Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Praca dyplomowa magisterska

na kierunku Informatyka Stosowana

w specjalności Inżynieria Danych

Szybkie metody znajdowania pól własnych osiowosymetrycznych rezonatorów mikrofalowych

inż. Patryk Guba

numer albumu 307347

promotor

dr inż. Bartosz Chaber

WARSZAWA 2024

Szybkie metody znajdowania pól własnych osiowosymetrycznych rezonatorów mikrofalowych

Streszczenie

Praca dotyczy szybkich metod znajdowania pól własnych za pomocą metod numerycznych opracowanych w języku Julia. Jest to zagadnienie wykorzystywane w modelowaniu wnęk rezonansowych wykorzystywanych w klistronach. W pierwszym rozdziale omówiono fizyczne i matematyczne podstawy poruszanego problemu oraz związek z uogólnionym problemem własnym.

W części praktycznej przeanalizowano kilka metod numerycznych dedykowanych rozwiązywaniu uogólnionego problemu własnego. Opisano domyślną funkcję rozwiązywania uogólnionego problemu własnego w bibliotece LinearAlgebra.jl, metody Arnoldiego w bibliotece Arpack.jl, metodę Kryłowa-Schura implementowaną w ArnoldiMethod.jl oraz podejście metody QZ Jacobi-Davidsona. Przedstawiono algorytmy i właściwości każdej z tych metod, zwracając uwagę na ich wydajność i zastosowanie w praktyce.

Kolejnym etapem pracy była ocena opracowanego oprogramowania. Wyniki uzyskane za pomocą implementacji w języku Julia porównano z wynikami komercyjnych rozwiązań, takich jak COMSOL Multiphysics czy Matlab. Przeprowadzone testy metod rozwiązywania problemów własnych zaimplementowanych w języku Julia wykazały lepsze rezultaty niż dla oprogramowania komercyjnego, oferując porównywalne wyniki przy krótszym czasie pracy.

Słowa kluczowe: wnęki rezonansowe, uogólniony problem własny, metoda Arnoldiego, metoda Kryłowa-Schura, język programowania Julia

Fast methods for finding eigenmodes of axisymmetric, microwave resonators

Abstract

The paper deals with fast methods for finding eigenmodes of axisymmetric, microwave resonators using numerical methods developed in the Julia language. It is an issue used in the modeling of resonant cavities used in klystrons. The first chapter introduces the theoretical background, discusses the physical and mathematical basis of the problem and the relationship to the generalized eigenproblem.

In the practical part, several numerical methods dedicated to solving the generalized eigenproblem were analyzed. Described methods include the default function for solving the generalized eigenproblem in the LinearAlgebra.jl library, Arnoldi methods in the Arpack.jl package, the Krylov-Schur method implemented in ArnoldiMethod.jl and the Jacobi-Davidson QZ method. The algorithms and properties of each of these methods were presented, highlighting their performance and practical application.

The next stage of the work was the evaluation of the developed software. The results obtained with the Julia implementation were compared with those of commercial solutions, such as COMSOL Multiphysics and Matlab. Tests conducted on the eigenvalue methods implemented in the Julia language showed better results than for commercial software, offering comparable numerical results with shorter running times.

Keywords: resonant cavities, generalized eigenproblem, Arnoldi method, Krylov-Schur method, Julia programming language

Spis treści

Dla Julii

Rozdział 1

Wstęp

Rezonatory mikrofalowe odgrywają istotną rolę w nowoczesnych technologiach, znajdując zastosowanie m.in. w systemach komunikacyjnych czy radarach. W szczególności rezonatory mikrofalowe znajdują zastosowanie w klistronach, w których odpowiadają za nadanie pobudzanemu sygnałowi odpowiedniej częstotliwości. Dostosowanie właściwości rezonatorów mikrofalowych wymaga szczegółowej analizy pól własnych, które determinują charakterystykę częstotliwościową i wydajność tych urządzeń. W niniejszej pracy przedstawiono metody wyznaczania pól własnych rezonatorów z wykorzystaniem uogólnionego problemu wartości własnych (ang. Generalized Eigenvalue Problem, GEP). Metody te umożliwiają modelowanie rezonatorów o pożądanej częstotliwości, co jest istotne z punktu widzenia projektowania i optymalizacji tego typu struktur.

1.1 Cel pracy

Celem tej pracy jest opracowanie kodu w Julia, który przy wykorzystaniu metody elementów skończonych pozwoli na szybkie i dokładne znajdowanie pól własnych oscylatorów mikrofalowych. Zadanie polega na znalezieniu metody pozwalającej na rozwiązanie ogólnego problemu własnego znajdując tylko wybrane pola własne we wnęce/rezonatorze. Uzyskane wyniki zostaną poddane weryfikacji z dostępnym oprogramowaniem komercyjnym.

1.2 Zakres i struktura pracy

W części teoretycznej zostaną omówione zagadnienia związane z wnękami rezonansowymi, metodą elementów skończonych oraz standardowym i uogólnionym problemem własnym wraz z powiązaniem fizycznej i matematycznej strony wymienionych zagadnień. Dalsza część pracy skupi się na analizie algorytmów rozwiązywania uogólnionego problemu własnego za pomocą specjalistycznych metod obliczeniowych w języku Julia. Omówione zostaną algorytmy takie jak rozkład QZ z biblioteki `LinearAlgebra.jl`, algorytm Arnoldiego z biblioteki `Arpack.jl`, metoda Krylova-Schura oraz metoda Jacobi-Davidsona QZ (JDQZ). Metody zostaną przeanalizowane pod kątem wydajności

i dopasowania do sformułowanego problemu. Zasadniczą częścią pracy będą badania porównawcze, które obejmą weryfikację wydajności opracowanych implementacji. Zostanie przeprowadzona analiza czasu działania metod zaimplementowanych w języku Julia oraz porównanie otrzymanych wyników z komercyjnymi narzędziami, takimi jak MATLAB i COMSOL Multiphysics. Pozwoli to na obiektywną ocenę skuteczności zaproponowanych rozwiązań.

1.3 Przegląd literatury i obecny stan wiedzy

Stan wiedzy i literatury w dziedzinie modelowania rezonatorów mikrofalowych i rozwiązywania problemu własnego dynamicznie się rozwija od połowy XX wieku aż do dzisiaj, ze szczególnym uwzględnieniem przełomu XX i XXI wieku. Kluczowe publikacje ostatnich dekad, począwszy od pracy Francisa o algorytmie QR [7], poprzez prace dotyczące metod opartych o podprzestrzenie Kryłowa [16, 17] czy algorytm Jacobi-Davidsona [6], systematycznie rozwijają techniki rozwiązywania uogólnionego problemu własnego wykorzystywanego w różnych dziedzinach nauki.

Julia, język programowania zapoczątkowany w 2012 roku, został stworzony z myślą o wysokowydajnych obliczeniach numerycznych [4]. Jego koncepcja narodziła się z potrzeby połączenia wydajności języków kompilowanych z elastycznością języków skryptowych. Ewolucja bibliotek numerycznych w Julii przebiegała równolegle z rozwojem samego języka. Kluczowe biblioteki do obliczeń numerycznych takie jak LinearAlgebra.jl były implementowane wraz z rozwojem języka, umożliwiając wydajne operacje algebraiczne.

Rozdział 2

Wprowadzenie teoretyczne

Przed przejściem do właściwego opisu działania metod pozwalających na znajdowanie pól własnych osiowosymetrycznych rezonatorów mikrofalowych, należy wprowadzić pojęcie wnęk rezonansowych, poruszyć kwestie teoretyczne związane z podstawami fizycznymi i metodami numerycznymi wykorzystywanymi w obliczeniach elektromagnetycznych oraz ich związku z uogólnionym problemem własnym.

2.1 Wnęki rezonansowe

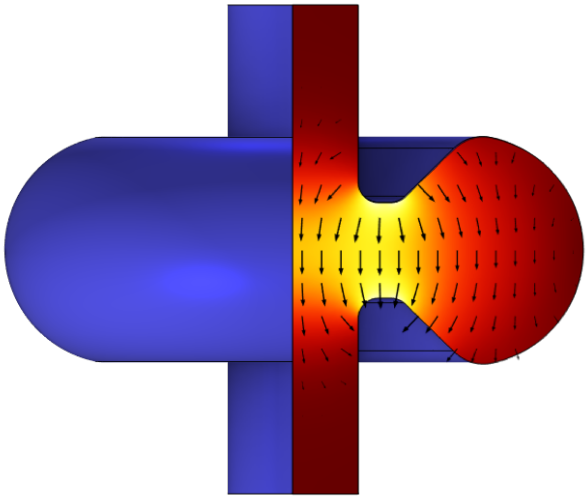
Wnękami rezonansowymi nazywamy struktury o metalowych ścianach, do których wprowadzany jest harmoniczny sygnał wzbudzający. W zależności od częstotliwości pobudzenia, wewnątrz wnęki pojawia się harmoniczne pole elektromagnetyczne o rozkładzie zależnym od częstotliwości oraz kształtu wnęki. Ponieważ metalowa powierzchnia wnęki jest w ogólności na jednakowym potencjale, dopuszczalnymi rodzajami pól są TM (ang. transverse-magnetic) oraz TE (ang. electric). Przekrój przykładowej osiowosymetrycznej wnęki przedstawia Rysunek 1.

