# Biblioteka RFem1D*

Zbigniew Romanowski†
*Institute of High Pressure Physics of the Polish Academy of Sciences*
*ul. Sokolowska 29/37, 01-142 Warsaw, Poland*
(Dated: 11 maja 2009)

Biblioteka `RFem1D` służy do: i) rozwiązywania jednowymiarowych eliptycznych równań różniczkowych drugiego rzędu, ii) znajdowania kilku najmniejszych wartości własnych i funkcji własnych jednowymiarowego eliptycznego zagadnienia własnego.

Biblioteka `RFem1D` została zaimplementowana w języku `C++`. Zastosowanym algorytmem rozwiązywania problemów jest metoda elementu skończonego według Galerkina. Jako funkcje bazowe zostały wybrane hierarchiczne funkcje Lobbato. Zastosowanie funkcji bazowych Lobbato wraz z metodą elementu skończonego sprowadza zagadnienie i) do układu równań z macierzą pasmowa a problem ii) do uogólnionego zagadnienia własnego, które rozwiązywane są za pomocą procedur z biblioteki `LAPACK`.

W pracy została opisana metoda od strony algorytmicznej oraz została podana funkcjonalność biblioteki od strony programistycznej. Ponadto, podane zostały przykłady rozwiązań numerycznych otrzymanych za pomocą biblioteki `RFem1D`. Przedyskutowano także dokładność otrzymanych wyników w funkcji liczby zastosowanych funkcji bazowych Lobatto.

Keywords: Równanie różniczkowe; Zagadnienie własne; Metoda elementu skończonego w jednym wymiarze; Funkcje bazowe Lobatto; C++.

**Contents**

**I. Wstęp** 2

**II. Definiecje problemów** 2
A. Równanie różniczkowe 2
B. Zagadnienie własne 2
C. Aproksymacja funkcji 3

**III. Opis klas biblioteki** 3
A. Klasa `Prob` 3
B. Klasa `OdeExProb` 3
C. Klasa `OdeProb` 4
D. Klasa `EigProb` 4
E. Klasa `Approx` 4

**IV. Wykorzystanie biblioteki** 5
A. Równanie różniczkowe 5
B. Zagadnienie własne 5
C. Aproksymacja funkcji 5

**V. Postać słaba** 5
A. Równanie różniczkowe 5
1. Zerowe warunki brzegowe Dirichleta 6
2. Niezerowe warunki brzegowe Dirichleta 6
B. Zagadnienie własne 6

**VI. Metoda Galerkina** 7
A. Tworzenie siatki 7
B. Tworzenie funkcji bazowych 7
1. Funkcje Lobatto $\{\psi_i\}$ 7
2. Pomocnicze funkcje bazowe $\{\phi_i^n\}$ 8
3. Funkcje bazowe $\{\phi_k\}$ 9
C. Dyskretyzacja 9
1. Dyskretyzacja równania różniczkowego 9
2. Dyskretyzacja zagadnienia własnego 10
3. Numerowanie funkcji bazowych 10
D. Analityczne obliczanie potrzebnych całek 11
1. Całki po funkcjach Lobatto 11
E. Kwadratury Gaussa 12
F. Algorytm adaptacyjny 12
1. Zagadnienie własne 12
2. Równanie różniczkowe 12

**VII. Aproksymacja wielomianami Lobatto** 13
A. Aproksymacja na jednym przedziale 13
B. Aproksymacja adaptacyjna 14

**VIII. Przykłady rozwiązań** 14
A. Równania różniczkowe 14
1. Przykład 1 14
2. Przykład 2 15
3. Przykład 3 15
4. Przykład 4 15
5. Przykład 5 16
B. Równania Schrödingera w jednym wymiarze 17
C. Radialne równania Schrödingera 18
1. Atomu wodoru 18
2. Potencjał Woodsa-Saxona, adaptacyjnie 18
3. Potencjał Morsea, adaptacyjnie 19
4. Potencjał Hellmanna, adaptacyjnie 20
D. Równanie Kohna-Shama dla atomu 20
1. Implementacja 21
2. Wyniki numeryczne 21

**Literatura** 22

*Wersja 1.3
†Electronic address: `romz@wp.pl`

## I. WSTĘP

W bibliotece `RFem1D` zostały zaimplementowane następujące algorytmy:

- Rozwiązywanie równań różniczkowych drugiego rzędu.
- Znajdowanie kilku najmniejszych wartości własnych zagadnienia własnego drugiego rzędu.
- Aproksymacja funkcji.

Rozwiązywane problemy są zdefiniowane w jednym wymiarze na skończonym przedziale. Prezentowane algorytmy są mumeryczne, adaptacyjne i są oparte na metodzie elementu skończonego [5–7, 18, 32–34] w sformułowaniu Galerkina [12]. W bibliotece `RFem1D` funkcjami bazowymi są funkcje Lobatto [32], które umożliwiają w prosty sposób budowanie baz wysokiego rzędu (ang. *high order basis*). Algorytmy zaimplementowane w `RFem1D` umożliwiają otrzymanie rozwiązania dla problemów zdefiniowanych na skończonym przedziale z nałożonymi warunkami brzegowymi Dirichleta.

Dokument ten zawiera następujące informacje. W sekcji II podane są definicje rozwiązywanych problemów w postaci różniczkowej. Podana została postać różniczkowa i postać słaba dla analizowanych problemów. Wyspecyfikowane zostały założenia dotyczące rozwiązywanych równań i funkcji w nich występujących. W sekcji III opisane są klasy zaimplementowane w tej bibliotece. W sekcji tej podane zostały deklaracje funkcji wraz z opisem ich parametrów. Opisano też przykładowe ich wykorzystanie. W sekcji V dla równań różniczkowych podana została postać słaba, która wykorzystywana jest do dyskretyzacji problemu. W sekcji VI podane są równania na elementy macierzowe, które uzyskuje się stosując metodę Galerkina. W sekcji tej dyskutowane jest również uwzględnienie warunków brzegowych. w ostatniej sekcji VIII podane są przykłady obrazujące dokładność opisanych algorytmów.

## II. DEFINIECJE PROBLEMÓW

W kolejnych podsekcjach zdefiniowane są problemy, które można rozwiązać numerycznie za pomocą biblioteki `RFem1D`.

### A. Równanie różniczkowe

**Problem 1** *Niech $\Omega = [a,b] \subset \mathbb{R}$ będzie skończonym przedziałem. Niech $f, g, h : \Omega \mapsto \mathbb{R}$ będą funkcjami ciągłymi spełniającymi warunki*

$$h(x) \geq \gamma > 0, \qquad g(x) \geq 0, \quad dla \quad x \in \Omega,$$

*gdzie $\gamma \in \mathbb{R}$. Znajdź funkcję $u : \Omega \mapsto \mathbb{R}$ spełniającą równanie różniczkowe drugiego rzędu*

$$-\big(h(x)u'(x)\big)' + g(x)u(x) = f(x) \tag{1}$$

*oraz warunki brzegowe Dirichleta*

$$u(a) = u_a, \qquad u(b) = u_b,$$

*gdzie $u_a, u_b \in \mathbb{R}$ są zadanymi liczbami.*

Szczególnym przypadkiem powyższego problemu jest sytuacja gdy funkcja $h(x) = \gamma > 0$ jest funkcją stałą. Wtedy równanie (1) ma szczególnie prostą postać

$$-\gamma u''(x) + g(x)u(x) = f(x). \tag{2}$$

Formalnie, równanie (2) jest szczególnym przypadkiem równania (1). Równania te są oddzielnie rozpatrywane ze względu na istnie bardziej efektywnego numerycznego algorytmu dla równania (2).

W ogólnej postaci równanie (2) może mieć wiele różnych warunków brzegowych. Jednakże, w bibliotece `RFem1D` zaimplementowane zostały tylko warunki brzegowe Dirichleta.

### B. Zagadnienie własne

**Problem 2** *Niech $\Omega = [a,b] \subset \mathbb{R}$ będzie skończonym przedziałem. Niech $\gamma > 0$ oraz niech $g : \Omega \mapsto \mathbb{R}$ będzie funkcją ciągłą taką, że $g(x) \geq 0$ dla $x \in \Omega$. Znajdź kilka najmniejszych wartości własnych $\lambda_n$ oraz odpowiadających im funkcji własnych $u_n : \Omega \mapsto \mathbb{R}$ spełniających zagadnienie własne drugiego rzędu*

$$-\gamma u_n''(x) + g(x)u_n(x) = \lambda_n u_n(x). \tag{3}$$

*Ponadto, funkcje własne muszą spełniać zerowe warunki brzegowe Dirichleta*

$$u_n(a) = u_n(b) = 0.$$

Powyższy problem można też sformułować dla funkcji $g$, która nie jest dodatnia, ale jest ograniczona z dołu, to znaczy

$$\exists_{C<0} \forall_{x\in\Omega} \quad g(x) > C.$$

W tym celu należy odjąć od obu stron równania (3) funkcję $Cu_n$. Wtedy otrzymujemy

$$-\gamma u_n''(x) + \tilde{g}(x)u_n(x) = \tilde{\lambda}_n u_n(x),$$

gdzie

$$\tilde{g}(x) = g(x) - C \quad \text{oraz} \quad \tilde{\lambda}_n = \lambda_n - C.$$

Ponieważ $\tilde{g}(x) \geq 0$ dla $x \in \Omega$, to dla funkcji $\tilde{g}$ można zastosować prezentowany algorytm. Jedyna trudność polega na podaniu rozsądnego ograniczenia dolnego $C$.

### C. Aproksymacja funkcji

**Problem 3** *Niech $\Omega = [a, b] \subset \mathbb{R}$ będzie skończonym odcinkiem. Niech $f : \Omega \mapsto \mathbb{R}$ będzie funkcją ciągłą. Znajdź wielomianową aproksymację $g$ funkcji $f$ tak, aby wartości funkcji aproksymowanej i funkcji aproksymującej na brzegach przedziału były sobie równe*

$$g(a) = f(a) \quad \text{oraz} \quad g(b) = f(b) \tag{4}$$

*Dodatkowo wielomian aproksymujący ma być stopnia $M$. Podaj błąd aproksymacji.*

Warunek równości funkcji aproksymowanej $f$ i funkcji aproksymującej $g$ jest bardzo pożądany. Na przykład, gdy podzieli się przedział $[a, b]$ na dwa przedziały $[a, b] = [a, x_0] \cup [x_0, b]$ oraz zostanie znaleziona aproksymacja $g_1$ na przedziale $[a, x_0]$ i aproksymacja $g_2$ na przedziale $[x_0, b]$, to funkcja

$$\tilde{g}(x) = \begin{cases} g_1(x) & \text{dla } x \in [a, x_0] \\ g_2(x) & \text{dla } x \in [x_0, b] \end{cases} \tag{5}$$

jest ciągła na przedziale $[a, b]$.

Do aproksymacji wielomianowej zastosowane zostały wielomiany Lobatto. Warunki na współczynniki rozwinięcia są uzyskiwane z otrogonalności residuum do funkcji bazowych. Błąd rozwiązania to całka z kwadratu różnicy funkcji aproksymowanej i aproksymującej.

Do rozwiązania tego problemu zastosowano algorytm $r$-adaptacyjny. Wtedy rozwiązaniem tego problemu jest ciąg liczb $a = r_0 < r_1 < \ldots < r_N = b$ taki, że na każdym przedziale $[r_i, r_{i+1}]$ funkcja aproksymująca jest wielomianem stopnia $M$ z błędem aproksymacji mniejszym niż $\Delta$.

## III. OPIS KLAS BIBLIOTEKI

Biblioteka `RFem1D` jest zaimplementowana w języku `C++`. Najważniejszymi składnikami tej biblioteki są klasy `OdeProb`, `OdeExProb`, `EigProb`, `Approx`:

- W klasie `OdeExProb` zaimplementowany jest algorytm rozwiązywania Problemu 1 z dowolnymi warunkami brzegowymi Dirichleta i z dowolna funkcją $h$. Wszystkie elementy macierzy obliczane są za pomocą kwadratur Gaussa.
- W klasie `OdeProb` zaimplementowany jest algorytm rozwiązywania Problemu 1 ze stała wartością funkcji $h$ i z zerowymi warunkami brzegowymi Dirichleta. Elementy macierzy powstałej z dyskretyzacji operatora różniczkowego obliczane są analitycznie.
- W klasie `EigProb` zaimplementowany jest algorytm rozwiązywania Problemu 2. Elementy macierzy powstałej z dyskretyzacji operatora różniczkowego i macierzy przekrycia obliczane są analitycznie.
- W klasie `Approx` zaimplementowany jest algorytm rozwiązywania Problemu 3.

Podział na dwie klasy `OdeExProb`, `OdeProb` wynika z możliwości uniknięcia całkowania numerycznego, a tym samym zmniejszenia czasu obliczeń i zwiększenia dokładności. Klasy `OdeProb`, `OdeExProb`, `EigProb` dziedziczą publicznie po klasie `Prob`.

Wykorzystanie klas `OdeProb`, `OdeExProb`, `EigProb` jest bardzo podobne. Aby z nich skorzystać należy wykonać następujące czynności:

1. Utworzy obiekt klasy.
2. Utworzyć początkową siatkę.
3. Rozwiązać problem wywołując funkcję `Solve()`.
4. Wykorzystać otrzymany wynik.

Pełny opis interfejsu klas, wraz z opisem ich parametrów, podany jest w kolejnych podsekcjach. Podany są także przykłady wykorzystania tych klas.

### A. Klasa `Prob`

Klasy `OdeProb`, `OdeExProb`, `EigProb` dziedziczą publicznie po klasie `Prob`. Klasa `Prob` ma interfejs składający się z następujących funkcji.

- `void Prob(void)`. Konstruktor domyślny.
- `void GenMeshLin(double a, double b, size_t nodeNo, size_t degree)`. Funkcja ta generuje siatkę składającą się z `nodeNo` węzłów z odcinkami o równej długości. Każdy odcinek jest stopnia `degree`.
- `void SetMesh(const std::vector<double>& x, const std::vector<size_t>& degree)`. Definiuje siatkę usuwając poprzednią. Węzły siatki znajdują się w wektorze `x`. Stopień wielomianów na elementach znajduje się w wektorze `degree`. Rozmiar wektora `degree` musi być o jeden większy od rozmiaru wektora `x`.
- `void AddToMesh(const std::vector<size_t>& eltToSplit)`. Dzieli elementy siatki o indeksach podanych w tablicy `eltToSplit` na połowy. Nowo utworzone elementy dodaje do siatki. Funkcja wykorzystywana w procedurze adaptacyjnej.

### B. Klasa `OdeExProb`

W klasie `OdeExProb` zaimplementowany jest algorytm rozwiązywania Problemu 1 z dowolnymi warunkami brzegowymi Dirichleta i z dowolna funkcją $h$. Elementy macierzy obliczane są za pomocą kwadratur Gaussa. Klasa `OdeExProb` ma interfejs składający się z następujących funkcji.

- `OdeExProb(void)`. Konstruktor domyślny.

- `void Define(double ua, double ub, const Fun1D* h, const Fun1D* g, const Fun1D* f)`. Definiuje Problem 1. Parametry `ua`, `ub` określają wartość warunku brzegowego Dirichleta w lewym i prawym końcu przedziału, odpowiednio. Parametry `h`, `g`, `f` są funkcjami z równania (1).
- `void Solve(void)`. Rozwiązuje równanie (1). Po zakończeniu działania tej funkcji otrzymane rozwiązanie jest gotowe do wykorzystania.
- `void SolveAdapt(double absMaxCoef)`. Rozwiązuje równania (1) adaptacyjnie. Po zakończeniu działania tej funkcji otrzymane rozwiązanie jest gotowe do wykorzystania. Siatka jest tak długo dzielona, aż najmniejszy z największych współczynników rozwinięcia na każdym elemencie siatki będzie mniejszy niż argument `absMaxCoef`.
- `double GetSol(double x) const`. Zwraca wartość obliczonego rozwiązania `u(x)`.
- `void WriteSol(const char* path, size_t pointNo) const`. Zapisuje otrzymane rozwiązanie do pliku `path`. W pliku zapisanych jest `pointNo` wartości z punktów leżących w równych odległościach.

### C. Klasa `OdeProb`

W klasie `OdeProb` zaimplementowany jest algorytm rozwiązywania Problemu 1 ze stała wartością funkcji $h = \gamma$ i z zerowymi warunkami brzegowymi Dirichleta. Niektóre elementy macierzy obliczane są analitycznie. Klasa ta jest szczególnie wydajna, gdy funkcja $g$ jest równa 0, co podaje się jako wskaźnik `NULL` w definicji problemu. Klasa `OdeProb` ma interfejs składający się z następujących funkcji.

