Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów

ALOPP

autor	**Filip Kudła**
prowadzący	**dr. inż. Wojciech Łabaj**
rok akademicki	**2022/2023**
kierunek	**informatyka**
rodzaj studiów	**SSI**
semestr	**2**
termin laboratorium	**środa, 13:30 - 15:00**
sekcja	**62**
termin oddania sprawozdania	**28.07.23r.**

1. Treść zadania

Napisać program konsolowy analizujący liniowe obwody prądu przemiennego. Obwody mogą się składać z elementów takich jak np.:

- źródła SEM (siły elektromotoryczne)
- źródła SPM (siły prądomotoryczne)
- rezystory
- kondensatory
- cewki

Dla każdego elementu oblicza prąd, napięcie, wydzieloną moc oraz częstotliwość rezonansową. Informacje są odczytywane i zapisywane do plików tekstowych.
Program jest uruchamiany z linii poleceń z wykorzystaniem przełączników:

- -i (dalej podawany jest plik wejściowy)
- -o (dalej podawany jest plik wyjściowy)

2. Analiza zadania

Sposób łączenia elementów jest dowolny, przy czym węzeł początkowy oznacza kierunek strzałki (do węzła) - dla źródeł napięcia lub prądu. Za pomocą tychże węzłów i zależności między nimi można łatwo skorzystać z metody potencjałów węzłowych i później policzyć układy równań za pomocą metody eliminacji Gaussa.

2.1 Struktury danych

Program korzysta z mapy i wektorów. Wektory wykorzystywane są m.in. do przechowywania wskaźników na elementy obwodu, natomiast mapy wykorzystywane m.in. do zamiany węzłów z pliku wejściowego na nowe, aby łatwiej użyć metody Coltriego (metody potencjałów węzłowych) do utworzenia macierzy admitancji obwodu. Program używa mapy także do przechowywania potencjału w danym węźle obwodu, gdzie kluczem jest numer węzła, co znacząco ułatwia dalsze obliczanie prądów i napięć.

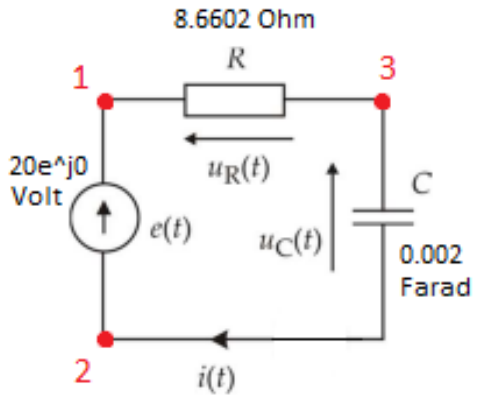
2.2 Algorytmy

Program używa metody eliminacji Gaussa-Jordana-Crouta do obliczenia potencjału w węzłach bazując na admitancji dodanej do macierzy metodą Coltriego. Metoda ta sprowadza macierz do postaci trójkątnej uzyskując zera nad i pod przekątną macierzy, oraz uzyskuje admitancje w postaci {1,0} na przekątnej macierzy, co pozwala w łatwy sposób uzyskać potencjały w węzłach w postaci ich impedancji.

3. Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: -i dla pliku wejściowego i -o dla pliku wyjściowego), np.

./alopp -i plik_wejscia.txt -o plik_wyjscia.txt



<typ><wezel_pocz><wezel_konc><wartosc>
W przypadku zrodel dodatkowo na koncu:
<przesuniecie_fazowe><czestotliwosc>
E 1 2 20 0 15.915494
R 1 3 8.6602
c 3 2 0.002

Uruchomienie programu bez parametru spowoduje wyświetlenie krótkiej pomocy jak użyć programu.

4. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.
W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (analizy obwodu).

4.1 Ogólna struktura programu

W funkcji głównej main() jest tworzony wektor wskaźników (shared_ptr), na którym wykonywane są wszystkie operacje. Jeśli wektor jest pusty, program od razu zakończa swoje zadanie. W przeciwnym wypadku wywoływana jest funkcja wektor_wezlow(), która tworzy algebraiczny wektor (set) węzłów z obwodu. Następnie wywoływana jest funkcja dodaj_rez_obok_sem(), która dodaje do wektora elementy ‘wirtualny’ węzeł o wartości -1, aby następnie gałąź ze źródłami elektromotorycznymi traktować jako gałąź o impedancji {1,0}. Następnie wyznaczana jest częstotliwość pracy obwodu bazując na dodanych źródłach do wektora elementy, która jest potrzebna do następnie wyznaczanej impedancji dla każdego elementu. Posiadając informację o impedancji elementu, można łatwo wyznaczyć ich impedancję oraz częstotliwość rezonansową (w przypadku kondensatora lub cewki). W dalszym etapie