Na rodzaje pola elektromagnetycznego można spojrzeć też z perspektywy analizy wartości i wektorów własnych macierzy wynikających z dyskretyzacji równań Maxwella przy pomocy Metody Elementów Skończonych (MES). Załóżmy trójwymiarową siatkę czworościenną, powstałą w wyniku dyskretyzacji metalowego cylindra [14]. Powierzchnia zewnętrzna cylindra jest idealnym przewodnikiem (ang. Perfect Electric Conductor – PEC), a wewnątrz wnęki znajduje się idealna próżnia.

Rozważmy sformułowanie równań Maxwella zakładające oscylujące z częstotliwością ω pole elektromagnetyczne:

$$\nabla \times \vec{\mathrm{E}} = -\mathrm{j}\omega\vec{\mathrm{B}}$$

$$\nabla \times \frac{\vec{\mathrm{B}}}{\mu_0} = \mathrm{j}\omega\epsilon_0\vec{\mathrm{E}}$$



Rysunek 1. Przekrój przez osiowosymetryczną wnękę rezonansową

gdzie $\vec{\mathrm{E}}$ oraz $\vec{\mathrm{B}}$ to odpowiednio pole elektryczne oraz pole indukcji magnetycznej, j to jednostka urojona, natomiast ϵ_0 oraz μ_0 to odpowiednio przenikalności elektryczne i magnetyczne próżni.

Z powyższych równań możliwe jest wyznaczenia równania falowego względem wybranego pola (elektrycznego lub magnetycznego). W dalszej części rozważań będziemy rozwiązywać równanie falowe względem pola elektrycznego w próżni, bez źródeł:

$$\nabla \times \nabla \times \vec{\mathrm{E}} - \mathrm{k}_0^2 \vec{\mathrm{E}} = 0,$$

gdzie $\mathrm{k}_0 = \frac{2\pi \mathrm{f}}{\mathrm{c}_0}$ to liczba falowa w próżni, dla fali o częstotliwości f i prędkości równej prędkości światła c_0.

Powyższe równania można zdyskretyzować przy pomocy Metody Elementów Skończonych, zakładając wykorzystanie elementów krawędziowych (tzn. z każdą krawędzią siatki skojarzony jest jeden stopień swobody). Krawędzie na powierzchni zewnętrznej wnęki mają zadany jednorodny warunek brzegowy Dirichleta, równoważny wyzerowaniu składowej stycznej do powierzchni $\hat{\mathrm{n}} \times \vec{\mathrm{E}} = 0$ (warunek PEC).

Efektem dyskretyzacji jest macierzowy układ równań:

$$[\mathrm{S}]\{\mathrm{e}\} = \mathrm{k}^2[\mathrm{T}]\{\mathrm{e}\},$$

gdzie $[\mathrm{S}]$ to macierz sztywności, $[\mathrm{T}]$ to macierz mas, natomiast $\{\mathrm{e}\}$ to wektor współczynników skojarzonych ze stopniami swobody, tj. krawędziami siatki. Powyższe równanie ma postać uogólnionego problemu własnego:

$$[\mathrm{A}]\{\mathrm{v}\} = \lambda[\mathrm{B}]\{\mathrm{v}\},$$

którego rozwiązaniami są pary wartości własnych λ i odpowiadających im wektorów własnych v.

2.2 Standardowy problem własny

Problem własny (ang. eigenvalue problem) polega na znalezieniu niezerowych wektorów, zwanych wektorami własnymi, oraz odpowiadających im skalarów, zwanych wartościami własnymi, które spełniają równanie macierzowe. Problem własny można zapisać w postaci:

$$\mathrm{A}\mathrm{v} = \lambda \mathrm{v}$$

gdzie:

- A jest daną macierzą kwadratową o wymiarach $\mathrm{N} \times \mathrm{N}$,
- v jest wektorem własnym o liczbie elementów N,
- λ jest wartością własną, liczbą zespoloną.

2.3 Uogólniony problem własny

Uogólniony problem własny jest rozszerzeniem klasycznego problemu własnego. W tym przypadku mamy do czynienia z dwoma macierzami, a celem jest znalezienie wartości własnych λ i odpowiadających im wektorów własnych v, które spełniają równanie:

$$\mathrm{A}\mathrm{v} = \lambda \mathrm{B}\mathrm{v}$$

gdzie:

- A i B są danymi macierzami kwadratowymi o wymiarach $\mathrm{N} \times \mathrm{N}$,
- v jest wektorem własnym o liczbie elementów N,
- λ jest wartością własną, liczbą zespoloną.

W opisywanym w tej pracy problemie, wartość własna $\lambda = \mathrm{k}_0^2$ odpowiada kwadratowi liczby falowej (pozwala wyznaczyć częstotliwość oscylacji), natomiast wektor własny $\mathrm{v} = \mathrm{e}$ pozwala wyznaczyć rozkład pola elektrycznego.

W rozpatrywanym uogólnionym problemie wartości własnych (A) i (B) są kwadratowymi macierzami o $\mathrm{N} \times \mathrm{N}$ elementach, gdzie N to liczba krawędzi siatki. Rozwiązaniem tego problemu jest więc N wartości własnych i N wektorów o N elementach. Ponieważ w opisywanym problemie nie modelujemy strat w ścianach wnęki, ani wewnątrz jej objętości (zakładamy $\sigma = 0\frac{\mathrm{S}}{\mathrm{m}}$), to jesteśmy zainteresowani jedynie rozwiązaniami, dla których wartości własne są liczbami rzeczywistymi. Rozwiązania, dla których część urojona λ jest niezerowa, charakteryzują się niefizycznym tłumieniem i wynikają z charakterystyki sformułowania MES. Spodziewamy się, że spośród N wartości własnych M z nich będzie niefizycznymi rodzajami pola. W [5] można znaleźć informację, że w powyższym sformułowaniu liczba fałszywych rodzajów pola M równa się liczbie węzłów siatki.

Przekształcenie wartości własnej $\mathrm{k} = \omega^2$ na faktyczną częstotliwość w hertzach (Hz) odbywa się za pomocą wzoru:

$$\mathrm{f} = \frac{\mathrm{c}_0\sqrt{\mathrm{k}}}{2\pi}.$$

Rozdział 3

Rozwiązywanie uogólnionego problemu własnego

Uogólniony problem własny stanowi wyzwanie w dziedzinie analizy numerycznej oraz modelowania matematycznego złożonych systemów fizycznych. Rozwiązywanie GEP obejmuje szerokie spektrum zastosowań – od mechaniki kwantowej, poprzez projektowanie struktur inżynieryjnych, po zaawansowane symulacje elektromagnetyczne.

Ze względu na naturę GEP jako rozszerzenia zwykłego problemu własnego, wiele metod wykorzystywanych dla problemu zwykłego ma swoją adaptację dla GEP. Spośród głównych nurtów współczesnych metod rozwiązywania GEP możemy wyróżnić [2, 12]:

- Metoda potęgowa: Prosta iteracyjna metoda, która może być używana do znajdowania największej (w sensie modułu) wartości własnej i odpowiadającego jej wektora własnego.
  - Odwrotna metoda potęgowa: odmiana metody potęgowej, która koncentruje się na znalezieniu wartości własnej najbliższej zadanej wartości, stosując odwrotność macierzy.
  - Metoda Rayleigha: łączy metodę potęgową z ilorazem Rayleigha, oferując szybszą zbieżność.
- Rozkład QZ (uogólniona dekompozycja Schura): rozwinięcie rozkładu QR dla GEP do obliczania wartości własnych poprzez przekształcenie macierzy w pary macierzy trójkątnych.
- Metody oparte o podprzestrzenie Kryłowa:
  - Metoda Arnoldiego: uogólnienie metody potęgowej, używane do znajdowania kilku wartości własnych i odpowiadających im wektorów własnych, szczególnie efektywne dla dużych, rzadkich macierzy.
  - Metoda Lanczosa: optymalizacja metody Arnoldiego dla symetrycznych macierzy, redukując potrzebę operacji i pamięci, zachowując efektywność w znajdowaniu wartości własnych.
  - Metoda Kryłowa-Schura: rozwinięcie metody Arnoldiego z restartami.

- Metoda Jacobi-Davidson QZ: metoda łącząca algorytm Jacobi-Davidsona z rozkładem QZ, efektywna w znajdowaniu wybranych wartości własnych i wektorów własnych w dużych, rzadkich układach.
- LOBPCG (Locally Optimal Block Preconditioned Conjugate Gradient Method): metoda łącząca poszukiwanie wartości własnych za pomocą ilorazu Rayleigha z prekondycjonowanym spadkiem gradientu.