- `OdeProb(void)`. Konstruktor domyślny.
- `void Define(double gamma, const Fun1D* g, const Fun1D* f)`. Definiuje Problem 1. Parametry `gamma`, `g`, `f` są parametrami z równania (2), gdzie funkcja $h$ jest funkcją stała i jej wartość wynosi `gamma`. Przyjmowane są zerowe warunki brzegowe Dirichleta.
- `void Solve(void)`. Rozwiązuje równanie (2) z parametrami zdefiniowanymi przez funkcję `Define`. Po zakończeniu działania tej funkcji otrzymane rozwiązanie jest gotowe do wykorzystania.
- `void SolveAdapt(double absMaxCoef)`. Rozwiązuje równania (2) adaptacyjnie z parametrami zdefiniowanymi przez funkcję `Define`. Po zakończeniu działania tej funkcji otrzymane rozwiązanie jest gotowe do wykorzystania. Siatka jest tak długo dzielona, aż najmniejszy z największych współczynników rozwinięcia na każdym elemencie siatki będzie mniejszy niż argument `absMaxCoef`.
- `double GetSol(double x) const`. Zwraca wartość obliczonego rozwiązania `u(x)`.
- `void WriteSol(const char* path, size_t pointNo) const`. Zapisuje otrzymane rozwiązanie do pliku `path`. W pliku zapisanych jest `pointNo` wartości z punktów leżących w równych odległościach.

### D. Klasa `EigProb`

W klasie `EigProb` zaimplementowany jest algorytm rozwiązywania Problemu 2. Elementy macierzy powstałej z dyskretyzacji operatora różniczkowania i macierz przekrycia obliczane są analitycznie. Gdy funkcja $g$ jest równa 0, to można ją zadeklarować jako wskaźnik `NULL`, co istotnie skraca czas obliczeń, gdyż nie jest wykonywane całkowanie numeryczne. Klasa `EigProb` ma interfejs składający się z następujących funkcji.

- `EigProb(void)`. Konstruktor domyślny.
- `void Define(double gamma, const Fun1D* g)`. Definiuje Problem 2. Argumenty `gamma`, `g` są parametrami z równania (3).
- `void Solve(size_t eigNo, double abstol)`. Rozwiązuje równanie (3). Znajduje `eigNo` najmniejszych wartości własnych i odpowiadających im funkcji własnych z dokładnością `abstol`. Po zakończeniu działania tej funkcji otrzymane wartości i funkcje własne są gotowe do wykorzystania.
- `void SolveAdapt(size_t eigNo, double abstol, double absMaxCoef)`. Rozwiązuje równanie (3) adaptacyjnie. Argumenty `eigNo`, `abstol` mają takie same znaczenie jak w funkcji `Solve`. Siatka, na której szukane jest rozwiązanie, jest zagęszczana tam gdzie współczynniki rozwinięcia są największe. Siatka jest tak długo dzielona, aż najmniejszy z największych współczynników rozwinięcia na każdym elemencie siatki będzie mniejszy niż argument `absMaxCoef`.
- `double GetEigVal(size_t eig) const`. Zwraca wartość własną `eig`.
- `double GetEigFun(size_t eig, double x) const`. Zwraca wartość funkcji własnej `eig` w punkcie `x`.
- `void WriteSol(const char* path, size_t pointNo) const`. Zapisuje otrzymane funkcje własne do pliku `path`. W pliku zapisanych jest `pointNo` wartości z punktów leżących w równych odległościach.

### E. Klasa `Approx`

W klasie `Approx` został zaimplementowany algorytm rozwiązywania Problemu 3. Klasa ta ma interfejs składający się z następujących funkcji.

- `Approx(size_t M, const Fun1D* f)`. Konstruktor. Argument `M` oznacza stopień aproksymacji. Natomiast argument `f` jest wskaźnikiem do interpolowanej funkcji.

- `double Solve(double a, double b)`. Rozwiązuje Problem 3. Argumenty `a` oraz `b`, wyznaczają przedział $[a, b]$. Funkcja ta zwraca błąd aproksymacji $\Delta$.
- `double FindB(double a, double delta, double h)`. Znajduje współrzędne węzła $r_0$ tak, że aproksymacja wielomianowa na przedziale $[a, r_0]$ zwraca błąd w przybliżeniu `delta`. Argument `h` oznacza długość pierwszego kroku.
- `void SolveAdapt(double a, double b, double delta, double h, std::list<double> & r)`. Rozwiązuje Problem 3 adaptacyjnie. Argumenty `a` oraz `b`, wyznaczają przedział $[a, b]$. Funkcja ta zwraca listę znalezionych współrzędnych ciągu $\{r_i\}$ takich, że na każdym odcinku $r_i, r_{i+1}$ aproksymacja ma błąd mniejszy niż `delta`. Argument `h` ma takie same znaczenie jak w funkcji `FindB`.
- `double GetSol(double x) const`. Zwraca wartość obliczonej funkcji aproksymującej dla argumentu $x \in [a, b]$.

## IV. WYKORZYSTANIE BIBLIOTEKI

W tej sekcji podane są typowe przykłady wykorzystania biblioteki `RFem1D`.

### A. Równanie różniczkowe

Poniżej podany jest fragment kodu, będącego przykładem typowego zastosowania klasy `OdeProb`.

```
FunF f;
FunG g;
FunH h;
double ua = 2, ub = 3;
OdeExProb ode;
ode.Define(ua, ub, &h, &g, &f);
ode.GenMeshLin(-1, 1, 5, 3);
ode.Solve();
ode.WriteSol("sol.dat", 200);
```

Powyższy program poszukuje przybliżonego rozwiązania Problemu 1. Równanie zdefiniowane jest na przedziale $[-1, 1]$, z warunkami brzegowymi Dirchleta `ua` oraz `ub`. Obiekty `f`, `g`, `h` klas `FunF`, `FunG`, `FunH` reprezentują funkcje $f, g, h$, odpowiednio. Siatka zawiera 5 węzłów, tworząc 4 elementy każdy o stopniu 3. Otrzymane rozwiązanie zapisywane jest do pliku `sol.dat` w 200 równo-odległych punktach.

### B. Zagadnienie własne

Poniżej podany jest fragment kodu, będącego przykładem typowego zastosowania klasy `EigProb`.

```
FunG g;
EigProb eig;
eig.Define(1, &g);
eig.GenMeshLin(-1, 1, 5, 3);
eig.Solve(4, 1e-6);
eig.WriteEigFun("sol.dat", 200);
```

Powyższy program poszukuje przybliżonego rozwiązania Problemu 2. Równanie zdefiniowane jest na przedziale $[-1, 1]$, gdzie `gamma` równa się 1, a funkcja $g$ reprezentowana jest przez klasę `FunG`. Siatka zawiera pięć węzłów, tworząc cztery elementy każdy o stopniu trzy. Na obu końcach przedziału niejawnie nałożone są zerowe warunki brzegowe Dirichleta. Otrzymane rozwiązanie (czyli cztery wartości własne i cztery wektory własne) zapisywane jest do pliku `sol.dat` w dwustu punktach.

### C. Aproksymacja funkcji

Poniżej podany jest fragment kodu, będącego przykładem typowego zastosowania klasy `Approx`.

```
FunF f;
Approx app(6, &f);
double delta = app.Solve(-1, 2);
double v = app.Get(0.1);
```

Powyższy program znajduje aproksymację rzędu $M = 6$ funkcji `f` na przedziale $[a, b] = [-1, 2]$ dla funkcji $f$. Obliczony błąd aproksymacji $\Delta$ zapisany jest w zmiennej `delta`. W zmiennej `v` przechowywana jest wartość funkcji aproksymującej dla argumentu 0.1.

## V. POSTAĆ SŁABA

Do rozwiązywania równania różniczkowego (Problem 1) i zagadnienia własnego (Problem 2) zastosowana została metoda elementu skończonego według Galerkina. Aby móc zastosować metodę elementu skończonego, należy sprowadzić postać różniczkową równania (1) i równania (3) do postaci całkowej nazywanej postacią słabą (ang. *weak form*) lub wariacyjną. Dowodzi się, że obie postacie są równoważne, jednakże w zastosowaniach numerycznych postać słaba jest wygodniejsza w użyciu. Ponadto, zaletą słabej postaci jest osłabienie warunków na funkcje $f, g, h, u$.

### A. Równanie różniczkowe

Równanie różniczkowe z Problemu 1 może mieć zerowe lub niezerowe warunki brzegowe Dirichleta. Ponieważ sformułowanie słabe dla zerowych warunków brzegowych Dirichleta jest istotnie prostsze od postaci słabej z niezerowymi warunkami brzegowymi Dirichleta, to zostały one podane w oddzielnych sekcjach.

### 1. Zerowe warunki brzegowe Dirichleta

Załóżmy, że rozwiązywany jest problem 1 z zerowymi warunkami brzegowymi Dirichleta

$$u(a) = 0, \qquad u(b) = 0.$$

Sformułowanie słabe tego problemu otrzymuje się w następujący sposób:

1. Pomnóż równanie (1) przez funkcję $v \in H_0^1(\Omega)$.
2. Scałkuj obustronnie po $\Omega$.
3. Zastosuj twierdzenie Greena-Gaussa aby zmniejszyć stopień różniczkowania.

Stosując powyższą procedurę, pokazuje się, że postać słaba równania (1) jest następująca.

**Problem 4** *Znajdź funkcję $u \in H_0^1(\Omega)$, która dla dowolnej funkcji $v \in H_0^1(\Omega)$ spełnia równanie całkowe*

$$\begin{aligned}\int_\Omega h(x)u'(x)v'(x)\mathrm{d}x + \int_\Omega g(x)u(x)v(x)\mathrm{d}x \\ = \int_\Omega f(x)v(x)\mathrm{d}x.\end{aligned} \tag{6}$$

Ponadto pokazuje się, że wystarczy aby $f \in L^2(\Omega)$ oraz $h, g \in L^\infty(\Omega)$.

### 2. Niezerowe warunki brzegowe Dirichleta

Aby uzyskać postać słabą dla równań (1) z niezerowymi warunkami brzegowymi Dirichleta, poszukiwana funkcja $u$ przedstawiona jest w postaci sumy dwóch funkcji

$$u(x) = \hat{u}(x) + \bar{u}(x). \tag{7}$$

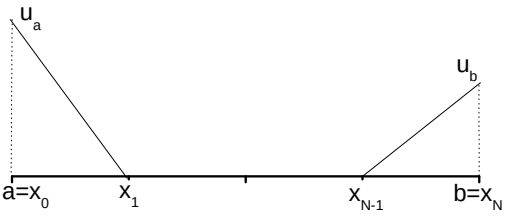
Funkcja $\hat{u}$ jest tak dobrana, że spełnia warunki brzegowe Dirichleta, to znaczy zachodzą warunki

$$\hat{u}(a) = u_a, \qquad \hat{u}(b) = u_b.$$

Z powyższego wynika, że funkcja $\bar{u}$ przyjmuje wartość zero na brzegach przedziału $\Omega = [a, b]$, to znaczy

$$\bar{u}(a) = \bar{u}(b) = 0.$$

Funkcję $\hat{u}(x)$ nazywana jest *Dirichlet lift* i można ją skonstruować na wiele sposobów. Jednakże zakłada się, że ma ona być kawałkami wielomianem, tak aby funkcja $u(x)$ też była kawałkami wielomianem. Przykładem takiej funkcji, która jest poprawna dla szerokiej klasy równań różniczkowych (i łatwo ją uogólnić dla problemów zdefiniowanych w $\mathbb{R}^2$ i $\mathbb{R}^3$) jest funkcja kawałkami liniowa, tak jak pokazano na Rys. 1.




Rysunek 1: Przykład funkcji $\hat{u}(x)$ (*Dirichlet lift*) dla siatki $a = x_0 < x_1 < \ldots < x_{N-1} < x_N = b$.

Wstawiając sumę (7) do równania (1) i grupując wyrazy otrzymujemy równanie na $\bar{u}$ analogiczne do (1) tylko ze zmienioną prawą stroną

$$-\big(h(x)\bar{u}'(x)\big)' + g(x)\bar{u}(x) = \tilde{f}(x), \tag{8}$$

gdzie funkcja $\tilde{f}$ ma postać

$$\tilde{f}(x) = f(x) + \big(h(x)\hat{u}'(x)\big)' - g(x)\hat{u}(x). \tag{9}$$

Stosując procedurę z poprzedniej sekcji, pokazuje się, że postać słaba równania (8) jest następująca.

**Problem 5** *Znajdź funkcję $\bar{u} \in H_0^1(\Omega)$, która dla dowolnej funkcji $v \in H_0^1(\Omega)$ spełnia równanie całkowe*

$$\begin{aligned}\int_\Omega h(x)\bar{u}'(x)v'(x)\mathrm{d}x + \int_\Omega g(x)\bar{u}(x)v(x)\mathrm{d}x \\ = \int_\Omega \big[f(x)v(x) - h(x)\hat{u}'(x)v'(x) - g(x)\hat{u}(x)v(x)\big]\mathrm{d}x.\end{aligned} \tag{10}$$

Ponadto pokazuje się, wystarczy aby $f \in L^2(\Omega)$ oraz $h, g \in L^\infty(\Omega)$ oraz $\hat{u} \in H_0^1(\Omega)$. Powyższe równanie jest postacią słabą Problemu 1 z niezerowymi warunkami brzegowymi Dirichleta. Poszukiwane rozwiązanie przedstawione jest w postaci sumy (7).

Zastosowanie metody elementu skończonego z bazą Lobatto do tego problemu sprowadza równania (6), (10) do liniowego układu równań z macierzą symetryczną i dodatnio określoną.

Aby uzyskać numeryczne rozwiązanie Problemu 4 ze stałą funkcją $h$ i zerowymi warunkami brzegowymi Dirichleta, to należy wykorzystać klasę `OdeProb` z biblioteki `RFem1D`. Gdy poszukiwane jest numeryczne rozwiązanie Problemu 5 dla niestałej funkcji $h$ i z niezerowymi warunkami brzegowymi, to należy wykorzystać klasę `OdeExProb` z biblioteki `RFem1D`.

## B. Zagadnienie własne

Pokazuje się, że postać słaba równania (3) jest następująca.

**Problem 6** *Znajdź funkcje $u_n \in H_0^1(\Omega)$ i odpowiadające im wartości własne $\lambda_n \in \mathbb{R}$, które dla dowolnej funkcji $v \in H_0^1(\Omega)$ spełniają równanie całkowe*

$$\gamma \int_\Omega u_n'(x)v'(x)\mathrm{d}x + \int_\Omega g(x)u_n(x)v(x)\mathrm{d}x = \lambda_n \int_\Omega u_n(x)v(x)\mathrm{d}x. \tag{11}$$

*Ponadto pokazuje się, wystarczy aby $f \in L^2(\Omega)$ oraz $h, g \in L^\infty(\Omega)$.*

Zastosowanie metody elementu skończonego z bazą Lobatto do tego problemu sprowadza równanie (11) do algebraicznego uogólnionego zagadnienia własnego z macierzami pasmowymi, symetrycznymi i dodatnio określonymi. Aby dla tego problemu uzyskać numerycznie kilka najmniejszych wartości własnych i odpowiadających im funkcji własnych, należy wykorzystać klasę `EigProb` z biblioteki `RFem1D`.

## VI. METODA GALERKINA

W tej sekcji została podana dyskretyzacja Problemów 4, 5, 6. Dyskretyzacja ta została wykonana na podstawie metody elementu skończonego sformułowanego według Galerkina. Pomimo, że Problemy 4, 5, 6 są odmienne, to ich dyskretyzacje są podobne, gdyż we wszystkich przypadkach poszukuje się rozwiązania w skończenie wymiarowej kawałkami wielomianowej przestrzeni $\mathcal{S}$. Przestrzeń $\mathcal{S}$ rozpięta jest na bazie funkcji $\{\phi_i\}_{i=1}^M$, to znaczy $\mathcal{S} = \mathrm{span}\{\phi_1, \ldots, \phi_M\}$, gdzie $M$ jest wymiarem przestrzeni $\mathcal{S}$. Funkcje $\{\phi_i\}_{i=1}^M$ są liniowo niezależne, ale nie muszą być ortogonalne.

W metodzie elementu skończonego zostało wprowadzonych wiele funkcji bazowych, które różnią się od siebie własnościami numerycznymi i łatwością ich konstruowania. Jednakże zwykle funkcje bazowe $\{\phi_i\}_{i=1}^M$ tworzone są na podstawie referencyjnych funkcji bazowych $\{\psi_i\}$.