tworzona jest macierz (wektor wektorów liczb zespolonych), do której dodawane są admitancje elementów. Po dodaniu do macierzy, obliczana jest impedancja każdego potencjału węzłowego korzystając z metody eliminacji Gaussa. W dalszym ciągu obliczany jest prąd, odłożone napięcie na każdym elemencie, oraz moc czynna i bierna dla każdego elementu. Ostatnia funkcja programu to funkcja wypisująca obliczone charakterystyki elementów do pliku tekstowego podanego przy uruchamianiu programu z wiersza poleceń. W tejże funkcji obliczany jest także bilans mocy (czynnej oraz biernej) i wypisywany do pliku tekstowego.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji oraz klas znajduje się w poniżej dołączonym Doxygenie

5. Testowanie

Program został przetestowany na różnego rodzaju przykładach. Zostały przetestowane różne obwody z różna kolejnością łączenia elementów. Dla każdego obliczane są poprawne wartości oraz bilans mocy zawsze jest równy zero, co jednoznacznie wskazuje, że wyliczone wartości (mocy pobranych dla odbiorników i mocy oddanych dla źródeł) są wyliczone poprawnie. Dla typu plików wejściowych ze źródłami posiadające różną częstotliwość, wczytywane jest pierwsze z pliku tekstowego i program kontynuuje swoje działanie dla wczytanych elementów, aż do źródła o innej częstotliwości. Wypisywane w pliku wejściowym wartości typu 1.4250e-14 to błędy numeryczne typu double – te wartości są równe zero. Program nie jest w pełni odporny na błędy, tzn. przy niepodaniu w pliku tekstowym wartości przesunięcia oraz częstotliwości dla źródeł kończy swe działanie bez wyniku. Przy podaniu wartości innych niż liczbowe, również kończy swoje działanie bez stosowanego komunikatu o błędzie. Program nie został sprawdzony pod kątem poprawności danych dla bardzo dużych plików wejściowych. Program został sprawdzony pod kątem wycieków pamięci.

6. Wnioski

Program ALOPP był programem skomplikowanym od strony elektrotechnicznej. Trzeba było rozważyć dużo różnych przypadków. Liczba klas pomimo tego, że jest niewielka, pozwala na łatwe zarządzanie, obliczanie i zastosowanie odpowiedniej metodyki przy analizie obwodów AC. Szczególnie trudne w zaimplementowaniu było zastosowanie metody potencjałów węzłowych oraz metody eliminacji Gaussa dla liczb zespolonych. Mimo korzystania z biblioteki <complex> implementacja tych funkcji w poprawny sposób zajęła wiele czasu. Podczas pisania projektu nauczyłem się we właściwy sposób korzystać z paradygmatu programowania obiektowego. Program jest napisany przeze mnie w sposób optymalny, starałem się tworzyć jak najmniej nowych zmiennych i zapełniać pamięć. Program korzysta z inteligentnych wskaźników pozwalających w efektywny sposób odnosić się do obiektów klasy pochodnych i obliczać dla nich różne wartości i właściwości.

ALOPP

v1.0

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 4

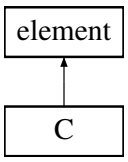
Dokumentacja klas

4.1 Dokumentacja klasy C

Klasa reprezentuje kondensatory, jest klasa pochodna elementu.

`#include <klasy.h>`

Diagram dziedziczenia dla C



Metody publiczne

- C (char typ, std::pair< int, int > miejsce, double wartosc)

  Konstruktor tworzacy kondensator.
- std::complex< double > wyznacz_Z (double &wartosc, double &freq)

  Funkcja wyznaczajaca impedancje kondensatora.
- std::complex< double > wyznacz_Y ()

  Funkcja wyznaczajaca admitancje elementu.
- void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca prad na galezi elementu.
- void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- void wyznacz_moce ()

  Funkcja wyznaczajaca moce dla kazdego elementu.
- double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- double czestotliwosc ()

  Funkcja zwracajaca skladowa freq dla kazdego elementu.
- ~C ()

  Destruktor elementu.

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.
- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dodatkowe Dziedziczone Składowe

Atrybuty publiczne dziedziczone z element

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.1.1 Opis szczegółowy

Klasa reprezentuje kondensatory, jest klasa pochodna elementu.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 C()

```
C::C (
            char typ,
            std::pair< int, int > miejsce,
            double wartosc )
```

Konstruktor tworzacy kondensator.