3.1 Metoda QZ - LinearAlgebra.jl

Algorytm QR

Algorytm QR[7] jest metodą numeryczną stosowaną do rozkładu macierzy kwadratowej $\mathrm{A} \in \mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ na iloczyn dwóch macierzy: unitarnej macierzy $\mathrm{Q} \in \mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ oraz macierzy trójkątnej górnej $\mathrm{R} \in \mathbb{C}^{\mathrm{n}\times\mathrm{n}}$. Rozkład ten można zapisać jako:

$$\mathrm{A} = \mathrm{QR}.$$

Rozkład macierzy A na macierze Q i R możemy przeprowadzić przy użyciu metod takich jak ortogonalizacja Grama-Schmidta, rotacje Givensa i odbicia Householdera. Na przekątnej macierzy R występują bloki 1×1, odpowiadające rzeczywistym wartościom własnym, lub bloki 2×2, odpowiadające parom sprzężonych wartości własnych.

Faktoryzacja QR za pomocą odbić Householdera

Jedną ze stosowanych metod dekompozycji QR jest metoda odbić Householdera.

Macierz odbicia Householdera jest reprezentowana jako H:

$$\mathrm{H} = \mathrm{I} - 2\frac{\mathrm{v}\mathrm{v}^{\mathrm{T}}}{\mathrm{v}^{\mathrm{T}}\mathrm{v}},$$

gdzie v to wektor, a I to macierz jednostkowa. Macierz H jest symetryczna i ortogonalna, co oznacza $\mathrm{H}^{\mathrm{T}} = \mathrm{H}$ oraz $\mathrm{H}^{\mathrm{T}}\mathrm{H} = \mathrm{I}$.

Algorytm dekompozycji QR z wykorzystaniem odbić Householdera polega na stopniowej transformacji macierzy A o wymiarach $\mathrm{n} \times \mathrm{n}$ do postaci górnotrójkątnej. Na podstawie pierwszej kolumny macierzy A tworzymy odbicie Householdera H_1, przez które mnożymy macierz A. Wynikiem jest macierz $\mathrm{H}_1\mathrm{A}$ z wyzerowaną kolumną aż do wartości diagonalnych.

$$\mathrm{H}_1\mathrm{A} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{\mathrm{n}} \\ 0 & & & \\ \vdots & & \mathrm{A}' & \\ 0 & & & \end{bmatrix}$$

gdzie A' jest macierzą o wymiarach $(\mathrm{n}-1)\times(\mathrm{n}-1)$, a $\alpha_1\alpha_2\cdots\alpha_\mathrm{n}$ to elementy pierwszego wiersza zmodyfikowanej macierzy $\mathrm{Q}_1\mathrm{A}$.

Kroki powtarzamy dla kolejnych kolumn. Po $\mathrm{n}-1$ krokach, A przekształca się w macierz trójkątną górną R. Macierz ortogonalna Q jest iloczynem wszystkich macierzy Householdera:

$$\mathrm{Q} = \mathrm{H}_1\mathrm{H}_2\cdots\mathrm{H}_{\mathrm{n}-1}$$

Rozkład Schura

Algorytm QR jest narzędziem umożliwiającym uzyskanie rozkładu Schura, Dla każdej macierzy $\mathrm{A}\in\mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ istnieje unitarna macierz $\mathrm{Q}\in\mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ taka, że:

$$\mathrm{Q}^*\mathrm{A}\mathrm{Q} = \mathrm{T},$$

gdzie T jest macierzą trójkątną górną, a Q^* oznacza sprzężenie hermitowskie macierzy Q. Rozkład taki nazywamy rozkład Schura.

Jeśli $\mathrm{A},\mathrm{B}\in\mathbb{C}^{\mathrm{n}\times\mathrm{n}}$, to istnieją unitarne macierze $\mathrm{Q},\mathrm{Z}\in\mathbb{C}^{\mathrm{n}\times\mathrm{n}}$ takie, że:

$$\mathrm{Q}^*\mathrm{A}\mathrm{Z} = \mathrm{T}^\mathrm{A},$$

$$\mathrm{Q}^*\mathrm{B}\mathrm{Z} = \mathrm{T}^\mathrm{B},$$

gdzie T^A i T^B są macierzami trójkątnymi górnymi. Jeśli dla pewnego k, $\mathrm{t}^\mathrm{A}_{\mathrm{kk}} = \mathrm{t}^\mathrm{B}_{\mathrm{kk}} = 0$, to $\sigma(\mathrm{A},\mathrm{B}) = \mathbb{C}$. W przeciwnym razie:

$$\sigma(\mathrm{A},\mathrm{B}) = \left\{ \frac{\mathrm{t}^\mathrm{A}_{\mathrm{ii}}}{\mathrm{t}^\mathrm{B}_{\mathrm{ii}}} \mid \mathrm{t}^\mathrm{B}_{\mathrm{ii}} \neq 0 \right\}.$$

Rozkład taki nazywamy uogólnionym rozkładem Schura.

Algorytm QZ

Algorytm QZ[13] jest analogiczną do algorytmu QR metodą numeryczną do obliczania macierzy Q i Z uogólnionego rozkładu Schura dla par macierzy (A,B). Algorytm QZ przebiega według następującego schematu:

1. Macierze A i B są najpierw jednocześnie redukowane do form skondensowanych za pomocą równoważnych transformacji unitarnych. Macierz A jest redukowana do macierzy Hessenberga, a B jest redukowana do górnej formy trójkątnej (formy Schura). Następnie celem jest redukcja A do górnej formy Schura, przy zachowaniu B w tej formie. To jest realizowane w następnym kroku.

2. Krok QR na AB^{-1} (bez formowania faktycznego iloczynu macierzy) jest realizowany za pomocą transformacji unitarnych Q i Z na parze macierzy A i B; to jest istota algorytmu QZ. Po odpowiedniej liczbie iteracji macierze zostają przekształcone do do trójkątnej lub quasi-trójkątnej formy (tj. z blokami 2×2 wzdłuż przekątnej, w przypadku gdy chcemy uniknąć arytmetyki zespolonej), przy zachowaniu trójkątnej struktury B. Po zakończeniu iteracji uzyskujemy uogólnioną formę Schura macierzy A i B, to znaczy, że obliczyliśmy macierze ortogonalne Q i Z tak, że QAZ i QBZ są macierzami górnymi trójkątnymi.

3. Wartości własne można obliczyć z przekątnych formy trójkątnej. Wektory własne można obliczyć jako wektory własne problemu trójkątnego, a następnie przekształcić z powrotem za pomocą Z do wektorów własnych oryginalnego problemu.

Algorytm QZ wymaga $\mathrm{O}(\mathrm{n}^3)$ operacji zmiennoprzecinkowych i $\mathrm{O}(\mathrm{n}^2)$ lokalizacji pamięci, gdzie n jest rzędem macierzy A i B. Dokładniej, wymaga około $30\mathrm{n}^3$ operacji zmiennoprzecinkowych do obliczenia samych wartości własnych [3].

Procedury numeryczne oparte o algorytm QZ dostępne są w bibliotece LAPACK. W przypadku programu MATLAB są one dostępne za pośrednictwem funkcji `eig(A,B)` [9]. W LAPACK[1] dostępne są następujące procedury sterujące do wykonywania różnych zadań:

- `gges` – Oblicza uogólnione wartości własne, uogólnioną postać Schura oraz opcjonalnie lewe i/lub prawe macierze wektorów Schura.
- `ggesx` – `gges` wraz z oszacowaniami uwarunkowania dla wartości własnych i podprzestrzeni deflacyjnych.
- `ggev` – Uogólnione wartości własne oraz opcjonalnie lewe i/lub prawe uogólnione wektory własne.
- `ggevx` – `ggev` wraz z oszacowaniami uwarunkowania dla wartości własnych i wektorów własnych.
- `ggev3` – Oblicza wartości własne (`eig`) z wykorzystaniem blokowej wersji algorytmu (od wersji LAPACK 3.6.0) [11].
- `gges3` – Oblicza uogólnioną postać Schura z wykorzystaniem blokowej wersji algorytmu (od wersji LAPACK 3.6.0).

W bibliotece `LinearAlgebra.jl` w języku Julia funkcja `eigen()` wykorzystuje rutynę LAPACK `ggev3`.

3.2 Metoda Arnoldiego zaimplementowana w bibliotece Arpack.jl

Znajdowanie wybranych wartości własnych

Metody QR i QZ obliczają wartości własne dla całego spektrum danej macierzy. Chociaż są one efektywne dla macierzy gęstych i o niewielkim rozmiarze, to przy dużych, rzadkich macierzach są

one kosztowne obliczeniowo. Wykonują one obliczenia potrzebne do uzyskania wszystkich wartości własnych, nawet jeśli interesują nas jedynie niektóre z nich.

W przypadku rozważanym w tej pracy, czyli poszukiwaniu częstotliwości rezonatorów mikrofalowych, interesują nas częstotliwości z zakresu mikrofalowego. Co więcej, macierze masy i sztywności w modelowanym problemie często są macierzami rzadkimi, symetrycznymi, o wartościach skupionych blisko głównej przekątnej. Mając na uwadze charakterystykę problemu będziemy chcieli korzystać z metod które znajdują wartości własne w interesującym nas zakresie, jednocześnie wykorzystując na korzyść charakterystykę badanego problemu, aby uzyskać jak najszybszy czas obliczeń.