W bibliotece `RFem1D` referencyjnymi funkcjami bazowymi $\{\psi_i\}$ są funkcji Lobatto [32], które umożliwiają (w prosty sposób) budować bazy wysokiego rzędu (ang. *high order basis*). Własności funkcji Lobatto opisane są szczegółowo w sekcji VI B 1. Natomiast sposób tworzenia funkcji bazowych $\{\phi_i\}$ na podstawie referencyjnych funkcji bazowych $\{\psi_i\}$ podany jest w Sekcji VI B.

Podstawą zastosowania metody elementu skończonego według Galerkina jest postać słaba problemu. Postać słaba dla równania różniczkowego i zagadnienia własnego podane są w Sekcjach II A, II B.

### A. Tworzenie siatki

W metodzie elementu skończonego, proces dzielenia dziedziny $\Omega$ na elementy jest nazywany tworzeniem siatki $\mathcal{T}_{1\mathrm{D}}$. Ponieważ dziedziną w obu problemach jest odcinek $\Omega = [a, b]$, więc utworzenie siatki $\mathcal{T}_{1\mathrm{D}}$ sprowadza się do wygenerowania ciągu $a = x_0 < x_2 < \ldots < x_N = b$. Niech $\Omega_n = [x_{n-1}, x_n]$ będzie $n$-tym odcinkiem (elementem), wtedy zbiór $\mathcal{T}_{1\mathrm{D}} = \{\Omega_n\}_{n=1}^N$ jest szukaną siatką, gdzie $N$ jest liczbą elementów w siatce. Z konstrukcji wynika, że zachodzi $\Omega = \bigcup_{n=1}^N \Omega_n$.

Dodatkowo, każdemu odcinkowi $\Omega_n$ przyporządkowana jest liczba naturalna $p_n > 0$. Liczba $p_n$ określa maksymalny stopień referencyjnej funkcji bazowej na odcinku $\Omega_n$.

### B. Tworzenie funkcji bazowych

Funkcje bazowe $\{\phi_i\}$ tworzy się w trzech krokach:

1. Wybierz referencyjne funkcje bazowe $\{\psi_i\}$. W bibliotece `RFem1D` zaimplementowane są funkcje Lobatto.
2. Utwórz funkcje pomocnicze $\phi_i^n$. Funkcje te są tworzone na podstawie referencyjnych funkcji bazowych $\psi_i$. Tworzenie tych funkcji jest niezależne dla każdego elementu $\Omega_n$.
3. Utwórz funkcje bazowe $\phi_i$. Funkcje te są tworzone na podstawie pomocniczych funkcji $\phi_i^n$.

W kolejnych sekcjach zostaną podane szczegóły istotne dla każdego kroku.

#### 1. *Funkcje Lobatto $\{\psi_i\}$*

W bibliotece `RFem1D` referencyjnymi funkcjami bazowymi, $\psi_i$, są hierarchiczne funkcje Lobatto [32]. Funkcje te (w przeciwieństwie do klasycznych funkcji bazowych Lagrange'a lub Hermite'a) charakteryzują się łatwością tworzenia baz wysokiego rzędu. Oznaczmy przez $\mathcal{B}_p = \{\psi_1, \psi_2, \ldots, \psi_p\}$, $p$ elementowy zbiór hierarchicznych funkcji bazowych Lobatto na odinku $[-1, 1]$. Wtedy zbiór $p+1$ elementowy $\mathcal{B}_{p+1}$ tworzy się następująco:

$$\mathcal{B}_{p+1} = \mathcal{B}_p \cup \{\psi_{p+1}\} \tag{12}$$

Oznacza to, że utworzenie bazy $p+1$ stopnia wymaga tylko utworzenia i dodania jednej funkcji $\psi_{p+1}$.

Definicja funkcji Lobatto jest następująca:

$$\psi_0(s) = (1-s)/2 \tag{13a}$$
$$\psi_1(s) = (1+s)/2 \tag{13b}$$
$$\psi_k(s) = \int_{-1}^{s} \tilde{P}_{k-1}(t)\mathrm{d}t \qquad \text{dla } k \geq 2 \tag{13c}$$

gdzie $\tilde{P}_k(t)$ jest znormalizowanym wielomianem Legendre'a:

$$\tilde{P}_k(t) = \frac{P_k(t)}{||P_k(t)||} \tag{14}$$

Tablica I: Jawne wzory funkcji Lobatto, $\psi_k(s)$, dla $k \leq 10$.

| $k$ | $\psi_k(s)$ |
|---|---|
| 0 | $(1-s)/2$ |
| 1 | $(1+s)/2$ |
| 2 | $\frac{1}{2}\sqrt{\frac{3}{2}}(s^2-1)$ |
| 3 | $\frac{1}{2}\sqrt{\frac{5}{2}}(s^2-1)s$ |
| 4 | $\frac{1}{8}\sqrt{\frac{7}{2}}(s^2-1)(5s^2-1)$ |
| 5 | $\frac{1}{8}\sqrt{\frac{9}{2}}(s^2-1)(7s^2-3)s$ |
| 6 | $\frac{1}{16}\sqrt{\frac{11}{2}}(s^2-1)(21s^4-14s^2+1)$ |
| 7 | $\frac{1}{16}\sqrt{\frac{13}{2}}(s^2-1)(33s^4-30s^2+5)s$ |
| 8 | $\frac{1}{128}\sqrt{\frac{15}{2}}(s^2-1)(429s^6-495s^4+135s^2-5)$ |
| 9 | $\frac{1}{128}\sqrt{\frac{17}{2}}(s^2-1)(715s^6-1001s^4+385s^2-35)s$ |
| 10 | $\frac{1}{256}\sqrt{\frac{19}{2}}(s^2-1)(2431s^8-4004s^6+2002s^4-308s^2+7)$ |

Tablica II: Jawne wzory pochodnej funkcji Lobatto, $\psi'_k(s)$, dla $k \leq 10$.

| $k$ | $\psi'_k(s)$ |
|---|---|
| 0 | $-\frac{1}{2}$ |
| 1 | $\frac{1}{2}$ |
| 2 | $\sqrt{\frac{3}{2}}s$ |
| 3 | $\frac{1}{2}\sqrt{\frac{5}{2}}(3s^2-1)$ |
| 4 | $\frac{1}{2}\sqrt{\frac{7}{2}}(5s^2-3)s$ |
| 5 | $\frac{1}{8}\sqrt{\frac{9}{2}}(35s^4-30s^2+3)$ |
| 6 | $\frac{1}{8}\sqrt{\frac{11}{2}}(63s^4-70s^2+15)s$ |
| 7 | $\frac{1}{16}\sqrt{\frac{13}{2}}(231s^6-315s^4+105s^2-5)$ |
| 8 | $\frac{1}{16}\sqrt{\frac{15}{2}}(429s^6-693s^4+315s^2-35)s$ |
| 9 | $\frac{1}{128}\sqrt{\frac{17}{2}}(6435s^8-12012s^6+6930s^4-1260s^2+35)$ |
| 10 | $\frac{1}{256}\sqrt{\frac{19}{2}}(12155s^8-25740s^6+18018s^4-4620s^2+315)s$ |

W powyższym wzorze $P_k(t)$ oznacza wielomian Legendre'a [1] a $||P_k||$ oznacza jego normę:

$$||P_k||^2 = \int_{-1}^{1} P_k^2(t)\mathrm{d}t \tag{15}$$

Ze wzoru (13) wynik, że $\psi_0$ oraz $\psi_1$ są funkcjami liniowymi, natomiast dla $k > 2$ są wielomianami stopnia $k$. Na podstawie jawnego wzoru na $P_k(t)$

$$P_k(t) = \frac{1}{2^k}\sum_{m=1}^{\lfloor k/2 \rfloor}(-1)^m \binom{k}{m}\binom{2k-2m}{2k} t^{k-2m} \tag{16}$$

łatwo podać jawne wzory na funkcje Lobatto. Dla wygody funkcje $\psi_k(s)$ dla $k \leq 10$ zostały podane w tablei I.

Tablica III: Jawne wzory drugiej pochodnej funkcji Lobatto, $\psi''_k(s)$, dla $k \leq 10$.

| $k$ | $\psi''_k(s)$ |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | $\sqrt{\frac{3}{2}}$ |
| 3 | $3\sqrt{\frac{5}{2}}s$ |
| 4 | $\frac{3}{2}\sqrt{\frac{7}{2}}(5s^2-1)$ |
| 5 | $\frac{5}{2}\sqrt{\frac{9}{2}}(7s^2-3)s$ |
| 6 | $\frac{15}{8}\sqrt{\frac{11}{2}}(21s^4-14s^2+1)$ |
| 7 | $\frac{21}{8}\sqrt{\frac{13}{2}}(33s^4-30s^2+5)s$ |
| 8 | $\frac{7}{16}\sqrt{\frac{15}{2}}(429s^6-495s^4+135s^2-5)$ |
| 9 | $\frac{9}{16}\sqrt{\frac{17}{2}}(715s^6-1001s^4+385s^2-35)s$ |
| 10 | $\frac{45}{128}\sqrt{\frac{19}{2}}(2431s^8-4004s^6+2002s^4-308s^2+7)$ |

Z definicji (13) wynika, że dla $k \geq 2$ zachodzi

$$\psi'_k(s) = \tilde{P}_{k-1}(s) \tag{17}$$

Oznacza, to że pochodną funkcji Lobatto dla $k \geq 2$ jest znormalizowany wielomian Legendre'a. Ponieważ wielomiany Legendre'a są ortogonalne na odcinku $[-1,1]$, to równość (17) ma wielkie znaczenie podczas obliczania elementów macierzowych. Dla wygody funkcje $\psi'_k(s)$ dla $k \leq 10$ zostały podane w tabeli II.

Czasami potrzebna jest druga pochodna po funkcji $f$ reprezentowanej jako kombinacja liniowa funkcji Lobatto. Taka sutuacja ma miejsce, gdy $f$ jest orbitalem atomowym i należy policzyć całki kinetyczne. W takim przypadku potrzebna jest druga pochodna po funkci Lobatto. Funkcje $\psi''_k(s)$ dla $k \leq 10$ zostały podane w tabeli III.

### 2. *Pomocnicze funkcje bazowe* $\{\phi_i^n\}$

Wprowadźmy przekształcenie

$$X_n : [-1,1] \mapsto \Omega_n, \tag{18}$$

które przeprowadza odcinek jednostkowy $[-1,1]$ na element $\Omega_n$. Łatwo pokazać, że $X_n$ jest przekształceniem liniowym (a tym samym bijekcją) zadanym wzorem

$$X_n(s) = c_{1,n} + sc_{2,n}, \tag{19}$$

gdzie współczynniki $c_{1,n}$, $c_{2,n}$ zadane są wzorami

$$c_{1,n} = \frac{x_n + x_{n-1}}{2}, \qquad c_{2,n} \equiv J_n = \frac{x_n - x_{n-1}}{2}, \tag{20}$$

a $J_n$ jest jakobianem elementu $\Omega_n$.

Korzystając z przekształcenia $X_n$, pomocnicze funkcje bazowe $\phi_i^n$ (dla $i = 0, \ldots, p_n$, gdzie $p_n$ jest stopniem elementu $\Omega_n$) na elemencie $\Omega_n$ definiuje się jako złożenie

funkcji

$$\phi_i^n(x) = \psi_i \circ X_n^{-1}(x) \equiv \psi_i(X_n^{-1}(x)). \tag{21}$$

Powyższa definicja jest zawsze poprawna, gdyż odwzorowanie $X_n$ jest bijekcją, a to oznacza, że zawsze istnieje odwzorowanie $X_n^{-1}$. Warto podkreślić, że w prezentowanym sformułowaniu metody elementu skończonego nie ma potrzeby jawnego tworzenia odwzorowania $X_n^{-1}$.

Z definicji przekształcenia $X_n$ oraz wzoru (13) wynikają następujące wnioski

$$\phi_0^n(x_{n-1}) = 1 \qquad \phi_0^n(x_n) = 0 \tag{22a}$$
$$\phi_1^n(x_{n-1}) = 0 \qquad \phi_1^n(x_n) = 1 \tag{22b}$$

Wnioski te zostaną wykorzystane w następnej sekcji do utworzenia funkcji bazowych.

#### 3. *Funkcje bazowe* $\{\phi_k\}$

Funkcje bazowe $\{\phi_k\}$ tworzy się na podstawie pomocniczych funkcji bazowych $\{\phi_i^n\}$. Należy podkreślić, że funkcje $\phi_i^n$ dla $i = 0, \ldots, p_n$ są zdefiniowane na elemencie $\Omega_n$ czyli odcinku $[x_{n-1}, x_n]$.

Niech $K : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{Q}$ będzie bijekcją, gdzie $\mathbb{Q} = \{1, \ldots, M\}$ oraz $M$ określone jest wzorem (25). Funkcja $K$ przyporządkowuje parze indeksów $(i, n)$ unikalny indeks $k$, gdzie indeks $i$ określa numer funkcji bazowej na elemencie $\Omega_n$, a indeks $k$ jest globalnym indeksem funkcji bazowej $\phi_k$.

Niech $k = K(i, n)$. Wtedy, funkcje bazowe dla $i \geq 2$ tworzymy według wzoru

$$\phi_k(x) = \begin{cases} \phi_i^n(x) & \text{dla } x \in \Omega_n \\ 0 & \text{w p.p.} \end{cases} \tag{23}$$

Tak zdefiniowane funkcje nazywamy *bubble shape functions*. Z powyższego wynika, że dziedziną funkcji $\phi_k$ dla $i \geq 2$ jest odcinek $\Omega_n$ i jest wielomianem stopnia $n$.

Dodatkowo, funkcje bazowe dla $i = \{0, 1\}$ tworzymy według wzoru

$$\phi_k(x) = \begin{cases} \phi_1^{n-1}(x) & \text{dla } x \in \Omega_{n-1} \\ \phi_0^n(x) & \text{dla } x \in \Omega_n \\ 0 & \text{w p.p.} \end{cases} \tag{24}$$

Tak zdefiniowane funkcje nazywamy *vertex shape functions*. Z powyższego wynika, że z każdym węzłem $x_n$ siatki $\mathcal{T}_{1\mathrm{D}}$ skojarzona jest jedna funkcja bazowa. Funkcja ta jest niezerowa na obszarze $\Omega_{n-1} \cup \Omega_n$, to znaczy na dwóch sąsiednich elementach. Ponadto, funkcja ta jest liniowa na odcinku $\Omega_{n-1}$ i liniowa na odcinku $\Omega_n$.

Na podstawie funkcji bazowych $\{\phi_k\}$ tworzona jest przestrzeń $\mathcal{S} = \mathrm{span}\{\phi_k\}_{k=1}^M$, w której szuka się rozwiązań. Wymiar przestrzeni $\mathcal{S}$ zadany jest wzorem

$$M = \underbrace{N-1}_{\mathrm{I}} + \underbrace{\sum_{n=1}^{N}(p_n - 1)}_{\mathrm{II}} = \sum_{n=1}^{N} p_n - 1 \tag{25}$$

gdzie $N$ jest liczbą elementów w siatce, a $p_n$ jest stopniem elementu $\Omega_n$. Człon $I$ oznacza funkcje bazowe pierwszego rzędu zadane wzorem (24). W zasadzie funkcji (24) jest $(N+1)$. Jednakże dwa warunki brzegowe Dirichleta powodują, że stopni swobody utworzonych przez te funkcje jest $(N-1)$. Człon $II$ oznacza funkcje bazowe wyższych rzędów, które zadane są wzorem (23).

### C. Dyskretyzacja

W tej sekcji zostanie podana dyskretna postać równań (10), (11). Podczas dyskretyzacji (zgodnie z sformułowaniem podanym przez Galerkina) zakłada się, że szukana funkcja $u$ oraz funkcja próbna $v$ należą do przestrzeni $\mathcal{S}$ rozpiętej na funkcjach bazowych $\{\phi_i\}_{i=1}^M$, gdzie $M$ jest wymiarem przestrzeni $\mathcal{S}$ zadanym wzorem (25). Oznacza to, że każdą funkcję $u \in \mathcal{S}$ można przedstawić w postaci kombinacji liniowej

$$u(x) = \sum_{i=1}^{M} c_i \phi_i(x) \tag{26}$$

gdzie $\mathbf{c} = [c_1, \ldots, c_M]^T \in \mathbb{R}^M$ jest wektorem współczynników. W kolejnych sekcjach przedstawione zostaną konsekwencje powyższego rozwinięcia.