Parametry

typ	Typ elementu
miejsce	Umiejscowienie elementu (wezel poczatkowy-koncowy)
wartosc	Wartosc elementu w Faradach

4.1.2.2 ~C()

```
C::~C ( )
```

Destruktor elementu.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 czestotliwosc()

```
double C::czestotliwosc ( )  [virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dla kondensatora zwroci NULL

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementuje element.

4.1.3.2 wyznacz_czest_rez()

```
double C::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy )  [virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

rezonans -> 1/sqrt(LC)

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementuje element.

4.1.3.3 wyznacz_I()

```
void C::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu i admitancje

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.1.3.4 wyznacz_moce()

```
void C::wyznacz_moce ( )  [virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje.

Parametry

brak	parametrow

Implementuje element.

4.1.3.5 wyznacz_V()

```
void C::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.1.3.6 wyznacz_Y()

```
std::complex< double > C::wyznacz_Y ( )  [virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementuje element.

4.1.3.7 wyznacz_Z()

```
std::complex< double > C::wyznacz_Z (
            double & wartosc,
            double & freq )  [virtual]
```

Funkcja wyznaczajaca impedancje kondensatora.

Parametry

wartosc	Kapacytancja
freq	Czestotliwosc pracy zrodel

Zwraca

impedancja

Implementuje element.

Dokumentacja dla tej klasy została wygenerowana z plików:

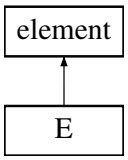
- klasy.h
- klasy.cpp

4.2 Dokumentacja klasy E

Klasa reprezentuje zrodla elektromotoryczne (SEM), jest klasa pochodna elementu.

`#include <klasy.h>`

Diagram dziedziczenia dla E



Metody publiczne

- E (char typ, std::pair< int, int > miejsce, double wartosc, double fi, double freq)

  Konstruktor tworzacy zrodlo SPM.
- std::complex< double > wyznacz_Z (double &wartosc, double &freq)

  Funkcja wyznaczajaca impedancje zrodla SEM.
- std::complex< double > wyznacz_Y ()

  Funkcja wyznaczajaca admitancje elementu.
- void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca prad na galezi elementu.
- void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- void wyznacz_moce ()

  Funkcja wyznaczajaca moce dla kazdego elementu.
- double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- double czestotliwosc ()

  Funkcja zwracajaca skladowa freq dla kazdego elementu.
- ~E ()

  Destruktor elementu.

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.

- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Atrybuty publiczne

- double fi
- double freq

Atrybuty publiczne dziedziczone z element

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.2.1 Opis szczegółowy

Klasa reprezentuje zrodla elektromotoryczne (SEM), jest klasa pochodna elementu.

Posiada dodatkowo zmienne double fi oznaczajaca przesuniecie fazowe zrodla oraz double freq oznaczajaca czestotliwosc zrodla.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 E()

```
E::E (
            char typ,
            std::pair< int, int > miejsce,
            double wartosc,
            double fi,
            double freq )
```

Konstruktor tworzacy zrodlo SPM.

Parametry

typ	Typ elementu
miejsce	Umiejscowienie elementu (wezel poczatkowy-koncowy)
wartosc	Wartosc skuteczna elementu w Voltach
fi	Przesuniecie fazowe zrodla
freq	Czestotliwosc pracy zrodla

4.2.2.2 ~E()

```
E::~E ( )
```

Destruktor elementu.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 czestotliwosc()

```
double E::czestotliwosc ( )  [virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementuje element.

4.2.3.2 wyznacz_czest_rez()

```
double E::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy )  [virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

Dla zrodla elektromotorycznego rezonans nie zachodzi

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementuje element.

4.2.3.3 wyznacz_I()

```
void E::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Wykorzystuje do tego obliczony prad na poprzednim dodanym rezystorze -1

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.2.3.4 wyznacz_moce()

```
void E::wyznacz_moce ( )  [virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje.

Parametry

brak	parametrow

Implementuje element.

4.2.3.5 wyznacz_V()

```
void E::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly ) [virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.2.3.6 wyznacz_Y()

```
std::complex< double > E::wyznacz_Y ( ) [virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementuje element.

4.2.3.7 wyznacz_Z()

```
std::complex< double > E::wyznacz_Z (
            double & wartosc,
            double & freq ) [virtual]
```

Funkcja wyznaczajaca impedancje zrodla SEM.

Wykorzystuje do tego przesuniecie fazowe zrodla (na plaszczyznie kartezjanskiej)

Parametry

wartosc	Wartosc skuteczna zrodla
freq	Czestotliwosc pracy zrodel

Implementuje element.

