to zaprvé nedovoluje rozsah práce, zadruhé na toto téma existuje dostatek vhodné literatury (např. (Demmel, 1997; Bai, 2000)). Spíše je cílem podat stručnou informaci o jednotlivých metodách, jež čtenáři umožní orientaci v metodách pro výpočet problému vlastních čísel.

Jelikož se pro různé typy problémů vlastních čísel užívají metody postavené na podobných myšlenkách, nebudeme strukturovat tuto kapitolu dle typů různých problémů, nýbrž dle ideí, na kterých jsou popisované metody založeny.

3.1 Dělení problémů vlastních čísel

Jak jsme již uvedli, problém vlastních čísel je vyjádřen rovnicí ve tvaru

$$A\mathbf{x} = \lambda B\mathbf{x}, \quad \text{kde } \mathbf{x} \in \mathbb{C},\ A, B \in \mathbb{C}^{n\times n},\ \mathbf{x} \neq \mathbf{0} \tag{3.1.1}$$

Konkrétní formulaci lze klasifikovat dle vlastností matic A a B.

Definice 26. *Jsou-li matice A a B a/nebo hledané řešení $\mathbf{x}$ komplexní, mluvíme o* komplexním problému vlastních čísel*. Jsou-li reálné, mluvíme o* reálném problému vlastních čísel*.*

Definice 27. *Je-li matice B maticí identita, problém se redukuje na* prostý problém vlastních čísel*:*

$$A\mathbf{x} = \lambda\mathbf{x}, \quad kde\ \mathbf{x} \in \mathbb{C},\ A \in \mathbb{C}^{n\times n},\ \mathbf{x} \neq \mathbf{0} \tag{3.1.2}$$

V opačném případě mluvíme o zobecněném problému vlastních čísel

Definice 28. *Jsou-li matice A a B symetrické (hermitovské), mluvíme o* symetrickém (hermitovském) problému vlastních čísel*.*

Algoritmy použitelné na symetrické matice jsou zpravidla vhodné (popř. s malou úpravou) i pro matice hermitovské a naopak.

Definice 29. *Vektory vyhovující rovnici (*3.1.2*) (popř. (*3.1.1*)) (zobecněného) problému vlastních čísel se nazývají* (zobecněné) vlastní vektory*, jim příslušné λ* vlastní číslo*. Dvojice vlastního vektoru se nazývá* vlastní pár*.*

3.2 Obecné postupy užívané při řešení problému vlastních čísel

V této kapitole se budeme věnovat problémům a technikám, které jsou společné pro většinu metod na počítání vlastních čísel, jimž se budeme věnovat. Do této kapitoly by také patřil *posun a inverze* a princip *blokových metod*, o nichž se však zmíníme až v další kapitole, neboť je budeme demonstrovat na konkrétních metodách.

3.2.1 Exaktní aritmetika a aritmetika s konečnou přesností

Protože se dnes k výpočtům užívají výhradně počítače, je třeba zmínit některé vlastnosti, které s sebou jejich užití přináší.

Dnes užívané počítače ukládají reálná čísla ve formátu *IEEE 754-1985* (IEEE Task P754, 1985). Tento formát ukládá jen omezený počet platných číslic, a proto při jeho užití dochází k zaokrouhlovacím chybám. Proto budeme dále rozlišovat *exaktní aritmetiku* a (počítačovou) *aritmetiku s konečnou přesností*. Vlastnosti a konvergenci algoritmů budeme vždy vztahovat k exaktní aritmetice, neboť rozbor těchto vlastností v aritmetice s konečnou přesností by byl příliš rozsáhlý či v některých případech není ani známý. Při implementaci algoritmu pak zmíníme, jak ovlivní aritmetika s konečnou přesností chování algoritmu a jak je ho popřípadě třeba modifikovat.

3.2.2 Metody užívající rozkladů matic

K řešení problémů vlastních čísel (stejně jako na řešení soustav lineárních rovnic) existují dvě skupiny metod. První z nich je založena na iteraci některé

formy rozkladu matic: např. LU rozkladu či dnes pravděpodobně nejčastěji užívaný QR algoritmus. Výsledkem těchto algoritmů je zpravidla diagonální matice, která tedy má na své diagonále vlastní čísla.

Tyto metody jsou vhodné v případě, kdy požadujeme výpočet celého spektra matice. V případě, že je požadována pouze malá část spektra a matice jsou velké, nejsou tyto metody efektivní, dále se jimi tedy nebudeme zabývat[1]

Druhou možností přístupu k řešení problému vlastních metod jsou metody s následující charakteristikou

- metody pracují v iteračním cyklu, ve kterém postupně zpřesňují nalezená řešení
- na rozdíl od předchozích matic zpravidla nepotřebují nijak manipulovat se samotnými maticemi (v některých případech je však vhodné spočíst LDL^T či LDU rozklad matice)
- lze je kdykoli přerušit a spokojit se s dosud spočteným výsledkem
- metody počítají vždy pouze několik vlastních čísel s vybranou charakteristikou (u většiny metod vlastní vektory odpovídající vlastním číslům s největší absolutní hodnotou)
- přestože některé z iteračních metod jsou teoreticky schopny vypočíst přesné řešení, užívají se k získání aproximace vlastního vektoru; v aritmetice s konečnou přesností se však tato vlastnost stírá (kde i „přesné“ řešení je nepřesné) a naopak v některých případech je výhodou moci vyřešit problém rychle, byť nepřesně
- algoritmy umí s velmi dobře využít užít řídkosti matic a zpravidla jsou dobře paralelizovatelné

Jelikož ve fyzikálních problémech zpravidla požadujeme výpočet pouze malé části spektra matic, budeme se dále v této práci věnovat pouze iteračním metodám.

3.2.3 Rayleighův koeficient

Rayleighův koeficient se užívá jako běžná aproximace vlastního čísla dané aproximace vlastního vektoru.

Definice 30. *(Hermitovský) Rayleighův koeficient pro matici $A \in \mathbb{C}^{n\times n}$ a vektor $\mathbf{x} \in \mathbb{C}^n$ je číslo $\theta \in \mathbb{C}$*

$$\theta = \frac{\langle \mathbf{x} | A\mathbf{x} \rangle}{\langle \mathbf{x} | \mathbf{x} \rangle} = \frac{\mathbf{x}^+ A\mathbf{x}}{\mathbf{x}^+\mathbf{x}} \tag{3.2.1}$$

Někteří autoři doporučují definovat Rayleighův koeficient v následující formě

Definice 31. *Transpoziční Rayleighův koeficient pro matici $A \in \mathbb{C}^{n\times n}$ a vektor $\mathbf{x} \in \mathbb{C}^n$ je číslo $\theta \in \mathbb{C}$*

$$\theta = \frac{\langle \mathbf{x} | A\mathbf{x} \rangle_T}{\langle \mathbf{x} | \mathbf{x} \rangle_T} = \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \tag{3.2.2}$$

[1] S výjimkou implicitního QR algoritmu, který je součástí implicitně restartované Arnoldiho metody

V oboru reálných čísel jsou tyto definice identické. V oboru komplexních čísel může být vhodné užití hermitovské definice, pokud se pracuje s normalizovaným vektorem (tzn. $\|\mathbf{x}\| = 1$), neboť v tomto případě $\theta = \langle \mathbf{x} | A\mathbf{x} \rangle$. Transpoziční definice je vhodná, pokud nám záleží na přesnosti odhadu, neboť transpoziční Rayleighův koeficient je asymptoticky přesnější než hermitovský. (Arbenz – Hochstenbach, 2004) Pokud neupřesníme, který z těchto dvou koeficientů máme na mysli, užíváme hermitovskou verzi. Ve většině případů je možno užít Rayleighova koeficientu spočteného podle obou definic.

Pro zobecněný problém vlastních čísel je definován Rayleighův koeficient obdobně:

Definice 32. *Zobecněný hermitovský (transpoziční) Rayleighův koeficient pro vektor* $\mathbf{x} \in \mathbb{C}^n$ *a dvojici matic* $A, B \in \mathbb{C}^{n \times n}$ *definující problém vlastních čísel* $A\mathbf{x} = \lambda B\mathbf{x}$ *je číslo* $\theta \in \mathbb{C}$ *definované*

$$\theta = \frac{\langle \mathbf{x} | A\mathbf{x} \rangle^{(T)}}{\langle \mathbf{x} | B\mathbf{x} \rangle^{(T)}} \tag{3.2.3}$$

3.2.4 Konvergence iteračních metod

Mluví-li se o vlastním vektoru matice, jde ve skutečnosti (vzhledem k linearitě zobrazení reprezentovaných maticemi) o *vlastní směr*.

Definice 33. *Vektory* $\mathbf{u}, \mathbf{v} \in \mathbb{C}^n$ *mají shodný* směr*, existuje-li* $a \in \mathbb{C} \setminus \{0\} : \mathbf{u} = a\mathbf{v}$.

Konvergencí metody k vektoru je tedy myšlena konvergence k určitému směru.

Definice 34. *Posloupnost vektorů* $\mathbf{x}_i \in \mathbb{C}^n$ *konverguje k vektoru* $\mathbf{v} \in \mathbb{C}^n \setminus \{0\}$*, existuje-li posloupnost* $a_n \in \mathbb{C}$:

$$\lim_{i \to \infty} \|a_i \mathbf{x}_i - \mathbf{v}\| = 0 \tag{3.2.4}$$

Metoda pro výpočet vlastních čísel konverguje*, konverguje-li posloupnost odhadů vlastních vektorů generovaných metodou.*

V praxi se pak konvergence kontroluje zpravidla pomocí následující podmínky.[2] V některých metodách (např. v Lanczosově metodě) se tato hodnota nemusí počítat, neboť je získána jako vedlejší výsledek prováděných operací.

Teorém 1. *Konverguje-li posloupnost vektorů* $\mathbf{x}_i \in \mathbb{C}^n, \|\mathbf{x}_i\| = 1$ *k vlastnímu vektoru* $\mathbf{v}$ *matice* A*, pak*

$$\lim_{i \to \infty} \|A\mathbf{x}_i - \langle A\mathbf{x}_i | \mathbf{x}_i \rangle \mathbf{x}_i\| = 0 \tag{3.2.5}$$

[2] Jelikož to je pouze nutná podmínka, konvergence posloupnosti $\|A\mathbf{x}_i - \langle A\mathbf{x}_i | \mathbf{x}_i \rangle \mathbf{x}_i\|$ konvergenci k vlastnímu vektoru nezaručuje. Vzhledem k tomu, že konvergence metod zaručená je, dá se tento test použít jako rychlá a poměrně spolehlivá heuristika. Může však selhat u lineární kombinace dvou vektorů se „skoro stejným“ vlastním číslem.

Aby nebyla požadovaná přesnost konvergence rozdílná pro různá vlastní čísla matice, kontroluje se konvergence pomocí následujícího kritéria

$$\left\| A\mathbf{x}_i \frac{\langle \mathbf{x}_i | \mathbf{x}_i \rangle}{\langle \mathbf{x}_i | A\mathbf{x}_i \rangle} - \mathbf{x}_i \right\| < e \tag{3.2.6}$$

kde e je požadovaná přesnost řešení. Odhad vlastního vektoru se tradičně nazývá *Ritzův vektor* a jemu příslušný odhad vlastního čísla počítaný jako *Rayleighův koeficient* se nazývá *Ritzovo číslo*.

3.2.5 Počítání dalších vlastních vektorů

Iterační metody konvergují k vlastnímu vektoru, který nejlépe splňuje danou charakteristiku[3] a není kolmý k Jordanovu bloku (viz definice 38) startovního vektoru. V jednom běhu lze spočítat pouze omezený počet vlastních vektorů. Chceme-li spočítat další vlastní vektory, můžeme metodu spustit opakovaně, vždy na startovní vektor, který je kolmý (*ortogonální*) na všechny dosud spočítané vektory: metoda pak zkonverguje k „nejlepšímu dalšímu“ vlastnímu vektoru.

Ztráta ortogonality, reortogonalizace a deflace

U nesymetrických matic však metody spočtou pouze vlastní vektor a nikoli jeho Jordanův řetězec – takže není možné vůči němu ortogonalizovat nový startovní vektor. Kvůli nepřesnému výpočtu vlastních vektorů a aritmetice s konečnou přesností se zpravidla během iterací ortogonalita ztrácí i u symetrických matic. Algoritmus pak má tendenci dokonvergovat k již spočtenému vlastnímu vektoru.

Tento problém se řeší dvěma postupy: první z nich je *reortogonalizace*: jednou za čas se iterovaný vektor zortogonalizuje ke všem vektorům, ke kterým má být kolmý.

Ztráta ortogonality zapříčiněná aritmetikou s konečnou přesností (a tedy nutnost reortogonalizací) se vyskytuje i v jiných případech.[4] Na toto riziko vždy upozorníme u konkrétní metody.

Wielandtova deflace

Druhou možností, jak zabránit konvergenci k již spočtenému vektoru jsou takzvané *deflační metody*: tyto metody upraví matici tak, že změní vlastní čísla u vybraných vlastních vektorů, zatímco zbylá vlastní čísla a vektory zůstanou nezměněny.

Jak dále ukážeme, rychlost konvergence závisí na rozložení vlastních čísel, vhodnou deflací lze tedy urychlit konvergenci metod na řešení problému vlastních čísel. Aplikací deflačních metod také můžeme snížit počet nutných reortogonalizací. Nevýhodou deflačních metod je možnost porušení řídkosti matice.

[3] Metody většinou konvergují k vektoru či vektorům s vlastním číslem v absolutní hodnotě maximálním, při aplikaci posunu a inverze pak k vlastnímu číslu nejbližšímu zvolené hodnotě.

[4] Např. v krylovovských algoritmech, které uvedeme dále, se ortogonalizuje báze prostoru modifikovaným Gram-Schmidtovým algoritmem. U větších dimenzí dochází ortogonalizací vůči dalším vektorům ke ztrátě ortogonality vůči prvním vektorům. Tato ztráta je zapříčiněna zaokrouhlovacími chybami.

Pro případ, kdy se chceme „zbavit“ jednoho vlastního vektoru, je vhodná *Wielandtova deflace*. Chceme-li v matici A zmenšit vlastní číslo vektoru $\mathbf{v}$ o hodnotu $\pi \in \mathbb{R}$, provedeme transformaci

$$A' = A - \pi(a\mathbf{v}\mathbf{v}^T), \quad \text{kde } a \in \mathbb{R} \colon \mathbf{v} * a\mathbf{v} = 1 \tag{3.2.7}$$

Matice A' pak bude mít vlastní čísla i vektory identické s maticí A, až na vlastní číslo vektoru $\mathbf{v}$, pro které platí:

$$\lambda_v^{'} = \lambda_v - \pi \tag{3.2.8}$$

Wielandtova-Schurova deflace

Definice 35. *Schurův (částečný) rozklad matice $A \in \mathbb{C}^{n\times m}, n > m$ je součin matic $Q^TAQ = R$, kde $Q \in \mathbb{C}^{n\times k}, Q^+Q = I$ a $R \in \mathbb{C}^{m\times m}$ je horní trojúhelníková. Je-li $k = m$, jde o Schurovu dekompozici (rozklad), je-li $k < m$, jde o částečný Schurův rozklad (řádu k).*

Je-li A čtvercová matice, diagonála R je složena z vlastních čísel matice A.

Pro reálné nesymetrické matice nemusí takovýto rozklad v oboru reálných čísel existovat. V tom případě je nutno povolit blokově trojúhelníkový tvar matice R, nebo její komplexnost.

Definice 36. *Vektory v matici Q Schurova rozkladu matice A se nazývají* Schurovy vektory *matice A.*

Wielandt-Schurova deflační metoda se hodí k *deflaci* více vektorů najednou. V tomto případě se matice upraví následujícím způsobem

$$A' = A - QDQ^+ \tag{3.2.9}$$

Q je ortonormální báze vlastního prostoru deflatovaných vektorů – obsahuje tedy Schurovy vektory matice vlastních vektorů. D je diagonální matice, kde $D_{i,i}$ udává posun vlastního čísla odpovídajícího i-tému Schurovu vektoru v matici Q. Důkaz a podrobnější rozbor lze nalézt zde (Saad, 1992, 1989).

3.3 Metody založené na iteraci vektoru I – mocninné metody

Dnes užívané iterační metody lze rozdělit do dvou skupin. První skupinu lze nazvat *metody založené na iteraci vektoru*. Metody této skupiny jsou založeny na snaze co nejlépe odhadnout řešení, při čemž se postupuje takto: vezmeme libovolný počáteční vektor jako odhad řešení a tento odhad postupně nějakým vhodným způsobem upřesňujeme. Z této skupiny metod se budeme věnovat *mocninné metodě* a metodám založeným na metodě největšího spádu.

Druhá skupina se nejčastěji nazývá *projekční metody*. Metody z této skupiny iterativně budují podprostor, do kterého promítají původní problém. Jelikož je dimenze budovaného podprostoru řádově menší než dimenze původního prostoru, je řešení v tomto podprostoru řádově levnější. Tento podprostor budují takovým způsobem, aby řešení promítnutého problému co nejlépe aproximovalo

řešení původního problému. Z této kategorie zmíníme *Arnoldiho metodu* a její variantu pro symetrické matice: metodu *Lanczosovu*, dále pak *Lanczosovu metodu pro nesymetrické matice* a *Jacobi-Davidsonovu* metodu, která je vhodná především pro nesymetrické matice, či v případě, kdy je k dispozici vhodné předpodmínění.

3.3.1 Mocninná metoda

Nejjednodušší z metod pro výpočet vlastních čísel jsou metody iterace vektoru založené na *mocninné metodě*. Přestože se dnes již mocninná metoda (ani metody z ní odvozené) pro problémy vlastních čísel vzniklé z fyzikálních výpočtů příliš neužívá, budeme se jí věnovat poněkud hlouběji, neboť je nejjednodušší ze všech metod a zároveň vykazuje vlastnosti, které ve větší či menší míře dědí i ostatní metody, ať již jde o vztah konvergence a rozložení spektra matice, chyby způsobené zaokrouhlovacími chybami atd...

Princip mocninné metody spočívá ve výpočtu vektoru $A^n\mathbf{x}$. Jak si dále ukážeme, pro dostatečně velké n je $A^n\mathbf{x}$ dobrým odhadem vlastního vektoru matice A. $A^n\mathbf{x}$ je počítáno iterativně: počáteční odhad vlastního vektoru $\mathbf{x}_0$ – *startovací vektor* – vynásobíme v každém kroku iterace maticí A:

$$\mathbf{x}_{i+1} = A\mathbf{x}_i \tag{3.3.1}$$

Implementace mocninné metody

Při implementaci v konečné aritmetice s konečnou přesností může vektor $\mathbf{x}_i$ růst do nekonečna či naopak klesat k nule, přičemž s příliš velkými či malými čísly nelze v této aritmetice pracovat. Proto se vektor $\mathbf{x}_i$ v každém kroku *normalizuje*: násobí se reálným číslem tak, aby se jeho norma rovnala jedné. K tomuto účelu lze použít libovolnou normu, protože se však zároveň pro kontrolu konvergence počítá Rayleighův koeficient, je vhodné užít 2-normu.[5]

Pro $\mathbf{x}_i$ tedy platí následující rekurentní vztah:

$$\mathbf{x}_{i+1} = A\mathbf{x}_i/\|A\mathbf{x}_i\| \tag{3.3.2}$$

Algoritmus 3.3.1: Algoritmus mocninné metody

1: x_0 = počáteční odhad vlastního vektoru
2: **for** i = 1,2... **do**
3: $\quad \mathbf{y}_i = \mathbf{x}_{i-1}/\|\mathbf{x}_{i-1}\|$ ▷ znormalizuj
4: $\quad \mathbf{x}_i = A\mathbf{y}_i$ ▷ iteruj
5: $\quad \theta_i = \langle \mathbf{x}_i | \mathbf{y}_i \rangle$ ▷ spočti odhad vlastního čísla
6: **end until** $\|\mathbf{y}_i - \mathbf{x}_i/\theta_i\| > e$ ▷ e je požadovaná přesnost
7: **return** $\mathbf{x}_i$ ▷ nalezená aproximace vlastního vektoru

[5]Pokud by se v optimalizované verzi v jednom kroku normovalo a zároveň se neprováděla kontrola konvergence, je výhodné v tomto kroku užít 1-normu či ∞-normu, které jsou jednodušší pro výpočet.

Přirozenou optimalizací tohoto algoritmu je, že normalizace a kontrola konvergence se nemusí provádět při každém kroku – přičemž zpočátku může být interval mezi kontrolami konvergence poměrně velký.

Konvergence mocninné metody

Pro porozumění metodám sloužícím k řešení problému vlastních čísel je důležité vědět, za jakých podmínek metody konvergují.

Definice 37. Dominantní/submisivní *vlastní číslo matice A je vlastní číslo matice A s největší/nejmenší absolutní hodnotou (v spektru matice A).*

Dominantní/submisivní vlastní vektor matice A je vlastní vektor příslušný k dominantnímu/submisivnímu vlastnímu číslu.

Je třeba upozornit na možný terminologický problém: mluví-li se o vlastních vektorech matice A, jde ve skutečnosti o vlastní vektory lineárního zobrazení vyjádřeného ve standardní bázi maticí A. Proto, budeme-li mluvit o matici A vyjádřené v bázi B, myslíme tím ve skutečnosti matice lineárního zobrazení v bázi B, které je ve standardní bázi vyjádřenéno maticí A.

Nyní dokážeme následující větu:

Věta 5. *Pokud matice $A \in \mathbb{C}^{n\times n}$ má právě jeden lineárně nezávislý dominantní vlastní vektor a vektor $\mathbf{x}_0$ není na tento vektor kolmý, pak posloupnost vektorů $\mathbf{x}_i$ generovaných mocninnou metodou pro matici A se startovacím vektorem $\mathbf{x}_0$ při normalizací q-normou konverguje k dominantnímu vlastnímu vektoru matice A.*

Důkaz. Důkaz konvergence nejprve provedeme na diagonalizovatelné matici. Následně ukážeme, jak jej rozšířit na matici obecnou. V důkazu budeme používat horní index pro označení vektoru z i-té iterace a dolní index pro označení složky vektoru.

Diagonalizovatelná matice

Diagonalizovatelná matice je taková matice A, pro kterou existuje báze B, v níž je matice A vyjádřena diagonální maticí. Taková báze je pak tvořena vlastními vektory A. Dále budeme pracovat v bázi B seřazené sestupně dle absolutních hodnot vlastních čísel[6].

V námi zvolené bázi platí pro libovolný vektor $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2 \dots]$[7] rovnost $A\mathbf{x} = [\lambda_1\mathbf{x}_1, \lambda_2\mathbf{x}_2 \dots]$ a tedy

$$\mathbf{x}^i = A^n\mathbf{x}_0 = [\lambda_1^i\mathbf{x}_1^0, \lambda_2^i\mathbf{x}_2^0, \dots, \lambda_n^i\mathbf{x}_n^0] \tag{3.3.3}$$

Definujeme-li q_i jako normu vektoru $\mathbf{x}^i$

$$q_i = \sqrt[q]{\sum_{j=1}^{n} |\lambda_j\mathbf{x}_j^i|^q} \tag{3.3.4}$$

pak

$$\mathbf{x}_1^{i+1} = \lambda_1\mathbf{x}_1^{i+1}/q_i \tag{3.3.5}$$

[6]Případ, kdy jsou všechna nulová, je jasný.

[7]Nyní budeme používat horní index pro index iterace a spodní index pro index báze.

Jelikož $\forall j > i : |\lambda_j| < |\lambda_1|$ a $\sum_{j=1}^{n} |\mathbf{x}_j^i|^q = 1$, tak $q_i = \lambda_1 \Leftrightarrow |\mathbf{x}_1^i| = 1$, v opačném případě $q_i < \lambda_1$. Proto je díky 3.3.5 posloupnost $|\mathbf{x}_1^i|$ neklesající, a má tedy limitu v intervalu $\langle 0, 1\rangle$.

$$\text{Byla-li by} \qquad \lim_{i\to\infty} |\mathbf{x}_1^i| = k < 1 \tag{3.3.6}$$

pak existuje $\overline{k}, \underline{k} \in (0,1) : \exists o : \forall p > o : \underline{k} < |\mathbf{x}_1^p| < \overline{k}$ a tedy

$$\frac{q_i}{\lambda_1} \leq \frac{\sqrt[q]{(\lambda_1\overline{k})^q + (\lambda_2(1-\underline{k}))^q}}{\lambda_1} \tag{3.3.7}$$

$$= \sqrt[q]{(\overline{k})^q + (\lambda_2/\lambda_1(1-\underline{k}))^q} \leq 1 \tag{3.3.8}$$

Posloupnost $|\mathbf{x}_1^i|$ by tedy byla omezena zdola geometrickou řadou, což by byl spor. Proto

$$\lim_{i\to\infty} |\mathbf{x}_1^i| = 1 \tag{3.3.9}$$

a vektor $\mathbf{x}_i$ tedy konverguje ke směru vektoru $[1, 0, 0 \dots]$, což je dominantní vlastní vektor matice A.

Obecná matice

Definice 38. Jordanův kanonický tvar *matice A je blokově diagonální matice $J = Q^T AQ$ s následujícími vlastnostmi:*

- *$QQ^T = I$*
- *kde každý blok matice J má na diagonále vlastní číslo matice, v diagonále nad hlavní diagonálou jedničky a v ostatních prvcích nuly.*

Tyto bloky se nazývají Jordanovy bloky*, báze Q se nazývá* Jordanova kanonická báze*. Vektory Jordanovy kanonické báze odpovídající jednomu Jordanovu kanonickému bloku se nazývají* Jordanův řetězec*.*

Z definice Jordanova kanonického tvaru plyne, že Jordanův řetězec odpovídající Jordanově bloku s vlastním číslem λ_k je konečná posloupnost vektorů, kde první vektor řetězce je vlastní vektor a další prvky posloupnosti jsou vektory $\mathbf{x}^i$, pro které platí $A\mathbf{x}^i = \lambda_k \mathbf{x}^i + \mathbf{x}^{i-1}$.

Nyní budeme pracovat v Jordanově kanonické bázi a složky vektoru $\mathbf{x}$ budeme nyní značit $\mathbf{x}_{i,j}$, kde i je číslo Jordanova bloku a j je pořadí vektoru v řetězci.

Lemma 1. *Nechť Jordanův blok k velikosti n má dimenzi l a $\mathbf{x}^i = A^i\mathbf{x}^0$. Je-li $\mathbf{x}^0_{k,l} \neq 0$ pak limita*

$$\lim_{i\to\infty} \left| \frac{\mathbf{x}^i_{k,l}}{\mathbf{x}^i_{k,l-1}} \right| = 0, \quad \textit{kde } \mathbf{x}^i = A^i\mathbf{x}^0, \textit{ pro } l \in \{2, \dots, n\} \tag{3.3.10}$$

Důkaz. Snadno se lze přesvědčit, že pro poslední prvek řetězce platí rovnost

$$\mathbf{x}^i_{k,l} = (\lambda_k)^i \mathbf{x}^0_{k,l} \tag{3.3.11}$$

a pro předposlední

$$|\mathbf{x}^i_{k,l-1}| \geq \sum_{j=1}^{i} |(\lambda_k)^j \mathbf{x}^0_{k,l} (\lambda_k)^{i-j-1}| = i|(\lambda_k)^{i-1}\mathbf{x}^0_{k,l}| \tag{3.3.12}$$

neboť předposlední prvek obdrží v každém kroku přírůstek od posledního prvku řetězce.

Z uvedených vztahů je zřejmé, že

$$\lim_{i\to\infty} \left| \frac{\mathbf{x}^i_{k,l}}{\mathbf{x}^i_{k,l-1}} \right| \leq \left| \frac{\lambda_k}{i} \right| = 0 \tag{3.3.13}$$

a tedy:

$$\lim_{i\to\infty} \left| \frac{\mathbf{x}^i_{k,l}}{\mathbf{x}^i_{k,l-1}} \right| = 0 \tag{3.3.14}$$

□

Principem indukce pak lze tuto limitu dokázat pro všechny sousední prvky řetězce (příspěvek od dalších členů řetězce se díky indukčnímu předpokladu odhadnou shora dostatečně malým číslem). Z toho pak okamžitě plyne

$$\lim_{i\to\infty} \left| \frac{\mathbf{x}^i_{k,j}}{\mathbf{x}^i_{k,1}} \right| = 0, \quad \text{kde } \mathbf{x}^i = A^i\mathbf{x} \text{ a } j \in \{2, 3, \cdots, l\} \tag{3.3.15}$$

Protože normalizace vektorů nemá na podíl (3.3.15) vliv, postupem iterací ve vektoru iterovaném mocninnou metodou vymizí (respektive budou zanedbatelně malé) všechny složky, které neodpovídají některému vlastnímu vektoru. U takto „pročištěného“ vektoru nyní můžeme použít stejný důkaz jako pro diagonalizovatelnou matici. □

Divergence mocninné metody Nyní si rozeberme případy, kdy podmínky věty nejsou splněny:

- Je-li $|\lambda_1| = |\lambda_2|$ (popř. $= |\lambda_3|$) …, je vcelku zřejmé, že vektor zkonverguje k lineární kombinaci dominantních vlastních vektorů.
- Pokud se jako startovací vektor vezme vektor kolmý na Jordanův kanonický blok dominantního vlastního vektoru, metoda dokonverguje k „dominantnímu obsaženému“ vektoru, neboť není-li v $\mathbf{x}$ obsažen vlastní vektor $\mathbf{v}$, pak není ani v $A\mathbf{x}$.