#### 1. *Dyskretyzacja równania różniczkowego*

Podstawiając równanie (26) do słabej postaci równania różniczkowego (6) otrzymujemy liniowy układ równań

$$\mathbf{S}\mathbf{c} = \mathbf{b}, \tag{27}$$

gdzie macierz $\mathbf{S}$ zadana jest jako suma macierzy

$$\mathbf{S} = \mathbf{F} + \mathbf{G}. \tag{28}$$

Elementy macierzy $\mathbf{F}$ zadane sa całką

$$\begin{aligned} \mathbf{F}_{i,j} &= \int_a^b h(x)\phi_i'(x)\phi_j'(x)\mathrm{d}x \\ &= \sum_{n=1}^{N} \int_{\Omega_n} h(x)\phi_i'(x)\phi_j'(x)\mathrm{d}x. \end{aligned} \tag{29}$$

W przypadku, gdy funkcja $h(x) = \gamma$ jest funkcją stałą, to powyższa całka redukuje się do postaci

$$\mathbf{F}_{i,j} = \gamma \sum_{n=1}^{N} \int_{\Omega_n} \phi_i'(x)\phi_j'(x)\mathrm{d}x, \tag{30}$$

którą można policzyć analitycznie, tak jak jest to dyskutowane w Sekcji VI D. Macierz $\mathbf{G}$ ze wzoru (28) jest całką z iloczynu trzech funkcji

$$\mathbf{G}_{i,j} = \sum_{n=1}^{N} \int_{\Omega_n} g(x)\phi_i(x)\phi_j(x)\mathrm{d}x. \tag{31}$$

We wzorze (27) wektor $\mathbf{b}$ wyrażony jest jako suma dwóch wektorów

$$\mathbf{b} = \mathbf{f} - \mathbf{g}, \tag{32}$$

gdzie elementy wektora $\mathbf{f}$ zadane są całką

$$\mathbf{f}_i = \sum_{n=1}^{N} \int_{\Omega_n} f(x)\phi_i(x)\mathrm{d}x. \tag{33}$$

W ogólnej postaci elementy wektora $\mathbf{g}$ wyrażone są jako

$$\mathbf{g}_i = \int_a^b \Big[h(x)\hat{u}'(x)\phi_i'(x) + g(x)\hat{u}(x)\phi_i(x)\Big]\mathrm{d}x \tag{34}$$

Ponieważ funkcja $\hat{u}(x)$ jest różna od zera tylko na elementach $\Omega_1, \Omega_N$, to powyższy wzór redukuje się do

$$\mathbf{g}_i = \sum_{n=\{1,N\}} \int_{\Omega_n} \Big[h(x)\hat{u}'(x)\phi_i'(x) + g(x)\hat{u}(x)\phi_i(x)\Big]\mathrm{d}x \tag{35}$$

gdzie całki obliczane są tylko po przedziałach $\Omega_1$, $\Omega_N$. W sekcji V A 2 zostało zaproponowane aby funkcja $\hat{u}(x)$ była liniowa na przedziałach $\Omega_1$ i $\Omega_N$, tak jak to zostało pokazane na Rys. 1. Łatwo zauważyć, że w takim przypadku funkcję $\hat{u}$ można zapisać za pomocą liniowych funkcji bazowych

$$\hat{u}(x) = \begin{cases} u_a\phi_{k1}(x) & \text{dla } x \in \Omega_1 \\ u_b\phi_{k2}(x) & \text{dla } x \in \Omega_N \\ 0 & \text{w p.p.} \end{cases} \tag{36}$$

gdzie $k1, k2$ są indeksami definiującymi stosowne funkcje bazowe tak jak to zostało opisane w sekcji VI C 3. Korzystając z powyższego wzoru oraz wzoru (35) otrzymujemy

$$\mathbf{g}_i = \begin{cases} G_a^i & \text{gdy dziedzina } \phi_i = \Omega_1 \\ G_b^i & \text{gdy dziedzina } \phi_i = \Omega_N \\ 0 & \text{w p.p.} \end{cases} \tag{37}$$

gdzie

$$G_a^i = u_a \int_{\Omega_1} \Big[h(x)\phi_{k1}'(x)\phi_i'(x) + g(x)\phi_{k1}(x)\phi_i(x)\Big]\mathrm{d}x$$

$$G_b^i = u_b \int_{\Omega_N} \Big[h(x)\phi_{k2}'(x)\phi_i'(x) + g(x)\phi_{k2}(x)\phi_i(x)\Big]\mathrm{d}x$$

Porównując powyższe wzory z wzorami (30) i (31) widzimy, że powyższe całki można policzyć w analogiczny sposób jak elementy macierzy $\mathbf{F}$ i $\mathbf{G}$.

W przypadku gdy rozważany jest Problem 4 z zerowymi warunkami brzegowymi Dirichleta, to $\mathbf{g} = 0$ i wtedy $\mathbf{b} = \mathbf{f}$.

Z powyższych wzorów wynika, że z powodu dowolności funkcji $g, f$ elementy macierzy $\mathbf{G}$ oraz wektorów $\mathbf{f}$, $\mathbf{g}$ muszą być obliczane numerycznie, na przykład za pomocą kwadratur Gaussa. Gdy funkcja $g$ lub funkcja $h$ jest funkcją stałą, to nie ma potrzeby stosowania kwadratur Gaussa do obliczenia macierzy $\mathbf{G}$ lub macierzy $\mathbf{F}$, gdyż można je policzyć analitycznie. Algorytm analitycznego obliczania macierzy $\mathbf{G}$ i $\mathbf{F}$ jest wyprowadzony w Sekcji VI D.

### *2. Dyskretyzacja zagadnienia własnego*

Podstawiając równanie (26) do słabej postaci zagadnienia własnego (11) otrzymujemy uogólnione zagadnienie własne

$$\mathbf{Sc} = \lambda\mathbf{Bc} \tag{38}$$

W powyższym równaniu macierz $\mathbf{S}$ jest zdefiniowana wzorem (28). Natomiast element macierzy $\mathbf{B}$ zdefiniowana jest jako całka

$$\mathbf{B}_{i,j} = \sum_{n=1}^{N} \int_{\Omega_n} \phi_i(x)\phi_j(x)\mathrm{d}x. \tag{39}$$

Analityczny algorytm obliczania tej całki pokazany jest w Sekcji VI D.

### *3. Numerowanie funkcji bazowych*

Funkcja $K$ zdefiniowana w Sekcji VI B 3 ustala kolejność funkcji bazowych $\{\phi_k\}$. Z punktu widzenia rozwinięcia elementu przestrzeni $\mathcal{S}$ jako kombinacji liniowej funkcji bazowych, kolejność funkcji bazowych nie ma znaczenia. Jednakże kolejność funkcji bazowych determinuje strukturę macierzy $\mathbf{S}$, $\mathbf{B}$, które otrzymuje się podczas dyskretyzacji równania różniczkowego lub zagadnienia własnego.

Jeżeli pominąć strukturę macierzy $\mathbf{S}$, $\mathbf{B}$, to jedynym warunkiem nałożonym na funkcję $K$ jest jednoznaczne przyporządkowanie parze indeksów $(i, n)$ liczby całkowitej z zakresu $1, \ldots, M$. Nie ma dodatkowych warunków na funkcje $K$ nie ma.

W bibliotece `RFem1D` funkcja $K$ jest tak dobrana, że macierz $\mathbf{B}$ i macierz $\mathbf{F}$ w przypadku stałej funkcji $h$ są pasmowe. Ponadto, szerokość pasma (to znaczy, liczba elementów pozadiagonalnych nie licząc diagonali) macierzy $\mathbf{B}$ i macierzy $\mathbf{S}$ wynosi:

$$\max_n\{p_n\} \qquad \text{gdzie} \qquad n = 1, \ldots, N \tag{40}$$

Wykorzystanie takiego przyporządkowania $K$ umożliwia stosowanie algorytmów specyficznych dla macierzy pasmowych [3, 14, 21, 27] i nie ma potrzeby korzystania z ogólnych algorytmów dla macierzy rzadkich [21, 24, 25].

Poniżej podany jest opis zastosowanego algorytmu przyporządkowania $K$ w bibliotecte `RFem1D`.

1. $k = 0$. Dla elementu $\Omega_1$ numeruj kolejno wszystkie funkcje *bubble*.
2. Dla każdego elementu $\Omega_n$ ze zbioru $\{\Omega_2, \ldots, \Omega_{N-1}\}$:
   - Numeruj funkcję *vertex* z węzła $x_{n-1}$.
   - Numeruj kolejno wszystkie funkcje *bubble*.
3. Dla elementu $\Omega_N$ numeruj funkcję *vertex* z węzła $x_{N-1}$ kolejno wszystkie funkcje *bubble*

W powyższym "numeruj" oznacza zwiększ $k$, to znaczy $k = k + 1$.

Informacje o numerowaniu funkcji bazowych przechowywane są w tablicy połączeń (ang. *connectivity array*). W tablicy połączeń, dla każdego elementu $\Omega_n$, przechowywane są indeksy referencyjnych funkcji bazowych (funkcji Lobatto). W bibliotece `RFem1D` z każdym elementem $\Omega_n$ skojarzona jest tablica o długości $p_n + 1$, gdzie $p_n$ jest rzędem elementu $\Omega_n$. Dla elementu $\Omega_n$ tablica wypełniona jest indeksami funkcji bazowych $\phi_k$ przestrzeni $\mathcal{S}$, które odpowiadają referencyjnym funkcjom bazowym $\psi_0, \psi_1, \ldots, \psi_{p_n}$ z elementu $\Omega_n$. Informacje zawarte w tablicy połączeń wykorzystywane są podczas procesu tworzenia macierzy $\mathbf{S}$ i $\mathbf{B}$.

### D. Analityczne obliczanie potrzebnych całek

W tej sekcji zostanie pokazane jak analitycznie policzyć całkę

$$I_1 = \int_{\Omega_n} \phi_i(x)\phi_j(x)\mathrm{d}x \tag{41}$$

oraz całkę

$$I_2 = \int_{\Omega_n} \phi_i'(x)\phi_j'(x)\mathrm{d}x \tag{42}$$

dla dowolnej pary $(i, j)$ indeksów i obszaru $\Omega_n$. Do obliczenia obu całek zostanie wykorzystane całkowanie przez podstawienie i wykorzystane przekształcenie $X_n$ zdefiniowane wzorem (19).

Z analizy matematycznej [11] wiadomo, że dla każdej całkowalnej funkcji $w : \Omega_n \mapsto \mathbb{R}$ zachodzi

$$\int_{\Omega_n} w(x)\mathrm{d}x = J_n \int_{-1}^{1} w(X_n(s))\mathrm{d}s \tag{43}$$

Ponadto z definicji przekształcenia $X_n$ (wzór (19)) dostajemy ważną tożsamość dla $s \in [-1, 1]$

$$\phi_i^n(X_n(s)) = \psi\Big(X_n^{-1}\big(X_n(s)\big)\Big) = \psi_i(s) \tag{44}$$

Korzystając z tej tożsamości dla funkcji $w = \phi_i^n$ mamy

$$I_1 = \int_{\Omega_n} \phi_i^n(x)\phi_j^n(x)\mathrm{d}x = J_n \int_{-1}^{1} \psi_i(s)\psi_j(s)\mathrm{d}s \tag{45}$$

Zadanie obliczania całki $I_1$ zostało więc sprowadzone do obliczania całek po referencyjnych funkcjach bazowych. Jak policzyć tę całkę jest pokazane w Sekcji VI D 1.

Aby obliczyć całkę $I_2$ należy najpierw wyrazić pochodną $[\phi_i^n]'(x)$ w zmiennej $s$ korzystając ze wzoru na pochodną funkcji złożonej. Na mocy wzoru (44) mamy

$$\phi_i^n(X_n(s)) = \psi_i(s) \tag{46}$$

Obliczając pochodną dostajemy

$$\psi_i'(s) \equiv [\phi_i^n(X_n(s))]' = J_n[\phi_i^n]'(x), \tag{47}$$

co daje wyrażenie na szukaną pochodną

$$[\phi_i^n]'(x) = \frac{1}{J_n}\psi_i'(s). \tag{48}$$

Podstawiając powyższy wzór do całki $I_2$ otrzymujemy

$$I_2 = \frac{1}{J_n}\int_{-1}^{1} \psi_i'(s)\psi_j'(s)\mathrm{d}s. \tag{49}$$

Zadanie obliczania całki $I_2$ zostało więc sprowadzone do obliczania całek z pochodnych po referencyjnych funkcjach bazowych. Jak policzyć tę całkę jest pokazane w Sekcji VI D 1.

#### 1. *Całki po funkcjach Lobatto*

Ważną właściwością funkcji Lobatto jest to, że w metodzie elementu skończonego prowadzą do macierzy rzadkich. W tej sekcjo zostanie opisane jak policzyć pewne całki pomiędzy tymi funkcjami.

Z definicji funkcji Lobatto (wzór 13) wiemy, że dla $k \geq 2$ zachodzi

$$\psi_k'(s) = \tilde{P}_{k-1}(s), \tag{50}$$

gdzie $\tilde{P}_{k-1}(s)$ jest unormowanym wielomianem Lenendre'a. Wykorzystując ortogonalność wielomianu Legendre'a dla $i, j \geq 2$ natychmiast otrzymujemy

$$\int_{-1}^{1} \psi_i'(s)\psi_j'(s)\mathrm{d}s = \int_{-1}^{1} \tilde{P}_{i-1}(s)\tilde{P}_{j-1}(s)\mathrm{d}s = \delta_{i,j} \tag{51}$$

Ponadto dla $j \geq 2$ mamy

$$\int_{-1}^{1} \psi_0'(s)\psi_j'(s)\mathrm{d}s = \int_{-1}^{1} \psi_1'(s)\psi_j'(s)\mathrm{d}s = 0 \tag{52}$$

Ponieważ $\psi_0$, $\psi_1$ są jawnie zadane, więc pozostałe całki dla $i, j = 0, 1$ mozna z łatwośi policzyć. Dla wygody wszystkie powyższe informacje zostałe przedstawione w postaci macierzy

$$\int_{-1}^{1} \psi_i'(s)\psi_j'(s)\mathrm{d}s = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 & \cdots & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & 0 & \cdots & 1 \end{bmatrix} \tag{53}$$

W tabeli I zostały podane jawne wzory na funkcje Lobatto $\psi_k(x)$ dla $k \leq 10$. Ponieważ funkcje $\psi_i(x)$ są wielomianami, to można łatwo policzyć całki z iloczynu tych funkcji

$$\mathbf{K}_{i,j} = \int_{-1}^{1} \psi_i(s)\psi_j(s)\mathrm{d}s \tag{54}$$

Wartości tych całek zostały obliczone za pomocą pakietu `Mathematica` i sa podane poniżej. Widać, że dużo elementów macierzy $\mathbf{K}$ jest równe zero.

$$\mathbf{K}_{i,j} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & \frac{-1}{\sqrt{6}} & \frac{1}{3\sqrt{10}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \frac{2}{3} & \frac{-1}{\sqrt{6}} & \frac{-1}{3\sqrt{10}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & \frac{2}{5} & 0 & \frac{-1}{5\sqrt{21}} & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & \frac{2}{21} & 0 & \frac{-1}{21\sqrt{5}} & 0 & 0 & 0 & 0 & 0 \\ & & & & \frac{2}{45} & 0 & \frac{-1}{9\sqrt{77}} & 0 & 0 & 0 & 0 \\ & & & & & \frac{2}{77} & 0 & \frac{-1}{33\sqrt{13}} & 0 & 0 & 0 \\ & & & & & & \frac{2}{117} & 0 & \frac{-1}{13\sqrt{165}} & 0 & 0 \\ & & & & & & & \frac{2}{165} & 0 & \frac{-1}{15\sqrt{221}} & 0 \\ & & & & & & & & \frac{2}{221} & 0 & \frac{-1}{17\sqrt{285}} \\ & & & & & & & & & \frac{2}{285} & 0 \\ & & & & & & & & & & \frac{2}{357} \end{bmatrix} \tag{55}$$

### E. Kwadratury Gaussa

Z powodu dowolności funkcji $f(x)$, elementy wektora $\mathbf{b}$, zdefiniowane wzorem (32), oblicza się za pomocą kwadratur Gaussa. Stopień kwadratury Gaussa równa się $2\max_n\{p_n\}$, gdzie $p_n$ jest stopniem elementu $\Omega_n$.