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 fi

```
double E::fi
```

4.2.4.2 freq

```
double E::freq
```

Dokumentacja dla tej klasy została wygenerowana z plików:

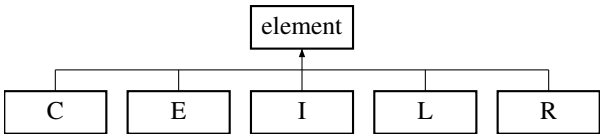
- klasy.h
- klasy.cpp

4.3 Dokumentacja klasy element

Wirtualna klasa bazowa reprezentujaca element i jego wartosci.

```
#include <klasy.h>
```

Diagram dziedziczenia dla element



Metody publiczne

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.
- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Atrybuty publiczne

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.3.1 Opis szczegółowy

Wirtualna klasa bazowa reprezentujaca element i jego wartosci.

Klasa przechowuje informacje o elemencie ukladu:

- typ (rezystor, zrodlo SEM, zrodlo SPM, kondensator, cewka)
- umiejscowienie (wezel poczatkowy, wezel koncowy)
- wartosc
- odlozone napiecie
- prad przeplywajacy przez element
- moc wydzielona Posiada funkcje wirtualne obliczajace impedancje, admitancje, prad, napiecie, czestotliwosc rezonansowa elementu.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 czestotliwosc()

```
virtual double element::czestotliwosc ( )  [pure virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementowany w E, I, R, C i L.

4.3.2.2 wyznacz_czest_rez()

```
virtual double element::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy ) [pure virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementowany w E, I, R, C i L.

4.3.2.3 wyznacz_I()

```
virtual void element::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly ) [pure virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementowany w E, I, R, C i L.

4.3.2.4 wyznacz_moce()

```
virtual void element::wyznacz_moce ( ) [pure virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje. Funkcja oblicza bilans mocy na podstawie mocy oddanej lub pobranej dla kazdego elementu.

Parametry

brak	perametrow

Implementowany w E, I, R, C i L.

4.3.2.5 wyznacz_V()

```
virtual void element::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly )  [pure virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementowany w E, I, R, C i L.

4.3.2.6 wyznacz_Y()

```
virtual std::complex< double > element::wyznacz_Y ( )  [pure virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementowany w E, I, R, C i L.

4.3.2.7 wyznacz_Z()

```
virtual std::complex< double > element::wyznacz_Z (
            double & wartosc,
            double & freq )  [pure virtual]
```

Funkcja wyznaczajaca impedancje elementu.

Parametry

wartosc	Wartosc parametru podana z pliku tekstowego
freq	Czestotliwosc zrodel w obwodzie

Zwraca

impedancja Zwraca impedancje elementu w postaci zespolonej

Implementowany w E, I, R, C i L.

4.3.3 Dokumentacja atrybutów składowych

4.3.3.1 admitancja

```
std::complex<double> element::admitancja
```

4.3.3.2 czest_rez

```
double element::czest_rez
```

4.3.3.3 impedancja

```
std::complex<double> element::impedancja
```

4.3.3.4 moc_bierna

```
double element::moc_bierna
```

4.3.3.5 moc_czynna

```
double element::moc_czynna
```

4.3.3.6 napiecie

```
std::complex<double> element::napiecie
```

4.3.3.7 prad

```
std::complex<double> element::prad
```

4.3.3.8 typ

```
char element::typ
```

4.3.3.9 umiejscowienie

```
std::pair<int, int> element::umiejscowienie
```

4.3.3.10 wartosc

```
double element::wartosc
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

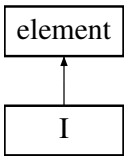
- klasy.h

4.4 Dokumentacja klasy I

Klasa reprezentuje zrodla pradomotoryczne (SPM), jest klasa pochodna elementu.

```
#include <klasy.h>
```

Diagram dziedziczenia dla I



Metody publiczne

- I (char typ, std::pair< int, int > miejsce, double wartosc, double fi, double freq)

  Konstruktor tworzacy zrodlo SEM.
- std::complex< double > wyznacz_Z (double &wartosc, double &freq)

  Funkcja wyznaczajaca impedancje zrodla SPM.
- std::complex< double > wyznacz_Y ()

  Funkcja wyznaczajaca admitancje elementu.
- void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca prad na galezi elementu.
- void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- void wyznacz_moce ()

  Funkcja wyznaczajaca moce dla kazdego elementu.
- double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- double czestotliwosc ()

  Funkcja zwracajaca skladowa freq dla kazdego elementu.
- ~I ()

  Destruktor elementu.

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.
- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Atrybuty publiczne

- double fi
- double freq

Atrybuty publiczne dziedziczone z element

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.4.1 Opis szczegółowy

Klasa reprezentuje zrodla pradomotoryczne (SPM), jest klasa pochodna elementu.

Posiada dodatkowo zmienne double fi oznaczajaca przesuniecie fazowe zrodla oraz double freq oznaczajaca czestotliwosc zrodla.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 I()