Analizę kolejnych metod, innych niż QZ, rozpoczynamy od problemu poszukiwania wartości własnych w pobliżu interesujących nas wartości docelowych. Zaczynamy od przykładu wykorzystującego metodę potęgową, która jest prostą iteracyjną metodą obliczania wartości własnej macierzy. W podstawowej formie rezultem działania algorytmu będą wartości własne o największym module. Metoda działa następująco: rozpoczyna się ona od pewnego wektora początkowego $\mathrm{x}^{(0)}$ i generuje ciąg kolejnych przybliżeń $\mathrm{x}^{(\mathrm{k})}$ zgodnie z iteracją:

$$\mathrm{x}^{(\mathrm{k}+1)} = \frac{\mathrm{A}\mathrm{x}^{(\mathrm{k})}}{||\mathrm{A}\mathrm{x}^{(\mathrm{k})}||}$$

W każdym kroku iteracji normalizujemy wektor $\mathrm{A}\mathrm{x}^{(\mathrm{k})}$, aby otrzymać następny wektor przybliżenia $\mathrm{x}^{(\mathrm{k}+1)}$. Ciąg ten zbieżny jest do wektora własnego macierzy A odpowiadającego wartości własnej o największym module. W przypadku macierzy posiadających wartości własne będące sprzężonymi liczbami zespolonymi istnieje ryzyko wystąpienia oscylacji i utraty zbieżności ze względu na wystąpienie dwóch wartości własnych o takich samych modułach.

Aby obliczyć wartość własną skojarzoną z danym wektorem własnym x, możemy wykorzystać iloraz Rayleigha:

$$\lambda \approx \frac{\mathrm{x}^{\mathrm{T}}\mathrm{A}\mathrm{x}}{\mathrm{x}^{\mathrm{T}}\mathrm{x}}.$$

Metodę potęgową możemy zmodyfikować tak, aby zamiast wartości własnych o największym module poszukiwać wartości własnych bliskich zadanej wartości celu. W tym celu możemy wykorzystać modyfikację polegającą na przesunięciu i odwróceniu macierzy A. Dla odwróconej macierzy A metoda potęgowa znajduje wartości własne macierzy A o najmniejszym module, a dodanie przesunięcia σ powoduje, że znajdowane są wartości własne bliskie wartości przesunięcia.

Krok iteracyjny dla zmodyfikowanej wersji wygląda następująco:

$$\mathrm{x}^{(\mathrm{k})} := (\mathrm{A} - \sigma\mathrm{I})^{-1}\mathrm{x}^{(\mathrm{k}-1)} \quad \Leftrightarrow \quad (\mathrm{A} - \sigma\mathrm{I})\mathrm{x}^{(\mathrm{k})} := \mathrm{x}^{(\mathrm{k}-1)}$$

Oznacza to, że w każdym kroku iteracji wyznaczamy następny wektor $\mathrm{x}^{(\mathrm{k})}$ przez rozwiązanie układu liniowego $(\mathrm{A} - \sigma\mathrm{I})\mathrm{x}^{(\mathrm{k})} = \mathrm{x}^{(\mathrm{k}-1)}$. Iteracja zbieżna jest do wektora własnego odpowiadającego wartości własnej najbliższej przesunięciu σ.

Zastosowanie metody potęgowej do uogólnionego problemu wartości własnych $\mathrm{Ax} = \lambda \mathrm{Bx}$ prowadzi do iteracji:

$$\mathrm{x}^{(\mathrm{k})} := \mathrm{B}^{-1}\mathrm{A}\mathrm{x}^{(\mathrm{k}-1)} \quad \Leftrightarrow \quad (\mathrm{A} - \sigma\mathrm{B})\mathrm{x}^{(\mathrm{k})} := \mathrm{B}\mathrm{x}^{(\mathrm{k}-1)}$$

Iterację można zinterpretować jako wykonywanie zwykłej metody potęgowej dla problemu wartości własnych dla macierzy zdefiniowanej jako $(\mathrm{A}-\sigma\mathrm{B})^{-1}\mathrm{B}\mathrm{x} = \mu\mathrm{x}$, gdzie $\mu = \frac{1}{\lambda-\sigma}$. Wówczas przybliżenie wartości własnej bliskiej wartości σ jest dane przez:

$$\lambda = \frac{1}{\mu} + \sigma$$

Warto zauważyć, że poruszany w tej pracy problem dotyczy macierzy symetrycznych. W związku z tym w celu obliczenia kolejnych wartości własnych $\lambda_2, \lambda_3, \ldots$ możemy skorzystać z faktu, że wektory własne macierzy symetrycznych są ortogonalne. Jeżeli znamy wektor własny u_1 odpowiadający najmniejszej wartości własnej λ_1, możemy go wykorzystać do obliczenia kolejnej pary $(\lambda_2, \mathrm{u}_2)$. W przypadku tej procedury wektor startowy $\mathrm{x}^{(0)}$ musi być ortogonalny do już znanych wektorów własnych $\mathrm{u}_1, \ldots, \mathrm{u}_\mathrm{j}$. Po zlokalizowaniu pierwszej wartości własnej i odpowiadającego jej wektora własnego, rozpoczynamy proces od takiego wektora $\mathrm{x}^{(0)}$, który jest ortogonalny do wszystkich wcześniej obliczonych wektorów własnych. W praktyce oznacza to, że każda nowa iteracja $\mathrm{x}^{(0)}$ jest aktualizowana tak, aby pozostawała ortogonalna do wcześniej obliczonych wektorów, co można zapisać jako:

$$\mathrm{x}^{(0)} \perp \mathrm{u}_1, \ldots, \mathrm{u}_{\mathrm{j}-1}.$$

Warto zwrócić uwagę, że w dokładnej arytmetyce warunek $\mathrm{u}_1^{(\mathrm{k})}(\mathrm{x}) = \ldots = \mathrm{u}_{\mathrm{j}-1}^{(\mathrm{k})}(\mathrm{x}) = 0$ implikuje, że wszystkie $\mathrm{x}^{(\mathrm{k})}$ są ortogonalne do $\mathrm{u}_1, \ldots, \mathrm{u}_{\mathrm{j}-1}$. Jednak w przypadku obliczeń numerycznych należy spodziewać się błędów zaokrąglania, które wprowadzają składowe w kierunkach $\mathrm{u}_1, \ldots, \mathrm{u}_{\mathrm{j}-1}$. Dlatego konieczne jest wymuszanie warunków ortogonalności podczas iteracji, co jednak przekłada się na większe koszty obliczeniowe. Rozwiązaniem tego problemu jest metoda iteracji podprzestrzeni, która dokonuje ortogonalizacji wszystkich dotychczasowo znalezionych wektorów własnych za pomocą algorytmu QR, bądź metody oparte o podprzestrzenie Kryłowa, takie jak metoda Arnoldiego czy Lanczosa.

Podprzestrzenie Kryłowa

Metoda Arnoldiego jest algorytmem iteracyjnym opartym o podprzestrzenie Kryłowa, który dla macierzy A oblicza ortogonalną bazę podprzestrzeni Kryłowa Q, a jednocześnie oblicza rzutowaną macierz H w sposób wymagający mniej obliczeń dzięki redukcji złożoności problemu.

Wykorzystanie podejścia projekcyjnego dla problemów własnych ma na celu obliczenie częściowego rozwiązania problemu wartości własnych przy znacznie mniej złożonej wymiarowości, to znaczy, że

dla danej macierzy kwadratowej A rzędu n, celem jest obliczenie niewielkiej liczby wartości własnych $\lambda_\mathrm{i}, \mathrm{z_i}$, $\mathrm{i} = 1, \ldots, \mathrm{k}$, przy czym $\mathrm{k} \ll \mathrm{n}$. Podstawową zasadą metod projekcji jest znalezienie najlepszych przybliżeń wartości własnych w danej podprzestrzeni o małym wymiarze.

W podstawowym założeniu metodę projekcyjną realizuje procedura Rayleigha-Ritza. Dla danej macierzy $\mathrm{n} \times \mathrm{m}$ A, $\mathrm{k} \leq \mathrm{m} \ll \mathrm{n}$, której kolumny $\mathrm{v_i}$ stanowią ortonormalną bazę danej podprzestrzeni $\mathcal{V}$, tzn. $\mathrm{V^T V} = \mathrm{I_m}$ dla $\mathcal{V} = \mathrm{span}\{\mathrm{v_1}, \mathrm{v_2}, \ldots, \mathrm{v_m}\}$, procedura Rayleigha-Ritza polega na obliczeniu $\mathrm{H} = \mathrm{V^T A V}$, która jest macierzą kwadratową rzędu m, a następnie rozwiązaniu związanego z nią problemu wartości własnych, tj. $\mathrm{H y_i} = \theta_\mathrm{i} \mathrm{y_i}$. Przybliżone wartości własne $\lambda_\mathrm{i}, \mathrm{x_i}$ oryginalnego problemu określone są jako $\lambda_\mathrm{i} = \theta_\mathrm{i}$ oraz $\mathrm{x_i} = \mathrm{V y_i}$ i nazywane są odpowiednio wartościami i wektorami Ritza. Należy zauważyć, że wektory Ritza należą do podprzestrzeni $\mathcal{V}$. Macierz H jest projekcją A na $\mathcal{V}$, stąd nazwa tej klasy metod.