Ponieważ funkcja $g(x)$ jest dowolna, więc kwadratury Gaussa także wykorzystuje się do obliczania elementów macierzy $\mathbf{G}$ zdefiniowanych wzorem (31). Gdy funkcja $g(x)$ jest funkcją stałą, to do jej obliczania można wykorzystać analityczne wzory podane do obliczania macierzy $\mathbf{B}$. Stosowanie analitycznych wzorów znacznie skraca czas wykonania i nie powoduje błądów wynikających z całkowania numerycznego.

### F. Algorytm adaptacyjny

W tej sekcji zostanie opisany algorytm adaptacyjny, który został zaimplementowany w tej bibliotece. Algorytm ten został wymyślony przez autora tego artykułu. Bazuje on na spostrzeżeniu, że współczynniki rozwinięcia szybko maleją. Innymi słowy, jeżeli najmniejszy współczynnik rozwinięcia na elemencie jest za duży to należy ten element podzielić na dwa mniejsze elementy. Ponadto, aby podziały miały największy wpływ na rozwiązanie, to do podziału wybierane są elementy o największym współczynniku wybranym ze zbioru najmniejszych współczynników.

Ta idea adaptacyjności jest wspólna dla algorytmu rozwiązywania zagadnienia własnego oraz równania różniczkowego. Implementacja dla tych dwóch przypadków różni się tylko szczegółami, które podane są poniżej.

#### 1. *Zagadnienie własne*

Rozważmy siatkę $\mathcal{T}$. Wykorzystując tę siatkę obliczamy kilka najmniejszych wartości własnych i odpowiadających im wektorów własnych algorytmem opisanym w poprzednich sekcjach. Ponieważ algorytm bazuje na metodzie elementu skończonego, to składowe wektorów własnych odpowiadają rozwinięciom na funkcje Lobatto. Dla każdego $i$-tego wektora własnego i dla każdego elementu siatki $\mathrm{e} \in \mathcal{T}$ można znaleźć najmniejszy współczynnik rozwinięcia. Oznaczmy ten współczynnik przez $c_{i,\mathrm{e}}$. Ze zbioru tych współczynników $\{c_{i,\mathrm{e}}\}$, dla zadanego $i$, największy współczynnik

$$c_i^* = \max_{\mathrm{e}\in\mathcal{T}} \{|c_{i,\mathrm{e}}|\} \tag{56}$$

jest wybierany. Ponieważ siatka $\mathcal{T}$ jest skończona, to dla każdej $i$-tej wartości własnej istnieje element $\mathrm{e}_i^*$ odpowiadający współczynnikowi $c_i^*$. W zaimplementowanym algorytmie wszystkie elementy $\mathrm{e}_i^*$ dla każdego $i$ są zaznaczone do podziału. Każdy zaznaczony element jest podzielony na dwie połowy i utworzona nowa siatka $\mathcal{T}'$. Obliczenia są powtórzone dla nowej siatki $\mathcal{T}'$. Proces ten jest tak długo powtarzany aż współczynniki $c_i^*$ są mniejsze niż ustalona wartość graniczna. W funkcji `EigProb::SolveAdapt` wartością graniczną jest argument `absMaxCoef`.

#### 2. *Równanie różniczkowe*

Dla równania różniczkowego algorytm adaptacyjny jest analogiczny jak dla zagadnienia własnego. Jeżeli w zagadnieniu własnym obliczany by był tylko jeden wektor

własny, to analogia byłaby jeszcze bardziej ścisła. Dla porządku podajemy ten algorytm jawnie.

Rozważmy siatkę $\mathcal{T}$. Wykorzystując siatkę $\mathcal{T}$ rozwiązujemy równanie różniczkowe algorytmem opisanym w poprzednich sekcjach. Ponieważ algorytm bazuje na metodzie elementu skończonego, to rozwiązaniem są współczynniki rozwinięciom na funkcje Lobatto. Dla każdego elementu siatki $\mathrm{e} \in \mathcal{T}$ znajdujemy najmniejszy współczynnik rozwinięcia. Oznaczmy ten współczynnik przez $c_\mathrm{e}$. Ze zbioru tych współczynników $\{c_\mathrm{e}\}$, jest wybierany największy współczynnik

$$c^* = \max_{\mathrm{e}\in\mathcal{T}} \{|c_\mathrm{e}|\} \tag{57}$$

Ponieważ siatka $\mathcal{T}$ jest skończona, to istnieje element $\mathrm{e}^*$ odpowiadający współczynnikowi $c^*$. W zaimplementowanym algorytmie element $\mathrm{e}^*$ dzielony jest na dwie połowy i tworzona jest nowa siatka $\mathcal{T}'$. Obliczenia są powtórzone dla nowej siatki $\mathcal{T}'$. Proces ten jest tak długo powtarzany aż współczynnik $c^*$ jest mniejszy niż ustalona wartość graniczna. W funkcji `OdeProb::SolveAdapt` wartością graniczną jest argument `absMaxCoef`.

## VII. APROKSYMACJA WIELOMIANAMI LOBATTO

W tej sekcji opisany jest proces aproksymacji funkcji ciągłej, zdefiniowanej na skończonym przedziale, wielomianami Lobatto. Analizowane są dwa przypadki. W pierwszym przypadku aproksymacja wielomianowa jest szukana na całej dziedzinie. W drugim przypadku aproksymacja jest szukana adaptacyjnie dzieląc dziedzinę na stosowne pod przedziały.

### A. Aproksymacja na jednym przedziale

W tej sekcji zostanie opisany algorytm rozwiązywania Problemu 3. Ponieważ w Problemie 3 zakładamy, że funkcje aproksymowana $f$ i funkcja aproksymująca $g$ muszą być równe to musi zachodzić

$$f(a) = g(a) \quad \text{and} \quad f(b) = g(b) \tag{58}$$

Aby spełnić ten warunek wprowadza się nową funkcję

$$\tilde{f}(x) = f(x) - f(a)\phi_0(x) - f(b)\phi_1(x) \tag{59}$$

Na podstawie równania (22) wiemy, że funkcja $\tilde{f}$ spełnia warunki

$$\tilde{f}(a) = \tilde{f}(b) = 0 \tag{60}$$

Ponieważ tylko funkcje $\phi_0(x)$, $\phi_1(x)$ przyjmują wartości różne od zera na brzegu przedziału $[a, b]$, to szukana funkcja aproksymująca $g$ dla funkcji $\tilde{f}$ nie musi zawierać tych dwóch funkcji. Oznacza to, że mamy

$$\tilde{f}(x) \approx g(x) = \sum_{i=2}^{M} c_i \phi_i(x) \tag{61}$$

gdzie $c_i$ są współczynnikami rozwinięcia. W zasadzie współczynniki $c_i$ zależą od $M$, jednakże nie jest to jawnie zaznaczone w powyższej notacji. Wprowadźmy residuum

$$R(x) = \tilde{f}(x) - g(x) \tag{62}$$

które jest miarą błędu aproksymacji. Analogicznie do metody elementu skończonego w sformułowaniu Galerkina współczynniki $\mathbf{c} = \{c_2, \ldots, c_M\}^T$ otrzymujemy z warunków ortogonalności

$$\int_a^b R(x)\phi_j(x)\mathrm{d}x = 0 \qquad \text{dla} \quad j = 2, \ldots, M. \tag{63}$$

Podstawiając równanie (62) do równania (63) otrzymujemy

$$\sum_{i=2}^{M} c_i \int_a^b \phi_i(x)\phi_j(x)\mathrm{d}x = \int_a^b \tilde{f}(x)\phi_j(x)\mathrm{d}x \tag{64}$$

Wprowadźmy wektor $\mathbf{b}$ jako

$$\mathbf{b}_j = \frac{1}{A}\int_a^b \tilde{f}(x)\phi_j(x)\mathrm{d}x = \int_{-1}^{1} \tilde{f}\big(X(s)\big)\psi_j(s)\mathrm{d}s \tag{65}$$

Wtedy równanie (64) może być zapisane jako układ równań w postaci macierzowej

$$A\,\mathbf{K}\mathbf{c} = A\,\mathbf{b} \tag{66}$$

co jest równoważne równaniu

$$\mathbf{K}\mathbf{c} = \mathbf{b} \tag{67}$$

Macierz $\mathbf{K}$ jest zadana równaniem (54) i nie zależy od funkcji $f$, więc można ją obliczyć i zapisać w pliku. Rozwiązując powyższe równanie z symetryczną i dodatnio określoną macierzą $\mathbf{K}$ otrzymujemy współczynniki $\mathbf{c}$ a tym samym szukaną aproksymację $g(x)$ zdefiniowaną wzorem (61).

Błąd aproksymacji zdefiniowany jest jako całka z kwadratu różnicy

$$\begin{aligned}
\Delta &= \int_a^b R^2(x)\mathrm{d}x = \int_a^b \big[\tilde{f}(x) - g(x)\big]^2\mathrm{d}x \qquad (68)\\
&= \int_a^b \Big[\tilde{f}(x) - \sum_{i=2}^{M} c_i\phi_i(s)\Big]^2\mathrm{d}x \\
&= \int_a^b \tilde{f}^2(x)\mathrm{d}x - 2\sum_{i=2}^{M} c_i \int_a^b \tilde{f}(x)\phi_i(x)\mathrm{d}x \\
&\quad + \sum_{i,j=2}^{M} c_i c_j \int_a^b \phi_i(x)\phi_j(x)\mathrm{d}x
\end{aligned}$$

Mnożąc równanie (64) prze $c_j$ oraz sumując po wszystkich $j$ dostajemy

$$\sum_{i,j=2}^{M} c_i c_j \int_a^b \phi_i(x)\phi_j(x)\mathrm{d}x = \sum_{j=2}^{M} c_j \int_a^b \tilde{f}(x)\phi_j(x)\mathrm{d}x \tag{69}$$

Podstawiając powyższe równanie do równania (68) mamy

$$\Delta = \int_a^b \tilde{f}^2(x)\mathrm{d}x - \sum_{i,j=2}^{M} c_i c_j \int_a^b \phi_i(x)\phi_j(x)\mathrm{d}x \qquad (70)$$

Dodatkowo, wykorzystując definicję całek przekrycia otrzymujemy

$$\begin{aligned} \Delta &= \int_a^b \tilde{f}^2(x)\mathrm{d}x - A\sum_{i,j=2}^{M} c_i c_j \mathbf{K}_{i,j} \qquad (71)\\ &= \int_a^b \tilde{f}^2(x)\mathrm{d}x - A\mathbf{c}\mathbf{K}\mathbf{c} \end{aligned}$$

Współczynnik $\Delta$ zależy od $M$. Aby obliczyć współczynnik $\Delta$ należy zastosować całkowanie numeryczne do funkcji $f^2$. Obliczanie $\Delta$ ze wzoru (71) jest bardziej stabilne numerycznie niż obliczanie go z równania (68). Dodatkowo, jeżeli aproksymacja jest wykonywana wielokrotnie dla ustalonego $f$ na ustalonym przedziale $[a, b]$ to obliczanie $\Delta$ wymaga tylko dwóch mnożeń macierzy przez wektor, co jest dużo szybsze niż całkowanie numeryczne.

### B. Aproksymacja adaptacyjna

W tej sekcji zostanie opisany algorytm rozwiązywania Problemu 3 algorytmem adaptacyjnym. Rozwiązanie Problemu 3 bazuje na efektywnej implementacji rozwiązania Problemu 3 i wykorzystuje metodę bisekcji. Najważniejszy w opisywanym algorytmie jest to, że każdy węzeł $r_i$ szukanego podziału można znaleźć niezależnie, co wielce upraszcza problem. Zadanie sprowadza się do znalezienie punktu $r_0$ takiego, że aproksymacja wielomianami stopnia $M$ z wykorzystaniem algorytmu opisanego w sekcji VII A takiego, że błąd aproksymacji jest mniejszy niż $\Delta$. Jeżeli $r_0$ zostanie znalezione to analogicznie szukany jest punkt $r_1$.

Węzeł $r_0$ szukany jest w następujący sposób.

1. Wybierz krok $h > 0$.
2. Oblicz błąd aproksymacji $\epsilon$ na przedziale $[a, x_0]$, gdzie $x_i = 2^i h$ dla $i = 0, 1, \ldots$.
3. Jeżeli $\epsilon < \Delta$, to $i = i+1$ i wykonaj poprzedni punk jeszcze raz.
4. Jeżeli $\epsilon \geq \Delta$, to oblicz metodą bisekcji $r_0$ takie, że $\epsilon \approx \Delta$. Bisekcje wykonaj na przedziale $[x_{i-1}, x_i]$.

Z powyższego opisu wynika, że algorytm z Sekcji VII A jest wywoływany wielokrotnie. Powyższy algorytm został zaimplementowany w funkcji `FindB`. Wielokrotne zastosowanie funkcji `FindB` pozwala otrzymać szukany ciąg $\{r_i\}$ i jest to zaimplementowane w funkcji `SolveAdapt`.

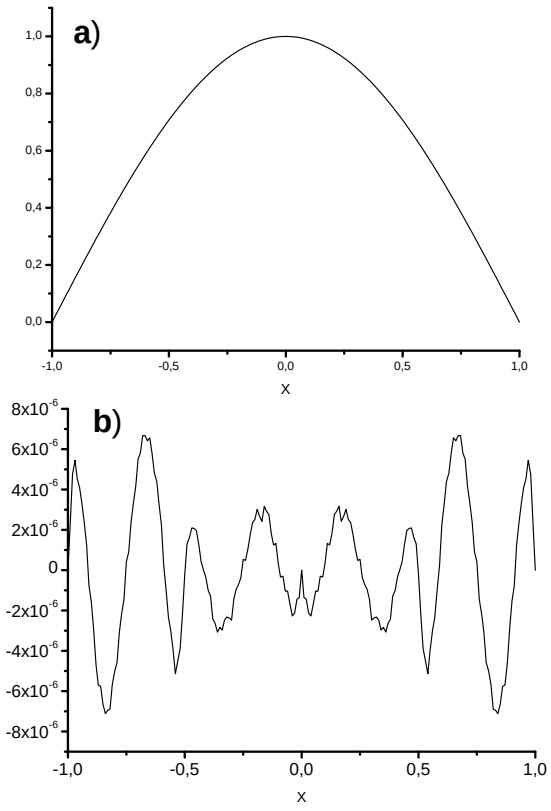
## VIII. PRZYKŁADY ROZWIĄZAŃ

W tej sekcji znajdują się przykłady zagadnień, dla których znane są rozwiązania analityczne. Podane przykłady demonstrują dokładność i efektywność zaimplementowanej metody.

### A. Równania różniczkowe

#### 1. *Przykład 1*

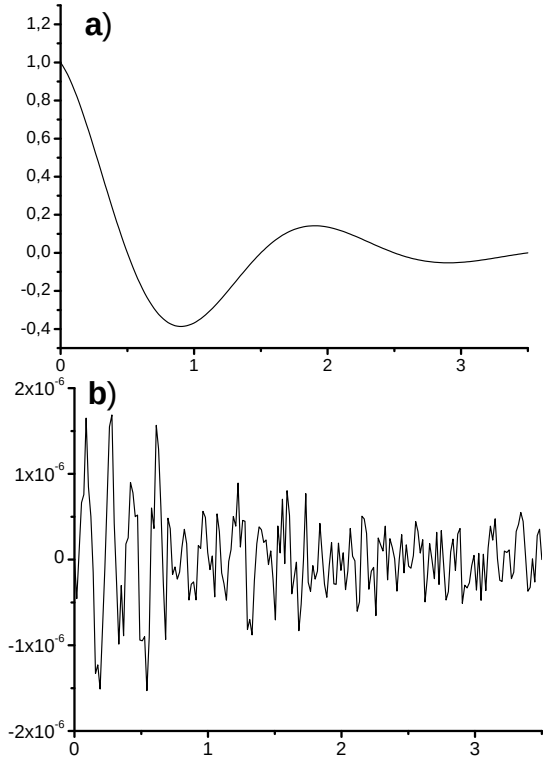
Rozważmy zagadnienie $-u''(x) = \frac{1}{4}\pi^2 \cos(\frac{1}{2}\pi x)$ na przedziale $x \in [-1, 1]$ z warunkami brzegowymi Dirichleta $u(-1) = u(1) = 0$. Rozwiązaniem dokładnym tego równania jest $u(x) = \cos(\frac{1}{2}\pi x)$, pokazane na rysunku 2. Przybliżone rozwiązanie $\tilde{u}(x)$ tego równania zostało otrzymane na siatce składającej się z $M = 4$ elementów o równej długości. Stopień każdego elementu jest $p = 4$. Rozmiar macierzy $\mathbf{S}$ (zgodnie ze wzorem (25)) wynosi $N = 19$. Na rysunku 2 podane jest różnica $\tilde{u}(x) - u(x)$. Z rysunku tego wynika, że $|\tilde{u}(x) - u(x)| < 8 \cdot 10^{-6}$, co dowodzi dużej dokładności przybliżonego rozwiązania.