  Metoda tedy bude počítat v podprostoru generovaném vlastními vektory obsaženými v $\mathbf{x}$ a na tomto podprostoru dokonverguje k dominantnímu vektoru (pokud v tomto podprostoru nejsou alespoň dva lineárně nezávislé dominantní vlastní vektory). I tato situace se však považuje za neúspěch, neboť je očekáván výpočet dominantního vlastního páru.

  V případě konečné artitmetiky se však vinou zaokrouhlovacích chyb zpravidla ztratí ortogonalita s dominantním vlastním vektorem. V tomto případě metoda i tak zkonverguje k dominantnímu vlastnímu vektoru.

Rychlost konvergence Z uvedeného důkazu také vyplývá toto pozorování: rychlost konvergence mocninné metody je zdola omezena hodnotou λ_1/λ_2. Jelikož všechny metody na počítání vlastních čísel jsou ve větší či menší míře založeny na násobení maticí A, lze formulovat následující tezi:

Rychlost konvergence iteračních metod závisí na tom, jak dobře jsou od sebe vlastní čísla oddělena.

Pro mocninnou metodu samotnou pak platí následující vztahy:

$$\{\|\mathbf{x}^i - \mathbf{x}\|\} \in \mathcal{O}\left(\left|\frac{\lambda_1}{\lambda_2}\right|\right), \quad \text{kde } \mathbf{x} \text{ je dominantní vlastní vektor} \tag{3.3.16}$$

$$\{|\theta_i - \lambda_1|\} \in \mathcal{O}\left(\left|\frac{\lambda_1}{\lambda_2}\right|^2\right) \tag{3.3.17}$$

$$\text{kde } \{a_i\} \in \mathcal{O}(c) \Rightarrow \left|\frac{a_i}{a_{i+1}}\right| \geq c$$

Výhody a nevýhody mocninné metody

Výhoda mocninné metody je v její velké jednoduchosti a snadné implementaci. Další výhodou této metody jsou velmi malé paměťové nároky a jednoduchá paralelizace. Největší nevýhodou této metody je její pomalá konvergence; její další nevýhodu – možnost spočítat pouze jeden vlastní pár – lze kompenzovat užitím blokové mocninné metody.

Proto je tato metoda vhodná především pro řešení rozsáhlých úloh, které jsou příliš rozsáhlé na aplikaci jiných metod a zároveň nepotřebují velkou přesnost řešení. Mocninná metoda se užívá například k počítání ohodnocení stránek (takzvané *pagerank*) v internetových vyhledávačích. (Langville – Meyer)

3.3.2 Bloková mocninná metoda (metoda iterace podprostoru)

Ke každé metodě pro výpočet vlastních čísel existuje bloková metoda. Příslušná bloková metoda provádí stejný algoritmus jako nebloková metoda, pouze místo s jedním vektorem pracuje s více vektory najednou. Na iterovaný „blok“ vektorů $\mathbf{v}_i, i \in \{1, 2 \ldots, m\}$, většinou popisovaný jako matice $V \in \mathbb{C}^{n\times m}$, můžeme pohlížet jako na bázi podprostoru. Proto se bloková mocninná metoda někdy nazývá *iterace podprostoru*.

Pokud bychom pouze prováděli iterace

$$V_{i+1} = AV_i \tag{3.3.18}$$

všechny vektory $\mathbf{v}_i$ by zkonvergovaly k dominantnímu vektoru matice A. V každém kroku algoritmu (popř. v optimalizované verzi jednou za několik kroků) je proto třeba iterované vektory $\mathbf{v}_i$ vůči sobě ortonormalizovat – spočíst ortonormalizovanou bázi prostoru V_i (např. pomocí modifikovaného Gram-Schmidtova algoritmu).

Algoritmus 3.3.2: Algoritmus blokové mocninné metody

1: V_0 = počáteční odhad vlastních vektorů
2: **for** i = 1,2... **do**
3: W_i = báze(V_{i-1}) ▷ spočti bázi iterovaného prostoru
4: $V_i = AW_i$ ▷ iteruj
5: **end until** konvergence
6: **return** V_i

Iterované vektory u této metody budou konvergovat k vlastním vektorům různou rychlostí. Nejprve zkonverguje dominantní vlastní vektor, a teprve pak začnou konvergovat „druhé“ dominantní a další vektory. Výpočtem přesného odhadu prvního vektoru odstraníme při ortogonalizaci z iterovaných vektorů jeho „rušící“ složku.

Proto se u této metody provádí jednou za čas kontrola konvergence a (dostatečně přesně) zkonvergované vektory se *zamknou*: již se neiterují, pouze se vůči nim dále ortogonalizují zbylé iterované vektory.

Konvergence blokové mocninné metody

Podobně jako u mocninné metody závisí rychlost konvergence (pro i-tý vlastní vektor) na podílu $\lambda_i/(\lambda_{n+1})$, kde n je velikost iterovaného podprostoru. Díky tomu lze snadno zvýšit rychlost konvergence mocninné metody, budeme-li iterovat podprostor větší dimenze, než kolik potřebujeme vypočítat vlastních vektorů (a algoritmus ukončíme v okamžiku, kdy zkonverguje námi chtěný počet vlastních vektorů). Tento postup může být obzvláště výhodný, jsou-li vlastní čísla matice sdružená ve shlucích, jejichž velikost můžeme předem odhadnout (což lze např. u diagonálně dominantní matice pomocí Greshgorinova teorému (Demmel, 1997)) – pak je výhodné zvolit za n velikost shluku vlastních čísel.

Další možnou optimalizací konvergence je užití Ritzovy (někdy zvané Rayleigh-Ritzova) procedury (SjÖstrÖm, 1996; Demmel, 1997) pro ortonormalizaci vektorů V. Ritzova procedura ortogonalizuje vektory V na vlastní vektory matice A kolmo promítnuté do prostoru generovaného vektory V. Díky tomu se nejen snadno provede kontrola konvergence, ale též se urychlí konvergence metody. Tato modifikace mocninné metody se nazývá *Ritzovou procedurou urychlená mocninná metoda* (*Ritz accelerated subspace iteration*).

Výhody a nevýhody blokové mocninné metody

Bloková mocninná metoda je přirozeným a efektivním rozšířením mocninné metody, platí pro ni to samé, co pro základní verzi metody. Oproti základní verzi lze urychlit výše zmíněnými postupy konvergenci, i tak však konverguje pomaleji než metody, které uvedeme dále. Tato metoda se tedy hodí na ty případy, kdy bychom použili základní *mocninnou metodu*, avšak potřebujeme spočíst více vlastních vektorů, nebo pokud vlastní čísla matice A vytvářejí shluky.

Další výhoda blokových metod se projeví v okamžiku, kdy operační paměť počítače nestačí pojmout matici A a ta je uložena na pevném disku. V tomto případě násobení jednoho vektoru maticí trvá řádově stejnou dobu, jako násobení n vektorů, neboť načtení matice z disku je řádově pomalejší než operace prováděné

s ní v operační paměti. Další výhoda blokových algoritmů spočívá ve snažší paralelizaci.

Nevýhodou blokových algoritmů může být větší rozsah jak algoritmu, tak i zpracovávaných dat a tím způsobená menší efektivita cache procesoru.

K počítání úloh vlastních čísel vzniklých z aplikace metody konečných prvků se někdy používá *Ritzovou procedurou urychlená mocninná metoda*, na přesné a dostatečně rychlé řešení problémů vlastních čísel velkých dimenzí jsou však vhodnější rychleji konvergující metody – na což ukazuje i nepříliš velký zájem současné odborné veřejností o tuto metodu.

3.3.3 Metody užívající posunu a inverze (shift and invert)

Jak je patrno, mocninná metoda je schopna spočíst pouze dominantní vlastní čísla. Ve velké většině úloh vedoucích k problému vlastních čísel je však požadavek spočíst vlastní čísla blízká nějakému číslu. V tom případě lze užít některou z metod užívajících principu *posunu a inverze* (*shift and invert*): místo matice A použijeme v algoritmu matici $A' = (A - \sigma I)^{-1}$. Vlastní čísla matice A a matice A' na sobě závisí: nechť má matice $(A - \sigma I)^{-1}$ vlastní pár $(\lambda, \mathbf{v})$, pak:

$$\begin{aligned} (A - \sigma I)^{-1}\mathbf{x} &= \lambda\mathbf{x} \\ \mathbf{x}/\lambda &= A\mathbf{x} - \sigma\mathbf{x} \\ A\mathbf{x} &= (1/\lambda - \sigma)\mathbf{x} \end{aligned} \tag{3.3.19}$$

Spočteme-li tedy dominantní vlastní vektory matice A', spočítali jsme zároveň takové vlastní vektory matice A, které mají vlastní čísla nejbližší k hodnotě σ.

Inverzní mocninná metoda (inverse power method)

Výpočet inverze matice $A' = (A - \sigma I)^{-1}$ je pro většinu matic příliš náročný. Jelikož se matice A' užívá pouze k násobení vektoru, u problémů malých a středních dimenzí se dá explicitní vypočtení inverze snadno obejít výpočtem LDU (či pro symetrické matice LDL^T (Duff, 2002)) rozkladu matice A. Tento rozklad pak lze použít k rychlému řešení rovnice

$$(A - \sigma I)^{-1}\mathbf{y} = \mathbf{x} \tag{3.3.20}$$

U problémů velkých dimenzí je někdy vhodné vyhnout se i výpočtu rozkladu (jak z důvodu výpočetní i paměťové náročnosti, tak i pro horší možnosti paralelizace metod pro výpočet rozkladu). Místo toho můžeme v každém kroku znovu řešit rovnici (3.3.20) pro neznámou y.

Ukazuje se, že rovnice (3.3.20) se nemusí řešit přesně (přičemž lze nejprve řešit velmi přibližně a přesnost řešení postupně zvětšovat). K řešení rovnice (3.3.20) se zpravidla užívají iterační řešiče lineárních rovnic. Pro zachování lineární konvergence metody stačí v průměru nevelký konstantní počet cyklů. (Golub – Ye, 1999). Tato varianta mocninné metody se nazývá *inverzní mocninná metoda*.

Druhá možnost, užívaná pro řešení rovnice (3.3.20), je užití pouze jednoho kroku iteračního řešiče s užitím předpodmínění – tato metoda se nazývá *předpodmíněná inverzní mocninná metoda*. (Neymeyr, 2000) I u ní lze s použitím

vhodného předpodmínění pro symetrickou pozitivně definitní matici A zachovat lineární konvergenci (Knyazev – Neymeyr, 2001). Jelikož je tato metoda velmi blízká metodě největšího spádu, která je citována častěji, uvedeme další podrobnosti týkající se předpodmínění v kapitole o metodě největšího spádu (Neymeyr, 2002). Všechny zde zmíněné metody se zpravidla provádějí v blokové formě.

Iterace Rayleighova koeficientu (Rayleigh quotient iteration)

Přirozeným vylepšením předchozí inverzní mocninné metody je metoda *iterace Rayleighova koeficientu* (např. (Fattebert)). Jak jsme ukázali v kapitole (3.3.1) o konvergenci, mocninná metoda konverguje tím rychleji, čím jsou dominantní vlastní čísla lépe oddělena. Čím se lépe „strefíme“ s odhadem σ k některému vlastnímu číslu, tím bude toto vlastní číslo lépe odděleno od svého nejbližšího souseda.

Inverzní mocninná metoda užívá k posunu pevně zvolené hodnoty σ. Iterace Rayleighova koeficientu urychluje svoji konvergenci tím, že v každém kroku užívá jinou hodnotu σ_i, a to odhad vlastního čísla získaný v předchozím kroku – tedy Rayleighův koeficient odhadu vlastního vektoru z předchozí iterace (lze užít jak hermitovský, tak transpoziční Rayleighův koeficient, vzhledem k velké výpočetní náročnosti inverze je však lepší užít přesnějšího transpozičního koeficientu).

Konvergence teto metody pro symetrické matice je kubická (van den Eshof, 2002), na rozdíl od inverzní mocninné metody však není zajištěna konvergence k vlastnímu číslu nejbližšímu k startovacímu odhadu σ_0 (Demmel, 1997; Parlett, 1980).

$$\{\|\mathbf{x}^i - \mathbf{x}^\infty\|\} = \mathcal{O}\left(\left|\frac{\lambda_1}{\lambda_2}\right|^3\right) \qquad \text{kde } \mathbf{x}^i \to \mathbf{x}^\infty \tag{3.3.21}$$

$$\{\|\theta_i - \lambda_1\|\} = \mathcal{O}\left(\left|\frac{\lambda_1}{\lambda_2}\right|^3\right) \tag{3.3.22}$$

Jelikož pro nesymetrické matice konverguje iterace Rayleighova koeficientu pouze kvadraticky, existuje pro tento případ následující modifikace počítající s odhadem levého a pravého vlastního vektoru, pro niž lze taktéž dokázat kubickou konvergenci (Ostrowski, 1959).

Algoritmus 3.3.3: Iterace Rayleighova koeficientu pro nesymetrické matice

1: $\hat{\mathbf{x}}_0 = \hat{\mathbf{y}}_0 =$ počáteční odhad levého a pravého vlastního vektoru
2: **for** i = 1,2... **do**
3: $\quad \mathbf{x}_i = \hat{\mathbf{x}}_{i-1}/\|\hat{\mathbf{x}}_{i-1}\|$ ▷ Znormalizuj
4: $\quad \mathbf{y}_i = \hat{\mathbf{y}}_{i-1}/\|\hat{\mathbf{y}}_{i-1}\|$
5: $\quad \theta_i = \dfrac{\langle \mathbf{x}_i | A\mathbf{y}_i \rangle}{\langle \mathbf{x}_i | \mathbf{y}_i \rangle}$ ▷ Spočti odhad vlastního čísla
6: $\quad$ **if** $\|A\mathbf{y}_i - \theta_i \mathbf{y}_i\| < e\theta_i$ **then**
7: $\quad\quad$ **break** ▷ Dostatečná konvergence
8: $\quad \hat{\mathbf{x}}_i^+ = \mathbf{x}_i^+(A - \theta_i I)^{-1}$ ▷ Iteruj levý...
9: $\quad \hat{\mathbf{y}}_i = (A - \theta_i I)^{-1}\mathbf{y}_i$ ▷ ...a pravý vlastní vektor

3.3.4 Mocninné metody a zobecněný problém vlastních čísel

Mocninnou metodou lze řešit pouze prostý problém vlastních čísel. Chceme-li řešit pomocí mocninné metody či některé její varianty zobecněný problém vlastních čísel

$$A\mathbf{x} = \lambda B\mathbf{x} \tag{3.3.23}$$

musíme ho převést na na prostý problém vlastních čísel

$$B^{-1}A\mathbf{x} = \lambda\mathbf{x} \tag{3.3.24}$$

Je-li možné levně spočíst inverzi B, a pokud inverze příliš neporuší řídkost matice B, lze použít tento přímý způsob. Vlastní vektory takto transformovaného problému jsou zároveň vlastními vektory problému původního. Tato varianta může být vhodná, je-li matice B (skoro) diagonální.

Není-li výhodné provést přímou inverzi matice B či rozklad, je možné podobně jako v inverzní mocniné metodě místo počítání inverze v každém kroku řešit rovnici:

$$B\mathbf{y} = A\mathbf{x} \qquad \text{pro neznámou } \mathbf{y} \tag{3.3.25}$$

I v tomto případě lze (při užití pouze malého počtu kroků vhodného iteračního řešiče) zachovat lineární konvergenci metody. (Golub – Ye, 1999) Užijeme-li na rovnici (3.3.25) iterační předpodmíněný řešič, získáme *předpodmíněnou mocninou metodu*. (Bai, 2000)

Mocninnou metodu pro zobecněný problém vlastních čísel lze aplikovat taktéž v blokové formě. Konvergenci lze urychlit užitím metody posunu a inverze: v tom případě převedeme zobecněný problém na prostý transformací

$$A' = (A - \sigma B)^{-1}B \tag{3.3.26}$$

Pokud neumíme provést levně inverzi matice B, je operace posunu a inverze obzvlášť vhodná, neboť výpočetní složitost „posunutého“ problému se příliš neliší od výpočetní složitosti problému převedeného dle rovnice (3.3.24).

Zhodnocení variant mocninné metody

O uvedených variantách mocninné metody lze říci zhruba totéž, co o mocninné metodě samotné: není-li velmi silný důvod pro jejich užití (ať již je to např. velmi snadná inverze matic tvaru $A - \theta I$, nebo naopak přílišný rozsah problému pro řešení pomocí jiných metod), bývají metody, které dále zmíníme, zpravidla rychlejší cestou k dosažení cíle.

To obzvláště platí pro zobecněný problém vlastních čísel, kde se zpravidla nelze vyhnout poměrně (vzhledem k náročnosti násobení matice vektorem) „drahému“ rozkladu matice.

3.4 Projekční metody I – krylovovské metody

Zatímco varianty mocninné metody se snažily o co nejlepší odhad vlastního vektoru, který v každém kroku zpřesňovaly, *projekční metody* na řešení vlastních čísel budují „vhodný malý“ podprostor, do kterého problém promítnou. „Vhodný malý“ znamená, že řešení promítnutého problému v tomto podprostoru musí být

snadno spočítatelné a zároveň pokud možno co nejbližší k řešení originálního problému.

Báze tohoto podprostoru se buduje inkrementálně, v prvním kroku tvoří bázi startovací vektor a v každém dalším kroku se k němu přidá další vektor, ortonormalizovaný vůči všem předešlým. Na rozdíl od metod iterace vektoru (nebereme-li v úvahu jejich blokové varianty) jsou tyto metody schopny spočítat více vlastních čísel najednou. Nejužívanější metody z této skupiny jsou založeny na *krylovovských prostorech* – budeme je tedy označovat *krylovovské metody*.

Krylovovské podprostory

Definice 39. Krylovovské podprostory matice A a vektoru $\mathbf{x}$ *jsou posloupností podprostorů* $K_n(A, \mathbf{x}), n \in 1, 2 \cdots, \infty$ *, kde* $K_n(A, \mathbf{x})$ *je generovaný bází*

$$B_{K_n(A,\mathbf{x})} = \text{ortonormalizuj}\{\mathbf{x}, A^1\mathbf{x}, A^2\mathbf{x}, \cdots, A^n\mathbf{x}\}$$

tyto báze se nazývají krylovovské báze.

Budou-li z kontextu jasné parametry A a $\mathbf{x}$, budeme tyto prostory a jejich báze značit pouze K_n, respektive B_{K_n}.

Metody založené na krylovovských prostorech jsou zobecněním mocninné metody (a v další kapitole uvedených minimalizačních metod) – stejně jako ony počítají $A^n\mathbf{x}_0$, zatímco však mocniná metoda užívá k výpočtu Ritzova vektoru pouze informaci z poslední iterace, metoda největšího spádu (kterou uvedeme dále) z posledních dvou, krylovovské metody užívají informace spočtené během všech iterací[8].

3.4.1 Arnoldiho metoda

Arnoldiho metoda je základní krylovovská iterační metoda na řešení problému vlastních čísel pro nesymetrické a nehermitovské matice. Jejím jádrem je *Arnoldiho faktorizace*: iterační procedura pro získání krylovovské báze a projekce matice A do podprostoru K_n v bázi B_{K_n}.

Arnoldiho faktorizace

První člen krylovovské báze v_0 je normalizovaný startovací vektor (vybraný náhodně nebo jako známá aproximace vlastního vektoru). Další členy báze $\mathbf{v}_i$ získáme ortonormalizací vektoru $A\mathbf{v}_{i-1}$ oproti předešlým vektorům báze.

Nyní odvodíme tvar matice A promítnuté do prostoru $K_n(A, \mathbf{x}_0)$ vzhledem k bázi $B_{K_n(A,\mathbf{x}_0)}$

$$H_n = V_n^H A V_n \tag{3.4.1}$$

Jelikož pro $i \in \{1, 2, \cdots, n\}$ platí

$$\mathbf{v}_i = A\mathbf{v}_{i-1} - \sum_{j=0}^{i-2} \alpha_j \mathbf{v}_j, \quad \text{kde } \alpha_j \in \mathbb{C} \tag{3.4.2}$$

[8]Přesněji všech iterací od posledního restartu, s implicitním restartem je situace ještě složitější.

matice H_n je v *horním Hessenbergově tvaru*: má nenulové prvky pouze v části nad hlavní diagonálou, na hlavní diagonále a v diagonále pod hlavní diagonálou.

Definice 40. *Matice $H \in \mathbb{C}^{n\times n}$ je v horním Hessenbergově tvaru, je-li $a_{i,j} = 0$ pro $i > j + 1$ (kde i označuje řádkový a j sloupcový index).*

Matici H_n budeme nazývat *Arnoldiho Hessenbergovou maticí*

Arnoldiho Hessenbergova matice Z konstrukce krylovovské báze vyplývá, že:

$$\forall i \in \{0, 1, \cdots, n-1\} : A\mathbf{v}_i \in K_n \tag{3.4.3}$$

Jelikož $K_n \subset K_{n+1}$, matice H_{n+1} lze získat z H_n přidáním jednoho řádku a jednoho sloupce. Pro matice H_n platí následující rekurentní vztah:

$$H_n = \left(\begin{array}{cccc|c} & & H_{n-1} & & \\ & & & & \\ 0 & 0 & \cdots & 0 & \alpha_n \end{array}\right) \left| V_n^H A\mathbf{v}^n \right), \quad \text{kde } \alpha_n = \langle \mathbf{v}_n \,|\, A\mathbf{v}_{n-1} \rangle \tag{3.4.4}$$

Vektor $(V_n)^H A\mathbf{v}^n$ tvořící poslední sloupec matice je vektorem skalárních součinů vektorů $\mathbf{v}_i$ s vektorem $A\mathbf{v}_n$.

Jelikož jsou tyto koeficienty matice H_n zároveň koeficienty užívanými v (popř. modifikovaném) Gram-Schmidtovu ortogonalizačním algoritmu, při výpočtu báze krylovovského prostoru se zároveň tvoří matice H_n.

Arnoldiho metoda

Arnoldiho metoda provádí n kroků Arnoldiho faktorizace, čímž zároveň počítá matici H_n. Jelikož pro vlastní vektor $\mathbf{u}^n \in R^n$ matice H_n platí

$$AV_n^T\mathbf{u} - \lambda V_n^T\mathbf{u} = (\alpha_{n+1}\mathbf{u}_n^n)\mathbf{v}_{n+1} \tag{3.4.5}$$

kde $\mathbf{u}_n^n$ je poslední složka vektoru $\mathbf{u}^n$, lze konvergenci kontrolovat pomocí podílu

$$\alpha_{n+1}/h_{j,j} \tag{3.4.6}$$

. Je-li tento podíl dostatečně malý, je krylovovský prostor dobrou aproximací invariantního prostoru matice A.

Pokud je v některém kroku Arnoldiho metody vektor $\mathbf{v}_n$ (a tedy koeficient $h_{j+1,j}$) nulový, $K_{n-1}(A, \mathbf{x}_0)$ je vlastní podprostor matice A. Z toho plyne, že vlastními vektory matice H_{n-1} jsou vlastní vektory matice A.

Algoritmus 3.4.1: Arnoldiho metoda

1: $\mathbf{v}_0$ = počáteční odhad vlastního vektoru, $|\mathbf{v}_0| = 1$
2: **for** $j = 0, 1, \ldots$ **do**
3: $\quad \mathbf{w}_i = A\mathbf{v}_i$
4: $\quad$ **for** $i = 0$ to j **do** ▷ Grand-Schmidtova ortogonalizace
5: $\quad\quad H_{j,i} = \langle \mathbf{w}_i \,|\, \mathbf{v}_j \rangle$ ▷ spočteme prvek z matice H_j
6: $\quad\quad \mathbf{w}_j = \mathbf{w}_j - h_{j,i}\mathbf{v}_i$ ▷ krok modifikovaného GS algoritmu
7: $\quad H_{j+1,j} = \|\mathbf{w}_j\|$ ▷ α z matice H_j
8: $\quad \mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$ ▷ přidáme ortonormalizované $\mathbf{w}_j$ do báze
9: **end until**$(\alpha_{j+1,j}/h_{j,j} < e)$

Takto prováděná ortogonalizace vektorů $\mathbf{v}_i$ využívá modifikovaného *Gram-Schmidtova* (dále MGS) algoritmu, který je numericky stabilnější než původní Gram-Schmidtův algoritmus.

3.4.2 Reortogonalizace

Při praktické implementaci se však ukazuje, že i při použití MGS dochází kvůli zaokrouhlovacím chybám ke ztrátě ortogonality vektorů $\mathbf{v}_i$. Tento problém se řeší reortogonalizacemi. (Parlett, 1980) Jelikož je ortogonalizace nejdražší částí iterace, existují testy, které zjišťují, zda je v tomto kroku reortogonalizace potřeba. Tato technika se nazývá selektivní reortogonalizace. (Daniel et al., 1976)

Je vcelku zřejmé (Parlett – Scott, 1979), že ztráta ortogonality vzrůstá s přibývajícími iteracemi. Největší defekty v ortogonalitě přitom mají směr nejlépe zkonvergovaných Ritzových vektorů (Simon, 1982). Proto jedna z účinných selektivních reortogonalizcí má následující schéma: Provede se m kroků Arnoldiho metody, spočte aproximace vlastních vektorů a z nich vybere ty, které aproximují vlastní vektory dostatečně přesně. Pak se provádí další kroky Arnoldiho algoritmu, ve kterých se nové bázové vektory reortogonalizují proti vybraným aproximacím vlastních vektorů. (Mazzia, 1998)

Další možností, jak omezit ztrátu ortogonality je místo MGS užít pro ortogonalizaci proceduru *Householder ortogonalization* (Walker, 1988).

Householder ortogonalization je výpočetně dražší ($\mathcal{O}(4m^2n - 4/3m^3)$, zatímco MGS má složitost $\mathcal{O}(2m^2n)$ až $\mathcal{O}(4m^2n)$, dle počtu reortogonalizací). Dle některých autorů je však numericky stabilnější a proto vhodná obzvláště pro Arnoldiho faktorizaci k řešení problémů vlastních čísel (Saad, 2003), v praxi se však householder ortogonalization pro tyto účely příliš nepoužívá, což svědčí spíše proti tomuto tvrzení. V praxi se spíše používá MGS spolu se selektivními reortogonalizacemi.

3.4.3 Arnoldiho metoda s částečnou ortogonalizací a restartovaná Arnoldiho metoda

Nevýhodou Arnoldiho metody je, že s přibývajícími iteracemi příliš vzrůstají paměťové nároky algoritmu: algoritmus musí uchovávat vektory $\mathbf{v}_i$ a matici H_n a pamětová náročnost algoritmu je tedy $\mathcal{O}(nm + m^2)$, kde n je dimenze matice A a m je počet iterací.

Jednou z metod na snížení paměťové i výpočetní složitosti problému je *Arnoldiho metoda s částečnou reortogonalizací*. (Saad, 2003) V té se provádí ortogonalizace pouze vůči posledním l vektorům. Paměťová složitost pak klesne na $\mathcal{O}(nl + m^2)$.

Explicitně restartovaná Arnoldiho metoda Druhou, častěji zmiňovanou variantou algoritmu snižující jeho paměťové nároky, je explicitně restartovaná Arnoldiho metoda.[9]

Explicitně restartovaná Arnoldiho metoda je tvořena cyklem, v jehož každém kroku se vždy provede k kroků Arnoldiho metody. Pak se zvolí vhodná lineární

[9] Jelikož se užívá spíše *implicitně restartovaná Arnoldiho metoda*, popíšeme explicitně restartovanou metodu jen stručně, přesnější vyjádření zde nalézt v (Bai, 2000)

kombinaci vektorů $\mathbf{x}_{j+1} = \sum_{i=0}^{k} \mathbf{v}_i$, která se použije jako startovací vektor Arnoldiho metody v příští iteraci. Tento cyklus probíhá, dokud není dosažena požadovaná přesnost vlastních vektorů.