Powyższe procedury działają lepiej, gdy podprzestrzeń $\mathcal{V}$ przybliża niezmienniczą podprzestrzeń A. Z tego powodu jako bazę ortonormalną wykorzystuje się podprzestrzeń Kryłowa związaną z macierzą A i danym początkowym wektorem $\mathrm{x_1}$, rozpiętą na wektorach $\mathrm{x_1}, \mathrm{A x_1}, \mathrm{A^2 x_1}, \ldots, \mathrm{A^{m-1}}$:

$$\mathcal{K}_\mathrm{m}(\mathrm{A}, \mathrm{x_1}) = \mathrm{span}\{\mathrm{x_1}, \mathrm{A x_1}, \mathrm{A^2 x_1}, \ldots, \mathrm{A^{m-1} x_1}\}.$$

Podprzestrzenie Kryłowa to generowane są przez kolejne potęgi macierzy A stosowane do wektora początkowego x. Warto zauważyć, że dla macierzy $\mathrm{n} \times \mathrm{n}$ kolumny macierzy Kryłowa $\mathrm{K^{n+1}(x)}$ są liniowo zależne, ponieważ podprzestrzeń $\mathrm{K_n(A, x_1)}$ nie może mieć wymiaru większego niż n. Jeśli u jest wektorem własnym odpowiadającym wartości własnej λ, to $\mathrm{Au} = \lambda \mathrm{u}$, a w konsekwencji $\mathrm{K^2(u)} = \mathrm{span}\{\mathrm{u}, \mathrm{Au}\} = \mathrm{span}\{\mathrm{u}\} = \mathrm{K^1(u)}$. Istnieje więc najmniejsze m, $1 \leq \mathrm{m} \leq \mathrm{n}$, zależne od x, takie że: $\mathrm{K^1(x)} \subsetneq \mathrm{K^2(x)} \subsetneq \cdots \subsetneq \mathrm{K^m(x)} = \mathrm{K^{m+1}(x)}$. Konstrukcja podprzestrzeni Kryłowa ułatwia obliczenia wartości własnych poprzez redukację problemu do równoważnego w podprzestrzeni o mniejszym wymiarze niż oryginalnej macierzy A.

Algorytm Arnoldiego

Algorytm Arnoldiego jest algorytmem numerycznym wykorzystywanym do rozwiązywania problemów własnych w oparciu o konstrukcję ortonormalnej bazy podprzestrzeni Kryłowa $\mathrm{K^m(x)}$ dla macierzy A i wektora początkowego x. Działanie zostało zaprezentowane w postaci pseudokodu jako Algorytm 1. W każdym kroku algorytm oblicza kolejny iloczyn $\mathrm{A v_j}$, a następnie dokonuje ortonormalizacji otrzymanego wektora względem macierzy dotychczas skonstruowanej bazy $\mathrm{v_j}$ za pomocą procesu Grama-Schmidta.

Algorytm zatrzymuje się po $\mathrm{h_{j+1,j}} = 0$, kiedy znaleziona zostaje podprzestrzeń niezmiennicza, bądź po m krokach. Kolejne iteracje algorytmu można interpretować jako dekompozycje rzędu m, zwane dekompozycją bądź relacją Arnoldiego, wyrażone jako:

$$\mathrm{A V_{m+1}} = \mathrm{V_m H_m} + \mathrm{f e_m^*}$$

gdzie:

- A jest macierzą początkową,
- $\mathrm{V_m}$ jest podprzestrzenią Kryłowa zawierającą ortonormalne wektory,
- $\mathrm{H_m}$ jest macierzą Hessenberga,
- f jest wektorem residualnym,
- $\mathrm{e_m^*}$ jest sprzężonym wektorem błędu.

Algorithm 1 Algorytm Arnoldiego

1: **Wejście:** Macierz A, liczba kroków m, oraz wektor początkowy $\mathrm{v_1}$ o normie 1
2: **Wyjście:** $(\mathrm{V_m}, \mathrm{H_m}, \mathrm{f}, \beta)$ takie, że $\mathrm{AV_m} = \mathrm{V_m H_m} + \mathrm{fe_m^*}$, $\beta = \|\mathrm{f}\|_2$
3: **for** $\mathrm{j} = 1, 2, \ldots, \mathrm{m}-1$ **do**
4: $\quad \mathrm{w} = \mathrm{Av_j}$
5: $\quad$ Ortogonalizuj w względem $\mathrm{V_j}$ (uzyskaj $\mathrm{h_{i,j}}$ dla $\mathrm{i} = 1, \ldots, \mathrm{j}$)
6: $\quad \mathrm{h_{j+1,j}} = \|\mathrm{w}\|_2$
7: $\quad$ **if** $\mathrm{h_{j+1,j}} = 0$ **then**
8: $\quad\quad$ zatrzymaj
9: $\quad$ **end if**
10: $\quad \mathrm{v_{j+1}} = \frac{\mathrm{w}}{\mathrm{h_{j+1,j}}}$
11: **end for**
12: $\mathrm{f} = \mathrm{Av_m}$
13: Ortogonalizuj f względem $\mathrm{V_m}$ (uzyskaj $\mathrm{h_{i,m}}$ dla $\mathrm{i} = 1, \ldots, \mathrm{m}$)
14: $\beta = \|\mathrm{f}\|_2$

Relacja ta pokazuje, jak w kolejnych iteracjach macierz A jest przybliżana przez budowaną podprzestrzeń ortogonalną $\mathrm{V_m}$. Wektor residualny f wskazuje kierunek różnicy między podprzestrzenią $\mathrm{V_m}$ otrzymaną w danym kroku a docelową podprzestrzenią niezmienniczą. Metoda ma tym lepszą zbieżność im bardziej wektor residualny f jest ukierunkowany na wektory własne, o czym mówi nam norma β.

W praktyce jednak wektor residualny często nie jest dobrze ukierunkowany i potrzebne jest wiele iteracji na osiągnięcie zadowalających rezultatów. Stanowi to problem, ponieważ im większą podprzestrzeń wykorzystujemy, tym większa jest złożoność pamięciowa i obliczeniowa naszego programu. Aby złagodzić ten problem, stosuje się restart, czyli przerwanie algorytmu po m iteracjach i jego ponowne uruchomienie z nowym wektorem początkowym $\mathrm{x_1}$, wyliczonym na podstawie dotychczas uzyskanych przybliżeń wartości własnych.

Jedną z metod restartu jest tzw. jawny restart, w której nowy wektor początkowy jest liniową kombinacją wybranych wektorów Ritza. W praktyce dobór odpowiednich parametrów do budowy nowego wektora początkowego jest trudny, co stanowi główną wadę tej metody [8].

Alternatywnym podejściem jest niejawny restart [16], w którym metoda Arnoldiego jest łączona z algorytmem QR z przesunięciami. Proces m-krokowy jest kompresowany do $(\mathrm{m}-\mathrm{d})$-krokowego procesu Arnoldiego, który następnie jest rozszerzany do m-krokowego. Kluczowym elementem jest to, że mała faktoryzacja zachowuje istotne informacje o wartościach własnych dużej faktoryzacji, co osiągane jest poprzez zastosowanie kilku kroków iteracji QR. Ten proces jest bardziej efektywny

niż jawny restart i można go interpretować jako niejawne zastosowanie wielomianu $\psi(\mathrm{A})$ do wektora początkowego. Technika niejawnego restartu została zaimplementowana w oprogramowaniu ARPACK.

Metoda Lanczosa (szczególny przypadek metody Arnoldiego dla macierzy symetrycznych) z grubymi restartami (ang. thick restart) jest kolejną z metod opartych o przestrzenie Kryłowa. Podejście grubych restartów pomaga zachować kluczowe informacje spektralne zmniejszając zarazem wpływ pozostałych wektorów Ritza i złożoność obliczeniową poprzez redukcję bazy podprzestrzeni Kryłowa. W porównaniu do prostych restartów, gdzie zaczynamy od jednego wektora Ritza obliczonego dla bazy sprzed restartu, w grubych restartach wybierane jest kilka wektorów. Na podstawie macierzy A obliczane są wektory Ritza stanowiące przybliżenie wektorów własnych. Następnie wybierana jest najkorzystniejsza część z nich według ustalonego kryterium (np. największych lub bliskich przesunięcia σ), która posłuży do utworzenia nowej bazy, a reszta jest usuwana [18].

Biblioteka Arpack.jl

Arpack.jl to biblioteka w ekosystemie Julia, która stanowi interfejs do biblioteki ARPACK. Pozwala ona na korzystanie z metod zaimplementowanych w języku FORTRAN bezpośrednio w środowisku Julii za pomocą natywnej składni języka.