Rysunek 2: Wyniki dla przykładu 1. a) Funkcja $u(x) = \cos(\frac{1}{2}\pi x)$, dokładne rozwiązanie. b) Funkcja $\tilde{u}(x) - u(x)$, różnica pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$.

### 2. Przykład 2

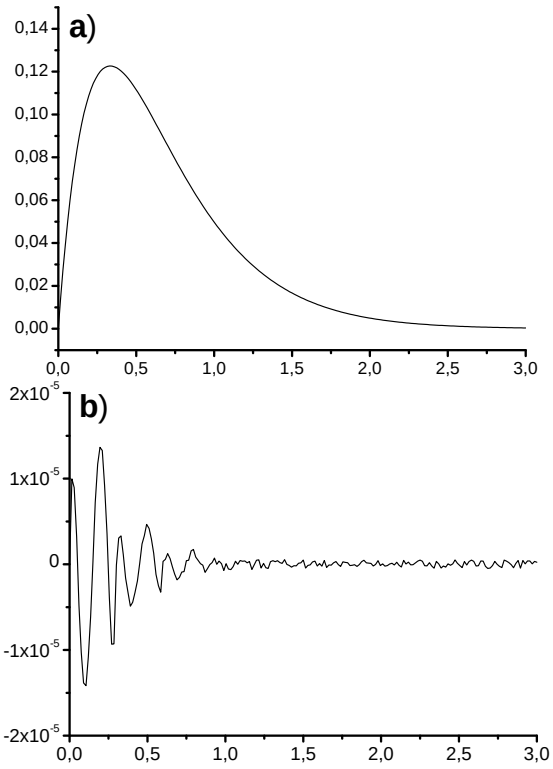
Rozważmy zagadnienie $-u''(x) + 2u(x) = \mathrm{e}^{-x}[(1 + \pi^2)\cos(\pi x) - 2\pi\sin(\pi x)]$ na przedziale $x \in [0, 3.5]$ z warunkami brzegowymi Dirichleta $u(0) = 1$ i $u(3.5) = 0$. Rozwiązaniem dokładnym tego równania jest $u(x) = \mathrm{e}^{-x}\cos(\pi x)$, pokazane na rysunku 3. Przybliżone rozwiązanie $\tilde{u}(x)$ tego równania zostało otrzymane na siatce składającej się z $M = 10$ elementów o równej długości. Stopień każdego elementu jest $p = 5$. Rozmiar macierzy $\mathbf{S}$ (zgodnie ze wzorem (25)) wynosi $N = 49$. Na rysunku 3 podane jest różnica $\tilde{u}(x) - u(x)$. Z rysunku tego wynika, że $|\tilde{u}(x) - u(x)| < 2 \cdot 10^{-6}$, co dowodzi dużej dokładności przybliżonego rozwiązania.




Rysunek 3: Wyniki dla przykładu 2. a) Funkcja $u(x) = \mathrm{e}^{-x}\cos(\pi x)$, dokładne rozwiązanie. b) Funkcja $\tilde{u}(x) - u(x)$, różnica pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$.

### 3. Przykład 3

Rozważmy zagadnienie $-u''(x) + 5u(x) = \mathrm{e}^{-3x}(6 - 4x)$ na przedziale $x \in [0, 3]$ z warunkiem brzegowym Neumanna $u'(0) = 1$ i Dirichleta $u(3) = 3\mathrm{e}^{-9}$. Rozwiązaniem dokładnym tego równania jest $u(x) = x\mathrm{e}^{-3x}$, pokazane na rysunku 4. Przybliżone rozwiązanie $\tilde{u}(x)$ tego równania zostało otrzymane na siatce składającej się z $M = 4$ elementów o równej długości. Stopień każdego elementu jest $p = 4$. Rozmiar macierzy $\mathbf{S}$ (zgodnie ze wzorem (25)) wynosi $N = 15$. Na rysunku 4 podane jest różnica $\tilde{u}(x) - u(x)$. Z rysunku tego wynika, że $|\tilde{u}(x) - u(x)| < 2 \cdot 10^{-5}$, co dowodzi dużej dokładności przybliżonego rozwiązania.



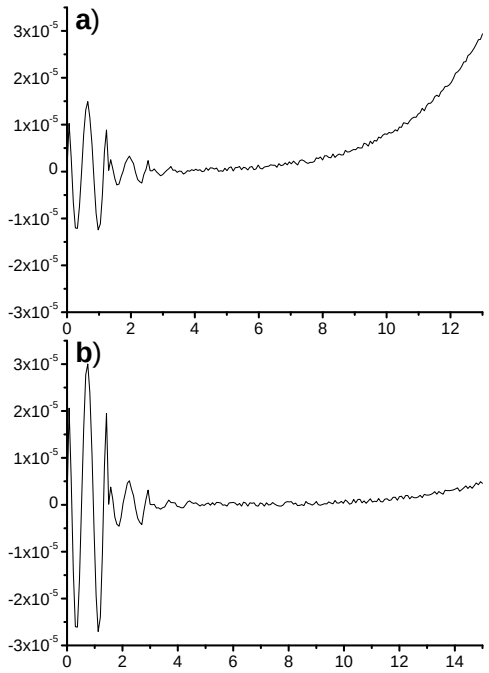

Rysunek 4: Wyniki dla przykładu 3. a) Funkcja $u(x) = x\mathrm{e}^{-3x}$, dokładne rozwiązanie. b) Funkcja $\tilde{u}(x) - u(x)$, różnica pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$.

### 4. Przykład 4

Rozważmy zagadnienie $-\frac{1}{2}u''(x) + \frac{1}{x}u(x) = -\frac{1}{2}(x - 4)\mathrm{e}^{-x}$. Dokładnym rozwiązaniem tego równania, z zerowymi warunkami brzegowym Dirichleta $u(0) = \lim_{x\to\infty} u(x) = 0$, jest funkcja $u(x) = x\mathrm{e}^{-x}$. Przybliżone rozwiązanie tego równania zostało otrzymane na siatce składającej się z $M = 10$ elementów o równej długości. Stopień każdego elementu jest $p = 5$. Rozmiar macierzy $\mathbf{S}$ wynosi $N = 49$.

Nieskończona dziedzina $[0, \infty)$ została zastąpiona skończonym przedziałem. Dodatkowo, z powodu osobliwości dla $x = 0$, za lewy brzeg przedziału wybrano mała liczbę większą od zera. Przybliżone rozwiązanie $\tilde{u}(x)$ otrzymano na dwóch przedziałach $[0, 13]$ oraz $[0, 15]$. W ten sposób warunek brzegowe na prawym końcu przedziału nie jest dokładny. Na rysunku 5 podane jest różnica $\tilde{u}(x) - u(x)$ otrzymana dla obu przedziałów. Z rysunku tego wynika, że $|\tilde{u}(x) - u(x)| < 3 \cdot 10^{-5}$. Ponadto błąd ten systematycznie narasta dla przedziału $[0, 13]$. Narastanie błędu jest konsekwencją przybliżonego warunku

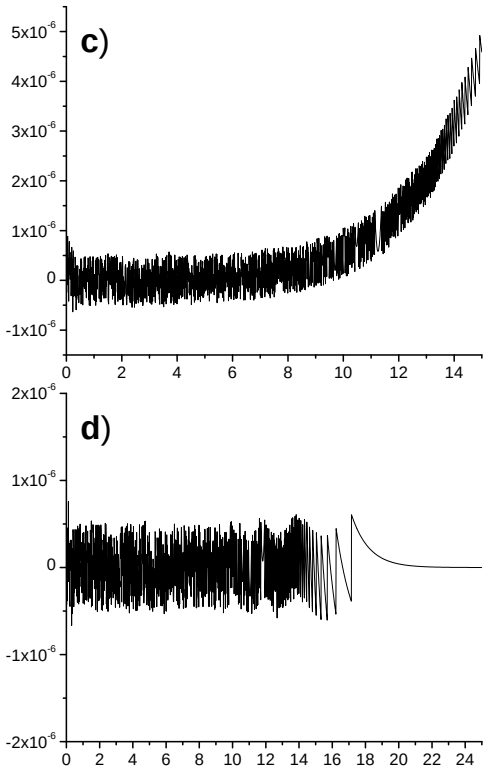
brzegowego w prawym końcu przedziału. Ponieważ, rozwiązanie $u(x)$ zanika wykładniczo, to błąd ten można zmniejszyć rozwiązując równanie na większym przedziale $[0, 15]$, co zostało przedstawione na rysunku 5 b).




Rysunek 5: Różnica $\tilde{u}(x) - u(x)$ pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$ z przykładu 4. a) Rozwiązanie otrzymane na przedziale $[0, 13]$. b) Rozwiązanie otrzymane na przedziale $[0, 15]$.

Na rysunku 6 pokazane jest różnica $\tilde{u}(x) - u(x)$, gdzie $\tilde{u}(x)$ otrzymano za pomocą algorytmu adaptacyjnego. Algortym adaptacyjny kontynuowano dopóki `absMaxCoef` w funkcji `OdeProb::SolveAdapt` mniejszy od $10^{-5}$. Siatka początkowa składała się z dwóch elementów. Wszystkie elementy były stopnia 5. Rozwiązanie na przedziale $[0, 15]$ otrzymano po 12 krokach adaptacyjnych co dało 14 elementów, a to odpowiada 69 stopniom swobody. Rozwiązanie na przedziale $[0, 25]$ otrzymano po 13 krokach adaptacyjnych co dało 15 elementów, a to odpowiada 74 stopniom swobody.

Z rysunku 6 widać, że algorytm adaptacyjny prowadzi do równomiernego rozłożenia błędów, to znaczy wartość błędu na całej dziedzinie jest w w przybliżeniu równa. Widać to szczególnie dobrze na rysunku 6 d). Na rysunku 6 c) widać systematyczny błąd, który jest konsekwencją wstawienia zerowego warunku brzegowego na prawym końcu dziedziny. Ponieważ dokładne rozwiązanie zanika exponencjalnie, to wpływ niepoprawnego warunku brzegowego na przedziale $[0, 25]$ jest znacznie mniejszy i dlatego na rysunku 6 d) tego zjawiska się nie obserwuje. Ponadto, wartość błędu dla algorytmu adaptacyjnego jest o jeden rząd mniejsza niż dla algorytmu z siatką jednorodną.




Rysunek 6: Różnica $\tilde{u}(x) - u(x)$ pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$ z przykładu 4. c) Rozwiązanie adaptacyjne otrzymane na przedziale $[0, 15]$. d) Rozwiązanie adaptacyjne otrzymane na przedziale $[0, 25]$.

### *5. Przykład 5*

Rozważmy równanie

$$-\big(h(x)u'(x)\big)' + g(x)u(x) = f(x)$$

na przedziale $x \in [-5, 10]$, gdzie

$$\begin{aligned} h(x) &= \left[\mathrm{e}^2 + \mathrm{e}^x\right]^2 \\ g(x) &= 1 + \mathrm{e}^{2-x} \\ f(x) &= 1 - \mathrm{e}^{2+x}. \end{aligned}$$

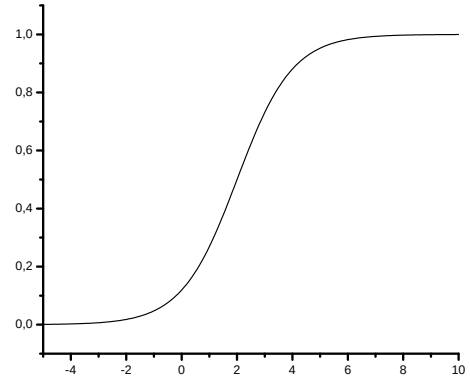
Rozwiązaniem tego równania z warunkami brzegowymi $\lim_{x\to-\infty} = 0$ oraz $\lim_{x\to\infty} = 1$ jest funkcja

$$u(x) = \left[1 + \mathrm{e}^{2-x}\right]^{-1}, \tag{72}$$

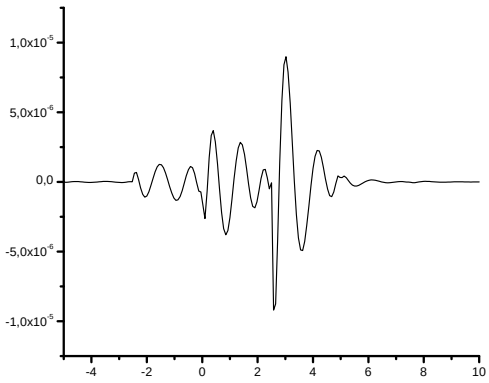
której wykres pokazany jest na Rys. 7. Z dokładnymi warunkami brzegowymi Dirichleta, przybliżone rozwiązanie $\tilde{u}$ tego równania zostało otrzymane na siatce składającej się z $M = 10$ elementów o równej długości. Stopień każdego elementu wynosi $p = 6$. Rozmiar macierzy $\mathbf{S}$ (zgodnie ze wzorem (25)) wynosi $N =$. Na rysunku 7 podane jest różnica $\tilde{u}(x) - u(x)$. Z rysunku tego wynika, że $|\tilde{u}(x) - u(x)| < 1 \cdot 10^{-5}$, co dowodzi dużej dokładności przybliżonego rozwiązania. Z rys. 8 widać,

że błąd jest wyraźnie największy dla $x \approx 0$. Oznacza to, że siatka jednorodna nie jest optymalna.

Na rys. 9 przedstawione jest błąd rozwiązania otrzymanego algorytmem adaptacyjnym. Algorytm adaptacyjny wykorzystywał wielomiany stopnia $p = 6$. Siatka początkowa składała się z dwóch elementów o równej długości. Procedura adaptacyjna była przerwana gdy absMaxCoef $< 10^{-4}$ do wygenerowało 6 elementów. Z rys. 9 wynika, że procedura adaptacyjna prowadzi do bardzo równomiernego rozłożenia błędu.




Rysunek 7: Wykres funkcji zadany równaniem (72).




Rysunek 8: Różnica $\tilde{u}(x) - u(x)$ pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ i dokładnym $u(x)$ z przykładu 5 dla $M = 7$ i $p = 6$.

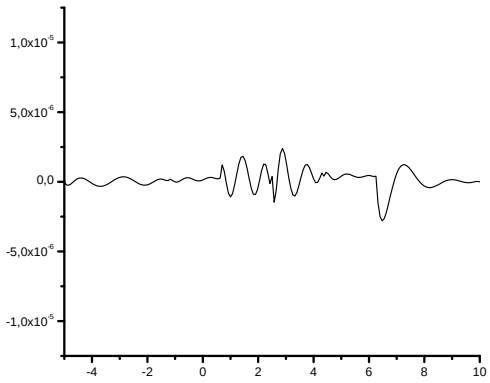
### B. Równania Schrödingera w jednym wymiarze

W tej sekcji analizowane będą wartości własne jednowymiarowego równania Schrödingera

$$\Big( - u_n''(x) + V(x)\Big) u_n(r) = \varepsilon_n u_n(r) \tag{73}$$

określone na dziedzinie $x \in (-\infty, \infty)$ z warunkiem brzegowym

$$\lim_{x \to -\infty} u(x) = \lim_{x \to +\infty} u(x) = 0 \tag{74}$$




Rysunek 9: Różnica $\tilde{u}(x) - u(x)$ pomiędzy rozwiązaniem przybliżonym $\tilde{u}(x)$ otrzymanym algorytmem adaptacyjnym i dokładnym $u(x)$ z przykładu 5 dla $p = 6$. Algorytm adaptacyjny wygenerował 6 elementów.