Existuje více metod, jak zkonstruovat nový startovací vektor. Jednou z možností je vzít lineární kombinaci Ritzových vektorů (odhadů vlastních vektorů), které jsou nejblíže žádaným vlastním číslům. (Saad, 1984)

Velmi často se na konstrukci lineární kombinace vektorů $\mathbf{v}_i$, jež je užita jako nový startovací vektor, pohlíží jako na vytvoření polynomu $\mathbf{p}(A)\mathbf{x}_j$. Některé metody explicitního restartu konstruují takový polynom, který z vektoru $\mathbf{x}_j$ odstraní „nežádoucí“ vlastní vektory (Saad, 2003). Vlastní páry matice H_i rozdělí na „žádané“ $[\mathbf{w}_i, \theta_i]$ a „nechtěné“ $[\tilde{w}_i, \tilde{\theta}_i]$. Polynom $\mathbf{p}$ se pak sestaví tak, aby

$$\mathbf{p}A(\mathbf{w}_i) \neq 0 \qquad \text{a zároveň} \qquad \mathbf{p}A(\tilde{w}_i) = 0 \tag{3.4.7}$$

Nejužívanější explicitní metody využívají na konstrukci polynomu $\mathbf{p}$ *Chebychevových polynomů*. (Chatelin – Ho, 1990; Saad, 1984)

Algoritmus 3.4.2: Explicitně restartovaná Arnoldiho metoda

1: $\mathbf{x}_0$ = počáteční odhad vlastního vektoru, $|\mathbf{x}_0| = 1$
2: **for** i = 0,1,2... **do**
3: Proveď k kroků Arnoldiho metody se startovacím vektorem $\mathbf{x}_i$
4: Spočti W^k = vlastní vektory H_i^k a zkontroluj konvergenci
5: **if** metoda konverguje **then**
6: **break**
7: Zkonstruuj polynom $\mathbf{p}$
8: $\mathbf{x}_{i+1} = \mathbf{p}(A)x_i$

3.4.4 Implicitně restartovaná Arnoldiho metoda

Častěji užívanou variantou Arnoldiho metody je implicitně restartovaná Arnoldiho metoda. V této variantě Arnoldiho metody se nejprve provede $m = k + l$ kroků Arnoldiho faktorizace. Pak je prováděn cyklus, v jehož j-tém kroku se neprve zkomprimuje spočtený krylovovský prostor $K_m(A, \mathbf{x}_0^{(j)})$ na krylovovský prostor $K_k(A, \mathbf{x}_0^{(j+1)})$ a následně opět provede l kroků Arnoldiho metody. Tento cyklus probíhá, dokud se ve spektru matice $H_m^{(j)}$ neobjeví dostatečně přesné vlastní vektory odpovídající požadovaným vlastním číslům.

Komprimace krylovovského prostoru spočívá ve vytvoření unitární matice $Q^{(j)}$ která transformuje krylovovskou bázi a matici $H_m^{(j)}$. Prvních k vektorů z transformované báze pak tvoří bázi pro další iterační krok. Matice $Q^{(j)}$ je tvořena l kroky *implicitního QR algoritmu* (*Implicitly Shifted QR Iteration* (Demmel, 1997))

Implicitní QR algoritmus Implicitní QR algoritmus je iterační procedurou pro výpočet (částečného) Schurova rozkladu matice H (viz definice 35) v horním Hessenbergově tvaru. Tento algoritmus začne s maticemi $M_0 = H, Q_0 = I$ a ve svém i-tém kroku vypočte matici $Q_i \in \mathbb{C}^{n\times n}$ a matici $M_i = Q_i^H M_{i-1} Q_i$, které splňují následující vlastnosti:

– $Q_i^H Q_i = 1$
– Prvních i řádek a sloupců matice M_i a prvních i sloupců matice Q_i tvoří částečný Schurův rozklad matice H, přičemž zbytek matice M_i zůstává v horním Hessenbergově tvaru.

Každý krok implicitního QR algoritmu tedy „odstraní“ prvek pod diagonálou v i-tém sloupci matice H, čímž na diagonále „vznikne“ vlastní číslo. Tuto transformaci provádí standardní implicitní QR algoritmus pomocí vnitřního iteračního cyklu, během kterého algoritmus „odhaduje“ hodnotu vlastního čísla na diagonále a se zpřesňujícím se odhadem konverguje prvek pod diagonálou k nule. Při použití v implicitně restartované Arnoldiho metodě je však hodnota tohoto vlastního čísla předem známá, proto tento cyklus zkonverguje v prvním kroku. (Demmel, 1997)

Implicitní restart Implicitní restart Arnoldiho metody rozdělí spektrum matice $H_m^{(j)}$ na na k chtěných a l nechtěných vlastních čísel. Nechtěná vlastní čísla se pak užijí k l krokům *QR iterace*, čímž získáme matici $Q^{(j)} = \prod_{i=0}^{l-1} Q_i$.

Matice $\bar{H}^{(j)} = (Q^{(j)})^+ H_m^{(j)} (Q^{(j)})$ má na diagonále od k plus prvního prvku nechtěná vlastní čísla (a pod nimi nuly). $H_k^{(j+1)}$ se utvoří jako levá horní čtvercová submatice tvaru $l \times l$ z matice $\bar{H}^{(j)}$. Díky QR algoritmu má matice $H_k^{(j+1)}$ ve svém spektru pouze chtěná vlastní čísla.

Věta 6. *Nechť $V_m^{(j)} \in \mathbb{C}^{n\times m}$ je krylovovská báze a $H_m^{(j)} \in \mathbb{C}^{m\times m}$ jí odpovídající Arnoldiho Hessenbergova matice. Nechť $Q^{(j)}$ matice tvořená l kroky implicitní QR iterace.*

Pak báze $V_k^{(j+1)}$ tvořená prvními k sloupci transformované báze $W_m^{(j)} = Q^{(j)} V_m^{(j)}$ a matice $H_k^{(j+1)}$ získaná jako levý horní blok velikosti $k \times k$ matice $(Q^{(j)})^+ H_m^{(j)} Q^{(j)}$ tvoří bázi krylovovského prostoru $K_k(A, \mathbf{x}_0^{(j+1)})$ a jí příslušnou Arnoldiho Hessenbergovu matici.

Důkaz. Ověříme, že $V^{(j+1)}$ splňuje definici krylovovské báze. Víme, že (pro jednoduchost vynecháme indexy $^{(j)}$):

$$AV_m = V_m H_m + \mathbf{w}\mathbf{e}_m^T \tag{3.4.8}$$

kde $\mathbf{e}_m$ je sloupcový vektor délky m, mající poslední prvek rovný jedné a ostatní nule a

$$\mathbf{w} = (A - V_m V_m^+ A)\mathbf{v}_m \tag{3.4.9}$$

$\mathbf{w}\mathbf{e}_m^T$ tedy vyjadřuje rozdíl mezi tím, jak krylovovskou bázi zobrazí matice A a jak její průmět do krylovovského prostoru. Díky struktuře krylovovského prostoru se tyto matice liší pouze v zobrazení posledního vektoru krylovovské báze a matice $\mathbf{w}\mathbf{e}_m^T$ má nenulový pouze poslední sloupec.

$$AV_m Q = V_m H_m Q + \mathbf{w}\mathbf{e}_m^T Q \tag{3.4.10}$$

Vezmeme-li $$W = V_n Q \tag{3.4.11}$$

pak $$AV_m Q = AV_m Q (Q)^+ H_m Q + \mathbf{w}\mathbf{e}_m^T Q \tag{3.4.12}$$

$$AW_m = W_m Q^+ H_m Q + \mathbf{w}\mathbf{e}_m^T Q \tag{3.4.13}$$

Jelikož matice Q_i generované implicitní QR iterací mají horní Hessenbegrovu strukturu, matice $\mathbf{we}_m^T Q = \mathbf{we}_m^T Q_1 Q_2 \cdots Q_k$ má prvních $k-1$ sloupců nulových (každé násobení maticí Q_i zmenší počet nulových sloupců o jeden).

Jelikož $H_m^{(j)}$ je v horním Hessenbergově tvaru a QR iterace tento tvar zachovává, je v tomto tvaru i matice $(Q^{(j)})^+ H_m^{(j)}(Q^{(j)})$. □

Implicitní restart je ekvivalentní explicitnímu restartu, kde:

$$\mathbf{x}_0^{(j+1)} = \mathbf{p}(A)\mathbf{x}_0^{(j)} \tag{3.4.14}$$

Polynom $\mathbf{p}(\alpha) = \prod_{i=1}^{l}(\alpha - \mu_i)$, kde koeficienty μ_i jsou posuny užité v QR iteraci (ušetří však provádění k kroků Arnoldiho metody).

Algoritmus 3.4.3: Implicitně restartovaná Arnoldiho metoda

1: Získej V_m, H_m pomocí m kroků Arnoldiho metody
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Spočti spektrum σ matice H_m
4: **if** konvegrence **then**
5: **break**
6: Rozděl σ na chtěná vlastní čísla σ^+ a nechtěná σ^-, kde $|\sigma^-| = l$
7: Q = I
8: **for** j = 1 to l **do**
9: Rozlož $H_m - \sigma_j^- I = Q_j R$ ▷ krok QR iterace
10: $Q = QQ_j$
11: $H_k = Q^T H_m Q(1:k, 1:k)$ ▷ Levý horní obdélník velikosti $k \times k$
12: $V_k = V_m Q(:, 1:k)$ ▷ Prvních k sloupců
13: Proveď dalších l kroků Arnoldiho metody s bází V_k a Hessenbergovou maticí H_k

Detailnější popis implicitní restartované metody lze najít v této literatuře: (Bai, 2000; Sorensen, 1996; Lehoucq et al., 1997; Sorensen, 1992).

Konvergence Arnoldiho metody a její užití

Konvergence Arnoldiho algoritmu je lineární. Tato metoda je zobecněním mocninné metody na metodu projekčního typu, dosahuje tedy přinejhorším stejného konvergenčního faktoru. V praxi dosahuje rychlejší konvergence a existují ostřejší odhady konvergence, které však už nelze jednoduše zapsat.

Jelikož je krylovovský prostor invariantní vůči posunu a násobení matice skalárem, rychlost konvergence není podmíněna podílem maximálního a minimálního vlastního čísla, nýbrž rozložením vlastních čísel matice. Algoritmus má přitom tendenci nejrychleji konvergovat k *vnějším vlastním vektorům* – tedy vektorům odpovídajícím vlastním číslům na pokraji spektra matice. Dobré „oddělení“ vnějších vlastních vektorů urychluje konvergenci. (**?**) Analýzu konvergence lze nalézt např. v Saadově práci. (Saad, 1992)

Výhody a nevýhody Arnoldiho metody Ze zde uvedených metod je nejnovější – a zároveň asi nejpoužívanější *implicitně restartovaná Arnoldiho metoda*.

Výhodou všech variant Arnoldiho algoritmu je dobrý teoretický základ, na kterém jsou postaveny spolehlivé a dobře ověřené knihovny.

Nevýhodou tohoto algoritmu je, že jeho činnost nelze příliš ovlivnit, a nelze tedy při výpočtu využít speciálních vlastností matice (je běžné, že matice vzniklé z různých zadání stejného problému mají podobné vlastnosti, které lze využít např. k tvorbě vhodného předpodmínění).

Tento algoritmus také (díky kvadraticky se zvětšující časové a paměťové náročnosti) nemusí dostačovat k řešení problémů velkých dimenzí.

V tomto algoritmu jsou v podstatě pouze tři místa, kde lze využít znalosti problému. Prvním je možnost (obzvláště pro zobecněný problém vlastních čísel) předpodmínění problému: např. místo problému $A\mathbf{x} = \lambda B\mathbf{x}$ můžeme řešit problém $CA\mathbf{x} = \lambda CB\mathbf{x}$, který může mít pro řešení pomocí Arnoldiho algoritmu lepší vlastnosti.

Druhou, a asi nejdůležitější, možností je výběr startovacího vektoru: je-li možné alespoň přibližně odhadnout žádaný vlastní vektor nebo alespoň vektor s nevelkým počtem nenulových složek v Jordanově kanonické bázi, algoritmus zkonverguje mnohem rychleji.

Třetí možností jak ovlivnit chování algoritmu je speciální výběr nového startovacího vektoru při restartu. Tato možnost je spíše teoretická, neboť již hotové knihovny si většinou řídí restart samy a nedávají uživateli možnost výběr tohoto vektoru ovlivnit.

Užití tohoto algoritmu je tedy v podstatě „sázka na jistotu“: algoritmus výsledek spočítá s jistotou a rychleji než mocninná metoda, pravděpodobně to však nebude nejrychlejší algoritmus na daný specifický problém.(Sadkane, 1992)

Bloková verze Arnoldiho algoritmů

Arnoldiho metoda a metody od něj odvozené (implicitně restartovaná Arnoldiho metoda, nesymetrická Lanczosova metoda...) lze implementovat i v blokové variantě. V tomto případě se místo startovacího vektoru $\mathbf{v}_1$ iteruje báze startovacího podprostoru V_1 a v každém kroku se buduje ortonormalizovaná báze podprostoru ortogonálního vůči všem předchozím prostorům. Matice H_n pak není Hessenbergova (popř. tridiagonální pro Lanczosovu metodu), ale blokově-Hessenbergova (popř. tridiagonální). Detaily k blokové implementaci lze nalézt v (Bai, 2000; Saad, 2003; Sadkane, 1992; Harrar II, 1998).

3.4.5 (Symetrická) Lanczosova metoda

Lanczosova metoda[10] je v podstatě Arnoldiho metoda pro případ hermitovské či symetrické matice. Dále budeme pracovat s variantou pro hermitovské matice, pro symetrickou matici bude algoritmus identický s výjimkou chybějícího operátoru komplexní sdruženosti v některých rovnicích, záměny operátoru $^+$ za operátor ^{T} a z této změny vyplývající změny znaménka, neboť

$$\mathbf{w}^T\mathbf{w} = \mathbf{w}\mathbf{w}^T, \text{ ale } \mathbf{w}^+\mathbf{w} = -\mathbf{w}\mathbf{w}^+ \tag{3.4.15}$$

Jelikož je matice A hermitovská, tak musí být i $H_i = V_i^+ AV$ hermitovská (tento fakt vychází z podobnosti těchto matic). Protože je však zároveň v horním

[10]Tato metoda je také nazývaná *symetrickou Lanczosovou metodou*, pro rozlišení s *nesymetrickou Lanczosovou metodou*, kterou zmíníme dále.

Hessenbergově tvaru, musí být tridiagonální. Z toho plyne, že vektor $A(\mathbf{v}_i) = \mathbf{v}_{i+1} + \alpha_i \mathbf{v}_i + \beta_i \mathbf{v}_{i-1}$. Díky tomu je třeba ortogonalizovat vektor $\mathbf{v}_i$ pouze vůči dvěma předchozím vektorům, což snižuje výpočetní složitost algoritmu. Zároveň díky hermitovskosti platí $A_{i-1,j} = (A_{i,j-1})^+$. Matice H_n má tedy následující tvar:

$$H_n = \begin{pmatrix} \alpha_1 & \overline{\beta}_1 & & & & \\ \beta_1 & \alpha_2 & \overline{\beta}_2 & & & \\ & \beta_2 & \alpha_3 & \cdots & & \\ & & \cdots & \cdots & \cdots & \\ & & & \cdots & \alpha_{n-1} & \overline{\beta}_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix} \tag{3.4.16}$$

Reortogonalizace a Implementace Lanczosova algoritmu a restart

Stejně jako v Arnoldiho algoritmu dochází i u Lanczosova algoritmu ke ztrátě ortogonality. To bohužel kazí jinak teoreticky velmi úspornou metodu, v které by v exaktní aritmetice stačilo pouze počítat koeficienty α_i a β_i a v paměti držet pouze poslední dva vektory. Vzhledem k nutnosti reortogonalizace to neplatí a musí se používat reortogonalizační techniky již popsané u Arnoldiho algoritmu.

Vzhledem k nutnosti provádět reortogonalizace a tedy uchovávat spočtené vlastní vektory je i u Lanczosova algoritmu někdy třeba provést restart. Některé implementace (Marques, 1997) používají jednoduché techniky, kdy se dostatečně zkonvergované vektory zamknou a vyberou se zcela nové startovací vektory. Je taktéž možno použít technik implicitního či explicitního restartu, používaného u Arnoldiho algoritmu: aplikace těchto technik se neliší, a tak se jimi nebudeme již hlouběji zabývat.

Implementace Lanczosovy metody

Lanczosův algoritmus buduje iteračně matici H_n a ukládá vektory $\mathbf{v}_i$ pro potřeby reortogonalizací. Kvůli nutnosti reortogonalizí může vzniknout i u této metody nutnost restartu – jelikož se pro restart používají stejné techniky jako u Arnoldiho metody, nebudeme se tématu více věnovat a uvedeme zde pouze základní schéma.

Algoritmus 3.4.4: Lanczosova metoda (bez restartu)

1: v_1 = počáteční odhad vlastního vektoru, $|\mathbf{v}_1| = 1$
2: **for** $j = 1, 2 \ldots$ **do**
3: $\quad \mathbf{w}_i = A\mathbf{v}_i$
4: $\quad \alpha_i = \langle \mathbf{w}_i | \mathbf{v}_i \rangle$
5: $\quad \beta_i = \|\mathbf{w}_i - \alpha_i \mathbf{v}_i\|$
6: $\quad \mathbf{v}_{i+1} = \mathbf{w}_i - \alpha_i \mathbf{v}_i / \beta_i$
7: $\quad$ Je-li třeba, reortogonalizuj
8: **end until** konvergence

Výhody a nevýhody Lanczosova algoritmu

Na rozdíl od Arnoldiho algoritmu má Lanczosův algoritmus bez reortogonalizací lineární časovou i paměťovou složitost, a je tedy vhodný i pro problémy

velkých dimenzí. V praxi se však ukazuje, že kvůli ztrátě konvergence jsou reortogonalizace nutností. Proto je paměťová náročnost stejná jako u Arnoldiho algoritmu, a to kvadratická. Pokud by se reortogonalizace prováděly v každém kroku algoritmu, byla by shodná i výpočetní složitost; vzhledem k tomu, že lze provádět pouze částečné a selektivní reortogonalizace a že jsou vypracovány vcelku spolehlivé heuristiky, kdy je nutné reortogonalizace provádět, což počet nutných reortogonalizací snižuje na minimum, jde o spolehlivý a ověřený algoritmus s rychlou a jistou konvergencí.

3.4.6 Nesymetrická Lanczosova metoda

Nesymetrická Lanczosova metoda je adaptací Lanczosova algoritmu na nesymetrické (a nehermitovské) matice. Stejně jako *symetrická Lanczosova metoda* tvoří krylovovskou bázi prostoru, do kterého se matice A promítne jako tridiagonální matice. Zatímco Arnoldiho a Lanczosova metoda pracovala s kolmou projekcí na prostor $K_n(A, \mathbf{x}_0)$, nesymetrická Lanczosova metoda užívá šikmou projekci na prostor $K_n(A, \mathbf{v}_0)$ s bází V_n ve směru kolmém k prostoru $K_n(A^T, \mathbf{w}_0)$ s bází W_n, přičemž udržuje báze V_n a W_n biortogonální.

Definice 41. *Báze V a W vektorového prostoru dimenze n jsou* biortogonální*, pokud*

$$\forall i \in \{1, 2, \cdots, n\} \ \langle \mathbf{v}_i | \mathbf{w}_i \rangle = 1 \text{ a zároveň}$$
$$\forall i \in \{1, 2, \cdots, n\}, \forall j \in \{1, 2, \cdots, n\}, j \neq i : \langle \mathbf{v}_i | \mathbf{w}_j \rangle = 0$$

Šikmý průmět matice A – matice H_n má tento tvar (Saad, 2003):

$$H_n = W_n^T A V_n \tag{3.4.17}$$

Věta 7. *Je-li $A \in \mathbb{C}^{d \times d}$ komplexní matice a V_n a W_n biortogonální báze krylovovských prostorů $K_n(A, \mathbf{v}_0)$ a $K_n(A^T, \mathbf{w}_0)$, pak je matice $H_n = W_n^T A V_n$ tridiagonální.*

Důkaz. Nechť dimenze matice A je d. Vezměme hermitovskou matici

$$B^{2d \times 2d} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \tag{3.4.18}$$

pak

$$H = \begin{pmatrix} W_n \\ V_n \end{pmatrix}^T B \begin{pmatrix} W_n \\ V_n \end{pmatrix} = \begin{pmatrix} W_n^T & V_n^T \end{pmatrix} \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} W_n \\ V_n \end{pmatrix} \tag{3.4.19}$$

$$= \begin{pmatrix} W_n^T & V_n^T \end{pmatrix} \begin{pmatrix} AV_n \\ A^T W_n \end{pmatrix} = W_n^T A V_n + V_n^T A^T W_n \tag{3.4.20}$$

Z definice krylovovské báze a biortogonality platí

$$\forall i, j \in \{1, 2, \cdots, n\}, j > i + 1 : \mathbf{w}_j A \mathbf{v}_i = 0 \tag{3.4.21}$$

obdobně pro A^T a $\mathbf{w}_i$. Matice $W_n^T A V_n$ a $V_n^T A^T W_n$ jsou tedy díky biortogonalitě bází V_n a W_n v horním Hessenbergově tvaru. Jelikož $(AB)^T = B^T A^T$, matice $V_n^T A^T W_n = (W_n^T A V_n)^T$ je zároveň v dolním Hessenbergově tvaru. Z toho plyne, že $W_n^T A V_n$ je tridiagonální. □

Uvedená věta zároveň ukazuje „myšlenku“ tohoto algoritmu: „zesymetričtění“ matice A za cenu zdvojnásobení její dimenze. Na této symetrické matici B se provádí pseudo-Lanczosův algoritmus, užívající „pseudo-krylovovské“ báze $\begin{pmatrix} W_n \\ V_n \end{pmatrix}$, kde W_n a V_n jsou krylovovské báze. Díky této volbě báze lze rozložit matici H_n na součet dvou matic, které jsou navzájem svojí transpozicí. Algoritmus tak stačí provádět pouze na jednom kvadrantu matice B při zachování tridiagonality matice H_n.

Implementace nesymetrického Lanczosova algoritmu

Nesymetrický Lanczosův algoritmus tedy vytváří nesymetrickou tridiagonální matici:

$$H_n = \begin{pmatrix} \alpha_1 & \gamma_2 & & & & \\ \beta_2 & \alpha_2 & \gamma_3 & & & \\ & \beta_3 & \alpha_3 & \cdots & & \\ & & \cdots & \cdots & \cdots & \\ & & & \cdots & \alpha_{n-1} & \gamma_n \\ & & & & \beta_n & \alpha_n \end{pmatrix} \tag{3.4.22}$$

Tato matice se tvoří následujícím algoritmem:

Algoritmus 3.4.5: Nesymetrická Lanczosova metoda

1: $\mathbf{v}_1, \mathbf{w}_1 =$ počáteční odhady vlastních vektorů matic A a A^+, $\langle \mathbf{v}_1 | \mathbf{w}_1 \rangle = 1$
2: $\beta_0 = \gamma_1 = 0$ ▷ První bázové vektory se neortogonalizují
3: **for** $j = 1, 2 \ldots$ **do**
4: $\quad \alpha_j = \langle A\mathbf{v}_j | \mathbf{w}_j \rangle$
5: $\quad \hat{\mathbf{v}}_{j+1} = A\mathbf{v}_j - \alpha_j \mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$ ▷ Biortogonalizace
6: $\quad \hat{\mathbf{w}}_{j+1} = A^T \mathbf{w}_j - \alpha_j \mathbf{w}_j - \gamma_j \mathbf{w}_{j-1}$
7: $\quad \pi = \langle \hat{\mathbf{v}}_{j+1} | \hat{\mathbf{w}}_{j+1} \rangle$
8: $\quad$ **If** $\pi = 0$ **then break**
9: $\quad \gamma_{j+1} = \sqrt{|\pi|}$ ▷ Volba β_{j+1} a γ_{j+1} tak,
10: $\quad \beta_{j+1} = \langle \hat{\mathbf{v}}_{j+1} | \hat{\mathbf{w}}_{j+1} \rangle / \gamma_{j+1}$ ▷ . . . že $\langle \mathbf{v}_{j+1} | \mathbf{w}_{j+1} \rangle = 1$
11: $\quad \mathbf{v}_{j+1} = \hat{\mathbf{v}}_{j+1} / \beta_{j+1}$
12: $\quad \mathbf{w}_{j+1} = \hat{\mathbf{w}}_{j+1} / \gamma_{j+1}$
13: **end until** konvergence

Na řádce 8 a 9 lze zvolit β_{j+1} a γ_{j+1} libovolně tak, aby byla zachována biortogonalita bází ($\langle \mathbf{v}_{j+1} | \mathbf{w}_{j+1} \rangle = 1$). Nejjednodušší je užít takovou volbu, aby pokud možno zachovávala vektory $\mathbf{v}_i$ a $\mathbf{w}_i$ „rozumně velké“ (kvůli numerické stabilitě v aritmetice s konečnou přesností).

Na rozdíl od Arnoldiho a Lanczosovy metody není u této metody zaručená konvergence. Pokud v některém kroku bude $\langle \hat{\mathbf{v}}_i | \hat{\mathbf{w}}_i \rangle = 0$, algoritmus zkolabuje, aniž by došlo ke konvergenci, neboť nebude možno korektně definovat vektory $\mathbf{v}_i, \mathbf{w}_i$. Tuto situaci lze řešit např. pomocí restartu.

Druhou variantou pro řešení tohoto problému, doporučovanou právě pro užití

nesymetrického Lanczosova algoritmu pro počítání vlastních čísel,[11] je *Lanczosův algoritmus s předpovědí* (*Look-ahead Lanczos Method*), kdy se algoritmus vyrovnává s neexistencí vektorů $\mathbf{v}_i, \mathbf{w}_i$ tak, že je přeskočí a snaží se definovat biortogonální vektory $\mathbf{v}_{i+1}, \mathbf{w}_{i+1}$, a pokud to nelze tak $\mathbf{v}_{i+2}, \mathbf{w}_{i+2}$ atd... (Saad, 2003) Další možnost je vektory $\mathbf{v}_i, \mathbf{w}_i$ do báze zahrnout, ale neortogonalizovat je vůči $\mathbf{v}_{i-1}$ respektive $\mathbf{w}_{i-1}$.(Freund – Nachtigal, 1992). Oba tyto postupy však vedou k tomu, že matice H_n nebude tridiagonální, nýbrž blokově tridiagonální. Při užití těchto technik není zaručeno, že se podaří nalézt vektory $\mathbf{v}_{i+n}, \mathbf{w}_{i+n}$, které vyhovují pro další pokračování algoritmu. V tom případě nezbývá, než provést některou formu restartu.

Problémy nastávají i pokud jsou vektory $\mathbf{v}_i, \mathbf{w}_i$ pouze „skoro“ kolmé

$$\langle \mathbf{v}_i \,|\, \mathbf{w}_i \rangle \doteq 0 \tag{3.4.23}$$

neboť se tím (v aritmetice s konečnou přesností) velmi zvýší nepřesnost vnášená zaokrouhlovacími chybami. I v tomto případě může být lepším řešením užít techniku *look-ahead* (a v případě selhání popř. provést restart).

Výhody a nevýhody nesymetrického Lanczosova algoritmu

K výhodám tohoto algoritmu patří především menší paměťová náročnost a z toho vyplývající možnost počítat s většími prostory – tím se sníží či přímo eliminuje počet nutných restartů algoritmu (čímž se sníží počet nutných operací); zároveň je jedna iterace kratší než u Arnoldiho algoritmu.

Nevýhodou tohoto algoritmu je zatím nedostatečné prozkoumání jeho vlastností, dále fakt, že není zaručena konvergence, a také větší náchylnost ke vzniku chyb způsobených aritmetikou s konečnou přesností, což zvyšuje četnost nutných reortogonalizací a restartů. Jde tedy o rychlý algoritmus vhodný i na problémy větších dimenzí, na některých typech problémů se ale mohou projevit uvedené negativní vlastnosti.

3.4.7 Krylovovské metody pro zobecněný problém vlastních čísel

Jelikož všechny krylovovské metody jsou vhodné především pro prostý problém vlastních čísel, jejich adaptace na zobecněný problém vlastních čísel pouze nějakým způsobem převádí zobecněný problém na prostý. Proto jsou tyto postupy společné pro všechny krylovovské algoritmy. Uvažujme tedy zobecněný problém vlastních čísel ve tvaru

$$A\mathbf{x} = \lambda B\mathbf{x}, \quad \text{kde } \mathbf{x} \in \mathbb{C}^n \setminus \{0\}, \quad A, B \in \mathbb{C}^{n\times n} \tag{3.4.24}$$

Je-li matice B hermitovská pozitivně definitní, pak je v uvedených variantách krylovovských metod pro zobecněný problém vlastních čísel výhodné užít *B-ortogonalizace* (viz kapitola 1.2.2). *Arnoldiho algoritmus s B-ortogonalizací* je standardní Arnoldiho algoritmus, v němž je skalární součin (v ortogonalizační

[11] Metody založené na budování tridiagonální matice lze s úspěchem použít i na řešení systému lineárních rovnic.

smyčce) nahrazen *B-skalárním součinem*. Díky tomu jsou bázové vektory krylovovského prostoru B-ortogonální a zobecněný problém vlastních čísel se v promítnutém prostoru redukuje na problém prostý.(Meerbergen – Spence, 1997; Booten et al., 1996)

Jednoduché adaptace krylovovských algoritmů na zobecněný problém

Nejjednodušší varianta Arnoldiho algoritmu pro zobecněný problém vlastních čísel se téměř neliší od varianty pro prostý problém, jediným rozdílem je, že v krylovovském prostoru se řeší zobecněný problém vlastních čísel. Tato metoda ovšem nevyužívá vlastností matice B a je tedy vhodná hlavně pro případy, kdy B je „blízká“ identické matici.