Biblioteka oferuje zestaw rutyn do obliczania wartości własnych i wektorów własnych dużych macierzy rzadkich. Poniżej opisano poszczególne rutyny oraz tryby pracy.

Rutyny ARPACK dla macierzy symetrycznych

- **XYaupd** – Wstępne obliczenia wartości własnych (rzeczywiste/zespolone) – wartości Ritza.
- **XYeupd** – Pełne obliczenia wartości i wektorów własnych po osiągnięciu zbieżności przez XYaupd.
- **X – typ zmiennych liczbowych:**
  - `s` – float32
  - `d` – float64
  - `c` – complex32
  - `z` – complex64
- **Y – typ macierzy:**
  - `n` – niesymetryczna
  - `s` – symetryczna

Tryby pracy XYaupd

W przypadku rutyny `XYaupd` mamy do wyboru kilka trybów pracy w zależności od problemu, który chcemy rozwiązać:

- Tryb standardowego problemu własnego.
- Tryb uogólnionego problemu własnego.

- Tryb przesunięcia i odwrócenia.

Możemy także wybrać kolejność w której będą szukane kolejne wartości własne:

- **LM (Largest Magnitude)**: Wartości własne o największym module.
- **SM (Smallest Magnitude)**: Wartości własne o najmniejszym module.
- **LR (Largest Real)**: Wartości własne o największej części rzeczywistej.
- **SR (Smallest Real)**: Wartości własne o najmniejszej części rzeczywistej.
- **SI (Shift-and-Invert)**: Znalezienie wartości własnych bliskich przesunięciu σ.
- **BE (Both Ends)**: Szukanie wartości własnych z obu stron spektrum.

3.3 Metoda Kryłowa-Schura - ArnoldiMethod.jl

Można wykazać, że każda dekompozycja Kryłowa jest równoznaczna z dekompozycją Arnoldiego, to jest, że obie mają te same przybliżenia Ritza [17]. Podejście metody Kryłowa-Schura opiera się na iteracyjnym rozwijaniu mniejszego problemu z wykorzystaniem procesu Arnoldiego do coraz większego, efektem czego jest dekompozycja Kryłowa-Schura.

Mając relację Arnoldiego:

$$\mathrm{AQ_m} = \mathrm{Q_m H_m} + \mathrm{r_m e_m^T},$$

gdzie $\mathrm{H_m}$ jest macierzą Hessenberga, niech $\mathrm{H_m} = \mathrm{S_m T_m S_m^*}$ będzie rozkładem Schura dla $\mathrm{H_m}$ z macierzą unitarną $\mathrm{S_m}$ i trójkątną $\mathrm{T_m}$. Wówczas mamy:

$$\mathrm{AY_m} = \mathrm{Y_m T_m} + \mathrm{r_m s^*}, \quad \mathrm{Y_m} = \mathrm{Q_m S_m}, \quad \mathrm{s^*} = \mathrm{e_m^T S_m}.$$

Trójkątna forma $\mathrm{T_m}$ ułatwia analizę poszczególnych par Ritza. W szczególności, pozwala na przesuwanie niechcianych wartości Ritza do dolnego, prawego narożnika $\mathrm{T_m}$. Podobnie jak w podejściu grubego restartu tworzymy kryterium selekcji i dzielimy wektory Ritza na dwie macierze: $\mathrm{Y_1}$ (pożądane wektory) oraz $\mathrm{Y_2}$ (niechciane wektory). W ten sposób uzyskujemy:

$$\mathrm{A}[\mathrm{Y_1}, \mathrm{Y_2}] = [\mathrm{Y_1}, \mathrm{Y_2}] \begin{bmatrix} \mathrm{T_{11}} & \mathrm{T_{12}} \\ 0 & \mathrm{T_{22}} \end{bmatrix} + \beta_{\mathrm{k+1}} \mathrm{q_{k+1}} [\mathrm{s_1^*}, \mathrm{s_2^*}].$$

Zachowując pierwszy zestaw wektorów Ritza i usuwając resztę, otrzymujemy:

$$\mathrm{AY_1} - \mathrm{Y_1 T_{11}} = \beta_{\mathrm{k+1}} \mathrm{q_{k+1}} \mathrm{s_1^*}.$$

W metodzie grubego restartu Lanczosa uzyskujemy parę własną gdy $\beta_{\mathrm{k+1}} |\mathrm{s_{1k}}|$ jest wystarczająco małe. W przypadku ogólnej metody Kryłowa-Schura musimy przekształcić $\mathrm{s_1}$ do formy:

$$\mathrm{s_1} = \begin{bmatrix} \mathrm{s_1'} \\ \mathrm{s_1''} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{s_1''} \end{bmatrix},$$

wówczas odnaleźliśmy podprzestrzeń niezmienniczą:

$$\mathrm{A}[\mathrm{Y}_1', \mathrm{Y}_1''] - [\mathrm{Y}_1', \mathrm{Y}_1''] \begin{bmatrix} \mathrm{T}_{11} & \mathrm{T}_{12} \\ 0 & \mathrm{T}_{22} \end{bmatrix} = \beta_{\mathrm{k}+1} \mathrm{q}_{\mathrm{k}+1} [0^{\mathrm{T}}, \mathrm{s}_1''^{\mathrm{T}}].$$

w efekcie czego:

$$\mathrm{AY}_1' = \mathrm{Y}_1' \mathrm{T}_{11}.$$

W oparciu o metodę Kryłowa-Schura działa funkcja `eigs()` w programie MATLAB [10], służąca do obliczania wybranych wartości własnych oraz wektorów własnych dużych i rzadkich macierzy. W Julii metoda jest zaimplementowana w ramach funkcji `partialschur()` biblioteki `ArnoldiMethod.jl`.

3.4 Jacobi-Davidson i JDQZ

Kolejną metodą wykorzystywaną do rozwiązywania problemu własnego jest metoda Jacobi-Davidsona [15]. Jest to rozwinięcie metody Davidsona, która opiera się na poszukiwaniu wektora własnego przy użyciu residuum:

$$\mathrm{r} = \mathrm{Av} - \lambda \mathrm{v}.$$

Wektor poprawy t jest definiowany jako:

$$\mathrm{t} = (\mathrm{D} - \theta \mathrm{I})^{-1} \mathrm{r},$$

gdzie θ jest parametrem przesunięcia, który jest często poszukiwaną wartością własną. Przybliżony wektor własny jest następnie aktualizowany jako:

$$\mathrm{v}_{\mathrm{nowy}} = \mathrm{v} + \mathrm{t}.$$

Modyfikacją w metodzie Jacobiego-Davidsona jest zastosowanie ortogonalnej korekcji Jacobiego. Jeśli $\mathrm{r} = \mathrm{Au} - \theta \mathrm{u}$ jest residuum przybliżenia wartości Ritza, poprawa wektora u jest wyznaczana z równania korekcyjnego:

$$(\mathrm{I} - \mathrm{uu}^{\mathrm{H}})(\mathrm{A} - \theta \mathrm{I})(\mathrm{I} - \mathrm{uu}^{\mathrm{H}})\mathrm{t} = -\mathrm{r},$$

gdzie $\mathrm{t} \perp \mathrm{u}$ i $(\mathrm{I} - \mathrm{uu}^{\mathrm{H}})$ to operator projekcji ortogonalnej na $\mathrm{u}^{\perp}$.

Rozwinięcie metody o algorytm QZ pozwala na wykorzystanie jej do rozwiązywania uogólnionego problemu własnego [6]. Metoda ta jest szczególnie skuteczna w przypadku dużych, rzadkich macierzy oraz kiedy szukamy wartości własnych w pobliżu zadanej wartości. W języku Julia dostępna jest biblioteka JacobiDavidson.jl implementująca tę metodę, jednak nie udało się uzyskać zadowalającej zbieżności programów dla badanego problemu. Program nie zbiegał się do żadnego rozwiązania pomimo przeprowadzania obliczeń przez wiele godzin.

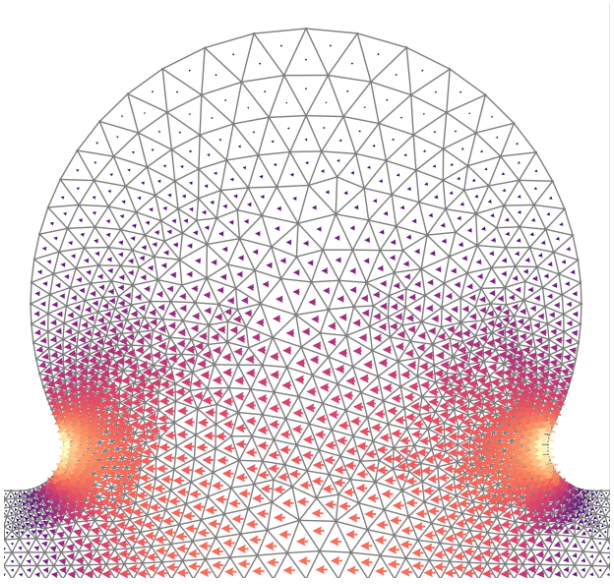
Rozdział 4

Testy oprogramowania

Implementacje metod wymienionych w poprzednim rozdziale zostały poddane testom wydajnościowym i porównane z oprogramowaniem komercyjnym. Do przeprowadzenia testów działania programu przygotowany został model wnęki osiowosymetrycznego rezonatora mikrofalowego. Model został opracowany dwuetapowo. Najpierw utworzony został model wnęki z wykorzystaniem metody elementów skończonych (Rys. 2), a następnie przekształcony do postaci uogólnionego problemu własnego jako macierze sztywności (S) i masy (T) (Rys. 3).