Tablica IV: Trzy najmniejsze wartości własnych $E_n$ dla potencjalu zadanego wzorem (75). W kolumnie $\Delta_n$ podana została suma ostatanich sześciu współczynników rozwinięcia. W kolumnie $N_n$ znajduje się liczba wygenerowanych elementów, a w kolumnie $M_n$ podany jest wymiar utworzonej przestrzeni. Dla kadego $n$, w drugim wierszu podane są wartości otrzymane przez Eid [10].

| $n$ | $E_n$ | $\Delta_n$ | $N_n$ | $M_n$ |
|---|---|---|---|---|
| 0 | 10.639 788 711 | 2.5[-13] | 24 | 143 |
| | 10.639 788 711 | | | |
| 1 | 38.086 833 459 | 9.4[-13] | 24 | 143 |
| | 38.086 833 459 | | | |
| 2 | 74.681 404 202 | 3.0[-12] | 24 | 143 |
| | 74.681 404 200 | | | |

W pracy Eid [10] analizowane było powyższe równanie dla wielu potencjałów $V(x)$. Tutaj podane zostaną wyniki dla

$$V(x) = 10^3 x^4 + x^2. \tag{75}$$

Równanie Schrödingera z tym potencjałem zostało rozwiązane adaptacyjnie na przedziale $\Omega = [-10, 10]$ z funkcjami Lobatto stopnia 6. Warunkiem stopu dla algorytmu adaptacyjnego było $\delta < 10^{-5}$. Jako siatkę początkową wybrano jednorodną siatkę z czterema elementami.

W tabeli IV podane sa otrzymane wyniki za pomocą biblioteki `RFem1D` wraz z wynikami otrzymanymi przez Eid [10]. Z porównania wynika, że nasze wyniki są podane z dokładności do 9 miejsc po przecinku.

### C. Radialne równania Schrödingera

W tej sekcji będą analizowane numeryczne otrzymane wartości własne dla radialnego równania Schrödingera

$$-\gamma u''_{\ell,n}(r)+\Big(\frac{\ell(\ell+1)}{2r^2}+V(r)\Big)u_{\ell,n}(r)=\varepsilon_{\ell,n}u_{\ell,n}(r) \quad (76)$$

okreslone na $\Omega=[0,\infty)$ z warunkami brzegowymi Dirichleta $u_{\ell,n}(0)=\lim_{r\to\infty}u_{\ell,n}(r)=0$. Dla ustalonego $\ell$, indeks $n=1,2,\ldots$ oznacza numer znalezionej pary $(\varepsilon,u(r))$ wartości własnej i funkcji własnej.

W kolejnych sekcjach analizowany będzie wpływ zastąpienia nieskończonego przedziału $\Omega=[0,\infty)$ skończonym przedziałem $[0,\xi]$.

#### *1. Atomu wodoru*

Dla atomu wodoru $\gamma=1/2$, a potencjał $V(r)$ w równaniu (76) zadany jest wzorem

$$V(r)=-\frac{1}{r} \quad (77)$$

Rozwiązaniem tego zagadnienie są funkcje znane w mechanice kwantowej [20] a wartości własne zadane są wzorem

$$\varepsilon_{\ell,n}=-\frac{1}{2(n+\ell)^2} \qquad \text{dla } n=1,2,\ldots \quad (78)$$

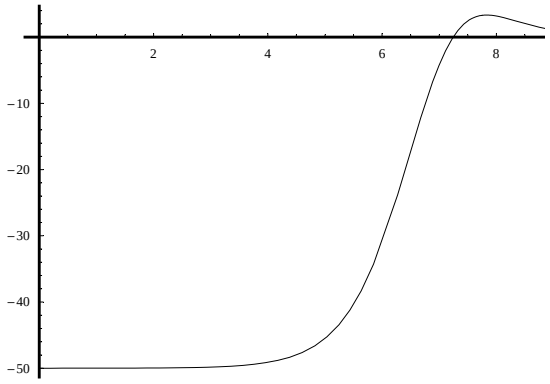
Kilka pierwszych funkcji własnych zostało podanych poniżej

$$\begin{aligned}
u_{0,1}(r) &= r\mathrm{e}^{-r}\\
u_{0,2}(r) &= r\Big(1-\tfrac{r}{2}\Big)\mathrm{e}^{-r/2}\\
u_{0,3}(r) &= r\Big(3-2r+\tfrac{2r^2}{9}\Big)\mathrm{e}^{-r/3}\\
u_{1,1}(r) &= r^2\mathrm{e}^{-r/2}\\
u_{1,2}(r) &= r^2\Big(1-\tfrac{r}{6}\Big)\mathrm{e}^{-r/3}\\
u_{2,1}(r) &= r^3\mathrm{e}^{-r/3}
\end{aligned}$$

Przybliżone rozwiązanie $\tilde{u}(x)$ tego zagadnienie zostało otrzymane na siatce składającej się z $M=13$ elementów o równej długości. Stopień każdego elementu jest $p=5$. Rozmiar macierzy $\mathbf{S}$ wynosi $N=64$. Nieskończona dziedzina $[0,\infty)$ została zastąpiona skończonym przedziałem. Dodatkowo, z powodu osobliwości dla $r=0$, lewy brzeg przedziału zastąpiono liczbą $10^{-6}$.

W tabeli V podane zostały obliczone wartości własne w funkcji długości przedziału $[10^{-6},\xi]$. Zostało policzonych siedem wartości własnych: trzy dla $\ell=0$, dwie dla $\ell=1$, jedna dla $\ell=2$. Każda z tych wartości własnych została obliczona dla pięciu równych długości przedziału $\xi$. Z podanych danych widać, że na dokładność obliczeń kluczowy wpływ ma poprawnie ustalone warunki brzegowe. Dla dostatecznie dużego $\xi$ wartości własne różnią sie do wyników analitycznych o mniej niż $10^{-5}$ hartree, co w większości zastosowań chemicznych jest wystarczające.

#### *2. Potencjał Woodsa-Saxona, adaptacyjnie*




Rysunek 10: Potencjał Woodsa-Saxona zadany wzorem (79) dla $u_0=-50$, $u_1=-u_0/a$, $a=0.6$, $r_0=7$.

Potencjał Woodsa-Saxona [16, 17, 26, 28, 29] $\gamma=1$ zadany jest wzorem

$$V(r)=\frac{u_0}{1+t}+\frac{u_1 t}{(1+t)^2}, \qquad t=\exp\Big(\frac{r-r_0}{a}\Big) \quad (79)$$

gdzie $u_0,u_1,a,r_0\in\mathbb{R}$ oraz $r_0>0$. Potencjał ten jest skończony dla $r\geq 0$. W szczególności dla $r=0$ mamy

$$V(0)=\frac{w\big(u_1+u_0(1+w)\big)}{(1+w)^2}, \qquad w=\exp\Big(\frac{r_0}{a}\Big). \quad (80)$$

Ponadto, z powodu funkcji exp() w granicy zachodzi

$$\lim_{r\to\infty}V(r)=0. \quad (81)$$

Wykres tej funkcji dla wybranych stałych pokazany jest na rys. 10. Ostatnio została opublikowana praca [4], gdzie zostały wyprowadzone analityczne wzory na wartości własne dla $\ell=0$.

Radialne równanie Schrödingera z tym potencjałem zostało rozwiązane adaptacyjnie na przedziale $\Omega=[0,20]$ z funkcjami Lobatto stopnia 6. Warunkiem stopu dla algorytmu adaptacyjnego było $\delta<10^{-5}$. Jako siatkę początkową wybrano jednorodną siatkę z czterema elementami.

W tabeli VI zostało podanych pięć najmniejsze wartości własnych dla $\ell=0,1,2$. W kolumnie $\Delta_n$ znajduje się suma sześciu ostatnich współczynników rozwinięcia na funkcje Lobatto. Z podanych danych widać, że $\Delta_n$ jest największe dla $n=5$, co jest zgodne z ogólnym przekonaniem, że funkcje własne dla dużych $n$ są bardziej rozciągłe niż dla małych $n$. Dodatkowo, wszystkie wartości $\Delta_n$ są mniejsze od $2\times10^{-8}$, co oznacza, że przedział $\Omega=[0,20]$ odpowiednio reprezentuje nieskończony przedział $[0,\infty)$.

Algorytm adaptacyjny dla $\ell=0,1,2$ wygenerował $N=24,25,23$ elementów, wymiar przestrzeni wynosił

Tablica V: Kilka pierwszych wartości własnych atomu wodoru. Wartości obliczone na przedziale $[10^{-6}, \xi]$, który został podzielony na 13 elementów, piątego stopnia każdy. Długość $\xi$ w borach, energia własna w hartree.

| | $\ell = 0$ | | | $\ell = 1$ | | $\ell = 2$ |
|---|---|---|---|---|---|---|
| $\xi$ | $n = 1$ | $n = 2$ | $n = 3$ | $n = 1$ | $n = 2$ | $n = 1$ |
| 10 | -0.499997 | -0.112806 | +0.091423 | -0.118860 | +0.049191 | -0.007093 |
| 15 | -0.499998 | -0.124499 | -0.026875 | -0.124771 | -0.035044 | -0.046643 |
| 20 | -0.499998 | -0.124987 | -0.049918 | -0.124995 | -0.051611 | -0.053968 |
| 30 | -0.499993 | -0.125000 | -0.055424 | -0.125000 | -0.055471 | -0.055528 |
| 40 | -0.499946 | -0.124999 | -0.055554 | -0.125000 | -0.055555 | -0.055555 |

Tablica VI: Pięć najmniejszych wartości własnych $E_n$ dla potencjalu Woodsa-Saxona, wzór (79), dla $\ell = 0, 1, 2$. W kolumnie $\Delta_n$ podana została suma ostatanich sześciu współczynników rozwinięcia. Dla $\ell = 0, 1, 2$ liczba wygenerowanych elementów wynosiła $N = 24, 25, 23$, a wymiar utworzonej przestrzeni wynosił $M = 143, 149, 137$, odpowiednio.

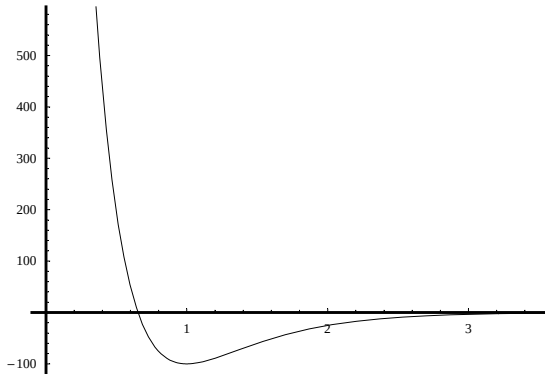
| | $\ell = 0$ | | $\ell = 1$ | | $\ell = 2$ | |
|---|---|---|---|---|---|---|
| $n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ |
| 0 | -49.457 788 723 | 8.8[-11] | -48.951 731 624 | 1.6[-10] | -48.349 481 052 | 2.6[-10] |
| 1 | -48.148 430 420 | 3.5[-10] | -47.341 691 702 | 5.9[-10] | -46.461 659 232 | 3.2[-10] |
| 2 | -46.290 753 955 | 1.1[-09] | -45.237 176 986 | 1.8[-09] | -44.121 537 377 | 2.7[-09] |
| 3 | -43.968 318 432 | 3.0[-09] | -42.698 002 625 | 4.7[-09] | -41.373 224 427 | 6.9[-09] |
| 4 | -41.232 607 772 | 7.6[-09] | -39.767 208 069 | 1.1[-08] | -38.253 426 536 | 1.6[-08] |

Tablica VII: Siedem najmniejszych (o parzystym indeksie $n$) wartości własnych $E_n$ dla potencjalu Woodsa-Saxon-a, wzór (79), dla $\ell = 0, 1, 2$. Wartości te zostały wyznaczone z dokładnością do 9 cyfr po przecinku w pracy [28].

| | $\ell = 0$ | $\ell = 1$ | $\ell = 2$ |
|---|---|---|---|
| 0 | -49.457 788 728 | -48.951 731 623 | -48.349 481 052 |
| 2 | -46.290 753 954 | -45.237 176 986 | -44.121 537 377 |
| 4 | -41.232 607 772 | -39.767 208 069 | -38.253 426 539 |
| 6 | -34.672 313 205 | -32.868 392 986 | -31.026 820 921 |
| 8 | -26.873 448 916 | -24.794 185 466 | -22.689 041 510 |
| 10 | -18.094 688 282 | -15.812 724 871 | -13.522 303 352 |
| 12 | -8.676 081 670 | -6.308 097 192 | -3.972 491 432 |

$M = 143, 149, 137$, a liczba wykonanych iteracji wynosiła $P = 7, 8, 7$, odpowiednio.

Dla porównania w tabeli VII zostały podane energie własne obliczone z dokładnością do 9 miejsc po przecinku [28]. Z porównania wartości zamieszczonych w tabeli VI i tablei VII wynika, że dla potencjału Woodsa-Saxona wartości własne otrzymane metodą zaimplementowaną w bibliotece `RFem1D` zgadzają się z wartościami referencyjnymi z dokładnością do 8 cyfr po przecinku.




Rysunek 11: Potencjał Morsea zadany wzorem (82) dla $D = 100$, $\alpha = 2$. Dla tych wartości $V(0) = 3982$.

### *3. Potencjał Morsea, adaptacyjnie*

Potencjał Morsea [28] $\gamma = 1$ zadany jest wzorem

$$V(r) = D(\mathrm{e}^{-2\alpha(r-1)} - 2\mathrm{e}^{-\alpha(r-1)}) \tag{82}$$

gdzie $D, \alpha > 0$. Potencjał ten jest skończony dla $r \geq 0$. W szczególności dla $r = 0$ mamy

$$V(0) = D(\mathrm{e}^{2\alpha} - 2\mathrm{e}^{\alpha}). \tag{83}$$

Ponadto, z powodu funkcji exp() w granicy zachodzi

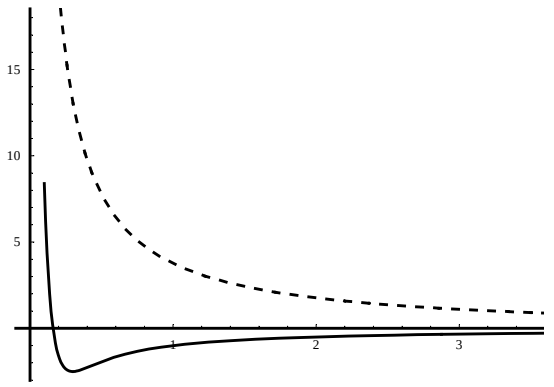
$$\lim_{r\to\infty} V(r) = 0. \tag{84}$$

Wykres tej funkcji dla wybranych stałych pokazany jest na rys. 11. Dla tego potencjału dla $\ell = 0$ wartości własne zadane są analitycznie [13].

Radialne równanie Schrödingera z tym potencjałem zostało rozwiązane adaptacyjnie na przedziale $\Omega = [0, 20]$ z funkcjami Lobatto stopnia 6. Warunkiem stopu dla algorytmu adaptacyjnego było $\delta < 10^{-4}$. Jako siatkę początkową wybrano jednorodną siatkę z czterema elementami.

Algorytm adaptacyjny dla $\ell = 0, 1, 2$ wygenerował $N = 17, 19, 18$ elementów, wymiar przestrzeni wynosił $M = 101, 113, 107$, a liczba wykonanych iteracji wynosiła $P = 7, 7, 7$, odpowiednio.

*4. Potencjał Hellmanna, adaptacyjnie*




Rysunek 12: Potencjał Hellmanna zadany wzorem (85). Dla $A = 1$, $B = 5$, $C = 0.05$ potencjał jest większy od zera i zaznaczony jest na wykresie linią przerywaną. Dla $A = 1$, $B = 5$, $C = 10$ potencjał ma minimum lokalne i zaznaczony jest na wykresie linią ciągłą. Dla obu przypadków $\lim_{r\to 0} V(0) = \infty$.

Potencjał Hellmanna [2, 15, 28] dla $\gamma = 1/2$ zadany jest wzorem

$$V(r) = -\frac{A}{r} + \frac{B\mathrm{e}^{-Cr}}{r} \tag{85}$$

gdzie $A, B, C \geq 0$. Parametry $A, B$ określają potencjał Coulomba oraz Yukawy, a parametr $C$ jest współczynnikiem ekranowania. Wartość potencjału dla $r = 0$ oraz $r \to \infty$ zależy od parametrów $A, B, C$. Wykres tej funkcji dla przykładowych wartości parametrów $A, B, C$ pokazany jest na rys. 12. Potencjał ten był szczegółowo analizowany przez Adamowskiego [2].

Radialne równanie Schrödingera z tym potencjałem zostało rozwiązane adaptacyjnie na przedziale $\Omega = [0, 20]$ z funkcjami Lobatto stopnia 6. Warunkiem stopu dla algorytmu adaptacyjnego było $\delta < 10^{-4}$. Jako siatkę początkową wybrano jednorodną siatkę z czterema elementami.