Častěji citovanou variantou Arnoldiho algoritmu pro řešení zobecněného problém vlastních čísel je provádět (explicitně či implicitně) restartovanou Arnoldiho metodu s krylovovskými prostory $K_m(A - \theta_i B, \mathbf{x}_i)$, kde $\mathbf{x}_i$ je dominantní vektor Arnoldiho Hessenbergovy matice z předchozí iterace (pro $i = 0$ libovolný odhad) a θ_i jemu odpovídající vlastní číslo. Pro symetrické matice A, B se tak při každém restartu maximalizuje Rayleighův koeficient na krylovovském prostoru. Tato varianta je zobecněním metody největšího sestupu a tudíž lze (pro symetrické matice) dokázat její konvergenci. Rychlost konvergence této metody lze vylepšit užitím předpodmínění. (Golub – Ye, 2002)

Arnoldiho metoda pro zobecněný problém vlastních čísel s purifikací

V případě, že A není singulární, je možné užít metody posunu a inverze: aplikovat příslušnou Arnoldiho metodu pro prostý problém vlastních čísel na matici (Lehoucq – Scott, 1997)

$$S = (A - \sigma B)^{-1} B \tag{3.4.25}$$

Parametr $\sigma \in \mathbb{C}$ je posun (shift), se stejným významem jako v předchozí variantě algoritmu. Vzhledem k náročnosti inverze (rozkladu) se zpravidla užívá jedna matice S pro všechny restarty.

Je-li B singulární, mohou se v promítnuté matici S objevit takzvané *podivné vlastní vektory* – takto někteří autoři nazývají vlastní vektory matice H_n, které odpovídají vlastním vektorům příslušným k nulovým vlastním číslům matice S (a tedy odpovídající „nezajímavým“ nekonečným vlastním vektorům původního problému). Kvůli zaokrouhlovacím chybám v aritmetice s konečnou přesností nemusí být vypočtené vlastní číslo náležící těmto vektorům nulové. Ze stejného důvodu se může objevit i vícenásobný výskyt jednoho vlastního vektoru původního problému. Je možné ukázát (Rommes, 2006), že užití B-ortogonalizace část těchto vektorů odstraní.

Pomocí B-ortogonalizace je možné odstranit pouze některé podivné vlastní vektory. Vezme-li se za startovací vektor vektor $S^2\mathbf{x}_0$, tyto vektory vymizí (zjednodušeně řečeno se ve vektoru $\hat{\mathbf{x}} = S^2\mathbf{x}_0$ zobrazí na nulový vektor). Aritmetika s konečnou přesností a s ní spojená ztráta ortogonality však může zapříčinit, že se tyto vektory ve spektru matice H_n opět objeví. Proto se spolu s B-ortgonalizací a násobením startovacího vektoru užívá *purifikace*.

Arnoldiho metoda s purifikací na konci algoritmu (respektive při každém restartu) spočítá vlastní páry $(\mathbf{y}_i, \theta_i)$ matice H_n. Ty odpovídají (až na podi-

vné vektory) vlastním vektorům $\mathbf{v}_i$ původního problému. Ty pak pročistí vynásobením maticí S (neboť tím opět „podivné“ vlastní vektory odpovídající jádru matice B vymizí). Díky struktuře krylovovské báze platí

$$SV_n\mathbf{y}_i = V_{n+1}\overline{H}_{n+1}\mathbf{y}_i \tag{3.4.26}$$

kde matice $\overline{H}_{n+1}$ je prvních n sloupců matice H_{n+1}. Purifikace tedy po spočtení vlastních vektorů spočte jeden krok Arnoldiho faktorizace navíc a maticí H_{n+1} pročistí vlastní vektory matice H_n. (Meerbergen – Spence, 1997; Sorensen, 1996)

Zhodnocení projekčních metod postavených na krylovovských prostorech

Pro jednoduchý problém vlastních čísel nabízí *Lanczosova* metoda rychlou a spolehlivou metodu pro řešení problému vlastních čísel, ve své symetrické formě navíc velmi dobře podloženou teoretickými základy. Tyto výtečné vlastnosti sice poněkud kazí fakt, že při použití na zobecněný problém vlastních čísel je třeba počítat rozklad matice, avšak vzhledem k tomu, že pro rychlou konvergenci je stejně vhodné provést operaci posunu a inverze, tato nevýhoda není tak velká, jak by se na první pohled mohlo zdát. Lanczosova metoda tak zůstává výtečnou volbou pro řešení jak prostého, tak zobecněného problému vlastních čísel.

3.5 Metody založené na iteraci vektoru II – minimalizační metody

Přestože se na minimalizační metody dá nahlížet jako na rozšíření mocninné metody, uvádíme je až nyní, neboť jedna z minimalizačních metod, *metoda sdružených gradientů*, má silnou vazbu na Lanczosovu metodu.

Největším problémem mocninných metod je silná závislost na zvoleném startovním vektoru: zvolíme-li počáteční odhad špatně, bude trvat velmi dlouho, než v iterovaném vektoru získá převahu složka odpovídající dominantnímu vlastnímu vektoru.

Minimalizační metody se snaží využít informaci, kterou nám výpočet přináší, a stanovovat tak „lepší odhad“ vlastního vektoru, než $A^n\mathbf{x}$ užívané mocninnou metodou. Tyto metody volí v každém kroku za odhad vlastního vektoru vektor z nějakého malého (typicky dimenze dvě až tři) podprostoru, který minimalizuje (popř. maximalizuje) danou veličinu.

Část autorů tyto metody uvádí pouze pro řešení lineárních systémů a jako metodám pro řešení problému vlastních čísel jim přikládá pouze malý význam; proto tyto metody aplikované na problém vlastních čísel nemají vlastní název a užívají stejné názvy jako korespondující metody pro řešení lineárních systémů.[12]

[12] Někteří autoři (Saad, 2003) nazývají tyto metody *projekčními metodami*, protože na prováděné minimalizace lze nahlížet jako na vhodnou projekci iterovaného vektoru. Tento termín je však zároveň (a častěji) užíván pro metody, které užívají k řešení projekce do podprostorů, proto budeme tyto metody označovat jako *minimalizační*.

3.5.1 Metoda největšího spádu

Nejjednodušší z minimalizačních metod je metoda největšího spádu, což je adaptace stejnojmenného algoritmu (užívaného v různých odvětvích matematiky) na řešení problému vlastních čísel. Tato metoda má zaručenou konvergenci pouze na symetrických pozitivně definitních maticích. Někteří autoři tuto metodu nazývají (aby se odlišila od metody největšího spádu pro řešení lineárních systémů) *metoda minimálního residua.*

Teoretický základ metody největšího spádu

Metoda největšího spádu je obecná iterativní procedura sloužící k nalezení minimalizace či maximalizace funkce $f : \mathbb{R}^n \to \mathbb{R}$. Iterace této metody je

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \alpha_i \nabla f \tag{3.5.1}$$

kde ∇f značí *gradient* funkce f – tj. směr, ve kterém se (lokálně) nejvíce mění její hodnota – a $\alpha_i \in \mathbb{R}$ minimalizuje hodnotu funkce f v prostoru[13].

$$P = \mathbf{x}^i - \beta \nabla f,\ \beta \in \mathbb{R} \tag{3.5.2}$$

Definujme funkci $f_A : \mathbb{R}^{n\times n} \to \mathbb{R}$ jako transpoziční Rayleighův koeficient:

$$f_A = \frac{\langle A\mathbf{x} | \mathbf{x} \rangle_T}{\langle \mathbf{x} | \mathbf{x} \rangle_T} \tag{3.5.3}$$

a dokážeme následující větu:

Věta 8. *Nechť matice $A^{n\times n}$ je symetrická. Pak $\forall \mathbf{x} \in \mathbb{R}^n$ platí*

$$\lambda_{min} \leq f_A(\mathbf{x}) \leq \lambda_{max} \tag{3.5.4}$$

kde λ_{min} a λ_{max} je minimální a maximální vlastní číslo matice.
Dále pak

$$\nabla f_A(\mathbf{x}) = A(\mathbf{x}) - f_A(\mathbf{x})\mathbf{x} \tag{3.5.5}$$

.

Důkaz. 1. Z linearity skalárního součinu i násobení matic plyne, že $f(\mathbf{x}) = f(c\mathbf{x})$ pro každé $\forall \mathbf{x} \in \mathbb{R}^n$ a $\forall c \in R \setminus \{0\}$, tvrzení tedy stačí dokázat pro $\mathbf{x} \in \mathbb{R}^n$ splňující $\langle \mathbf{x} | \mathbf{x} \rangle_T = 1$. Pro takovéto vektory je f_A ekvivalentní funkci

$$g_A = \langle A\mathbf{x} | \mathbf{x} \rangle_T \qquad \mathbf{x} \in \mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x} | \mathbf{x} \rangle_T = 1\} \tag{3.5.6}$$

Jelikož je funkce g spojitá a prostor $\mathcal{S}$ je kompaktní, musí na něm funkce g_A dosahovat extrémů.

Ze známé věty o Lagrangeových multiplikátorech a vázaných extrémech víme, že g může mít vázaný extrém pouze v bodě, jež splňuje pro $i \in \{1, \cdots, n\}$

$$\frac{\delta g_A}{\delta \mathbf{x}_i} - \lambda \frac{\delta(\langle \mathbf{x} | \mathbf{x} \rangle_T - 1)}{\delta \mathbf{x}_i} = 0 \qquad \text{zderivujeme} \tag{3.5.7}$$

$$A^T\mathbf{x} + A\mathbf{x} - \lambda 2\mathbf{x} = 0 \qquad \text{jelikož } A^T = A \tag{3.5.8}$$

$$A\mathbf{x} - \lambda\mathbf{x} = 0 \tag{3.5.9}$$

[13] Někteří autoři v případě metody největšího spádu pro počítání vlastních čísel uvádějí prostor $\nabla f - \beta \mathbf{x}^i$, věta 9 ukazuje, že jde (z hlediska vlastních vektorů) o stejný prostor.

Funkce g_A tedy může dosahovat extrémů pouze na vlastních vektorech. Jelikož pro každý vlastní pár $\mathbf{v}_i, \lambda_i : g(\mathbf{v}_i) = \lambda_i$, musí platit

$$\forall \mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\| = 1 : \lambda_{min} \leq g(x) \leq \lambda_{max} \tag{3.5.10}$$

.

2. Gradient

$$\nabla f_A = [\frac{\delta f(\mathbf{x})}{\delta \mathbf{x}_1}, \frac{\delta f(\mathbf{x})}{\delta \mathbf{x}_2}, \cdots, \frac{\delta f(\mathbf{x})}{\delta \mathbf{x}_n}] \tag{3.5.11}$$

Z pravidel pro derivace plyne:

$$\nabla f_A = \frac{\nabla \langle \mathbf{x} | A\mathbf{x} \rangle_T \langle \mathbf{x} | \mathbf{x} \rangle_T - \langle \mathbf{x} | A\mathbf{x} \rangle_T \nabla \langle \mathbf{x} | \mathbf{x} \rangle_T}{(\langle \mathbf{x} | \mathbf{x} \rangle_T)^2} \tag{3.5.12}$$

$$= \frac{\langle \mathbf{x} | \nabla A\mathbf{x} \rangle_T + \langle \nabla \mathbf{x} | A\mathbf{x} \rangle_T - 2 f_A(\mathbf{x}) \mathbf{x}}{\langle \mathbf{x} | \mathbf{x} \rangle_T} \tag{3.5.13}$$

$$= \frac{A\mathbf{x} + A^T\mathbf{x} - 2 f_A(\mathbf{x}) \mathbf{x}}{\langle \mathbf{x} | \mathbf{x} \rangle_T} \tag{3.5.14}$$

$$= c\, \mathbf{r}_i \tag{3.5.15}$$

$$\text{kde} \quad \mathbf{r}_i = A\mathbf{x}_i - f_A(\mathbf{x}_i)\mathbf{x}_i \quad \text{a} \quad c = \frac{2}{\langle \mathbf{x}_i | \mathbf{x}_i \rangle_T} \tag{3.5.16}$$

□

Směr gradientu $\mathbf{r}_i$ budeme dále nazývat *residuum*. Jelikož je konstanta c pro směr residua nepodstatná, budeme normovat vektor $\mathbf{x}_i$ tak, aby byla rovna jedné.

Nyní dokážeme následující jednoduchou větu, která umožňuje hledat vektor maximalizující funkci f jako vlastní vektor matice promítnuté do prostoru $\mathcal{V} = span(\mathbf{x}_i, \mathbf{r}_i)$.

Věta 9. *Nechť matice A_V je symetrická matice $A \in \mathbb{R}^{n \times n}$ kolmo promítnutá do prostoru*

$$\mathcal{V} = span(\mathbf{x}_i, \mathbf{r}_i)$$

a vektor a jemu příslušné residuum

$$\mathbf{x}_i,\ \mathbf{r}_i \in \mathbb{R}^n,\ \mathbf{x}_i \neq 0$$

Pak

$$\exists \mathbf{u} \in \mathcal{P} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}_i - \alpha \nabla f_A\} \tag{3.5.17}$$

jenž je vlastním vektorem matice A_V náležejícím k většímu z vlastních čísel matice A_V. Tento vektor maximalizuje f_A na prostoru $\mathcal{P}$ (i na prostoru $\mathcal{V}$).

Důkaz. Důkaz budeme budovat po krocích z jednotlivých tvrzení:

1. Vlastní vektor A_V nemůže mít směr $\mathbf{r}_i$.

   Měl-li by vlastní vektor matice A_V s vlastním číslem λ směr $\mathbf{r}_i$, musela by mít A_V (v normalizované bázi $\mathbf{x}_i, \mathbf{r}_i$) pravý horní prvek nulový. Protože je A symetrická, je symetrická i její kolmá projekce do prostoru s ortonormální bází. A_V by tedy musela mít v této projekci nulový i levý dolní prvek, $\mathbf{x}_i$ by byl tedy také vlastním vektorem A, a $\mathbf{r}_i$ by tedy byl nulový, což je spor.

2. Je-li

$$\overline{\mathcal{V}} = \mathcal{V} \setminus span(\mathbf{r}_i) \tag{3.5.18}$$

pak

$$\forall \mathbf{v} \in \overline{\mathcal{V}},\ \exists \mathbf{w} \in \mathcal{P} \text{ a } \beta \in \mathbb{R} \setminus \{0\},\ \text{tak že}$$
$$\mathbf{w} = \beta \mathbf{v} \ \text{ a } \ f_A(\mathbf{v}) = f_A(\mathbf{w}) \tag{3.5.19}$$

Obě části tvrzení jsou zřejmé: z předpokladu plyne, že

$$\mathbf{v} = \gamma \mathbf{x}_i + e\mathbf{r}_i \quad \text{kde} \quad \gamma \in \mathbb{R} \setminus \{0\} \ \text{ a } \ \epsilon \in \mathbb{R} \tag{3.5.20}$$

Pro nalezení vektoru $\mathbf{w}$ stačí položit $\beta = 1/\gamma$. Druhá část tvrzení je důsledek faktu, že hodnota f_A je závislá pouze na směru, nikoli velikosti vektoru, což lze snadno odvodit dosazením do rovnice 3.5.3.

Tímto a předešlým tvrzením jsme dokázali existenci vektoru $\mathbf{u}$ z předpokladu věty.

3. Jelikož A_V je symetrická, funkce f_A je dle věty 8 v prostoru $\mathcal{V}$ maximalizována vlastním vektorem $\mathbf{v}$ matice A_V. Jelikož $\mathcal{P} \subset \mathcal{V}$ a f_A, vektor $\mathbf{u}$ maximalizuje f_A na prostoru P.

□

Algoritmus a implementace

Věta 9 ukazuje jak snadno nalézt maximum funkce f na prohledávaném prostoru – jelikož záleží pouze na směru iterovaného vektoru, stačí najít vlastní vektor matice

$$A_V = \begin{pmatrix} f_A(\mathbf{x}_i) & \xi \\ 1 & f_A(\mathbf{r}_i) \end{pmatrix} \qquad \xi = \frac{\langle A\mathbf{r}_i \,|\, \mathbf{x}_i \rangle_T}{\langle \mathbf{x}_i \,|\, \mathbf{x}_i \rangle_T} \tag{3.5.21}$$

Vlastní vektory matice A_V spočteme pomocí rovnice

$$\det |A_V - \lambda I| = 0 \tag{3.5.22}$$

Díky své dimenzi bude tato rovnice kvadratická, a tedy snadno řešitelná.

Algoritmus 3.5.1: Algoritmus metody největšího spádu

1: $\mathbf{x}_0$ = počáteční odhad vlastního vektoru, $\langle \mathbf{x}_0 \,|\, \mathbf{x}_0 \rangle_T = 1$
2: **for** i = 1,2... **do**
3: $\quad \mathbf{p}_i = A\mathbf{x}_i$
4: $\quad \theta_i = \langle \mathbf{p}_i \,|\, \mathbf{x}_i \rangle_T$ ▷ $\theta_i = f_A(\mathbf{x}_i)$
5: $\quad \mathbf{r}_i = \mathbf{p}_i - \theta_i \mathbf{x}_i$
6: $\quad$ Spočti matici A_V ▷ Dle (3.5.21)
7: $\quad \mathbf{x}_{i+1}$ = největší vlastní vektor matice A_V, $\langle \mathbf{x}_{i+1} \,|\, \mathbf{x}_{i+1} \rangle_T = 1$
8: **end until** $\|\mathbf{r}_i\| > \epsilon\theta_i$ ▷ ϵ je kritérium konvergence
9: **return** $\mathbf{x}_i$

Ze věty 9 vyplývá, že p_{i+1} má směr $(\mathbf{p}_i - \alpha A\mathbf{r}_i)$. Jelikož $A\mathbf{r}_i$ počítáme při sestavování matice A_V, můžeme na řádce 3 ušetřit jedno násobení matice vektorem a počítat $\mathbf{p}_i$ iterativně ze vztahu

$$\mathbf{p}_{i+1} = \frac{\mathbf{p}_i - \alpha A\mathbf{r}_i}{\|\mathbf{p}_i + \alpha A\mathbf{r}_i\|} \tag{3.5.23}$$

Takto počítané $\mathbf{p}_i$ však bude vlivem aritmetiky s konečnou přesností zatíženo zaokrouhlovacími chybami, proto je třeba „jednou za čas“ přepočítat $\mathbf{p}_i$ z definice $\mathbf{p}_i = A\mathbf{x}_i$ znovu.

3.5.2 Metoda největšího spádu pro zobecněný problém vlastních čísel

Definujeme-li funkci $f_{A,B}$ jako zobecněný transpoziční Rayleighův koeficient,

$$f_{A,B} = \frac{\langle \mathbf{x} | A\mathbf{x} \rangle}{\langle \mathbf{x} | B\mathbf{x} \rangle} \tag{3.5.24}$$

je její gradient opět residuum a nabývá svého minima a maxima na minimálních a maximálních vlastních číslech a vlastních vektorech (vše pro zobecněný problém).

Algoritmus největšího spádu pro zobecněný problém vlastních čísel se tedy liší výpočtem residua a výpočtem vlastních čísel matice A_V: vektor $\mathbf{x}_{i+1}$ se spočte jako (dominantní) vlastní vektor zobecněného problému $A_V\mathbf{x} = \lambda B_V\mathbf{x}$.

Tyto uvedené varianty největšího spádu jsou velmi podobné metodě sdružených gradientů. Jelikož metoda sdružených gradientů dosahuje v praktickém užití rychlejší konvergence za cenu nevelkého zesložitění výpočtu, nebudeme se implementacemi těchto metod příliš zabývat, případný zájemce si snadno odvodí konkrétní implementaci z příslušné varianty metody sdružených gradientů.

Konvergence metody a předpodmíněná metoda největšího spádu

Z Kantorovičovy nerovnosti (Saad, 2003) plyne, že metoda největšího spádu má zaručenou konvergenci pro symetrické pozitivně definitní (dále *SPD*) matice. Lze ji tedy užít k vyhledání dominantního či submisivního vlastního čísla SPD matice (funkce f_A se maximalizuje či minimalizuje).

Pro konvergenci metody největšího spádu na problém vlastních čísel (v případě počítání submisivního vlastního čísla) platí následující asymptotické vztahy (Ovtchinnikov, 2002; Saad, 2003):

$$\{\|\mathbf{r}_i\|_A\} = \mathcal{O}\left(\frac{\phi_{max} - \phi_{min}}{\phi_{max} + \phi_{min}}\right) \tag{3.5.25}$$

$$\{\theta_i - \lambda\} = \mathcal{O}\left(\frac{\phi_{max} - \phi_{min}}{\phi_{max} + \phi_{min}}\right)^2 \tag{3.5.26}$$

ϕ_{min} a ϕ_{max} vyjadřují minimální pozitivní (tedy díky SPD druhé nejmenší) a maximimální vlastní číslo matice $M = (A - \lambda_{min}B)$, kde λ_{min} je nejmenší vlastní číslo matice A.

Rychlost konvergence metody lze zvýšit pomocí předpodmínění. Zjednodušeně řečeno, místo problému $A\mathbf{x} = \lambda B\mathbf{x}$ budeme řešit problém

$$KA\mathbf{x} = \lambda KB\mathbf{x} \tag{3.5.27}$$

kde K je *předpodmiňující matice*, SPD matice velikosti $n \times n$. Ze vztahu (3.5.25) vyplývá, že čím je větší rozdíl mezi maximálním a minimálním vlastním číslem v poměru k velikosti maximálního vlastního čísla, tím je konvergence metody rychlejší. Pro prostý problém vlastních čísel je tedy jako předpodmiňující matice vhodná matice K, jež odhaduje inverzi matice A (někdy nazývanou *pseudoinverze matice*)

$$\forall \mathbf{x} \in \mathbb{R}^n : a\langle \mathbf{x} | K\mathbf{x} \rangle \leq \left\langle \mathbf{x} \middle| A^{-1}\mathbf{x} \right\rangle \leq b\langle \mathbf{x} | K\mathbf{x} \rangle \tag{3.5.28}$$

$a, b \in \mathbb{R}^+$ vyjadřují „kvalitu“ předpodmínění. (Knyazev, 1991) Čím jsou blíže k jedné, tím předpodmiňující matice lépe aproximuje inverzi A a předpodmínění je kvalitnější.

Na aplikaci skalárního součinu lze také hledět jako na nahrazení skalárního součinu v definici f_A *předpodmíněným skalárním součinem* (Bai, 2000)

$$\langle \mathbf{x} | \mathbf{y} \rangle_{K^{-1}} = \left\langle K^{-1}\mathbf{x} \middle| \mathbf{y} \right\rangle \tag{3.5.29}$$

Z použití odlišného skalárního součinu plyne v podstatě jediná odlišnost: gradient předpodmíněné metody má tvar $K\mathbf{r}_i$ a tedy $\mathbf{x}_{i+1}$ bude vlastním vektorem podprostoru generovaným vektory $\mathbf{x}_i$ a $K\mathbf{r}_i$.

Bloková varianta metody největšího spádu

V blokové adaptaci této metody je výhodné pro nalezení dominantních vektorů užít Ritzovu proceduru. Tato modifikace steepest descent je pak velmi blízká metodě Ritz-accelerated subspace iteration. Je-li V_i iterovaný blok vektorů, pak bloková mocninná metoda užívá Ritzovu proceduru na vektory AV_i, zatímco steepest descent na vektory $\{V_i, AV_i\}$.

Blokovou metodu největšího spádu lze analogicky rozšířit pro řešení zobecněného problému vlastních čísel či urychlit její konvergenci užitím předpodmínění.

3.5.3 Metoda sdružených gradientů

Přirozeným zobecněním metody největšího spádu je metoda, která v i-tém kroku minimalizuje Rayleighův koeficient přes prostor

$$\mathcal{P} = span(\mathbf{x}^{i-j}, \mathbf{x}^{i-j+1}, \cdots, \mathbf{x}^i, \mathbf{r}_i) \tag{3.5.30}$$

Pokud $j = 0$, jde o metodu největšího spádu, pokud $j = i$, pak získáme metodu ekvivalentní Lanczosově metodě. Ačkoli zatím není k dispozici teoretické zdůvodnění, z numerických experimentů vyplývá, že zvolíme-li parametr j větší než jedna, konvergence se neurychlí. Zvolíme-li $j = 1$, získaná metoda se nazývá *Metoda sdružených gradientů pro problém vlastních čísel*.[14]

Jelikož se metoda sdružených gradientů pro výpočet vlastních čísel užívá v praxi výhradně pouze v blokové variantě s předpodmíněním (a její schéma je v podstatě totožné s metodou největšího spádu), uvedeme zde rovnou nejobecnější případ: předpodmíněnou blokovou metodu pro zobecněný problém vlastních čísel.

[14]Zde se nabízí paralela s metodou sdružených gradientů pro řešení lineárních systémů. V tomto případě lze díky tridiagonální struktuře Lanczosovy matice dokázat, že pro symetrickou pozitivně definitní matici je optimalizace přes prostor $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{r}_i$ ekvivalentní optimalizaci přes celý krylovovský prostor. (Saad, 1992) V případě metody sdružených gradientů pro řešení problému vlastních čísel není zatím tento fakt teoreticky podložen. (Knyazev)

Lokálně optimální bloková předpodmíněná metoda sdružených gradientů

Lokálně optimální bloková varianta metody sdružených gradientů (*Localy Optimal Block Preconditioned Conjugate Gradient* dále *LOBPCG*) je nejcitovanější bloková varianta metody sdružených gradientů. *LOBPCG* v každém kroku získává vektory X^{i+1} Rayleigh-Ritzovou procedurou aplikovanou na prostor

$$\mathcal{P} = span(X^{i-1}, X^i, KR_i) \tag{3.5.31}$$

kde K je předpodmiňující matice. (Knyazev, 2002)

Algoritmus 3.5.2: Algoritmus LOBPCG

1: $X^{(0)} \in \mathbb{C}^{n\times m}$ ▷ počáteční odhad vlastních vektorů, $\|\mathbf{x}_m\| = 1$
2: **for** $i = 0, 1 \ldots$ **do**
3: **for** $j = 1$ **to** m **do**
4: $\theta_j^{(i)} = \dfrac{\left\langle \mathbf{x}_j^{(i)} \middle| A\mathbf{x}_j^{(i)} \right\rangle_T}{\left\langle \mathbf{x}_j^{(i)} \middle| B\mathbf{x}_j^{(i)} \right\rangle_T}$
5: $\mathbf{r}_j^{(i)} = B\mathbf{x}_j^{(i)} - \theta_j^{(i)} A\mathbf{x}_j^{(i)}$
6: $S^{(i)} = KR^{(i)}$ ▷ K je předpodmiňující matice
7: $X^{(i+1)} = Rayleigh\text{-}Ritz(X^{(i-1)}, X^{(i)}, S^{(i)})(:, 1:m)$ ▷ Prvních m normalizovaných vlastních vektorů (seřazeno dle vlastních čísel)
8: **end until** konvergence

Existují i jiné možnosti jak aplikovat blokový princip na metodu sdružených gradentů, např metody *PCG* a *PCG-XR* (Tomov et al., 2005b). Ty provádějí algoritmus sdružených gradientů na každém vektoru zvlášť, pouze je vůči sobě ortogonalizují, Rayleigh-Ritzovu proceduru přes celý generovaný prostor (řádek 6 v algoritmu *LOBPCG*) pak provádějí jen jednou za k kroků. Podle numerických experimentů však nedosahují lepších výsledků než algoritmus *LOBPCG*. (Tomov et al., 2005a)

Konvergence a užití minimalizačních metod

Odhady metody sdružených gradientů jsou nyní předmětem výzkumu, je dokázána konvergence pro symetrické matice. Tato metoda konverguje nejméně tak rychle jako metoda největšího sestupu, k dispozici jsou i ostřejší odhady konvergence (Knyazev – Neymeyr, 2001; Knyazev – Neymeyr). V numerických experimentech se ukazuje, že v praxi tato metoda konverguje rychleji než metoda největšího sestupu (kvůli tomu se metoda největšího spádu v praxi neužívá).

Tato metoda není tolik citovaná jako metody krylovovských prostorů či dále zmíněná Jacobi-Davidsonova metoda. Její užití by mohlo být vhodné, pokud chceme spočíst předem známý a ne příliš velký počet vlastních čísel a máme-li možnost odhadnout inverzi matice A a tento odhad použít jako předpodmínění.