Obliczenia zostały przeprowadzone na maszynach o następujących parametrach

- Maszyna 1: Windows 10 Pro 22H2, CPU Intel Core i5-7500 @ 3.40GHz, 16 GB RAM
- Maszyna 2: Windows 11 Pro 23H2, CPU Intel Core i9-11900H @ 2.50GHz, 16 GB RAM.



Rysunek 2. Model wnęki rezonatora w MES z zaznaczoną siatką kwantyzacji i wektorami pola.

Rysunek 3. Wizualizacja występowania wartości macierzy rzadkiej sztywności modelowanego problemu własnego wnęki rezonatora. Wartości są skupione wokół głównej przekątnej.

4.1 Rezultaty metod opracowanych w Julii

Pierwsze badanie dotyczyło analizy porównawczej nowych metod znajdowania pól własnych ze standardową implementacją w Julii i programie MATLAB [10]. Porównane zostały metody:

- `eigs()` z biblioteki Arpack.jl w Julii (metoda Arnoldiego z niejawnymi restartami),
- `partialscur()` z biblioteki ArnoldiMethod.jl w Julii (metoda Kryłowa-Schura),
- `eigen()` z biblioteki LinearAlgebra.jl w Julii (metoda QZ),
- `eigs()` w programie MATLAB (metoda Kryłowa-Schura).

Metody był uruchamiane na Maszynie 1. Jako typ zmiennoprzecinkowy metod używany był Float64 (double). Tolerancja zbieżności została ustalona na poziomie $\texttt{tol} = 1 \times 10^{-14}$. Poszukiwane było 36 wartości własnych w pobliżu wartości $\sigma = 1760$. W przypadku biblioteki Arpack.jl ustawiany był tryb LR (largest real) – poszukiwane były wartości własne o największej części rzeczywistej. Wartość przesunięcia σ została wybrana poprzez przekształcenie częstotliwości 2 GHz do wartości własnej za pomocą wzoru:

$$\mathrm{k} = \left(\frac{2\pi \mathrm{f}}{\mathrm{c}_0}\right)^2$$

i arbitralne wybranie bliskiej wartości otrzymanego wyniku. W ramach testu każda z metod była uruchamiana 10 razy. Statystyki czasu pracy każdej z metod zawiera Tabela 1, a 16 pierwszych wartości własnych zostało zamieszczone w Tabeli 2.

	Arpack.jl	ArnoldiMethod.jl	LinearAlgebra.jl	MATLAB
średni czas pracy (ms)	1980	3313	5481	29262
odchylenie standardowe (ms)	190	519	209	1722

Tabela 1. Średni czas pracy badanych metod

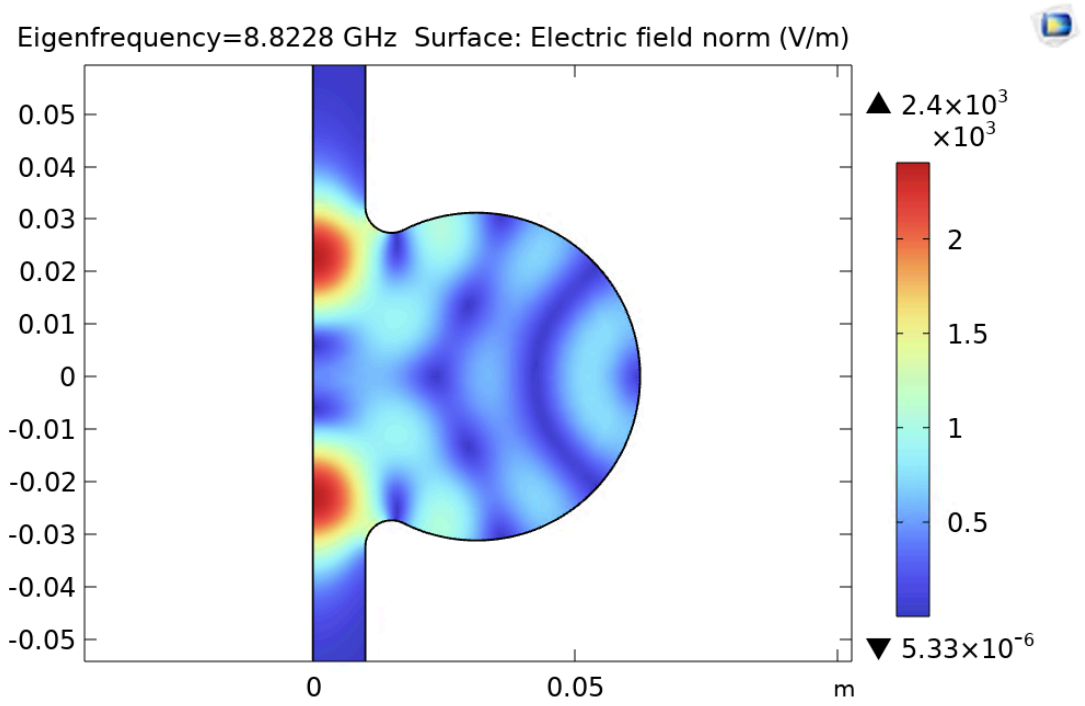
Arpack.jl	LinearAlgebra.jl	ArnoldiMethod.jl	MATLAB
1760.810	1760.810	3.419e-10	1.946e-09
5323.330	5323.330	3.419e-10	2.070e-09
8337.853	8337.853	4.329e-10	2.136e-09
13956.172	13956.172	1.632e-09	2.299e-09
14233.206	14233.206	1.632e-09	2.521e-09
19942.799	19942.799	2.975e-08	2.838e-09
24672.979	24672.979	1760.810	1760.809
25750.798	25750.798	5323.330	5323.330
30766.389	30766.389	8337.853	8337.853
34299.913	34299.913	13956.172	13956.172
37886.465	37886.465	14233.206	14233.206
41423.720	41423.720	19942.799	19942.799
46894.781	46894.781	24672.979	24672.979
48135.654	48135.654	25750.798	25750.798
48151.474	48151.474	30766.389	30766.389
48499.796	48499.796	34299.913	34299.913
50528.225	50528.225	37886.465	37886.465
52003.097	52003.097	41423.720	41423.720

Tabela 2. Część rzeczywista uzyskanych wartości własnych dla badanych metod

Wyniki uzyskane przez wszystkie z programów pokrywają się. W przypadku programów opartych o metodę Kryłowa-Schura (ArnoldiMethod.jl i MATLAB) możemy zaobserwować wystąpienie wartości własnych rzędu poniżej 10^{-7}. Są to wartości własne odpowiadające niefizycznym rodzajom pola. Najlepszy czas spośród testowanych metod ma metoda z biblioteki Arpack.jl – jest on ponad 10 razy szybszy niż obliczenia wykonywane w MATLAB. Jako przyczynę można wskazać wykorzystanie rutyn napisanych w języku Fortran, które pozwalają na przeprowadzenie obliczeń w optymalny sposób. Metoda Kryłowa-Schura z biblioteki ArnoldiMethod.jl również osiągnęła lepszy czas niż domyślna metoda z biblioteki LinearAlgebra.jl. Przeprowadzone badanie pokazało, że wybranie metody dopasowanej do problemu może pozwolić na przyspieszenie obliczeń przy zachowaniu takich samych wyników. Przeprowadzenie obliczeń dla rzadkich, symetrycznych macierzy z wykorzystaniem metod opartych o przestrzenie Kryłowa może być lepszym rezultatem niż korzystanie z wszechstronnej, lecz bardziej złożonej obliczeniowo metody QZ.

4.2 Porównanie rezultatów z oprogramowaniem COMSOL Multiphysics

Obok badań porównujących rezultat działania metod numerycznych w Julii z programem MATLAB, przeprowadzono także badania z wykorzystaniem dedykowanego oprogramowania do obliczeń fizycznych – COMSOL Multiphysics (Rysunek 4). Programy były uruchamiane na Maszynie 2, pozostałe parametry były takie same jak w pierwszym badaniu.



Rysunek 4. Wizualizacja pól własnych we wnęce rezonansowej w programie COMSOL Multiphysics.