Dla $C = 10$ algorytm adaptacyjny dla $\ell = 0, 1, 2$ wygenerował $N = 22, 22, 17$ elementów, wymiar przestrzeni wynosił $M = 131, 131, 101$, a liczba wykonanych iteracji wynosiła $P = 10, 9, 7$, odpowiednio. Dla $\ell = 0$ równanie było rozwiązywane na przedziale $\Omega = [0, 300]$ dla $\ell = 1$ na przedziale $\Omega = [0, 500]$ a dla $\ell = 2$ na przedziale $\Omega = [0, 600]$.

Dla $C = 0.05$ algorytm adaptacyjny dla $\ell = 0, 1, 2$ wygenerował $N = 22, 20, 17$ elementów, wymiar przestrzeni wynosił $M = 131, 119, 101$, a liczba wykonanych iteracji wynosiła $P = 9, 7, 5$, odpowiednio. Dla $\ell = 0, 1, 2$ równanie było rozwiązywane na przedziale $\Omega = [0, 1000]$.

Wartości własne podane w tabeli X zgadzają się z wartościami podanymi w pracy Vanden berghe [28] tylko dla $\ell = 0$. Dla $\ell = 1, 2$ wartości własne otrzymane za pomocą programu `RFem1D` są inne niż w cytowanej pracy. Podejrzewam, że jest to problem z przeskalowaniem (wyrażeniem w innych jednostkach) radialnego równania Schrödingera.

### D. Równanie Kohna-Shama dla atomu

Zastosowanie teorii funkcjonału gęstości (DFT) dla sferycznie symetrycznego atomu [22, 23] prowadzi do zagadnienia własnego

$$\Big[ -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{\ell(\ell+1)}{2r^2} + V(r)\Big] R_{n,\ell}(r) = \epsilon_{n,\ell} R_{n,\ell}(r) \tag{86}$$

Formalnie, w powyższym wzorze funkcja $R_{n,\ell}$ jest określona na przedziale $[0, \infty)$. Aby móc rozwiązać to równanie, nieskończony przedział $[0, \infty)$ zamieniany jest na skończony $[0, W)$. Dodatkowo nakłada się na funkcję $R_{n,\ell}$ warunek brzegowe Dirichleta

$$R_{n,\ell}(0) = R_{n,\ell}(W) = 0. \tag{87}$$

We wzorze (86) poboczna liczba kwantowa przyjmuje wartości całkowite $\ell = 0, 1, 2, \ldots$ i zależy od rozpatrywanego atomu. Ponadto funkcja $V(r)$ nazywana jest potencjałem efektywnym i wyznaczana jest jako suma

$$V(r) = -\frac{Z}{r} + V_{\mathrm{h}}(r) + V_{\mathrm{xc}}(r) \tag{88}$$

gdzie $Z$ jest ładunkiem jądra atomowego, $V_{\mathrm{h}}(r)$ jest potencjałem elektrostatycznym generowanym przez chmurę elektronów o gęstości $\varrho(r)$, natomiast $V_{\mathrm{xc}}(r)$ jest potencjałem korelacji-wymiany. Potencjał $V_{\mathrm{xc}}(r)$ jest funkcja gęstości elektronowej $\varrho(r)$ i wyznaczany jest na podstawie zaproponowanych w literaturze przybliżeń [8, 19]. Gęstość elektronową wyznacza się ze wzoru

$$\varrho(r) = \frac{1}{2\pi}\sum_{n,\ell}(2\ell+1) f_{n,\ell}\frac{R^2_{n,\ell}(r)}{r^2} \tag{89}$$

gdzie liczby $0 \leq f_\mu \leq 1$ oznaczają współczynniki obsadzeń stanów elektronowych i sumują się do liczby elektronów w atomie

$$N = \sum_\mu f_\mu \tag{90}$$

Tablica VIII: Pięć najmniejszych wartości własnych $E_n$ dla potencjalu Morsea, wzór (82), dla $\ell = 0, 1, 2$. W kolumnie $\Delta_n$ podana została suma ostatanich sześciu współczynników rozwinięcia. Dla $\ell = 0, 1, 2$ liczba wygenerowanych elementów wynosiła $N = 17, 19, 18$, a wymiar utworzonej przestrzeni wynosił $M = 101, 113, 107$, odpowiednio.

| | $\ell = 0$ | | $\ell = 1$ | | $\ell = 2$ | |
|---|---|---|---|---|---|---|
| $n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ |
| 0 | -80.999 999 994 | 1.4[-14] | -79.161 799 962 | 3.4[-17] | -75.544 697 966 | 2.2[-14] |
| 1 | -48.999 999 981 | 1.9[-12] | -47.482 803 564 | 1.2[-15] | -44.508 123 795 | 3.9[-12] |
| 2 | -24.999 999 992 | 7.2[-10] | -23.827 466 483 | 4.0[-14] | -21.543 363 512 | 1.4[-09] |
| 3 | -8.999 999 925 | 1.5[-08] | -8.204 425 589 | 5.4[-12] | -6.679 971 813 | 9.8[-09] |
| 4 | -0.999 999 920 | 2.0[-05] | -0.648 872 424 | 1.3[-04] | -0.062 471 854 | 1.3[-02] |

Tablica IX: Pięć najmniejszych wartości własnych $E_n$ dla potencjalu Morse, wzór (82), dla $\ell = 0, 1, 2$. Wartości własne dla $\ell = 1, 2$ zostały wyznaczone z dokładnością do 9 cyfr po przecinku w pracy [28]. Wartości dla $\ell = 0$ są dokładne.

| | $\ell = 0$ | $\ell = 1$ | $\ell = 2$ |
|---|---|---|---|
| 0 | -81.0 | -79.161 799 963 | -75.544 697 966 |
| 1 | -49.0 | -47.482 803 567 | -44.508 123 798 |
| 2 | -25.0 | -23.827 466 489 | -21.543 363 518 |
| 3 | -9.0 | -8.204 425 593 | -6.679 971 857 |
| 4 | -1.0 | -0.648 872 427 | - |

Potencjał elektrostatyczny $V_\mathrm{h}(r)$ otrzymuje się rozwiązując równanie Poissona. Dla układu sferycznie symetrycznego równanie Poissona redukuje się do

$$\frac{1}{r}\frac{\mathrm{d}^2}{\mathrm{d}r^2}\big[rV_\mathrm{h}(r)\big] = -4\pi\varrho(r) \tag{91}$$

Dla powyższego równania można podać analityczne rozwiązanie [9]:

$$V_\mathrm{h}(r) = 4\pi\Big[\frac{1}{r}\int_0^r \varrho(t)t^2\mathrm{d}t + \int_r^\infty t\varrho(t)\mathrm{d}t\Big]. \tag{92}$$

Jeżeli wprowadzi się funkcję pomocniczą $U_\mathrm{h}(r) = rV_\mathrm{h}(r)$, to równanie (91) sprowadza się do postaci

$$\frac{\mathrm{d}^2U_\mathrm{h}(r)}{\mathrm{d}r^2} = -4\pi r\varrho(r) \tag{93}$$

z warunkiem brzegowymi Dirichleta dla $r = 0$

$$U_\mathrm{h}(0) = 0 \tag{94}$$

Warunek brzegowy dla $r = W$ oblicza się ze wzoru (92)

$$U_\mathrm{h}(W) = 4\pi\int_0^W t^2\varrho(t)\mathrm{d}t = E \tag{95}$$

gdzie zostało wykorzystane całkowanie po kuli o promieniu $W$ we współrzędnych sferycznych.

Powyższe równania stanowią nieliniowy układ równań różniczkowo-całkowych połączony z zagadnieniem własnym. Do rozwiązania tych równań stosuje się metodę iteracyjną:

1. Wybież początkową gęstość elektronową $\varrho(r)$ tak aby spełniała warunek $4\pi\int_0^W r^2\varrho(r)\mathrm{d}r = E$ oraz była większa od zera: $\varrho(r) > 0$ dla $r \in [0, W]$.

2. Znając $\varrho(r)$ oblicz $V_\mathrm{h}(r)$ rozwiązując równanie (93). W zasadzie z równania (93) otrzymuje się się $U_\mathrm{h}(r)$. Aby uzyskać szukany wynik wystarczy wykonać jedno dzielenie $V_\mathrm{h}(r) = U_\mathrm{h}(r)/r$.

3. Oblicz potencjał oddziaływania $V(r)$ ze wzoru (88). Aby obliczyć $V(r)$ należy zastosować przybliżenie na korelacje-wymianę. W tej pracy potencjał wymiany przybliżany jest wyrażeniem Slatera natomiast potencjał korelacji obliczany jest ze wzoru podanego przez Vosko, Wilk, Nusair (VWN) [30, 31].

4. Znajdź kilka najmniejszych wartości własnych zagadnienia (86) dla stosownych $\ell$.

5. Oblicz współczynniki obsadzeń $f_\mu$ i nową gęstość elektronową $\varrho_\mathrm{new}$.

6. Oblicz enrgię całkowią $E_\mathrm{new}$ i jej składowe.

7. Jeżeli $|E_\mathrm{new} - E| < \epsilon$ oraz $||\varrho_\mathrm{new} - \varrho|| < \delta$, to koniec. W przeciwnym przypadku $\varrho = \varrho_\mathrm{new}$ i ponów powyższe kroki zaczynając od punktu 2.

W kolejnych sekcjach zostanie opisana implementacja tego problemu.

### *1. Implementacja*

### *2. Wyniki numeryczne*

Tablica X: Pięć najmniejszych wartości własnych $E_n$ dla potencjalu Hellmanna, wzór (85), dla $A = 1$, $B = 5$, $C = 0.05$ oraz $C = 10$ dla $\ell = 0, 1, 2$. W kolumnie $\Delta_n$ podana została suma ostatanich sześciu współczynników rozwinięcia.

| | $\ell = 0$ | | $\ell = 1$ | | $\ell = 2$ | |
|---|---|---|---|---|---|---|
| $n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ | $E_n$ | $\Delta_n$ |
| | | | $C = 0.05$ | | | |
| 0 | -0.010 994 420 | 1.3[-18] | -0.010 520 442 | 3.3[-18] | -0.009 652 661 | 2.2[-17] |
| 1 | -0.008 589 028 | 1.0[-15] | -0.008 228 548 | 2.4[-15] | -0.007 571 358 | 1.1[-14] |
| 2 | -0.006 817 379 | 1.8[-13] | -0.006 544 716 | 3.3[-13] | -0.006 048 648 | 1.0[-12] |
| 3 | -0.005 509 422 | 8.1[-12] | -0.005 302 280 | 1.3[-11] | -0.004 925 306 | 3.9[-11] |
| 4 | -0.004 531 786 | 3.2[-10] | -0.004 372 556 | 6.0[-10] | -0.004 082 210 | 2.1[-09] |
| | | | $C = 10$ | | | |
| 0 | -0.421 975 160 | 6.6[-25] | -0.076 200 395 | 2.1[-23] | -0.031 245 000 | 1.4[-17] |
| 1 | -0.114 644 973 | 1.7[-18] | -0.039 417 168 | 8.0[-21] | -0.020 000 000 | 5.9[-15] |
| 2 | -0.052 422 683 | 2.8[-15] | -0.240 292 157 | 1.8[-16] | -0.013 888 889 | 1.0[-13] |
| 3 | -0.029 914 218 | 1.3[-12] | -0.016 164 921 | 1.6[-13] | -0.010 204 082 | 5.5[-12] |
| 4 | -0.019 311 671 | 1.1[-09] | -0.011 613 230 | 7.2[-11] | -0.007 812 500 | 1.0[-09] |


[1] Abramowitz, M. and I. A. Stegun: 1972, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover Publications. `http://www.math.sfu.ca/~cbm/aands/toc.htm`.

[2] Adamowski, J.: 1985, 'Bound eigenstates for the superposition of the Coulomb and the Yukawa potentials'. *Phys. Rev. A* **31**, 43–50.

[3] Anderson, E., Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen: 1999, *The LAPACK Users' Guide*. SIAM. `http://www.netlib.org/lapack/`.

[4] Ayse, B., B. Cuneyt, and S. Ramazan: 2006, 'Eigenvalues and eigenfunctions of Woods-Saxon potential in PT-symmetric quantum mechanics'. *Mod. Phys. Lett. A* **21**, 2087–2097.

[5] Babuška, I. and T. Strouboulis: 2001, *The Finite Element Method and Its Reliability*. Oxford: Oxford University Press.

[6] Brenner, S. C. and L. R. Scott: 1996, *The Mathematical Theory of Finite Element Methods*. New York: Springer.

[7] Ciarlet, P. G.: 1978, *The Finite Element Method for Elliptic Problems*. Holland: North-Holland Publishing Company.

[8] Cramer, C. J.: 2004, *Essentials of Computational Chemistry*. Wiltshire: Wiley.

[9] Delley, B.: 1990, 'An all-electron numerical method for solving the local density functional for polyatomic molecules'. *J. Chem. Phys.* **92**, 508–517.

[10] Eid, R.: 1999, 'Higher order finite element solution of the one-dimensional Schrödinger equation'. *Int. J. Quant. Chem.* **71**, 147–152.

[11] Fichtenholz, G. M.: 1990, *Differential- und Integralrechnung*. Berlin: Harri.

[12] Fletcher, C. A. J.: 1984, *Computational Galerkin Methods*. Berlin: Springer.

[13] Flügge, S.: 1974, *Practical Quantum Mechanics*. New York: Springer.

[14] Golub, G. H. and C. F. Van Loan: 1996, *Matrix Computation*. Baltimore: The Johns Hopkins University Press.

[15] Hellmann, H.: 1935, 'A new approximation method in the problem of many electrons'. *J. Chem. Phys.* **3**, 61–61.

[16] Ixaru, L. and M. Rizea: 1980, 'A Numerov-like scheme for the numerical solution of the Schrödinger equation in the deep continuum spectrum of energies'. *Comp. Phys. Commun.* **19**, 23–27.

[17] Ixaru, L. and M. Rizea: 1985, 'Comparison of some four-step methods for the numerical solution of the Schrödinger equation'. *Comp. Phys. Comm.* **38**, 329–337.

[18] Johnson, C.: 1987, *Numerical solution of partial differential equations by the finite element method*. Cambridge: Cambridge University Press.

[19] Koch, W. and M. C. Holthausen: 2000, *A Chemist's Guide to Density Functional Theory*. New York: Wiley.

[20] Liboff, R. L.: 1987, *Introductory Quantum Mechanics*. New York: Addison Wesley.

[21] Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: 1990, *Numerical Recipes in C*. Cambridge: Cambridge University Press.

[22] Romanowski, Z.: 2007, 'Numerical solution of Kohn-Sham equation for atom'. *Acta Phys. Pol. B* **38**, 3263–3285.

[23] Romanowski, Z.: 2008, 'B-Spline finite element solution of Kohn-Sham equation for atom'. *Modelling Simul. Mater. Sci. Eng.* **16**, 015003.

[24] Saad, Y.: 2003, *Iterative Methods for Sparse Linear Systems*. New York: SIAM. `http://www-users.cs.umn.edu/~saad/books.html`.

[25] Saad, Y.: 2007, *Sparskit. A basic tool-kit for sparse matrix computations*. University of Minnesota.

`http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html`.

[26] Simosa, T. and P. Williams: 1999, 'On finite difference methods for the solution of the Schrödinger equation'. *Computers and Chemistry* **23**, 513–554.

[27] Stoer, J. and R. Bulirsch: 2004, *Introduction to Numerical Analysis*. New York: Springer.

[28] Vanden Berghe, G., V. Fack, and H. E. de Meyer: 1989, 'Numerical methods for solving radial Schrödinger equations'. *J. Comput. Appl. Math.* **28**, 391–401.

[29] Vigo-Aguiar, J. and T. E. Simos: 2005, 'Review of multistep methods for the numerical solution of the radial Schrödinger equation'. *Int. J. Quant. Chem.* **103**, 278–290.

[30] Vosko, S. and L. Wilk: 1980, 'Influence of an improved local-spin-density correlation-energy functional on the cohesive energy of alkali metals'. *Phys. Rev. B* **22**, 3812–3815.

[31] Vosko, S. H., L. Wilk, and M. Nusair: 1980, 'Accurate spin dependent electron liquid correlation energies for local spin density calculations: A critical analysis'. *Can. J. Phys.* **58**, 1200–1211.

[32] Šolín, P.: 2006, *Partial Differential Equations and the Finite Element Method*. Hoboken: Wiley Interscience.

[33] Šolín, P., K. Segeth, and I. Doležel: 2004, *High-Order Finite Element Method*. London: CHAPMAN & HALL/CRC.

[34] Wait, R. and A. R. Mitchell: 1986, *Finite Element Analysis and Applications*. New York: Wiley.