Další možné užití této metody je v závěru selfkonvergenčního cyklu, kdy opravujeme u mnoha vlastních čísel, předchozími iteracemi již velmi dobře odhadnutých, pouze malou chybu vzniklou nevelkou aktualizací potenciálů generujících

matice problémů vlastních čísel. V tomto případě může být i jinak méně spolehlivá či rychlá metoda rychlejší než projekční metody, které zpravidla budují projekční podprostor z několika málo vektorů a tak nemusí být schopny využít všech informací získaných během předchozích iterací selfkonzistentního cyklu. Projekční algoritmy také potřebují několik prvních iterací pouze k tomu, aby v projekčním prostoru zpětně oddělily odhady vlastních vektorů z předešlé iterace, poskytnuté jako lineární kombinace v startovacím bloku vektorů, zatímco projekční algoritmy pracují se všemi vstupními vektory jednotlivě.

Přestože tedy v současné době naše volba na metody z této skupiny nepadla, jedno z potenciálně nadějných použití řešičů z této skupiny se nabízí: použít v projektu řešiče dva. Jeden projekční k rychlému získání odhadu vlastních čísel a jeden iterační, sloužící k upřesnění vlastních vektorů s malou chybou.

3.6 Jacobi-Davidsonova metoda

Jacobi-Davidsonova metoda na řešení problému vlastních čísel je metodou projekční. Z určitého pohledu se dá považovat za zobecnění Arnoldiho metody (v podobném smyslu, jako je metoda největšího sestupu zobecněním mocninné metody). Zároveň lze na tuto metodu hledět jako na projekční variantu metody iterace Rayleighova koeficientu. Jacobi-Davidsonova metoda však byla odvozena z Davidsonovy metody pro výpočet vlastních čísel silně dominantní symetrické matice. (Davidson, Jan. 1975)

3.6.1 Davidsonova metoda

Definice 42. *Matice $A \in \mathbb{C}^{n\times n}$ je silně diagonálně dominantní, pokud*

$$\forall i \in \{1, 2, \cdots, n\}: \quad |a_{i,i}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}| \quad \& \quad |a_{i,i}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{j,i}|$$

Davidsonova metoda je projekční metoda, která neužívá krylovovské báze, nýbrž rozšiřuje prohledávaný prostor o *opravu* odhadu vlastního vektoru. Tuto opravu získá pomocí matice $(D_A - \theta_i I)$ (D_A je diagonála matice A), kterou aproximuje matici $(A - \theta_i I)$.

Algoritmus 3.6.1: Davidsonova metoda

1: $\mathbf{x}^{(0)} \in \mathbb{C}^n$ ▷ počáteční odhad vlastních vektorů, $\|\mathbf{x}_0\| = 1$
2: **for** $i = 0, 1 \ldots$ **do**
3: $[\mathbf{y}_i, \theta_i]$ = „nejvhodnější“ vlastní pár matice A v prostoru $span(\mathbf{x}_0, \ldots, \mathbf{x}_i)$
4: $\mathbf{u}_i = X_i \mathbf{y}_i$
5: $\mathbf{r}_i = A\mathbf{u}_i - \theta_i \mathbf{u}_i$
6: **If** $\|\mathbf{r}_i\| < \theta_i e$ **then break**
7: $\mathbf{v}_i = (D_A - \theta_i I)^{-1} \mathbf{r}_i$
8: $\mathbf{x}_{i+1} = orotonormalizuj(\mathbf{v}_i$ vůči $\{\mathbf{x}_0, \ldots, \mathbf{x}_i\})$

Na matici $M_i = (D_A - \theta_i I)$ lze hledět jako na předpodmínění. Byla-li by rovna identické matici, jednalo by se o ekvivalent Arnoldiho algoritmu. Matice M_i se zdá

být aproximací matice $(A - \lambda I)$, pokud však vezmeme přesný odhad, je zřejmé, že tato interpretace předpodmínění není správná, neboť $(A - \lambda I)^{-1}\mathbf{r}_i = \mathbf{u}_i$. Krok s takovýmto předpodmíněním by tedy nerozšířil prohledávaný podprostor.

Lze ukázat, že Davidsonova metoda má velmi blízko k staré Jacobiho iterační metodě; Davidsonova metoda je projekční verzí minimalizační metody vynalezené Jacobim, ve které se vektor $\mathbf{x}_{i+1}$ počítá pouze jako lineární kombinace vektorů $\mathbf{x}_i, \mathbf{u}_i$.(Sleijpen – der Vorst)

Sleijpen a van der Vorst ukazují (Sleijpen – van der Vorst, 1994), že pro silně diagonální matici je Davidsonova aproximace dobrou aproximací matice M_i. To vysvětluje i konvergenci této metody pro silně diagonálně dominantní matice.

3.6.2 Jacobi-Davidsonova metoda

Jacobi-Davidsonova metoda je zobecněním Davidsonovy metody pro obecné matice. Tato metoda se liší od Davidsonovy metody užitou předpodmiňující maticí. Jacobi-Davidsonova metoda užívá jako předpodmínění matici $(A - \theta_i I)$ promítnutou do prostoru kolmého k prostoru generovanému odhadem vlastního vektoru $\mathbf{u}_i$.

$$M_i = (I - \mathbf{u}_i\mathbf{u}_i^T)(A - \theta_i I)(I - \mathbf{u}_i\mathbf{u}_i^T) \tag{3.6.1}$$

Jelikož nelze jednoduše spočíst inverzi matice M_i, v každém kroku je nutno vyřešit *korekční rovnici*. (Sleijpen et al., 1999)

$$(I - \mathbf{u}_i\mathbf{u}_i^T)(A - \theta_i I)(I - \mathbf{u}_i\mathbf{u}_i^T)\mathbf{v}_i = (A - \theta_i I)\mathbf{u}_i = \mathbf{r}_i \tag{3.6.2}$$

Odvození této rovnice uvedeme u zobecněného problému. Tato rovnice se řeší iterativním řešičem, neboť z numerických experimentů vyplývá, že rovnici není třeba řešit příliš přesně (v praxi se užívá strategie řešit rovnici v i-tém kroku s přesností $2^{(-i)}$).

Matice M_i se nemusí explicitně vytvářet. Užije-li se krylovovský řešič na matici $(A - \theta_i I)$ se startovacím vektorem kolmým k $\mathbf{u}_i$ (nejlépe $\mathbf{r}_i$) a ortogonalizuje-li se krylovovská báze vůči $\mathbf{u}_i$, bude řešení splňovat dané „ortogonální“ podmínky. (Sleijpen et al., 1999)

Algoritmus 3.6.2: Jacobi-Davidsonova metoda

1: $\mathbf{v}^{(0)} \in \mathbb{C}^n$ ▷ počáteční odhad vlastních vektorů
2: **for** $i = 0, 1 \ldots$ **do**
3: **for** $j = 1$ **to** $i - 1$ **do**
4: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \langle \mathbf{v}_j \big| \mathbf{x}_i \rangle \mathbf{x}_i$ ▷ Ortogonalizace $\mathbf{x}_i$ vůči $\mathbf{x}_0 \ldots \mathbf{x}_{i-1}$
5: $\mathbf{v}_i = \mathbf{x}_i / \|\mathbf{x}_i\|$
6: $\mathbf{v}_i^A = A\mathbf{v}_i$
7: **for** $j = 1$ **to** $i - 1$ **do**
8: $H_{i,j} = \left\langle \mathbf{v}_i^A \middle| \mathbf{v}_j \right\rangle$ ▷ spočti poslední sloupec...
9: $H_{j,i} = \left\langle \mathbf{v}_j^A \middle| \mathbf{v}_i \right\rangle$ ▷ ... a poslední řádek matice $H = VAV^T$
10: $H_{i,i} = \left\langle \mathbf{v}_i^A \middle| \mathbf{v}_i \right\rangle$
11: Spočti dekompozici $HS_i = S_i\Theta$ ▷ Např. QR algoritmem
12: $[\mathbf{y}_i, \theta_i]$ = vlastní pár matice H nejbližší žádanému vlastnímu číslu
13: $\mathbf{u}_i = X_i\mathbf{y}_i$ ▷ X_i je matice $(\mathbf{x}_0|\mathbf{x}_1|\ldots|\mathbf{x}_i)$
14: $\mathbf{r}_i = A\mathbf{u}_i - \theta_i\mathbf{u}_i$
15: **If** $\|\mathbf{r}_i\| < \theta_i e$ **then break**
16: Vyřeš rovnici $(I - \mathbf{u}_i\mathbf{u}_i^T)(A - \theta_i I)(I - \mathbf{u}_i\mathbf{u}_i^T)\mathbf{x}_{i+1} = \mathbf{r}_i$

Jelikož se v praxi příliš nemění odhad θ_i, je výhodné použít pro řešení korekční rovnice předpodmínění (které se díky tomu nemusí počítat v každém kroku znovu). (Sleijpen et al., 1998, 1999) Nemáme-li k dispozici lepší předpodmínění, lze užít předpodmínění užívané v Davidsonově algoritmu $(D_A - \theta_i I)^{-1}$.

Rychlost konvergence je při přesném řešení korekční rovnice pro symetrické matice kubická, pro obecné kvadratická (Arbenz – Hochstenbach, 2004). Řešíme-li korekční rovnici iteračním řešičem s dostatečnou přesností, metoda si zachovává kvadratickou konvergenci. (Sleijpen – van der Vorst, 1994)

Metoda velmi dobře konverguje k vektorům s vlastními čísly na okraji spektra, pro vektory s vlastními čísly uvnitř spektra je lepší užít variantu počítající s harmonickými Ritzovými čísly.

3.6.3 Techniky užívané variantami Jacobi-Davidsonova algoritmu

Následující techniky jsou společné pro všechny varianty Jacobi-Davidsonova algoritmu.

Výpočet více vlastních vektorů Chceme-li spočíst více vlastních vektorů, nemusíme pro každý počítaný vlastní vektor budovat podprostor od začátku. V prostoru generovaném vektory $\mathbf{v}_i$ konvergují vlastní vektory s blízkými vlastními čísly podobnou rychlostí. Pokud tedy v tomto prostoru zkonvergoval vlastní vektor, ostatní vlastní vektory matice H budou dobrými odhady vlastních vektorů matice A. Proto je vhodné pro výpočet dalšího vlastního čísla použít jako startovací prostor vlastní vektory matice H kromě zkonvergovaného vektoru. (Sleijpen et al., 1999)

Deflace hermitovského problému Počítáme-li další vlastní vektory s vlastním číslem blízkým vlastním číslům již spočtených vektorů, v matici H se začnou objevovat složky těchto spočtených vektorů. Na jejich odstranění je vhodné užít deflaci.[15]

V případě hermitovského problému vlastních čísel jsou vlastní vektory na sebe kolmé. Budeme-li řešit korekční rovnici s podmínkou ortogonality vůči již spočteným vlastním vektorům, bude báze V_i vůči těmto vektorům ortogonální. Tato ortogonalita dostačuje, aby vektory (respektive jejich vlastní čísla) odstraněné deflací „nerušila“ spektrum matice $V_i^T A V_i$. (Sleijpen et al., 1999) Jsou-li vlastní vektory v matici G, deflatovaná korekční rovnice má tvar:

$$(I - GG^T)(I - \mathbf{u}_i\mathbf{u}_i^T)(A - \theta_i I)(I - \mathbf{u}_i\mathbf{u}_i^T)(I - GG^T)\mathbf{x}_{i+1} = \mathbf{r}_i \tag{3.6.3}$$

Označíme-li matici $\tilde{G} = [G, \mathbf{u}_i]$, pak lze operátor $(I - GG^T)(I - \mathbf{u}_i\mathbf{u}_i^T)$ napsat ve tvaru $(I - \tilde{G}\tilde{G}^T)$ a korekční rovnice bude mít tvar (Bai, 2000; Sleijpen et al., 1999)

$$(I - \tilde{G}\tilde{G}^T)(A - \theta_i I)(I - \tilde{G}\tilde{G}^T)\mathbf{x}_{i+1} = \mathbf{r}_i \tag{3.6.4}$$

Deflace obecného problému V nehermitovském problému nemusí být vlastní vektory na sebe kolmé. Chceme-li užít deflace, místo vlastních vektorů budeme počítat částečný Schurův rozklad (viz definice 35)

$$AQ_i = Q_i R_i \tag{3.6.5}$$

kde $Q_i Q_i^T = I$ a R_i je horní trojúhelníková, popř. v $\mathbb{R}$ blokově trojúhelníková. Schurovy vektory Q_i jsou navzájem ortogonální a zároveň lze z matic Q_i, R_i snadno spočíst vlastní vektory. (Fokkema et al., 1999; Sleijpen et al., 1999)

Restart Pokud vzroste dimenze prostoru, do kterého se matice A projektuje, je zapotřebí provést restart algoritmu. Jelikož není na bázi V_i žádný požadavek, není třeba žádných implicitních schémat (narozdíl od krylovovských algoritmů).

Z numerických experimentů vyplývá, že nejefektivnější je provést restart se startovacím prostorem složeným z vlastních (u hermitovského) či Schurových (u nehermitovského problému) vektorů odpovídajících žádaným vlastním číslům matice H. (Sleijpen et al., 1999)

Bloková verze I v Jacobi-Davidsonově metodě lze použít blokovou variantu algoritmu. V blokové variantě se řeší korekční rovnice pro l „nejvhodnějších“ vlastních vektorů a prohledávaný prostor se rozšiřuje najednou o l vypočtených oprav. (Booten et al., 1994)

Na rozdíl od jiných algoritmů nepřináší bloková verze zrychlení konvergence. V případě metod iterace vektoru měly blokové varianty výhodu v tom, že bylo snazší spočíst více vlastních vektorů. U projekčních algoritmů je toho však schopna i základní (nebloková) verze, a tak tato výhoda také odpadá. Bloková verze má však význam v případě implementace na výpočetních clusterech, neboť se velmi snadno paralelizuje.

[15] Deflaci ukážeme na metodě pro prostý problém vlastních čísel, na ostatní algoritmy se aplikuje analogicky.

3.6.4 Jacobi-Davidsonova metoda s harmonickými Ritzovými čísly

Má-li vektor Ritzovovo číslo uprostřed spektra matice, nemusí být dobrou aproximací vlastního vektoru. Proto je pro výpočet vlastních vektorů s vlastním číslem uprostřed spektra matice A vhodné užít principu posunu a inverze.

Definice 43. Harmonický Ritzův vektor $\mathbf{x}$ *a jemu odpovídající* harmonické Ritzovo číslo ϑ *pro invertibilní matici* $A \in \mathbb{C}^{n\times n}$ *(popř.* $\mathbb{R}^{n\times n}$*) a podprostor s bází* $V \in \mathbb{C}^{k\times n}$ *je vektor, splňující*

$$V^+A^{-1}V\mathbf{x} = \vartheta\mathbf{x}$$

Z definice harmonických Ritzových vektorů plyne, že harmonické Ritzovy vektory matice A jsou aproximací vlastních vektorů této matice. Harmonická Ritzova čísla jsou pak aproximací převrácené hodnoty vlastních čísel matice A. Aplikace harmonických Ritzových čísel je tedy modifikací metody posunu a inverze, přičemž se „invertuje“ až promítnutá matice (neboť tato inverze je řádově levnější). Proto metody užívající harmonická Ritzova čísla konvergují k submisivním vlastním vektorům.

Promítneme-li matici A^{-1} do prostoru generovaného vektory AV, pak bude mít tvar

$$(AV)^+A^{-1}AV = (AV)^+V \tag{3.6.6}$$

Definujme-li $W = AV$, pak problém vlastních čísel v prostoru AV lze vyjádřit ve tvaru

$$W^TV\mathbf{y} = \vartheta W^TW\mathbf{y} \tag{3.6.7}$$

Budeme-li budovat bázi W ortonormální, pak se problém redukuje na prostý problém vlastních čísel

$$W^TV\mathbf{y} = \vartheta\mathbf{y} \tag{3.6.8}$$

Korekční rovnice v této modifikaci zůstává stejná, neboť i v této variantě budeme hledat opravu vektoru kolmou k vypočtené aproximaci vlastního vektoru. Algoritmus a jeho podrobnější rozbor lze nalézt zde: (Sleijpen – der Vorst; Sleijpen et al., 1999). Ideu harmonických Ritzových vektorů lze implementovat i do jiných projekčních metod k řešení problému vlastních čísel (např. (Morgan – Zeng, 1998)).

3.6.5 B-ortogonální Jacobi-Davidsonova metoda pro zobecněný problém vlastních čísel

Je-li matice B symetrická pozitivně definitní, lze vybudovat bázi V_i B-ortogonální. Tím zredukujeme (v projekčním prostoru) zobecněný problém vlastních čísel na problém prostý.

Nyní si odvodíme korekční rovnici pro B-ortogonální bázi. Mějme pár $[\mathbf{u}, \theta]$, který je aproximací vlastního páru $[\mathbf{x}, \lambda]$

$$\langle (A - \theta B)\mathbf{u} \,|\, \mathbf{u}\rangle_T = 0, \quad \langle \mathbf{u} \,|\, \mathbf{u}\rangle_T = 1 \tag{3.6.9}$$

Pak lze vlastní vektor napsat jako součet aproximace a „optimální“ opravy $\mathbf{s}$

$$\mathbf{x} = (\mathbf{u} + \mathbf{s}) : \quad A\mathbf{x} = \lambda B\mathbf{x}, \quad \langle \mathbf{u} \,|\, \mathbf{s}\rangle_T = 0 \tag{3.6.10}$$

Pro tento vlastní vektor platí:

$$(A - \theta B)(\mathbf{u} + \mathbf{s}) = (\lambda - \theta)B(\mathbf{u} + \mathbf{s}) \tag{3.6.11}$$

$$(A - \theta B)\mathbf{s} = -(A - \theta B)\mathbf{u} + (\lambda - \theta)B\mathbf{u} + (\lambda - \theta)B\mathbf{s} \tag{3.6.12}$$

Přesným řešením této rovnice je vlastní vektor. Kubická, respektive kvadratická konvergence Jacobi-Davidsonovy metody vyplývá ze zanedbání třetí složky rovnice (3.6.12), neboť tato (pro symetrické matice) klesá se třetí mocninou[16]. (Arbenz – Hochstenbach, 2004).

Zanedbejme tedy poslední složku a vynásobme rovnici (3.6.12) projektorem $(I - B\mathbf{u}\mathbf{u}^T)$. Residuum

$$r = (A - \theta B)\mathbf{u} \perp^T \mathbf{u} \tag{3.6.13}$$

bude projektorem nezměněno. Naopak druhý člen rovnice (3.6.12) vymizí, neboť $B\mathbf{u}\mathbf{u}^T B\mathbf{u} = B\mathbf{u}$. Pro opravu odhadu vlastního vektoru tedy platí rovnice

$$(I - B\mathbf{u}\mathbf{u}^T)(A - \theta B)\mathbf{s} = -\mathbf{r} \tag{3.6.14}$$

Jelikož budujeme B-ortogonální bázi, chceme, aby $\mathbf{s}$ bylo B-ortogonální s $\mathbf{u}$. Proto je výsledná rovnice

$$(I - B\mathbf{u}\mathbf{u}^T)(A - \theta B)(I - \mathbf{u}\mathbf{u}^T B)\mathbf{s} = -\mathbf{r} \tag{3.6.15}$$

V B-ortogonální bázi je matice B identická, B-ortogonální Jacobi-Davidsonův algoritmus pro zobecněný problém vlastních čísel se tedy liší od prostého problému vlastních čísel pouze uvedenou korekční rovnicí, výpočtem residua a procesem ortogonalizace (báze se místo ortogonalizace B-ortogonalizuje).

3.6.6 QZ Jacobi-Davidsonova metoda pro zobecněný problém vlastních čísel

Není-li matice B symetrická pozitivně definitní, je vhodné ji z důvodu větší numerické stability řešit ve tvaru (Fokkema et al., 1999)

$$\eta A\mathbf{x} - \zeta B\mathbf{x} = 0 \tag{3.6.16}$$

I v této variantě metody se kvůli snadné deflaci a restartu místo vlastních vektorů počítají Schurovy vektory.

Definice 44. *Zobecněný (částečný) Schurův rozklad matic $A, B \in \mathbb{C}^{n\times n}$ je rozklad matic ve tvaru*

$$AQ_k = Z_k R_k^A \qquad BQ_k = Z_k R_k^B \tag{3.6.17}$$

kde matice $Q_k, Z_k \in \mathbb{C}^{n\times k}$ jsou ortonormální a $R_k^A, R_k^B \in \mathbb{C}^{k\times k}$ jsou horní trojúhelníkové. Je-li $k < n$ jde o částečný rozklad, je-li $k = n$ jde o (úplný) rozklad. Sloupce matice Q_k se nazývají zobecněné Schurovy vektory.

[16] Řešení této rovnice se zanedbaným třetím členem je v podstatě ekvivalentní kroku metody Rayleighova koeficientu. (Arbenz – Hochstenbach, 2004)

Pokud nebude hrozit záměna pojmů, budeme slovo zobecněný vynechávat.

Ekvivalentně lze definovat rozklad pro reálné symetrické matice, pro nesymetrické matice lze definic modifikovat s použitím blokově tridiagonální matice.

V této variantě Jacobi-Davidsonovy metody budeme problém vlastních čísel šikmo promítat do *prohledávaného prostoru* s bází V tak, aby residuum bylo kolmé k *testovacímu prostoru* s bází W. Promítnutý problém má tedy rovnici

$$(\eta W^+AV - \zeta W^+BV)\mathbf{s} = 0 \tag{3.6.18}$$

Definice 45. Petrovova čísla *je dvojice čísel* η, ζ*, pro které existuje vektor* $\mathbf{s}$*, splňující rovnici* (3.6.18)*. Tento vektor se nazývá* Petrovův vektor *příslušný k číslům* η, ζ*.*

Provedeme-li *zobecněný Schurův rozklad* (Moler – Stewart, 1973) promítnutého problému pomocí QZ algoritmu,[17] získáme rozklad

$$S^L(W^+AV)S^R = T^A \qquad S^L(W^+BV)S^R = T^B \tag{3.6.19}$$

První sloupec matice S^R tvoří Petrovův vektor a čísla $T^A_{1,1}, T^B_{1,1}$ jsou jemu příslušná Petrovova čísla. Bázi prohledávaného prostoru V budeme opět budovat z ortogonalizovaných odhadů vlastních vektorů. K vypočtení vlastního čísla nejbližšího hodnotě τ je optimální volba testovacího prostoru

$$W = \nu AV + \mu BV \tag{3.6.20}$$

$$\text{kde} \quad \nu = 1/\sqrt{1 + |\tau|^2} \quad \text{a} \quad \mu = -\tau\nu \tag{3.6.21}$$

neboť volba převádí zobecněný problém vlastních čísel na výpočet minimálních harmonických Ritzových čísel matice $A - \tau I$. (Fokkema et al., 1999)

$$(A - \tau B)^{-1}(\tau^+A + B)\mathbf{x} = \theta\mathbf{x} \tag{3.6.22}$$

Korekční rovnice má v i-tém kroku tvar

$$\left(I - \frac{\mathbf{p}_i\mathbf{p}_i^+}{\mathbf{p}_i^+\mathbf{p}_i}\right)(\eta_i A - \zeta_i B)(I - \mathbf{u}_i\mathbf{u}_i^T) \tag{3.6.23}$$

kde $\mathbf{p}_i = \nu A\mathbf{u}_i + \mu B\mathbf{u}_i$. Jelikož $\mathbf{u}_i = V_i\mathbf{s}_i$ a $W_i = \nu AV_i + \mu BV$, lze snáze spočíst $\mathbf{p}_i = W_i\mathbf{s}_i$.

Deflace Máme-li spočten částečný Schurův rozklad

$$AQ^{(j)} = Z^{(j)}R^{(j)} \quad \text{a} \quad BQ^{(j)} = Z^{(j)}S^{(j)} \tag{3.6.24}$$

a počítáme-li další Schurovy vektory, lze užít deflace. V tomto případě má korekční rovnice tvar

$$\left(I - \tilde{Z}_i\tilde{Z}_i^T\right)(\eta_i A - \zeta_i B)(I - \tilde{Q}_i\tilde{Q}_i^T) \tag{3.6.25}$$

$$\text{kde } \tilde{Q}_i = [Q^{(j)}, \mathbf{u}] \text{ a } \tilde{Z}_i = [Z^{(j)}, \mathbf{p}/\langle\mathbf{p}|\,\mathbf{p}\rangle_T]$$

Je-li $\mathbf{v}_m$ ortogonální k již spočteným vlastním vektorům, jeho protějšek $\mathbf{w}_m$ ortogonální být nemusí. Proto je třeba vůči deflatovaným vektorům ortogonalizovat i bázi W.

[17] Adaptace QR algoritmu pro zobecněný problém. (Moler – Stewart, 1973)

Algoritmus 3.6.3: QZ Jacobi-Davidsonova metoda s užitím deflace pro výpočet k_{max} vlastních čísel blízkých k τ

1: $\mathbf{v}_0 \in \mathbb{C}^n, \quad k = 0 \quad m = 0$ ▷ startovací vektor, počet spočtených vektorů, a dimenze V
2: $\nu = 1/\sqrt{1 + |\tau|^2}, \quad \mu = -\tau\nu$ ▷
3: $Q = [], Z = [], R^A = [], R^B = []$ ▷ Matice částečného Schurova rozkladu
4: $M^A = [], M^B = []$ ▷ Matice promítnutého problému
5: **while** $k < k_{max}$ **do**
6: **for** $i = 1$ **to** m **do** ▷ Ortogonalizace nového bázového vektoru vůči bázi
7: $\mathbf{v}_m \leftarrow \mathbf{v}_m - \langle \mathbf{v}_i | \mathbf{v}_m \rangle \mathbf{v}_i$
8: $\mathbf{v}_m = \mathbf{x}_m / \|v_m\|$ ▷ Normalizace
9: $m = m + 1$ ▷ Zvětš bázi
10: $\mathbf{v}_m^A = A\mathbf{v}_m,\ \mathbf{v}_m^B = B\mathbf{v}_m,\ \mathbf{w}_m = \nu\mathbf{v}_m^A + \mu\mathbf{v}_m^B$ ▷ Obrazy nového bázového vektoru
11: **for** $i = 1$ **to** k **do** ▷ Deflace $\mathbf{w}_m$ vůči „zamknutým“ vektorům
12: $\mathbf{w}_m = \mathbf{w}_m - \langle \mathbf{w}_m | z_i \rangle z_i$
13: **for** $i = 1$ **to** m **do** ▷ Ortogonalizace $\mathbf{w}_m$ vůči W
14: $\mathbf{w}_m = \mathbf{w}_m - \langle \mathbf{w}_m | \mathbf{w}_i \rangle \mathbf{w}_i$
15: $\mathbf{w}_m = \mathbf{w}_m / \|\mathbf{w}_m\|$ ▷ Normalizace
16: **for** $i = 1$ **to** $m - 1$ **do** ▷ Výpočet sloupce a řádky matic M^A, M^B
17: $M^A_{i,m} = \left\langle \mathbf{w}_i \middle| \mathbf{v}_m^A \right\rangle, \quad M^A_{i,m} = \left\langle \mathbf{w}_m \middle| \mathbf{v}_i^A \right\rangle$
18: $M^B_{i,m} = \left\langle \mathbf{w}_i \middle| \mathbf{v}_m^B \right\rangle, \quad M^B_{i,m} = \left\langle \mathbf{w}_m \middle| \mathbf{v}_i^B \right\rangle$
19: $M^A_{m,m} = \left\langle \mathbf{w}_m \middle| \mathbf{v}_m^A \right\rangle, M^B_{m,m} = \left\langle \mathbf{w}_m \middle| \mathbf{v}_m^B \right\rangle$
20: Spočti částečný Schurův rozklad
$$S^L(W^+AV)S^R = T^A, \qquad S^L(W^+BV)S^R = T^B$$
„seřazený“ dle vlastních čísel: tzn. aby
$$\forall i \in \{2, \ldots, m\} : |T^A_{i-1,i-1}/T^B_{i-1,i-1}| \le |T^A_{i,i}/T^B_{i+1,i+1}|$$
21: $\mathbf{u} = VS_1^R, \qquad \mathbf{p} = WS_1^L$ ▷ První sloupce matic S tvoří odhad vlastního vektoru a jeho obraz
22: OTESTUJVEKTORY() ▷ Otestuje a přijme všechny dostatečně přesné vektory
23: **if** $m > m_{max}$ **then** ▷ Báze příliš velká ⇒ Restart
24: $m = m_{min}$ ▷ Nová velikost báze
25: $\mathbf{v}_1 = \mathbf{u},\ \mathbf{w}_1 = \mathbf{p}$ ▷ První prvek báze je již spočten
26: $\mathbf{v}_1^A = V^A S^R_{1,1},\ \mathbf{v}_1^B = V^B S^R_{1,1}$
27: **for** $i = 2$ **to** m **do** ▷ Další spočteme z rozkladu
28: $\mathbf{v}_i = VS_i^R,\ \mathbf{v}_i^A = V^A S_i^R,\ \mathbf{v}_i^B = V^B S_i^R,\ \mathbf{w}_i = WS_i^L$
$M^A = T^A(1:m, 1:m),\ M^B = T^B(1:m, 1:m)$ ▷ A upravíme matice
29: $\tilde{Q} = [Q, \mathbf{u}], \tilde{Z} = [Z, \mathbf{p}]$
30: Řeš rovnici: $(I - \tilde{Z}\tilde{Z}^T)(\nu A - \mu B)(I - \tilde{Q}\tilde{Q}^T)\mathbf{v}_m = -\tilde{r}$

Implementace Jacobi-Davidsonova algoritmu se dá rozdělit na tři části: samotný Jacobi-Davidsonův algoritmus, implementace restartu, kdy se snižuje dimenze

prohledávaného prostoru (jako nový prohledávaný prostor se užijí nejvhodnější Schurovy vektory promítnutého problému) a část, kterou lze nazvat „testování a přijetí vektoru“.