Czas obliczeń dla 36 wartości własnych dla COMSOL wynosił średnio 6 sekund, z kolei dla 9 wartości własnych średnio 4 sekundy. Czas obliczeń dla 36 wartości własnych dla funkcji `eigen()` z biblioteki LinearAlgebra.jl wynosił średnio 3,084 sekundy z odchyleniem standardowym 0,318 sekundy, dla funkcji `eigs()` z biblioteki Arpack.jl wynosił średnio 2,037 sekundy z odchyleniem standardowym 1,078 sekundy. Czas obliczeń dla 9 wartości własnych dla funkcji `partialschur()` z biblioteki ArnoldiMethod.jl wynosił średnio 1,331 sekundy z odchyleniem standardowym 1,736 sekundy. Czas wykonywania metod w języku Julia był szybszy niż czas obliczeń w programie COMSOL Multiphysics, należy jednak pamiętać, że w przypadku programu COMSOL Multiphysics wpływ na czas potrzebny do ukończenia obliczeń mogą mieć pozostałe elementy programu takie jak np. GUI, które nie są obecne w przypadku programów w Julii.

Rezultaty obliczeń w COMSOL Multiphysics były podawane bezpośrednio w postaci częstotliwości. W Tabeli 3 zaprezentowano otrzymane częstotliwości i porównano z wartościami otrzymanymi

za pośrednictwem metod w Julii. Ze względu na pokrywanie się wyników w Julii podano wartości tylko z jednej z metod z biblioteki Arnoldi.jl.

Warto zwrócić uwagę, że wartości otrzymane przez program COMSOL Multiphysics jak i programy w Julii posiadają zarówno podobne (zaznaczone pogrubioną czcionką) jak i nowe częstotliwości, nie występujące w drugiej metodzie. Może to być skutek różnic pomiędzy sposobem modelowania problemu MES przez COMSOL, a modelem wykorzystanym w Julii.

f (GHz) COMSOL Multiphysics	wartości własne k w Julia	f (GHz) w Julia
2.000	1760,810	2,002
3.477	5323,330	3,481
3.986	8337,853	4,357
4.350	13956,172	5,637
5.628	14233,206	5,692
5.690	19942,799	6,738
5.997	24672,979	7,495
6.171	25750,798	7,657
6.728	30766,389	8,369
7.485	34299,913	8,837
7.652	37886,465	9,287
7.884	41423,720	9,711
8.125	46894,781	10,332
8.364	48135,654	10,468
8.534	48151,474	10,470
8.822	48499,796	10,508
9.278	50528,225	10,725
9.708	52003,097	10,881
9.778	52062,212	10,887
9.894	54723,577	11,162
10.330	56163,846	11,308
10.496	56186,032	11,310
10.659	59638,498	11,652
10.713	59708,451	11,659
10.726	61588,392	11,841
11.153	62444,325	11,923
11.494	64670,158	12,134
11.494	65817,459	12,241
11.608	67067,983	12,357
11.611	68538,286	12,491
11.645	70910,504	12,706
11.743	72623,681	12,858
11.791	73735,292	12,956
11.800	75639,347	13,122
11.901	78453,478	13,364
11.981	79599,768	13,462

Tabela 3. Częstotliwości rezonansowe otrzymane przez program COMSOL Multiphysics i programy w języku Julia

Rozdział 5

Podsumowanie

5.1 Wnioski z przeprowadzonych badań

Wyniki przeprowadzonych badań są pozytywne. Metody zaimplementowane w Julii pozwalały na uzyskanie porównywalnych wartości częstotliwości rezonansowych co narzędzia oferowane przez oprogramowanie komercyjne. Co więcej, czas działania metod w Julii był średnio szybszy niż w przypadku oprogramowania komercyjnego. Implementacja metod znajdowania pól własnych osiowosymetrycznych rezonatorów mikrofalowych w Julii oferuje przez to nie tylko możliwość modyfikacji kodu źródłowego, lecz ma także potencjał na szybsze przeprowadzenie obliczeń w porównaniu do oprogramowania komercyjnego.

5.2 Ograniczenia pracy i możliwości dalszych usprawnień

Interpretując rezultaty badań należy mieć na uwadze różnice w sposobie wykonywania obliczeń między implementacjami w języku Julia, a tymi w oprogramowaniu komercyjnym. Chociaż wyniki testów wskazują na przewagę wydajnościową kodu w języku Julia, należy zauważyć, że narzędzia te, ze względu na obecność pozostałych elementów programów takich jak graficzne interfejsy użytkownika (GUI) oraz wbudowane dodatkowe funkcjonalności, mogą działać wolniej w porównaniu z minimalnymi implementacjami w języku Julia. Elementy te mogą generować dodatkowe obciążenie systemu, co może wpływać na czas wykonywania obliczeń. Pełne porównanie efektywności powinno uwzględniać te różnice.

Istotnym ograniczeniem jest jakość obecnej implementacji metody Jacobi-Davidsona (JDQZ) w bibliotece JacobiDavidson.jl. W trakcie testów okazało się, że metoda ta nie zawsze zapewniała odpowiednią zbieżność, co znacząco wpływało na uzyskane wyniki. Możliwe przyczyny tego problemu mogą wynikać zarówno z ograniczeń samej implementacji, jak i jej parametrów domyślnych. W przyszłości warto rozważyć opracowanie lepszej wersji tej metody, dostosowanej do specyfiki analizowanych problemów. Ponadto, dalsze usprawnienia mogą obejmować optymalizację istniejących metod, na przykład poprzez wykorzystanie architektury CUDA do opracowania zrównoleglonych

wersji programów. Możliwe jest także rozszerzenie obecnych metod działających dla modeli osiowosymetrycznych o możliwość analizy modeli trójwymiarowych.

Bibliografia

[1] Anderson, E. i in., *LAPACK Users' Guide*, Third. Philadelphia, PA: Society for Industrial i Applied Mathematics, 1999, ISBN: 0-89871-447-8 (paperback).

[2] Arbenz, P., *Lecture Notes on Solving Large Scale Eigenvalue Problems*, Spring 2016.

[3] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. i Vorst, H. van der, *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000.

[4] Bezanson, J., Edelman, A., Karpinski, S. i Shah, V. B., „Julia: A fresh approach to numerical computing", *SIAM Review*, t. 59, nr. 1, s. 65–98, 2017. DOI: 10.1137/141000671. adr.: https://epubs.siam.org/doi/10.1137/141000671.

[5] Davidson, D. B., *Computational electromagnetics for RF and microwave engineering*, 2 wyd. Cambridge University Press, 2011, ISBN: 978-0–512-51891-8.

[6] Fokkema, D. R., Sleijpen, G. L. i Van der Vorst, H. A., „Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils", *SIAM journal on scientific computing*, t. 20, nr. 1, s. 94–125, 1998.

[7] Francis, J. G., „The QR transformation—part 2", *The Computer Journal*, t. 4, nr. 4, s. 332–345, 1962.

[8] Hernandez, V., Roman, J., Tomas, A. i Vidal, V., „Arnoldi methods in SLEPc", *SLEPc Technical Report STR-4*, 2007.

[9] Inc., T. M., *MATLAB Version: 24.1.0.2653294 (R2024a) Update 5, eig - Eigenvalues and eigenvectors*, Natick, Massachusetts, United States, 2024. adr.: https://www.mathworks.com/help/matlab/ref/eig.html.

[10] Inc., T. M., *MATLAB Version: 24.1.0.2653294 (R2024a) Update 5, eigs - Subset of eigenvalues and eigenvectors*, Natick, Massachusetts, United States, 2024. adr.: https://www.mathworks.com/help/matlab/ref/eigs.html.

[11] Jim Demmel Jack Dongarra, J. L. e. a. „LAPACK 3.6.0". (), adr.: https://www.netlib.org/lapack/lapack-3.6.0.html.

[12] Kressner, D., *Numerical Methods for General and Structured Eigenvalue Problems* (Lecture Notes in Computational Science and Engineering), eng, 1. Aufl. Berlin, Heidelberg: Springer--Verlag, 2005, t. 46, ISBN: 3540245464.

[13] Moler, C. B. i Stewart, G. W., „An algorithm for generalized matrix eigenvalue problems", *SIAM Journal on Numerical Analysis*, t. 10, nr. 2, s. 241–256, 1973.

[14] Rylander, T. i Par Ingelström, A. B., *Computational Electromagnetics* (Texts in Applied Mathematics). Springer New York, 2013, ISBN: 978-1-4614-5350-5.

[15] Sleijpen, G. L. i Van der Vorst, H. A., „A Jacobi–Davidson iteration method for linear eigenvalue problems", *SIAM review*, t. 42, nr. 2, s. 267–293, 2000.

[16] Sorensen, D. C., „Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations", w *Parallel Numerical Algorithms*, Springer, 1997, s. 119–165.

[17] Stewart, G. W., „A Krylov–Schur algorithm for large eigenproblems", *SIAM Journal on Matrix Analysis and Applications*, t. 23, nr. 3, s. 601–614, 2002.

[18] Wu, K., Canning, A., Simon, H. i Wang, L.-W., „Thick-Restart Lanczos Method for Electronic Structure Calculations", eng, *Journal of computational physics*, t. 154, nr. 1, s. 156–173, 1999, ISSN: 0021-9991.

Spis rysunków

Spis tabel