V té se testuje první Schurův vektor (ten je tedy zároveň vlastním vektorem) promítnutého problému, zda je dostatečně přesnou aproximací Schurova vektoru původního problému. Je-li tomu tak, vektor se přijme mezi vypočtené (a deflatované) vektory, utvoří se nové báze V a W (a všechny další potřebné proměnné) ze zbylých Schurových vektorů matice H_i a opět se otestuje nejvhodnější Schurův vektor.

1: **procedure** OTESTUJVEKTORY

2: **while** $m > 0$ **do** ▷ Dokud jsou nějaké vektory

3: $\eta = T^A_{1,1},\ \zeta = T^B_{1,1}$ ▷ Petrovův pár

4: $\mathbf{u}^A = A\mathbf{u},\ \mathbf{u}^B = B\mathbf{u},\ \mathbf{r} = \eta\mathbf{u}_A - \zeta\mathbf{u}_B$ ▷ Residuum

5: $\tilde{a} = Z^T\mathbf{u}^A,\ \tilde{b} = Z^T\mathbf{u}^B,\ \tilde{\mathbf{r}} = \mathbf{r} - Z(\eta\tilde{\mathbf{a}} - \zeta\tilde{\mathbf{b}})$ ▷ Deflace residua

6: **If** $\|\tilde{r}\| > e$ **then return** ▷ Vektor nevyhovuje, odejdi

7: $Q = [Q, \mathbf{u}],\ Z = [Z, \mathbf{p}],\ k = k + 1$ ▷ Přijetí vektoru $\mathbf{u}$

8: $R^A = \begin{pmatrix} R^A & \tilde{\mathbf{a}} \\ 0 & \eta \end{pmatrix}, \quad R^B = \begin{pmatrix} R^B & \tilde{\mathbf{b}} \\ 0 & \zeta \end{pmatrix}$ ▷ Schurovy matice pro deflaci, jsou-li třeba

9: **If** $k = k_{max}$ **then exit()** ▷ Spočteny všechny vektory

10: **for** $i = 1$ **to** $m - 1$ **do** ▷ Vytvoření nové báze

11: $\hat{\mathbf{v}}_i = VS^R_{i+1},\ \hat{\mathbf{v}}^A_i = V^AS^R_{i+1},\ \hat{\mathbf{v}}^B_i = V^BS^R_{i+1},\ \hat{\mathbf{w}}_i = WS^L_{i+1}$

12: $V = \hat{V},\ W = \hat{W},\ S^R = S^L = I$

13: $M^A = T^A(2:m, 2:m),\ M^B = T^B(2:m, 2:m),\ m = m - 1$

14: $\mathbf{u} = \mathbf{v}_1,\ \mathbf{p} = \mathbf{w}_1,\ \mathbf{u}^A = \mathbf{v}^A_1,\ \mathbf{u}^B = \mathbf{v}^B_1,\ \eta = T^A_{1,1},\ \zeta = T^B_{1,1}$

3.6.7 Zhodnocení Jacobi-Davidsonovy metody

Jacobi-Davidsonova metoda je z uvedených metod k řešení problému vlastních čísel nejnovější. Má z nich nejsložitější iterační krok a zároveň i nejrychlejší konvergenci. Hodí se obzvlášť k řešení nesymetrického zobecněného problému vlastních čísel, neboť (na rozdíl např. od krylovovských algoritmů) tyto problémy řeší řádově stejně rychle jako symetrický prostý problém.

Tato metoda díky své rychlé kvadratické konvergenci a z ní vyplývajícího menšího počtu kroků, nebude tak paměťově náročná jako jiné metody (či se spokojí s menším počtem restartů). Nevýhodou této metody je již zmíněná velká náročnost jednoho kroku. Pokud jsme schopni sestavit dobré předpodmínění pro matice $\eta A - \zeta B$, a díky tomu řešit tuto rovnici rychle, může být tento algoritmus správnou volbou.

Rychlost konvergence Jacobi-Davidsonova algoritmu lze ovlivnit přesností řešení korekční rovnice a již zmíněným předpodmíněním užitým k řešení této rovnice.

Na počátku vývoje naší metody byla jedna z variant formulace problému, jež vedla k nesymetrickému zobecněnému problému vlastních čísel: pak by byla Jacobi-Davidsonova metoda velmi zajímavou, možná i nejlepší volbou. Při současné formulaci problému ve formě symetrického zobecněného problému s aktualizacemi řádu již výhody Jacobi-Davidsonova algoritmu tolik nevynikají, což je důvodem, proč jsme dali přednost jinému algoritmu. To však nevylučuje možnost, že pokud se podaří bližším studiem matic vznikajících v naší metodě vytvořit kvalitní předpodmínění a bude k dispozici dobrá implementace tohoto řešiče, může být zajímavé zvážit možnost otestování tohoto algoritmu a jeho porovnání s naším současným řešičem.

4. Implementace

V předchozí kapitole jsme shrnuli nejpoužívanější metody na řešení problému vlastních čísel. Jak vyplývá z analýzy konkrétního námi řešeného problému, která byla provedena v kapitole 2.1.1, nejvhodnější formou diskretizace Khon-Shamových rovnic s nelokálními pseudopotenciály je jejich formulace v separovaném tvaru – neboť tato formulace Kohn-Shamových rovnic neporušuje ani symetrii matic, ani jejich řídkost. Obě tyto vlastnosti jsou přitom podstatné pro snadné řešení problému vlastních čísel.

Důležitosti řídkosti matice pro rychlost řešení jsme se již věnovali. Důležitost symetrie lze snadno posoudit, porovnáme-li složitost symetrických a nesymetrických metod vlastních čísel popsaných v předešlé kapitole, i to, jak dobře jsou tyto metody teoreticky prozkoumané.

Další výhodou naší formulace problému je obor hodnot, neboť se nám podařilo problém formulovat v oboru reálných čísel. Principiální složitost metod pro řešení komplexního problému vlastních čísel se sice neliší od reálného problému vlastních čísel, formulace v komplexních číslech však vyžaduje dvojnásobek paměti i (přinejmenším dvojnásobek) výpočetního času. Proti komplexním číslům mluví i užší nabídka dostupných řešičů.

4.1 Výběr řešiče

Pokud jsme vybrali konkrétní formu a způsob diskretizace vedoucí k zobecněnému problému vlastních čísel s aktualizacemi řádu n, je třeba zvolit vhodnou metodu řešení tohoto problému a implementovat do programu vlastní patřičný řešič. Z metod popsaných v předešlé kapitole jsme zvolili *blokovou Lanczosovu metodu*, a to z následujících důvodů:

+ Využití symetrie matice a z toho plynoucí malé paměťové nároky.
+ Oproti minimalizačním metodám je častěji citovaná, je dobře prozkoumáno její chování i v aritmetice s konečnou přesností a jsou vyvinuty metody, jak opravit případné chyby touto aritmetikou způsobené.
+ Oproti mocninným metodám rychleji konverguje.
+ Jsou dostupné i volně použitelné řešiče.
+ Blokovost umožňuje případnou snadnější paralelizaci řešení.

Pro náš problém Lanczosova metoda pouze dvě podstatnější nevýhody:

– Nutnost provést při řešení zobecněného problému vlastních čísel transformaci na prostý problém pomocí operace posunu a inverze. Oproti Jacobi-Davidsonově metodě se však v každém kroku užívá inverze stejné matice[1],

[1] Ve skutečnosti může být nutno kvůli špatném odhadu koeficientů pro posun a inverzi spočíst tuto inverzi vícerkát, počet inverzí však nijak nezávisí na dimenzi matice a lze ho tedy brát za konstantní.

proto lze na počátku Lanczosova algoritmu provést LDL^T rozklad této matice (Duff, 2002) a tím alespoň značně urychlit řešení soustavy lineárních rovnic.

– Relativně malá velikost startovacího bloku vektorů oproti očekávanému počtu potřebných vektorů, což vede k určitým obtížím při využití informací z předešlé iterace. V našich testech se však ukázalo, že Lanczosova metoda se dobře vyrovná i se startovním vektorem setaveným z lineární kombinace odhad; vlastních vektorů. Tato oblast si ovšem zaslouží ještě bližší pozornost.

Pro řešení zobecněného problému vlastních čísel s aktualizací řádu k neexistují volně dostupné řešiče, proto jsme zvolili cestu rozšíření existujícího řešiče. Z dostupných implementací Lanczosovy metody jsme pro zvolili knihovnu Blzpack (Marques, 1997), která je díky svojí *reverse communication strategy* vhodná pro úpravu na řešení problémů vlastních čísel s aktualizací řádu k. Tato *reverse communication strategy* spočívá v navracení výpočtu zpět volající funkci vždy, když je třeba provést násobení vektoru maticí. Volající funkce provede požadovanou operaci a opětovným voláním řídící funkce Blzpacku nechá tuto knihovnu pokračovat ve výpočtu. Díky této architektuře řešiče má uživatel této knihovny volnost ve způsobu, jakým násobení matic implementuje: což umožňuje i snadnou implementaci řešiče problému vlastních čísel s aktualizací řádu k.

Náš řešič tak mohl být navrhnut a implementován jako nadstavba nad knihovnou Blzpack, aniž by bylo (až na jednu dále zmíněnou výjimku) třeba modifikovat samotný řešič. Zmíněná nadstavba se stará o patřičné algoritmy na násobení matic (včetně aktualizací), provádění LDL^T rozkladu matice sloužícího k transformaci pomocí knihovny MA57 (Duff, 2002) a aplikaci Sherman-Morrison-Woodburyho věty na řešení získané pomocí tohoto rozkladu (viz dále). Nadstavba dále zajišťuje komunikaci s okolním prostředím, potřebné konverze formátů a nabízí rozhraní pro snadné volání metod řešiče z Fortranu i Pythonu. Tato nadstavba je psána ve Fortranu 90 a částečně v Pythonu, interface je vygenerováno pomocí f2py[2].

V následujících kapitolách se budeme detailněji věnovat několika nejzajímavějším momentům řešiče. V této kapitole budeme řešený problém vlastních čísel z rovnice (2.4.15) zapisovat s použitím značení běžného v oblasti řešičů vlastních čísel, tj.

$$\left(A + UDU^+\right)\mathbf{x} = \lambda B\mathbf{x} \tag{4.1.1}$$

Násobení rank-k-update maticí Pro výpočet operace násobení matice s rank-k-update není nutné formovat hustou matici, provede-li se výpočet v následujícím pořadí:

$$(A + UDU^+)\mathbf{x} = A\mathbf{x} + UDU^+\mathbf{x} = A\mathbf{x} + U(DU^+\mathbf{x}) \tag{4.1.2}$$

Pro jedno vynásobení vektoru rank-k-update maticí je tedy třeba provést jedno násobení řídkou maticí, k skalárních součinů, k násobení vektorů skalárem a $k$součtů vektorů. Vzhledem k tomu, že dle očekávání i dle reálných výsledků je dimenze matice zhruba třicetkrát menší než počet nenulových prvků, je režie

[2] `http://cens.ioc.ee/projects/f2py2e/`

vzniklá díky rank-k-update srovnatelná s režií standardního problému i při velkém k.

Pro velký počet atomů pak režie rank-k-update problémů převáží – jelikož se updaty od jednotlivých atomů nepřekrývají (viz kapitola 2.2), lze v tomto případě implementovat „kompresi“ více updatů do jednoho vektoru. V tom případě, vzhledem k tomu, že vektorů od jednoho atomového jádra je cca do desíti a pro každý vektor se provádí tři operace, je čas pro provedení rank-k-update operace srovnatelný s časem pro vynásobení vektoru maticí.

Inverze Blzpack provádí transformaci zobecněného problému na prostý pomocí principu posunu a inverze ve tvaru

$$A' = B(A + UDU^{+} - \sigma B)^{-1}B \tag{4.1.3}$$

Tato transformace vyžaduje provedení inverze respektive LDL^T rozkladu matice $A + UDU^{+} - \sigma B$, tedy matice s aktualizacemi řádu k. Pro spočtení tohoto rozkladu je použita Sherman-Morrison-Woodburyho věta:

$$\left(A + UCV^{+}\right)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + V^{+}A^{-1}U\right)^{-1}V^{+}A^{-1} \tag{4.1.4}$$

Pomocí této věty je zmíněný rozklad převeden na součet inverze A^{-1} nahraditelný klasickým LDL^T rozkladem řídké matice A a korekci řešení získaných pomocí tohoto rozkladu. Na začátku každého iteračního cyklu Lanczosovy metody tedy provedeme rozklad matice $A - \sigma B$ pomocí knihovny MA57 (Duff, 2002) a spočteme

$$U_{A^{-}} = A^{-1}U \quad \text{a} \quad X = I + U^{+}U_{A^{-1}} \tag{4.1.5}$$

Pro výpočet řešení soustavy lineárních rovnic $(A + UU^{+} - \sigma B)\mathbf{x} = \mathbf{y}$ je pak třeba kromě vyřešení rovnice $(A - \sigma B)\mathbf{x} = \mathbf{y}'$ navíc vypočítat

$$\mathbf{y} = \mathbf{y}' - U_{A^{-}}(X(U_{A^{-}}\mathbf{y}')) \tag{4.1.6}$$

Celkem je tedy třeba pro jedno násobení inverzí matice s aktualizací vyřešit jednu soustavu rovnic a provést k skalárních součinů, k^2 násobení skalárů, k násobení vektoru skalárem a k součtů vektorů. Z dosud provedených testů zcela jasně vyplývá, že největší podíl času stráví řešič při rozkladu matice $(A - \sigma B)$ a dále pak při řešení soustavy rovnic pro získání vektoru $A^{-1}\mathbf{x}$, výpočet (4.1.5) i (4.1.6) zabírá pouze malou část doby výpočtu. Proto lze konstatovat, že takto implementace aktualizace řádu k nepřináší výrazné zpomalení řešení problému vlastních čísel.

Jelikož je prováděný rozklad nejpomalejší částí výpočtu, jako nadějná možnost se jeví implementovat do řešiče B-ortogonalitu (popř. užít jiný řešič, který již toto umí) a tím se zbavit nutnosti rozkládat matici A, popř. pro výpočet $(A-\sigma B)^{-1}\mathbf{x}$ použít nějaký iterativní řešič. Po začlenění knihovny na rozklad matic MA57 do projektu místo náhrady za starší MA47 však stoupla rychlost prováděného rozkladu několikrát a spotřebovaný čas při provádění rozkladu není tak neúměrný oproti ostatním částem řešiče. Zavedení B-ortogonálního řešiče a tím odstranění nutnosti rozkladu matice B ovšem skýtá i nevýhodu: odstraněním výpočtu rozkladu z metody odstraníme princip posunu a inverze a tím přijdeme o možnost řídit, ke které části spektra bude řešič konvergovat.

Rank-k-update a počet záporných vlastních čísel Originální Blzpack při výpočtu používá jako jedno z kritérií počet záporných vlastních čísel, získaných při rozkladu matice $(A - \sigma B)$. Pomocí této informace Blzpack volí σ pro provedení operace posunu a inverze a v každém běhu pak kontroluje, zdali „neminul“ některé vlastní číslo. Vzhledem k tomu, že se jako startovní vektory berou zkonvergované vektory z předchozí iterace, je riziko, že bude Blzpack počítat v prostoru ortogonálním k některému ze chtěných vlastních vektorů, mizivé.

Proto se dle praktických testů ukazuje, že je bezpečné Blzpacku nesdělit přesnou informaci o počtu negativních vlastních čísel. Aby nepokračoval v hledání „neexistujícího“ vlastního čísla, pak bylo třeba upravit rutinu `LZCHECK` v Blzpacku tak, aby tuto informaci nebrala v úvahu a pro ukončení daného běhu se řídila pouze celkovým počtem zkonvergovaných vlastních čísel, nikoli zvlášť počtem zkonvergovaných pozitivních a negativních vlastních čísel. Pokud by se u složitých systémů ukázala znalost této informace jako nutná, jsou dvě možnosti, jak toto řešit: buďto tuto informaci získat z LDL^T rozkladu matice, nebo, pokud by stačil poměrně přesný odhad, z předchozích iterací řešiče. Jelikož se současné řešení prokázalo jako dostatečné, nebylo v tomto směru vyvíjeno další úsilí.

4.2 Popis dalších části řešení

Schéma kompletního výpočtu naší metody pro výpočet elektronové struktury je na obrázku 4.1. Detailnější popis všech částí programu, na kterém spolupracovalo několik lidí a zahrnuje různé komponenty napsané v různých dalších projektech by byl příliš rozsáhlý a nad rámec této práce: Zmíníme se tedy pouze o nejdůležitějších použitých komponentech.

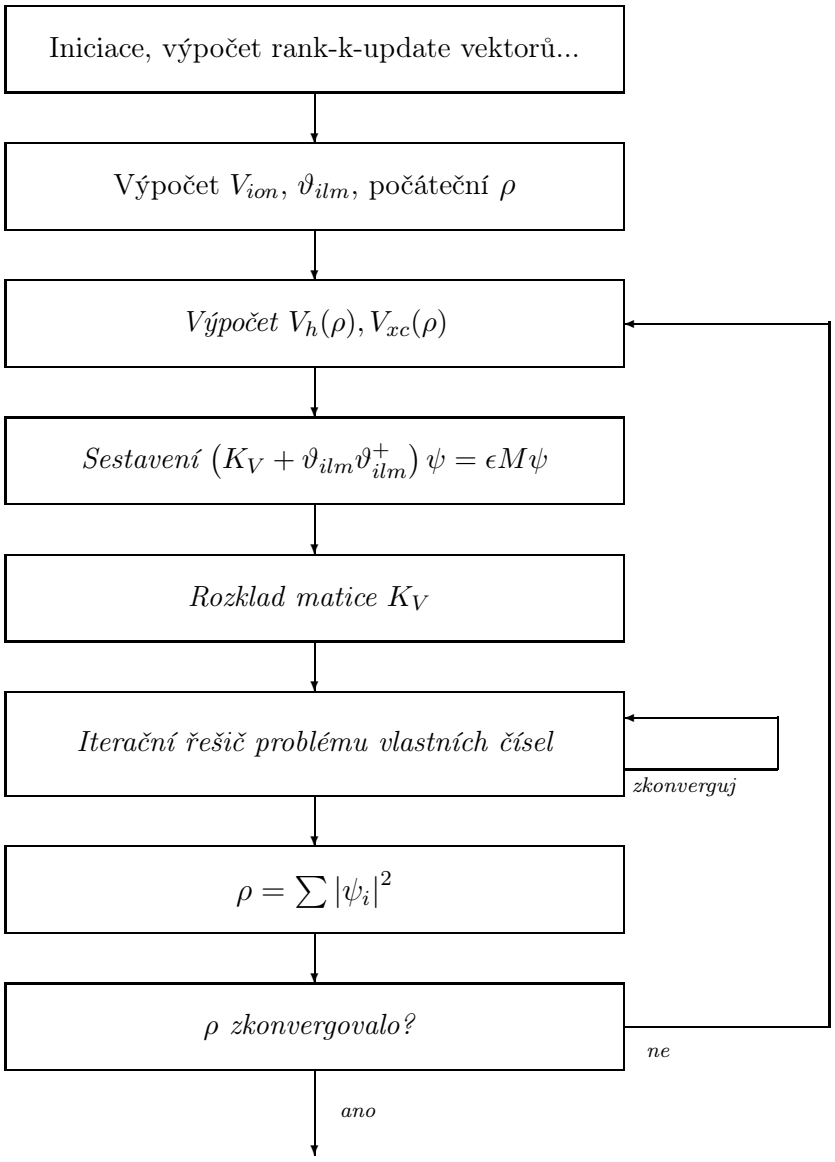
Na integrování slabé formy Schrödingerovy rovnice je použit framework Sfepy (Cimrman et al., 2011), psaný v jazyce Python. V tomtéž jazyce je pak napsána řídící smyčka algoritmu, znázorněná na obrázku 4.1. Tyto části programu jsou společným dílem s kolegy Ondřejem Čertíkem, který implementoval hlavní části iterační smyčky, a Robertem Cimrmanem, autorem Sfepy, který program odladil, vylepšil a dále pracuje na jeho vývoji.

Pro generování sítě konečných prvků používáme interní prostředky Sfepy nebo pro složitější sítě gmsh (Geuzaine – Remacle, 2003).

Pro výpočet pseudopotenciálů a vlnových funkcí použitých ke generování vektorů aktualizací matice Hamiltonova operátoru se používají programy vyvinuté Jiřím Vackářem (VACKÁŘ et al., 1998). Program generující radiální vlnové funkce napsaný ve fortranu jsem opatřil vhodným rozhraním pro volání z Pythonu a rozšířil o nadstavbu v Pythonu, umožňující generovat l-dependent pseudopotenciály a výpočet parciálních do l-podprostorů projektovaných nábojů příslušných k příslušnému atomovému centru, potřebných k modifikaci l-dependent pseudopotenciálů dle počítaného prostředí. (VACKÁŘ et al., 1996)

Poznámky k implementaci

Při běhu programu je prováděna iterační konvergenční smyčka, v níž se v každém kroku řeší problém vlastních čísel. Proto lze v druhém a dalším kroku využít jako startovní vektory výsledky z předchozích iterací. To má pozitivní vliv na



Obrázek 4.1: Schéma DFT metody s pseudopotenciály

počet iterací nutných ke konvergenci Lanczosovy metody. Vektorové operace (násobení maticí, skalární součiny vektorů etc.) využívají optimalizovaný Lapack[3] GotoBLAS2[4], který se v numerických testech na našich strojích ukázal být rychlejší než Atlas (Whaley – Dongarra, 1997). Jelikož Python kvůli své architektuře používá GIL (global interpret lock) a je pouze jednothreadový, nastavuje si afinitu na konkrétní procesorové jádro. Pro optimální využití multithreadové knihovny GotoBLAS2 bylo třeba tuto nastavenou masku pro procesorové jádro opět zrušit.

Formát matice, rank-k-update operace a jejich efektivita

Jelikož knihovna MA57 používá souřadnicový formát, pro efektivní využití paměti na počátku převedeme matice do tohoto tvaru. Algoritmy a datové struktury užívané pro výpočty s řídkými maticemi jsou dostatečně známy, proto se jim zde nebudeme blíže věnovat (podrobnosti lze nalézt např. v (Demmel, 1997)).

[3]optimalizovaná knihovna pro vektorové operace

[4]`http://www.tacc.utexas.edu/tacc-projects/gotoblas2/` [2011]

5. Dosažené výsledky

Modifikovaný řešič spolu s komunikačními rutinami pro Python, stejně jako rutiny pro generování a výpočet rank-k-update jsou zcela funkční a jsou zapojeny do celého programu řešícího Kohn-Shamovy rovnice. Nebudeme se zde věnovat samotné fyzikální přesnosti zvolené metody, neboť to je jednak mimo záběr této práce, jednak je metoda stále ve vývoji – v současné době dochází k implementaci kvadratických prvků, které dle zkušenosti z podobných metod (Pask et al., 2009) vedou k lepší konvergenci, řešení Poissonovy rovnice na jemnější síti, hledání vhodných pseudopotenciálů etc. Již současná implementace je však schopna snadno spočítat vlastní čísla a jim příslušné orbitaly atomu či molekuly dusíku, jak je vidět na obrázcích 5.1 a 5. My se však zde zaměříme pouze na analýzu samotného řešiče a na vliv jednotlivých parametrů na výkon a přesnost řešení.

5.1 Modelový problém

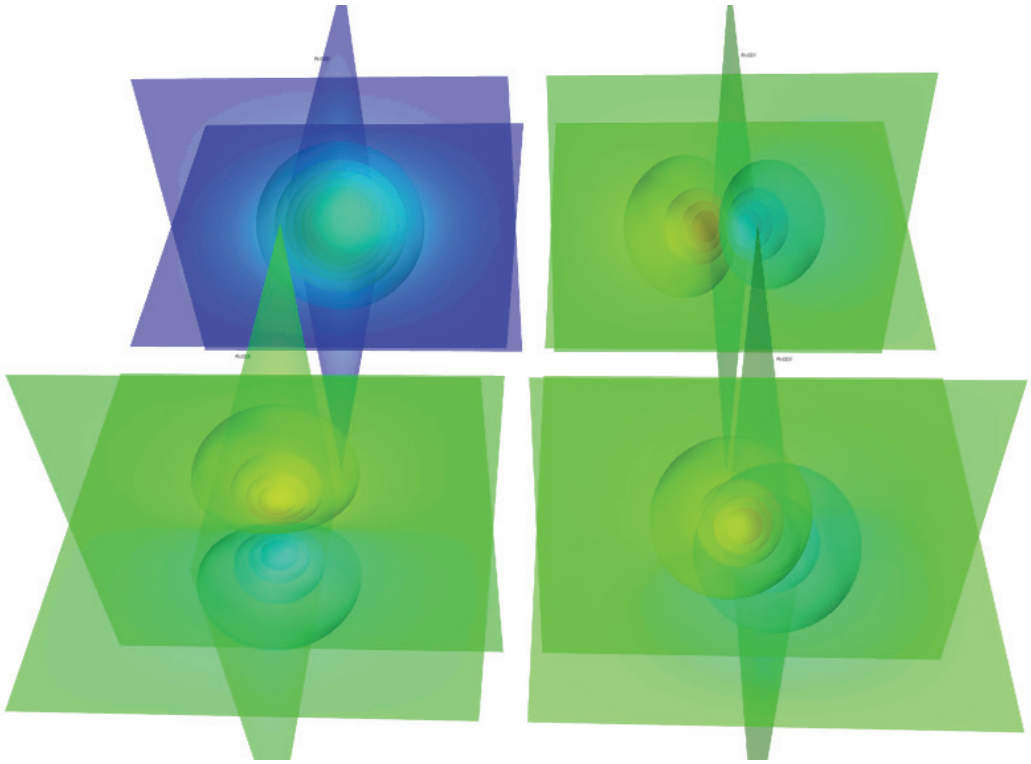
Testy rank-k-update řešiče byly prováděny na sestavě se čtyřmi procesory *Quad-Core AMD Opteron(tm) Processor 2360 SE* na frekvenci 2.5Ghz, obsazený 32GB paměti. Vzhledem k tomu, že řešič slouží k výpočtům DFT iterací a je tedy důležité vzít v úvahu i schopnost řešiče znovu použít již spočtené údaje, nebudeme měřit výkon řešiče pouze na jedné iteraci, ale na celém DFT selfkonzistentním cyklu. Aby byly zachovány stejné podmínky pro různá nastavení problému, bylo pro účely testů pevně nastaveno provádění 14 iterací selfkonzistentního cyklu. Pokud není řečeno jinak, požadovaná přesnost počítaných vlastních čísel byla v první iteraci $0,01$ a v každé další se zvětšovala násobením $0,11$ až do maximální hodnoty 1×10^{-10} a velikost bloku byla rovna čtyřem.

Síť byla vždy krychle s uniformní hustotou nodů s délkou hrany 10 atomových jednotek, počítaný atom či molekula byly umístěny v centru krychle. Elementy sítě byly taktéž krychlové, velikostí sítě budeme v dalším textu označovat počet nodů tvořících síť konečných prvků na hraně této krychle. Za poznámku stojí, že velkého zpřesnění a/či zrychlení výpočtu lze dosáhnout již jen volbou vhodné neuniformní sítě (popř. adaptivitou). Zde nám však nejde o co nejrychlejší spočtení problému, ale o prozkoumání vlastností řešiče, proto ponecháváme takto neoptimální síť, neboť ta se díky své uniformitě chová neutrálně.

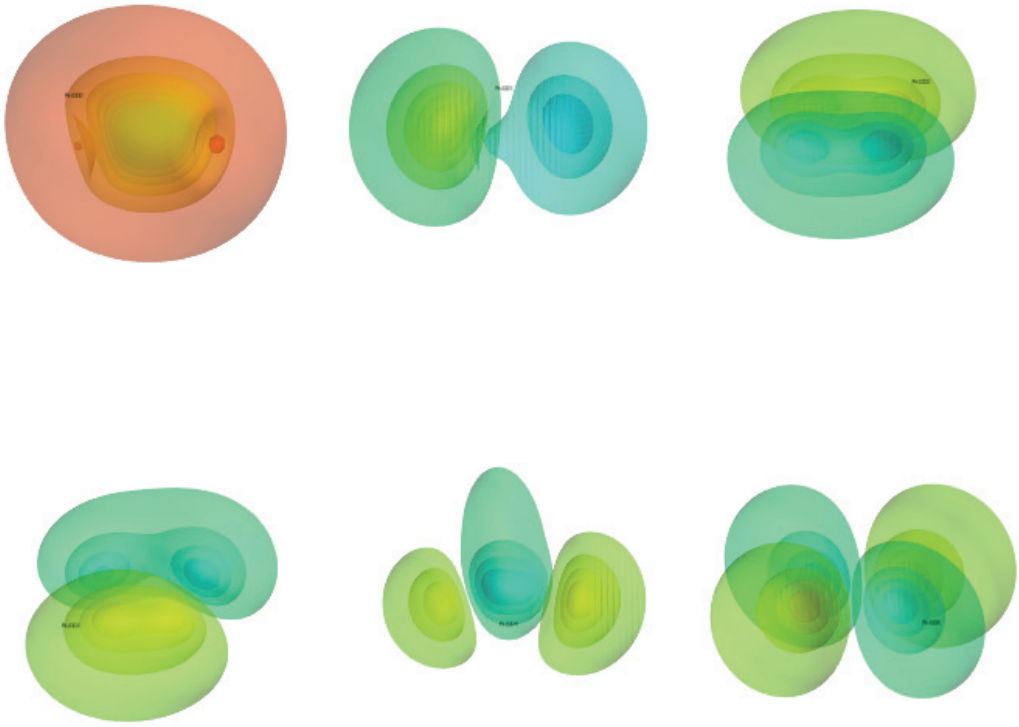
Čas budeme dále uvádět ve tvaru *minuty* : *sekundy*, N nechť značí dimenzi matice a NE počet nenulových elementů matice. Kromě času potřebného na dokončení čtrnácti iterací budeme také sledovat *přesnost konvergence*: normu rozdílu vektorů potenciálů působící na atomy. Byly počítány dva modelové problémy:

Atom dusíku Atom dusíku má pět valenčních elektronů, pro něž jsou relevantní celkem čtyři orbitaly (jeden $1s$ a tři $1p$). Pro větší přesnost a také větší vliv rank-k-update operací na výpočet budeme počítat i s $2s$ a $1d$ orbitalem, celkem tedy získáme rank-10-update. Díky dvěma možnostem orientace spinu je třeba spočíst tři vlastní čísla. Od řešiče je požadováno devět vlastních čísel.

Obrázek 5.1: Vlnové funkce valenčních elektronů atomu dusíku



Obrázek 5.2: Orbitaly valenčních elektronů molekuly dusíku N_2 (5 obsazených, jeden v neexcitovaném stavu neobsazený)



Tabulka 5.1: Reálné fill-faktory uniformních sítí

Velikost sítě	**Dimenze matice (n)**	**Nenulové položky (m)**	**Fill faktor**	**Poměr** n/m
$41 \times 41 \times 41$	59.319	1.520.875	$4.32‰$	0,039
$81 \times 81 \times 81$	493.039	12.977.875	$0.543‰$	0,0378
$100 \times 100 \times 100$	941.192	24.897.088	$0.0281‰$	0,0378

Tabulka 5.2: Závislost velikosti problému a počtu jader na času řešení (atom dusíku)

Velikost sítě	**N**	**NE**	**1 jádro**	**2 jádra**	**4 jádra**	**8 jader**
41	59 319	1 520 875	7:59	8:27	7:56	7:32
51	117 649	3 048 625	20:24	19:58	18:17	17:39
71	328 509	8 615 125	90:03	81:49	68:38	65:35
101	970 299	25 672 375	693:34	570:24	429:18	395:59

Molekula dusíku N_2 V molekule dusíku je celkem deset valenčních elektronů, obsazující celkem pět orbitalů; každý atom přispěje deseti rank-k-update vektory. Od řešiče je požadováno minimálně dvanáct vlastních čísel.

Faktor zaplnění matice Vzhledem k tomu, že veškeré operátory formující matice problému vlastních čísel se podařilo formulovat v lokálním tvaru, je počet nenulových prvků úměrný dimenzi matice. To potvrzují i výsledky vzniklé během výpočtů, zachycené v tabulce 5.1.

5.2 Analýza výkonu a přesnosti konvergence

5.2.1 Paralelizace

První věcí, které si povšimneme v tabulce 5.2, je celkově nepříliš přesvědčivé škálování. To je očekávaná vlastnost, neboť v současné době je v řešiči implementována paralelizace pouze za pomocí multithreadové knihovny GotoBLAS, sloužící k provádění optimalizovaných operací vektor–vektor, vektor–matice a matice–matice. Tyto operace je výhodné provádět paralelně, pracuje-li se s velkými bloky dat (tedy především u problémů velkých dimenzí). Proto také největší počítaný problém škáluje nejlépe.

Pro současné účely je toto řešení dostatečné a počítá s uspokojivou rychlostí – celé řešení zatím kvůli implementaci v CPythonu[1] s tzv. *global interpret lock* není určeno k nasazení na masivně paralelním stroji. Nejčastějším úkolem tohoto software bude počítat minimalizace totálních energií, kdy je třeba provést DFT selfkonzistentní cyklus v mnoha pozicích atomů: proto je daleko snažší paralelizace

[1]Standardní implementací standardu jazyka Python. `http://www.python.org/getit/`

Tabulka 5.3: Závislost požadované přesnosti a konvergence
(atom dusíku, velikost sítě = 51, blok velikosti 4)

Přesnost		**Čas**	**Chyba**	
1. iterace	**maximální**		**konvergence**	**4. vlast. čísla**
0,1	$1,00 \times 10^{-13}$	0:16:31	$2,00907 \times 10^{-06}$	2.126×10^{-08}
0,01	$1,00 \times 10^{-8}$	0:17:11	$7,58313 \times 10^{-07}$	$-1.536 \times 10^{-07*}$
0,01	$1,00 \times 10^{-10}$	0:17:22	$7,52644 \times 10^{-07}$	$4.080 \times 10^{-08*}$
0,01	$1,00 \times 10^{-12}$	0:17:37	$7,58316 \times 10^{-07}$	-1.136×10^{-08}
0,01	$1,00 \times 10^{-13}$	0:17:39	$7,58379 \times 10^{-07}$	-1.135×10^{-08}
0,001	$1,00 \times 10^{-13}$	0:19:34	$7,58435 \times 10^{-07}$	-1.136×10^{-08}

*) V poslední iteraci toto vlastní číslo nebylo spočteno, rozdíl je z předposlední iterace.

na této úrovni než na úrovni řešení vlastních čísel. Proto jsme také prozatím zvolili komponenty, které neposkytují sofistikovanější paralelizaci, oproti paralelním knihovnám však zpravidla zaručují snažší a tedy rychlejší vývoj.

Cesta k paralelizaci tím však nebyla uzavřena, neboť jsme volili knihovny a nástroje, které je možno v případě potřeby snadno paralelizovat: Blzpack poskytuje možnost paralelizace pomocí MPI. Použitý přímý řešič MA57 sice paralelizovat sám o sobě nelze, rank-k-update řešič však byl postaven modulárně, takže zahrnutí některého multithreadového řešiče (např. HSL_MA86 (Hogg – Scott, 2011)) je poměrně triviální záležitostí (viz. dokumentace k řešiči v příloze 1). Velké Sfepy pak právě prochází vývojem, přiněmž jsou jeho nejkritičtější části přepisovány do Cythonu[2], který snadnou paralelizaci umožňuje.[3]

5.2.2 Přesnost řešení

V tabulce 5.3 je zachycen vliv požadované přesnosti na dobu a přesnost výpočtu – připomeňme si, že požadovaná přesnost se v každém kroku násobí $0,11$. Ve sloupci „*chyba konvergence*“ je norma rozdílu mezi vektory potenciálu působící na elektrony ve dvou posledních iteracích a v posledním sloupci je relativní změna čtvrtého vlastního čísla, které mělo při výpočtu největší chybu. V posledním sloupci je rozdíl mezi desátým vlastním číslem v předposlední a poslední iteraci. Přesnost aritmetiky s konečnou přesností, ve které Blzpack na dané architektuře při použití typu double precision počítá, je řádově 1×10^{-16}.

Výpočet s různou přesností startovní konvergence ukazuje, že pro dosažení nejrychlejšího výpočtu je optimální počáteční přesnost $0,01$, neboť její zvýšení nepřináší výrazný zisk, pouze prodlužuje výpočet. Snížení počáteční přesnosti pak sice vede k urychlení výpočtu, ale za cenu přílišného snížení přesnosti. Vzhledem

[2]Implementace jazyka postaveného nad Pythonem, umožňujícího snadno kombinovat výhody snadného vývoje v dynamicky typovaném jazyce s rychlostí silně typovaného jazyka a snadnou interakci s funkcemi napsanými v jazyce C (a dalších kompilovaných jazycích). `http://cython.org/`

[3]Pomíjíme zde možnosti paralelismu v samotném CPythonu, např. modul multiprocessing – neboť pro účely numerických výpočtů je daleko vhodnější paralelismus na úrovni threadů, nikoli procesů.

Tabulka 5.4: Prodloužení výpočtu díky rank-k-update vektorům (molekula N_2)

Velikost sítě:	**51**		**71**	
	Čas	**Přesnost**	**Čas**	**Přesnost**
rank-0-update	16:31	$1,366346 \times 10^{-06}$	66:18	$6,233717 \times 10^{-05}$
rank-20-update	17:39	$7,583128 \times 10^{-07}$	77:09	$2,3042 \times 10^{-04}$

k tomu, že se rozdíly dosažených časů pohybují cca v rozmezí délky jedné iterace, je možné z pozorování vyvodit následující závěry:

- Nižší maximální přesnost než 1×10^{-12} ke konci konvergence je nevhodná, neboť při této hodnotě se začínají ztrácet vlastní čísla.
- Blzpack není příliš senzitivní na požadovanou přesnost, je tedy lepší volit spíše přesnost o něco vyšší.
- Ideální se zdá být začít s přesností $0,01$ a skončit na přesnosti 1×10^{-12}.
- Volba vyšší přesnosti ani tak neovlivňuje přesnost samotných „chtěných“ vlastních čísel, jako spíše poskytuje Blzpacku dostatek prostoru k nalezení těch správných. [4]

5.2.3 Vliv aktualizace řádu k na dobu výpočtu

Teoretické odhady, kterak přidání aktualizace řádu k zvýší výpočetní náročnost (jak byly provedeny v kapitole) lze snadno potvrdit testem. Pro molekulu dusíku byly spočítány dva případy: jeden, kdy každý atom generuje pět rank-k-update vektorů, druhý, kdy byly z výpočtu semilokální pseudopotenciály vyřazeny. Jak je vidět z tabulky 5.4, rozdíl mezi dobou výpočtu problému s aktualizacemi řádu 10 a bez nich je cca 6%. Dosažené časy shrnuje tabulka – je zřejmé, že rank-k-update prodlužuje dobu výpočtu sice zaznamenatelně, ale vzhledem k celkové době výpočtu nevýznamně. V nové verzi Sfepy postavené na Cythonu pak bude možno paralelizovat samotný výpočet rank-k-update vektorů, což dosažené časy ještě sblíží. V případě potřeby je možno implementovat optimalizační techniky popsané v kapitole 5.2.3.

5.2.4 Velikost bloku

V Lanczosově metodě nepřináší blokovost sama o sobě urychlení, neboť Lanczosova promítnutá matice se užitím blokovosti stane blokově tridiagonální, a tedy řešení promítnutého problému vlastních čísel potrvá delší dobu. Proto je třeba při výpočtu volit správnou velikost bloku. Blokové zpracování má tyto výhody a nevýhody:

[4] Doslova – větší požadovaná přesnost i pro další (z hlediska výpočtu defakto nezajímavá) vlastní čísla za hranicí Fermiho meze zvětšuje prohledávaný prostor a tím umožňuje Blzpacku neminout chtěná vlastní čísla.

+ umožňuje najednou pracovat s většími bloky dat a tím umožňuje efektivnější činnost paralelního BLASu,
+ při násobných (či velmi blízkých) vlastních číslech je umožňuje zachytit najednou a tím odstranit nežádoucí jev, kdy výpočet mezi nimi osciluje
+ umožňuje použít větší strartovací prostor, a tedy i větší množství informací z předchozích běhů
- výsledná matice je blokově tridiagonální a řešení projektovaného problému je náročnější
- je třeba vůči sobě ortogonalizovat více vektorů

Autor Blzpacku (Marques, 1997) doporučuje relativně malou velikost bloku – tak, aby aplikace násobení bloku vektorů netrvalo déle, než jednaapůlnásobek času potřebného pro násobení jednoho vektoru, čemuž by dle počtu jader odpovídal blok maximálně velikosti čtyři.

V tabulce 5.5 je zachycena závislost trvání výpočtu na velikosti bloku a počtu jader na výpočtu se podílejících. Podle výsledků by se zdálo, že je blokovost Lanczosovy metody spíše na škodu. V reálných problémech, kdy bude požadováno násobně více vlastních čísel, však bude hrát větší roli nutnost oddělit tato vlastní čísla do sebe, stejně jako bude potřeba uložit do startovního bloku více vektorů z předchozích iterací, čímž daleko více vynikne výhoda blokového zpracování.

Velký nárůst času u bloku velikosti osm a dvanáct zaslouží, abychom se mu podrobněji věnovali. Podíváme-li se do tabulky 5.6, vidíme, že tyto velikosti bloků zpočátku příliš nezaostávají – blok velikosti osm dokonce dosahuje často nejlepšího času, ovšem v jedné z iterací nastane skoro dvojnásobný nárůst časů: k tomu dojde v okamžiku, kdy nezkonverguje dostatečný počet vlastních čísel a Blzpack provede restart. Tento restart je ovšem poměrně drahou záležitostí, jelikož v dalším běhu se počítá málo vlastních čísel (ostatní jsou zkonvergovaná) za pomocí velkého bloku.

Současná implementace používá jako startovní vektory lineární kombinaci všech vektorů spočtených v minulé iteraci: při restartu pak má Blzpack problémy s deflací již zkonvergovaných vektorů. Při dalším vývoji řešiče tedy bude vhodné se na tuto oblast zaměřit a vyzkoušet více strategií, jak využít informace o zkonvergovaných vektorech z předešlé iterace.

Tabulka 5.5: Závislost přesnosti konvergence a času výpočtu na velikosti bloku a počtu jader na času (molekula N_2, velikost sítě 71)

Velikost	**Počet jader**				**Přesnost**
bloku	**1**	**2**	**4**	**8**	**konvergence**
1	104:18	90:06	76:57	72:51	1.7311×10^{-04}
2	106:47	96:32	84:53	80:45	1.8931×10^{-04}
4	106:14	91:46	78:32	77:09	2.3042×10^{-04}
6	101:51	91:44	77:31	76:20	7.7294×10^{-05}
8	136:17	120:23	105:04	98:23	1.9247×10^{-05}
12	158:28	138:56	121:01	117:42	9.2370×10^{-05}

Obecně se dá konstatovat, že pro malý počet požadovaných vlastních čísel menšího problému, kde jsou vlastní čísla elektronových stavů dobře oddělená, je vhodná velikost bloku jedna. Budeme-li počítat větší počet vlastních čísel, zpravidla dojde k většímu shlukování vlastních čísel bude vhodnější použít blokovou variantu. Podle provedených testů je ale zřejmé, že optimální velikost bloku se může lišit dle požadované přesnosti a kvality informací z předešlého běhu.

5.2.5 Startovací vektory

Je vcelku překvapivé, že počet vektorů z předešlé iterace nemá tak velký vliv na dobu konvergence, jak by se dalo očekávat. Je ovšem třeba podotknout, že velká část činností (LDL rozklad matice, výpočty v Pythonu) vykonávaných během iterací na tomto parametru nezávisí a že z důvodu velkého množství provedených testů byl tento proveden pouze na nepříliš velké síti, kde se rozdíly stírají. I tak je patrný rozdíl mezi případem, kdy Blzpacku nejsou dány žádné vektory, a kdy mu je dáno prvních n vektorů – nejenže výpočet trval déle, ale navíc i hůře konvergoval.

Zajímavým pozorováním je fakt, že pro dosažení nejlepšího času musíme mezi iteracemi uchovávat více vektorů, než kolik jich po řešiči požadujeme. Tato vlastnost je vcelku pochopitelná, neboť konvergenci vektorů na okraji počítaného části spektra ruší vektory ležící těsně za hranicí této části. Znalost vektorů „za hranicemi“ pomáhá k dobrému oddělení vektorů, ležících v žádané části spektra. Obecně se dá uzavřít, že dle provedeného testu není důvod neuchovávat všechny

Tabulka 5.6: Délka trvání iterací v sekundách v závislosti na velikosti bloku (molekula N_2, velikost sítě 71, 8 jader)

	Velikost bloku					
Iterace	**1**	**2**	**4**	**6**	**8**	**12**
1	241	265	254	252	243	275
2	246	261	265	253	251	262
3	265	280	287	290	272	266
4	286	302	284	283	281	293
5	286	328	300	298	285	293
6	303	312	304	316	296	312
7	300	337	317	310	312	653
8	306	351	330	309	311	643
9	311	346	322	326	324	636
10	311	355	331	345	322	633
11	311	369	344	329	662	633
12	315	357	330	331	658	637
13	314	362	340	336	661	638
14	437	481	477	463	884	748

Tabulka 5.7: Závislost času a přesnosti konvergence na počtu vstupních vektorů z předešlé iterace (molekula N_2, velikost sítě 51, velikost bloku 4)

Uchovaných vstupních vektorů	**Čas**	**Přesnost konvergence**
0	0:22:03	5.70191E-04
4	0:21:10	1.74335E-05
8	0:21:18	2.25486E-05
12	0:21:12	1.29572E-05
13	0:20:28	1.70248E-05
14	0:20:01	1.70249E-05
16	0:20:07	1.70248E-05
všechny	0:19:59	1.70248E-05

vektory spočtené řešičem, i když jsou mezi nimi vektory, o které „nemáme zájem“, neboť tyto konvergenci dominantních vlastních vektorů neruší, ale naopak podporují.

V souvislosti s předchozí kapitolou se ovšem naskýtá otázka, zda je zcela optimálně zvolena forma, jakou jsou Blzpacku tyto startovní vektory poskytovány. Navržená strategie poskytnout Blzpacku lineární kombinaci všech dostupných vstupních vektorů se ukazuje jako účinná v případě, kdy Blzpacku stačí jeden běh na spočítání všech vlastních čísel. V případě, že je však Blzpack přinucen k restartu, neprovádí deflaci natolik efektivně, aby z této kombinace dokázal odstranit již zkonvergované vlastní vektory. Proto bude při dalším rozvoji řešiče vhodné se na tuto oblast zaměřit a otestovat jiné strategie, např. manuální deflaci již spočtených vektorů, nebo nevytvářet lineární kombinaci, ale postupně (v každém běhu) poskytovat Blzpacku novou sadu vlastních vektorů.

5.2.6 Závěr z provedené analýzy

Provedené testovací výpočty s řešičem ukazují, že řešič je dostatečně stabilní a konverguje spolehlivě. Poněkud horší – ovšem očekávané – výsledky dává pouze test paralelního škálování řešiče – jak už bylo zmíněno, paralelizace není současnou prioritou a řešič nebyl od počátku komponován jako paralelní, při jeho návrhu však bylo myšleno na to, aby šla paralelizace snadno do řešiče doplnit. Vzhledem k tomu, že výpočet lze paralelizovat i jiným postupem a že prioritou bylo získání funkčního řešiče v co nejkratším čase, aby bylo možné testovat další komponenty podílející se na výpočtu elektronové struktury, nelze to považovat za nevýhodu. Další oblastí, kde lze očekávat možné suboptimální chování řešiče, je způsob využití informací (zkonvergovaných vlastních vektorů) získaných z předešlých iterací.

Výkon řešiče je dostatečný: vzhledem k neexistenci volně dostupných řešičů zobecněného problému vlastních čísel s aktualizací řádu k, nelze udělat exaktní porovnání, ale doba výpočtu problémů bez této aktualizace je porovnatelná

s dalšími volně dostupnými řešiči (např. prostřednictvím balíčku SciPy v Pythonu) a podle předběžných testů je několikanásobně rychlejší než jediný nám známý řešič, který umí řešit tento typ problému vlastních čísel. Vzhledem k tomu, že jsme ho obdrželi nedlouho před dokončením této práce, nelze jeho implementaci do Sfepy prohlásit za zcela otestovanou, a k publikaci výsledků řešiče nemáme svolení autora, nebudeme zde uvádět konkrétní výsledky testů.

Tento konkurenční řešič využívá metodu největšího sestupu: je tedy možné, že každému řešiči vyhovuje jiný typ problému: zatímco Blzpack umí velmi rychle zkonvergovat malé množství vektorů, ale je možné, že jeho schopnost využít informace o dobře zkonvergovaných vektorech z předchozích iterací je omezená; naopak iterační řešiče konvergují pravděpodobně pomaleji, ale umí lépe využít velké množství vstupní informace. Pokud by se tato teorie potvrdila, je možné že nejvhodnějším postupem bude během DFT iterace řešiče střídat: na rychlé přiblížení k hledanému řešení použít Blzpack a následně iteračním řešičem získané výsledky zpřesňovat.

6. Závěr

Cílem této práce bylo seznámit se se způsobem, jakým vzniká zobecněný problém vlastních čísel ze Schrödingerovy rovnice v rámci density functional theory (teorie funkcionálu hustoty) s použitím semilokálních pseudopotenciálů a metody konečných prvků, a seznámit se s jeho vlastnostmi, navrhnout a implementovat převedení pseudopotenciálu do pro numerické výpočty vhodného tvaru a zvolit a implementovat (s použitím již dostupných řešičů) vhodnou metodu pro řešení získaného zobecněného problému vlastních čísel.

V rámci vývoje metody bylo třeba se seznámit s poznatky z mnoha oborů a zpracovat je: ať již z oblasti kvantové fyziky a teorie funkcionálu hustoty, z teorie konečných prvků a metody pro řešení problému vlastních čísel – tedy z oblasti lineární algebry, či z teorie paralelních výpočtů a High Performance Computing z oblasti informatiky (přestože nyní metoda není cílena na paralelní výpočet, je navrhována tak, aby v případě potřeby šly její jednotlivé komponenty snadno upravit pro paralelní zpracování).

V práci jsme ukázali jsme, že převedení nelokálního pseudopotenciálu do separabilního tvaru pomocí B-ortogonální báze je vhodnou cestou k vyřešení nelokality potenciálů vzniklé zavedením pseudopotenciálů, a že výsledný problém lze dobře diskretizovat a v diskretizované podobě řešit, přičemž zavedení aktualizace řádu k neprodlužuje významně dobu výpočtu.

Dále jsme se věnovali různým metodám řešení problému vlastních čísel. Jelikož různé způsoby diskretizace mohou vést k různým formulacím problému vlastních čísel, neomezili jsme se pouze na symetrický zobecněný problém, ale věnovali jsme se i problému nesymetrickému a prostému. Z dostupných metod k řešení zobecněného problému vlastních čísel jsme pak vybrali blokovou Lanczosovu metodu, neboť je velmi vhodná pro řešení problému tohoto typu a je s ní mnoho dobrých teoretických i praktických zkušeností.

Byla nalezena její vhodná, výkonná a zároveň dobře modifikovatelná implementace – knihovna Blzpack (Marques, 1997), a ta byla rozšířena o možnosti řešení rank-k-update problémů, doplněna o metody umožňující snadné volání z Pythonu a implementována do našeho řešení na výpočet elektronové struktury. Zvolený řešič (spolu s přímým řešičem MA57 (Duff, 2002) k provádění operace posunu a inverze) splnil naše očekávání: je stabilní se spolehlivou konvergencí a jeho rychlost naprosto dostačuje pro řešení problémů.

Jak Blzpack tak i MA57 nabízí možnosti pro lepší paralelizaci, v případě potřeby je tedy možné dále posouvat hranici „spočítatelných problémů“. Další příjemnou vlastností řešiče je jeho relativní necitlivost na zvolené parametry. Díky tomu se budou moci fyzikální týmy používající náš software na výpočet elektronové struktury zcela soustředit na fyzikální aspekty počítaného problému a nebudou muset hledat vhodné parametry pro výpočet daného problému.

Zároveň jsme v této práci prozkoumali základní charakteristiky zvoleného řešiče a námi provedené modifikace a implementace do frameworku Sfepy, což nám umožnilo zvolit vhodná standardní nastavení pro výpočet a využívat vhodný počet procesorových jader pro paralelizaci. Z analýzy vyplynulo několik oblastí, na které bude vhodné se v dalším vývoji zaměřit a pokusit se výkon řešiče ještě vylepšit: jsou jimi již zmíněná paralelizace samotného řešiče i využití paraleli-

zované knihovny na LDL rozklad matice a problém optimálního využití vektorů z předešlého kroku DFT iterace.

Možnosti širšího využití námi vyvíjené metody na výpočet elektronové struktury nyní závisí na dořešení problémů v jiných oblastech, např. odladění implementace kvadratických prvků ve frameworku Sfepy, zpřesnění řešení Poissonovy rovnice pro lepší konvergenci DFT iterací a implementaci Hellman-Feynmanových sil pro možnost efektivněji provádět minimalizace totálních energií za využití minimalizačních metod využívajících derivace minimalizovaných funkcí. Vývoj metody v těchto oblastech jde rychle kupředu, a proto není v současné době další práce na samotném řešiči prioritou. Dá se konstatovat, že přestože lze výkon řešiče dále vylepšovat, implementace řešiče a jeho výkon splnily očekávání.

Literatura

Physical Review B. ISSN 1098-0121.

ARBENZ, P. et al. *Software for Numerical Linear Algebra*. ETH, Eidgenössische Technische Hochschule Zürich [Institut für Wissenschaftliches Rechnen], 2006
Zdroj: `http://e-collection.library.ethz.ch/eserv/eth:28724/eth-28724-01.pdf` [13. dubna 2012].

ARBENZ, P. – HOCHSTENBACH, M. E. A Jacobi–Davidson Method for Solving Complex Symmetric Eigenvalue Problems. *SIAM Journal on Scientific Computing*. May 2004, 25, 5, s. 1655–1673. ISSN 1064-8275. doi: 10.1137/S1064827502410992
Zdroj: `http://dx.doi.org/10.1137/S1064827502410992` [13. dubna 2012].

BAI, Z. *Templates for the Solution of Algebraic Eigenvalue Problems*. Software, Environments, Tools. Society for Industrial and Applied Mathematics, 2000. ISBN 9780898714715.

BOOTEN, J. et al. Jacobi-Davidson Methods for Generalized MHD-Eigenvalue Problems, 1996
Zdroj: `citeseer.ist.psu.edu/booten96jacobidavidson.html` [13. dubna 2012].

BOOTEN, J. G. L. et al. A Preconditioned Jacobi-Davidson Method for Solving Large Generalized Eigenvalue Problems. Technical Report NM-R9414, Universiteit Utrecht, Department of Mathematics, P. O. Box 4079, 1009 AB Amsterdam, The Netherlands, 1994
Zdroj: `citeseer.ist.psu.edu/booten94preconditioned.html` [13. dubna 2012].

BRAESS, D. *Finite Elements: Theory, Fast Solvers, and Applications in Elasticity Theory*. Cambridge University Press, 2007. ISBN 9780521705189.

CHATELIN, F. – HO, D. Arnoldi-Tschebychev Procedure for Large Scale Nonsymetric Matrices. *Mathematics Modeling and Numerical Analysis*. 1990, 24, s. 53 – 65. ISSN 0764-583X.

CIARLET, P. – LIONS, J. *Finite Element Methods*. Č. 1 v Handbook of Numerical Analysis / hlavní editor: P. G. Ciarlet. North-Holland, 1991. ISBN 9780444703651.

CIMRMAN, R. et al. SfePy Documentation, 2011
Zdroj: `http://docs.sfepy.org/doc/_downloads/sfepy_manual.pdf` [13. dubna 2012].

DANIEL, J. W. et al. Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization. *Mathematics of Computation*. 1976, 30, s. 772 – 795. ISSN 0025-5718.

DAVIDSON, E. R. The Iterative Calculation of a Few of The Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-symmetric Matrices. *Journal of Computational Physics*. Jan. 1975, vol. 17, 1, s. pp. 87–94. ISSN 0021-9991.

DEMMEL, J. W. *Applied Numerical Linear Algebra*. University of California, Berkley, California : SIAM, Philadelphia, 1997. ISBN 9780898713893.

DHATT, G. – TOUZOT, G. *The Finite Element Method Displayed*. (A Wiley-Interscience publication). Wiley, 1984. ISBN 9780471901105.

DREIZLER, R. M. – GROSS, E. K. U. *Density Functional Theory*. Springer-Verlag, 1990.

DUFF, I. S. MA57 – A New Code for the Solution of Sparse Symmetric Definite and Indefinite system, 2002.

ČERTíK, O. Calculation of Electron Structure in the Framework of DFT in Real Space. Master's thesis, Karlova Univerzita, 2008.

FATTEBERT, J.-L. A Block Rayleigh Quotient Iteration with Local Quadratic Convergence. *Electronic Transactions on Numerical Analysis*. ISSN 1097-4067.

FOKKEMA, D. R. – SLEIJPEN, G. L. G. – VORST, H. A. V. Jacobi–Davidson Style QR and QZ Algorithms for the Reduction of Matrix Pencils. *SIAM Journal on Scientific Computing*. 1999, 20, 1, s. 94–125. ISSN 1064-8275
Zdroj: `http://citeseer.ist.psu.edu/article/fokkema96jacobidavidson.html` [13. dubna 2012].

FORMÁNEK, J. *Úvod do kvantové teorie*. Academia, 2004. ISBN 9788020011763.

FREUND, R. W. – NACHTIGAL, N. M. QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems. In BEAUWENS, R. – GROEN, P. (Ed.) *Iterative Methods in Linear Algebra*. Elsevier Science Publishers, 1992. s. 151–154.

FREY, P. – GEORGE, P. *Mesh Generation: Application to Finite Elements*. Hermès Science, 2000. ISBN 9781903398005.

GEUZAINE, C. – REMACLE, J.-F. *Gmsh Reference Manual*. http://www.geuz.org/gmsh, 1.12 edition, aug 2003.

GOLUB, G. H. – YE, Q. An Inverse Free Preconditioned Krylov Subspace Method for Symmetric Generalized Eigenvalue Problems. *SIAM Journal on Scientific Computing*. January 2002, 24, s. 312–334. ISSN 1064-8275. doi: http://dx.doi.org/10.1137/S1064827500382579
Zdroj: `http://dx.doi.org/10.1137/S1064827500382579` [13. dubna 2012].

GOLUB, G. H. – YE, Q. Inexact Inverse Iterations for the Generalized Eigenvalue Problems. *Behaviour & Information Technology*. 1999, 40, s. 672–684.

HARRAR II, D. L. A Block Arnoldi Method for Large Nonsymmetric Eigenvalue Problems. Technical report, Centre for Mathematics and its Applications School of Mathematical Sciences, Australian National University, 1998.

HOGG, J. D. – SCOTT, J. A. HSL_MA86 – Sparse Symmetric Indefinite System Using OpenMP, 2011
Zdroj: `http://www.hsl.rl.ac.uk/specs/hsl_ma86.pdf` [13. dubna 2012].

HOHENBERG, P. – KOHN, W. Inhomogeneous Electron Gas. *Phys. Rev.* Nov 1964, 136, s. B864–B871. doi: 10.1103/PhysRev.136.B864
Zdroj: http://link.aps.org/doi/10.1103/PhysRev.136.B864 [13. dubna 2012].

IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. : , August 1985. Revised 1990. ISBN 1-55937-653-8.

KLEINMAN, L. – BYLANDER, D. M. Efficacious Form for Model Pseudopotentials. *Phys. Rev. Lett.* May 1982, 48, s. 1425–1428. doi: 10.1103/PhysRevLett.48.1425
Zdroj: http://link.aps.org/doi/10.1103/PhysRevLett.48.1425 [13. dubna 2012].

KNYAZEV, A. V. A Preconditioned Conjugate Gradient Method for Eigenvalue Problems and Its Implementation in a Subspace. In *International Ser. Numerical Mathematics, v. 96, Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung, Oberwolfach, 1990.*, s. 143–154, 1991.

KNYAZEV, A. V. Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method. *SIAM Journal on Scientific Computing*. 2002, 23, 2, s. 517–541. ISSN 1064-8275.

KNYAZEV, A. V. Preconditioned Eigensolvers – an Oxymoron? *Electronic Transactions on Numerical Analysis*. ISSN 1097-4067.

KNYAZEV, A. V. – NEYMEYR, K. A Geometric Theory for Preconditioned Inverse Iteration. III: A Short and Sharp Convergence Estimate for Generalized Eigenvalue Problems. Technical Report UCD-CCM-173, Center for Computational Mathematics, University of Colorado at Denver, 30 2001
Zdroj: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.297 [13. dubna 2012].

KNYAZEV, A. V. – NEYMEYR, K. Efficient Solution of Symmetric Eigenvalue Problems Using Multigrid Preconditioners in the Locally Optimal Conjugate Gradient Method. *Electronic Transactions on Numerical Analysis*. ISSN 1097-4067.

KOHN, W. – SHAM, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*. Nov 1965, 140, s. A1133–A1138. ISSN 1943-2879. doi: 10.1103/PhysRev.140.A1133
Zdroj: http://link.aps.org/doi/10.1103/PhysRev.140.A1133 [13. dubna 2012].

LANGVILLE, A. N. – MEYER, C. D. The Use of the Linear Algebra by Web Search Engines.

LEHOUCQ, R. – SCOTT, J. Implicitly Restarted Arnoldi Methods and Eigenvalues of the Discretized Navier-Stokes Equations, 1997.

LEHOUCQ, R. – SORENSEN, D. – YANG, C. Arpack Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, 1997
Zdroj: http://www.caam.rice.edu/software/ARPACK/ [13. dubna 2012].

MARQUES, O. A. BLZPACK: Description and User's Guide, 1997.

MARTIN, R. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004. ISBN 9780521782852.

MARZARI, N. Ab-initio Molecular Dynamics for Metallic Systems, 1996.

MAZZIA, F. Loss of Biorthogonality and Linear System Solvers, 1998.

MEERBERGEN, K. – SPENCE, A. Implicitly Restarted Arnoldi with Purification for the Shift-Invert Transformation. *Mathematics of Computation*. 1997, 66, 218, s. 667–689. ISSN 0025-5718. doi: http://dx.doi.org/10.1090/S0025-5718-97-00844-2.

MOLER, C. – STEWART, G. An Algorithm for Generalized Matrix Eigenvalue Problems. *SIAM Journal on Numerical Analysis*. 1973, 10, s. 241–256. ISSN 1064-8275.

MORGAN, R. – ZENG, M. Harmonic Projection Methods for Large Non-symmetric Eigenvalue Problems. *Numerical Linear Algebra with Applications*. 1998, , 5, s. 33–55. ISSN 1099-1506.

NEYMEYR, K. A Geometric Theory For Preconditioned Inverse Iteration. i: Extremal Properties of the Rayleigh quotient. *Mathematics of Computation*. 2000. ISSN 0025-5718.

NEYMEYR, K. A Geometric Theory for Preconditioned Inverse Iteration Applied to a Subspace. *Mathematics of Computation*. 2002, 71, 237, s. 197–216. ISSN 0025-5718.

OSTROWSKI, A. M. On the Convergence of the Rayleigh Quotient Iteration for the Computation of the Characteristic Roots and Vectors. III (Generalized Rayleigh Quotient and Characteristic Roots with Linear Elementary Divisors). *Archive for Rational Mechanics and Analysis*. 1959, 3, s. 325–240. ISSN 0003-9527.

OVTCHINNIKOV, E. Convergence Estimates for Preconditioned Gradient Subspace Iteration Eigensolvers. Technical report, Universiteit Utrecht, Department of Mathematics, 2002.

PARLETT, B. – SCOTT, D. The Lanczos Algorithm with Selective Orthogonalization. *Mathematics of Computation*. 1979, 33, s. 217–238. ISSN 0025-5718.

PARLETT, B. N. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ : Reprinted as Classics in Applied Mathematics 20, SIAM, Philadelphia, 1997, 1980. ISBN 9780138800475.

PASK, J. E. et al. Partition-of-Unity Finite Elements for Large, Accurate Quantum Mechanical Materials Calculations, 2009
Zdroj: `http://temple.birs.ca/~11w5121/Pask.pdf` [13. dubna 2012].

PICKETT, W. Pseudopotential Methods in Condensed Matter Applications. *Computer Physics Reports*. April 1989, 9, s. 115–197. doi: 10.1016/0167-7977(89)90002-6.

ROMMES, J. Arnoldi and Jacobi-Davidson methods for generalized eigenvalue problems $Ax = \lambda Bx$ with singular B. Technical report, Universiteit Utrecht, 2006.

SAAD, Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003. ISBN 9780898715347.

SAAD, Y. Chebyshev Acceleration Techniques for Solving Nonsymetric Eigenvalue Problems. *Mathematics of Computation*. 1984, 42, s. 567 – 588. ISSN 0025-5718.

SAAD, Y. Numerical Solution of Large Nonsymmetric Eigenvalue Problems. *Computer Physics Communications*. 1989, 53, s. 71–90. ISSN 0010-4655
Zdroj: `http://citeseer.ist.psu.edu/article/saad89numerical.html` [13. dubna 2012].

SAAD, Y. *Numerical Methods for Large Eigenvalue Problem*. New York : Hasted Press – John Wiley & Sons Inc, 1992. ISBN 9780470218207.

SADKANE, M. Block Arnoldi and Davidson Methods for Unsymmetric Large Eigenvalue Problem. Technical report, Institut National de Recherche en Informatique et en Automatique, France, 1992.

SIMON, H. *The Lanczos Algorithm for Solving Symmetric Linear Systems*. PhD thesis, 1982. AAI8300656.

SJÖSTRÖM, E. *Singular Value Computations for Toeplitz Matrices*. PhD thesis, Linkoping University, Linkoping, Sweden, 1996.

SLEIJPEN, G. – VORST, H. A Generalized Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems, 1994
Zdroj: `citeseer.ifi.unizh.ch/article/sleijpen94generalized.html` [13. dubna 2012].

SLEIJPEN, G. – VORST, H. – MEIJERINK, E. Efficient Expansion of Subspaces in the Jacobi-Davidson Method for Standard and Generalized Eigenproblems, 1998
Zdroj: `citeseer.ist.psu.edu/sleijpen98efficient.html` [13. dubna 2012].

SLEIJPEN, G. – VORST, H. – BAI, Z. Jacobi-Davidson Algorithms for Various Eigenproblems, 1999
Zdroj: `citeseer.ifi.unizh.ch/article/sleijpen99jacobidavidson.html` [13. dubna 2012].

SLEIJPEN, G. L. G. – VORST, H. A. V. A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems.

SORENSEN, D. C. Implicit Application of Polynomial Filters in a k-step Arnoldi Method. *SIAM Journal on Matrix Analysis and Applications*. 1992, 13, s. 357–385. ISSN 0895-4798.

SORENSEN, D. C. Implicitly Restarted Arnoldi/Lanczos Method for Large Scale Eigenvalue Calculation. Technical Report TR-96-40, Department of Computational and Applied Mathematics, Rice University, Huston, Texas,

1996
Zdroj: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.6613 [13. dubna 2012].

TOMOV, S. et al. Comparison of Nonlinear Conjugate-Gradient Methods for Computing the Electronic Properties of Nanostructure Architectures. In SUNDERMAN, V. S. et al. (Ed.) *the Proceedings of the 5th International Conference on Computational Science (ICCS). Springer's Lecture Notes in Computer Science 3514, Part III*, s. 317–235, Atlanta, GA, USA, May 2005a. Springer Verlag.

TOMOV, S. et al. Conjugate-gradient Eigenvalue Solvers in Computing Electronic Properties of Nanostructure Architectures. Technical Report UT-CS-05-559, The University of Tennessee, Computer Science Department, June 2005b. accepted for publication in International Journal of Computational Science and Engineering (Special Issue on Computational Methods and Techniques for Nanoscale Technology Computer Aided Design).

VACKÁŘ, J. – HYŤHA, M. – ŠIMŮNEK, A. All-Electron Pseudopotentials. *Physical Review B (Condensed Matter and Materials Physics)*. nov 1998, 581, s. 12712–12720. doi: 10.1103/PhysRevB.58.12712.

VACKÁŘ, J. – ŠIMŮNEK, A. – PODLOUCKY, R. *Ab Initio* Pseudopotentials For Interacting Atoms. *Phys. Rev. B*. Mar 1996, 53, s. 7727–7730. doi: 10.1103/PhysRevB.53.7727
Zdroj: http://link.aps.org/doi/10.1103/PhysRevB.53.7727 [13. dubna 2012].

ESHOF, J. The Convergence of Jacobi–Davidson Iterations for Hermitian Eigenproblems, 2002. ISSN 1099-1506
Zdroj: http://dx.doi.org/10.1002/nla.266 [13. dubna 2012]. http://www.math.uu.nl/publications/preprints/1165.pdf.gz.

WALKER, H. F. Implementation of the GMRES Method Using Householder Transformation. *SIAM Journal on Scientific Computing*. 1988. ISSN 1064-8275.

WHALEY, R. C. – DONGARRA, J. Automatically Tuned Linear Algebra Software. Technical Report UT-CS-97-366, University of Tennessee, December 1997
Zdroj: http://www.netlib.org/lapack/lawns/lawn131.ps+ [13. dubna 2012].

Seznam tabulek

5.1 Reálné fill-faktory uniformních sítí 80
5.2 Závislost velikosti problému a počtu jader na času řešení (atom dusíku) 80
5.3 Závislost požadované přesnosti a konvergence (atom dusíku, velikost sítě = 51, blok velikosti 4) 81
5.4 Prodloužení výpočtu díky rank-k-update vektorům (molekula N_2) 82
5.5 Závislost přesnosti konvergence a času výpočtu na velikosti bloku a počtu jader na času (molekula N_2, velikost sítě 71) 83
5.6 Délka trvání iterací v sekundách v závislosti na velikosti bloku (molekula N_2, velikost sítě 71, 8 jader) 84
5.7 Závislost času a přesnosti konvergence na počtu vstupních vektorů z předešlé iterace (molekula N_2, velikost sítě 51, velikost bloku 4) 85

Seznam algoritmů

2.1.1 DFT cyklus 15
3.3.1 Algoritmus mocninné metody 34
3.3.2 Algoritmus blokové mocninné metody 38
3.3.3 Iterace Rayleighova koeficientu pro nesymetrické matice 41
3.4.1 Arnoldiho metoda 44
3.4.2 Explicitně restartovaná Arnoldiho metoda 46
3.4.3 Implicitně restartovaná Arnoldiho metoda 48
3.4.4 Lanczosova metoda (bez restartu) 50
3.4.5 Nesymetrická Lanczosova metoda 52
3.5.1 Algoritmus metody největšího spádu 58
3.5.2 Algoritmus LOBPCG 61
3.6.1 Davidsonova metoda 62
3.6.2 Jacobi-Davidsonova metoda 63
3.6.3 QZ Jacobi-Davidsonova metoda s užitím deflace pro výpočet k_{max} vlastních čísel blízkých k τ 69

7. Příloha 1 - dokumentace k řešiči zobecněného problému s aktualizací řádu k

Tato knihovna slouží pro řešení zobecněného reálného problému reálných vlastních čísel v „double precision“ s aktualizací řádu k. Poskytuje rozhraní ve Fortranu 90 a integrační plugin v Pyhtonu do Sfepy snadno upravitelný pro jiné účely. Samotný řešič je součástí elektronické verze práce.

7.1 Požadavky

Tento řešič vyžaduje:

- překladač pro Fortran 90
- knihovnu LAPACK (doporučujeme optimalizovanou, např. MKL, GotoBLAS2 nebo ATLAS)
- Python knihovna numpy včetně nástroje f2py (není nutné pro volání z Fortranu či C)
- řešič MA57 či jeho ekvivalent: řešič je svými volbami připraven na knihovnu ma57, k dispozici je ale i wrapper pro MA47 a lze snadno napsat i komunikační rozhraní pro jinou knihovnu

7.2 Licence

Tento řešič je poskytován pro akademické účely zdarma, pro tyto účely ho taktéž lze volně šířit. Je postaven na řešiči Blzpack autora Osni Marquese, jenž podléhá následující BSD licenci:

```
BLZPACK Copyright (c) 2005, The Regents of the University of California,
through Lawrence Berkeley National Laboratory (subject to receipt of any
required approvals from the U.S. Dept. of Energy).  All rights reserved.

Redistribution and use in source and binary forms, with or without modi-
fication, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

(3) Neither the name of the University of California, Lawrence Berkeley
National Laboratory, U.S. Dept. of Energy nor the names of its
```

7.3 Instalace

- rozbalte archiv
  `tar -xzf rank-n-update.tgz`
- jděte do rozbaleného adresáře
  `cd rank-n-update`
- upravte Makefile – zkontrolujte, že je zde uveden správný kompilátor, f2py a LAPACK, a popř. pokud nepoužíváte MA57, tak knihovny vašeho řešiče: tom případě musíte vytvořit patřičné rozhraní pro komunikaci s programem (viz dále).
  `vi etc/MACROS.release`
- používáte-li MA57
  `mv ma57/Makefile ma57/Makefile.bak`
  `cp ...cesta k ma57.../* ma57/Makefile`
  `mv ma57/Makefile.bak ma57/Makefile`
  v opačném případě nainstalujte ekvivalent tohoto řešiče a patřičně upravte Makefile.release.
- přeložte
  `make`

7.4 Dokumentace exportovaných funkcí ve fortranu

Většina veřejných funkcí programu se řídí nastaveními, danými ve dvou polích: PARAMETERSD[3] a PARAMETERSI[25]. Tyto pole mají stejný význam, jako integer a real storage Blzpacku, s výjimkou následujících indexů u PARAMETERSI

1 počet kroků iterative refinement v ma57 (popř. parametr předaný jinému řešiči). Nedoporučeno používat, příliš zpomaluje.

4 úroveň ladících informací pro ma57, viz manuál ma57

14 pole ROWS/COLS jsou indexovány od tohoto čísla (tzn. 0 při volání z pythonu, 1 z Fortranu)

15 nulové, pokud matice obsahují obě poloviny, jinak nenulové

17 pokud je větší než nula, uloží vstup, tento je možno znovu řešit pomocí utility resolve, překládané spolu s řešičem. Pokud je větší než jedna, nepokračuje v řešení.

18 po násobení maticí za pomocí řešiče a spočtení Sherman-Morissonovy věty provede kontrolu chyby zpětným přenásobením a varuje, pokud je příliš velká

19 Počet kroků iterative refinement pro rank-k-update vektory. Doporučeno.

20 Pokud není nula, už byl inicializován solver. Pokud je tři, solver má rozloženou matici B.

21 Pouze v rutině načítání: je-li nula, je problém 32bit, jinak 64bit

22 Pouze v rutině načítání: je-li nula, je problém CSR, jinak je v RCI formátu

Poskytované funkce mají stejné parametry:

INTEGER N dimenze matic

INTEGER NE počet nenulových elementů

DOUBLE PRECISION A[NE] Matice A

DOUBLE PRECISION B[NE] Matice B

INTEGER IRN[N+1/NE] dle formátu matice index do indexu sloupců nebo řádkový index matice, u 64 bitového problému musí být 64bitové.

INTEGER JCN[NE] sloupcový index matice, u 64 bitového problému musí být 64bitové

INTEGER UNE problém je rank-UNE-update

DOUBLE PRECISION UPDATE[N,UNE] rank-UNE-update

DOUBLE PRECISION UCOEFS[UNE] diagonála matice D z rank-UNE-update VDV^T

NEIGS dimenze pro ukládání výsledků

DOUBLE PRECISION A[N, NEIGS] INOUT vlastní vektory (spočtené či pro deflaci, viz manuál k Blzpacku)

DOUBLE PRECISION A[2, NEIGS] INOUT vlastní čísla a jejich residua

FILENAME CHARACTER[] soubor pro uložení problému, program vytvoří navíc soubor FILENAME + '.bin'.

OUTPUT CHARACTER[] soubor pro uložení výsledků

RU_EIGENPROBLEM_CSR(N, NE, A, B, IRN, JCN, UNE, UPDATE, UCOEFS, NEIGS, PARAMETERSI, PARAMETERSD, EIGVEC, EIGEN, FILENAME) Vyřeší (popř. uloží k dalšímu řešení, viz PARAMETERSI[17]) problém ve tvaru CSR.

RU_EIGENPROBLEM_RCI(N, NE, A, B, IRN, JCN, UNE, UPDATE, UCOEFS, NEIGS, PARAMETERSI, PARAMETERSD, EIGVEC, EIGEN, FILENAME) Vyřeší (popř. uloží k dalšímu řešení, viz PARAMETERSI[17]) problém ve tvaru RCI.

STORE_PROBLEM(N, NE, A, B, IRN, JCN, UNE, UPDATE, UCOEFS, NEIGS, PARAMETERSI, PARAMETERSD, EIGVEC, EIGEN, FILENAME) Uloží k dalšímu řešení problému.

STORE_PROBLEM_32(N, NE, A, B, IRN, JCN, UNE, UPDATE, UCOEFS, NEIGS, PARAMETERSI, PARAMETERSD, EIGVEC, EIGEN, FILENAME) Uloží k dalšímu řešení 32 bitový problém.

SOLVE_STORED_PROBLEM(FILENAME, OUTPUT) Přečte definici problému, vyřeší a uloží výsledek.

READ_RESULT(OUTPUT, N, NEIGS,PARAMETERSI, PARAMETERSD, EIGEN, EIGVEC) Přečte předem uložený výsledek pomocí solve stored problému.

7.5 Dokumentace třídy v Pythonu

V Pythonu byly v rámci řešiče implementovány třídy `RefineEigenvalueSolver` zastřešující řešiče, které umí s přibývajícími iteracemi zvětšovat přesnost řešení. Tato třída je potomkem standardní třídy ze Sfepy `EigenvalueSolver`. Tato třída a její potomci mají čtyři konfigurační volby

precision_start Počáteční požadovaná přesnost řešení, standardně $0,01$

precision_wait Počet iterací, po které se přesnost nebude měnit, standardně 0

precision_factor Poté se v každé iteraci požadovaná přesnost vynásobí tímto číslem, standardně $0,11$

precision_max Avšak nikdy nepřesáhne toto číslo, standardně 1×10^{-13}

Přesnost lze obnovit na původní hodnoty voláním metody `restart_precision`.

Potomkem této třídy je BlzpackSolver, který volá výše uvedené fortranovské rutiny. Volá se jako standardní řešiče ve Sfepy s možností uvedení argumentů `rank_update` a `rank_update_coefs` pro rank-k-updaty a nabízí následující možnosti konfigurace Nabízí následující konfiguraci (vypnuto znamená nula, volba se zapne nenulovou integer hodnotou)

eig_estimation počáteční odhad nejmenšího vlastního čísla, standardně 0

eig_estimation_mixing jako další odhad nejmenšího vlastního čísla se vezme $nov \times eig\_estimation\_mixing + star(1 - eig\_estimation\_mixing)$, standardně $0,99$

rank_update

rank_update_coefs rank-k-update lze zadat i skrze konfiguraci

max_restarts pokud výpočet skončí s chybou, pokusí se řešič ještě o nový pokus či pokusy s pozměněnými parametry, standardně 2

block_size velikost bloku blokového Lanczosova algoritmu, standardně 4

max_steps maximální počet kroků algoritmu před restartem, standardní 0 ponechá rozhodnutí na Blzpacku

additional_store_ratio tímto číslem se vynásobí počet chtěných vektorů a získá se maximální počet vlastních vektorů, které lze od Blzpacku získat, standardně $1,2$

spectrum_slicing viz nápověda k Blzpacku, standardně vypnuto

purification viz nápověda k Blzpacku, standardně vypnuto

debug

debug_debug_factorize zapne vypisování ladících informací pro Blzpack respektive řešič pro shift and invert operace, standardně vypnuto

eigen_obsfucator příliš přesná vlastní hodnota může vést k numerickým problémům, proto se před předáním Bzpacku posune vynásobením touto hodnotou, standardně $1,3$

check_matrix_multiply je-li zapnuto, kontroluje přesnost inverze a Sherman-Morisnovy formule. Standardně vypnuto

refine_solver čím větší integer, tím přesněji a pomaleji je při iteracích řešiče řešen problém $A\mathbf{y} = \mathbf{x}$, standardně vypnuto

refine_solver_update čím větší integer, tím přesněji a pomaleji je počítán na počátku výpočtu výraz $A^{-1}U$, kde U je rank-k-update, standardně 10

store_problem je-li zapnuto, neřeší problém přímo, ale uloží ho a řešení počítá v samostatném procesu (brání se tak např. memory leakům apod.) a zároveň umožňuje zopakovat a ladit výpočet daného problému, standardně vypnuto

problem_file názvy souborů, kam se ukládají definice problémů a jejich výsledky, je-li předchozí volba zapnuta. Otazník v názvu souboru zamění za pořadové číslo řešeného problému, standardně `problem_file` a `result_file`

result_file viz *problem_file*

solve_mode 0 řeší problém ve tvaru $B^{-1}A$, 1 a 2 odpovídá zobecněnému a „buckling“ problému vlastních čísel popsaných v manuálu Blzpacku. Numericky stabilní se ukazuje pouze defaultní hodnota 1

7.6 Implementace vlastního řešiče

Pro nahrazení knihovny MA57 implementací vlastního řešiče soustavy lineárních rovnic je třeba vytvořit jednoduchý wrapper. Tento wrapper nazvěte EQS_nazev.f90, uložte ho do adresáře rank-k-update a v Makefile.release nastavte toto

```
SOLVER=nazev
SOLVER_DEP=
SOLVER_LIB=vaše knihovna
SOLVERLIB_DEP=
SOLVERLIB_LIB=
```

Wrapper je fortranovský modul, který musí mít následující funkce. Jejich význam je vysvětlen komentářem, zároveň je rovnou doplněna doporučená společná část (pro různé řešiče) implementace těchto funkcí.

```
MODULE eqs_solver

  !store for matrix
  INTEGER*8 N,NE
  INTEGER*8, POINTER :: IRN(:), JCN(:)
  DOUBLE PRECISION, POINTER :: MATRIX(:)

  !parametr for solving (if "non-ex" functions is called)
  INTEGER*4  REFINE

!Initialize solver
SUBROUTINE EQS_PREPARE(NN, NNE, IRNN, JCNN, PARAMETERSI )
      INTEGER*4 NN, NNE             !dimension, !number of nonzeros
      INTEGER*8, TARGET :: IRNN(:),JCNN(:)  !matrix coordinates
      INTEGER*4 PARAMETERSI(:)      !parameters, can be ignored or
                          ! used for passing some parameters to solver

      !remember useful information
      N=NN
      NE=NNE
      IRN=>IRNN
      JCN=>JCNN
      REFINE=PARAMETERSI(1)

      ! ###PREPARE####
END SUBROUTINE

! Prepare for solving with given matrix, e.g. factorize it.
FUNCTION EQS_FACTORIZE(MATRIXX) RESULT(NNEIGS)
       DOUBLE PRECISION, TARGET :: MATRIXX(:) !matrix datas
       INTEGER*4 NNEIGS           !return number of negative eigenvalues

       !remember useful information
       MATRIX => MATRIXX

       ! ###FACTORIZE####
END FUNCTION
```

```
!Solve block of vectors: A V = U
SUBROUTINE EQS_SOLVE_EX(NVOPU, U, V, REFIN)
    INTEGER*4 I, NVOPU               !block size
    DOUBLE PRECISION V(:,:), U(:,:)  !result, right side
    INTEGER*4 REFIN     !precision of solving, the higher number (from zero),
                        the greater precision is requested. can be ignored

    ! ###SOLVE####
END SUBROUTINE

!Solve block of vectors inplace A U = V, then set V=U
SUBROUTINE EQS_SOLVE_INPLACE_EX(NVOPU, V, REF)
    INTEGER*4 I, NVOPU            !block size
    DOUBLE PRECISION V(:,:)       !in - right side, out - result
    INTEGER*4 REFIN               !precision of solving, can be ignored

    DOUBLE PRECISION TMP(N)

    ! ###SOLVE####
END SUBROUTINE

!Clean all stuff, allocated by previous functions
SUBROUTINE EQS_FREE()
    ! ###CLEAN####
END SUBROUTINE

!shortcuts, can be let as is
SUBROUTINE EQS_SOLVE(NVOPU, U, V)
    INTEGER*4 I, NVOPU
    DOUBLE PRECISION V(:,:), U(:,:)
    CALL EQS_SOLVE_EX(NVOPU, U, V, REFINE)
END SUBROUTINE

SUBROUTINE EQS_SOLVE_INPLACE(NVOPU, V)
    INTEGER*4 I, NVOPU
    DOUBLE PRECISION V(:,:)
    DOUBLE PRECISION TMP(N)
    CALL EQS_SOLVE_INPLACE_EX(NVOPU, V, REFINE)
END SUBROUTINE
```