```
I::I (
            char typ,
            std::pair< int, int > miejsce,
            double wartosc,
            double fi,
            double freq )
```

Konstruktor tworzacy zrodlo SEM.

Parametry

typ	Typ elementu
miejsce	Umiejscowienie elementu (wezel poczatkowy-koncowy)
wartosc	Wartosc skuteczna elementu w Amperach
fi	Przesuniecie fazowe zrodla
freq	Czestotliwosc pracy zrodla

4.4.2.2 ∼I()

```
I::∼I ( )
```

Destruktor elementu.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 czestotliwosc()

```
double I::czestotliwosc ( )  [virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementuje element.

4.4.3.2 wyznacz_czest_rez()

```
double I::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy )  [virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

Rezonans nie zachodzi na zrodla SPM

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementuje element.

4.4.3.3 wyznacz_I()

```
void I::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Prad na galezi z SPM jest taki sam jak jego impedancja z przeciwnym znakiem

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.4.3.4 wyznacz_moce()

```
void I::wyznacz_moce ( )  [virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje.

Parametry

brak	parametrow

Implementuje element.

4.4.3.5 wyznacz_V()

```
void I::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.4.3.6 wyznacz_Y()

```
std::complex< double > I::wyznacz_Y ( )  [virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementuje element.

4.4.3.7 wyznacz_Z()

```
std::complex< double > I::wyznacz_Z (
            double & wartosc,
            double & freq )  [virtual]
```

Funkcja wyznaczajaca impedancje zrodla SPM.

Wykorzystuje do tego przesuniecie fazowe zrodla (na plaszczyznie kartezjanskiej)

Parametry

wartosc	Wartosc skuteczna zrodla
freq	Czestotliwosc pracy zrodel

Zwraca

impedancja

Implementuje element.

4.4.4 Dokumentacja atrybutów składowych

4.4.4.1 fi

```
double I::fi
```

4.4.4.2 freq

```
double I::freq
```

Dokumentacja dla tej klasy została wygenerowana z plików:

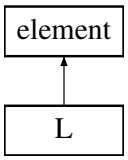
- klasy.h
- klasy.cpp

4.5 Dokumentacja klasy L

Klasa reprezentuje cewki, jest klasa pochodna elementu.

`#include <klasy.h>`

Diagram dziedziczenia dla L



Metody publiczne

- L (char typ, std::pair< int, int > miejsce, double wartosc)

  Konstruktor tworzacy cewke.
- std::complex< double > wyznacz_Z (double &wartosc, double &freq)

  Funkcja wyznaczajaca impedancje cewki.
- std::complex< double > wyznacz_Y ()

  Funkcja wyznaczajaca admitancje elementu.
- void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca prad na galezi elementu.
- void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- void wyznacz_moce ()

  Funkcja wyznaczajaca moce dla kazdego elementu.
- double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- double czestotliwosc ()

  Funkcja zwracajaca skladowa freq dla kazdego elementu.
- ~L ()

  Destruktor elementu.

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.
- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dodatkowe Dziedziczone Składowe

Atrybuty publiczne dziedziczone z element

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.5.1 Opis szczegółowy

Klasa reprezentuje cewki, jest klasa pochodna elementu.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 L()

```
L::L (
            char typ,
            std::pair< int, int > miejsce,
            double wartosc )
```

Konstruktor tworzacy cewke.

Parametry

typ	Typ elementu
miejsce	Umiejscowienie elementu (wezel poczatkowy-koncowy)
wartosc	Wartosc elementu w Henrach

4.5.2.2 ~L()

```
L::~L ( )
```

Destruktor elementu.

4.5.3 Dokumentacja funkcji składowych

4.5.3.1 czestotliwosc()

```
double L::czestotliwosc ( )  [virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dla cewki zwroci NULL

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementuje element.

4.5.3.2 wyznacz_czest_rez()

```
double L::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy )  [virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

rezonans -> 1/sqrt(LC)

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementuje element.

4.5.3.3 wyznacz_I()

```
void L::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu i admitancje

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.5.3.4 wyznacz_moce()

```
void L::wyznacz_moce ( )  [virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje.

Parametry

brak	parametrow

Implementuje element.

4.5.3.5 wyznacz_V()

```
void L::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.5.3.6 wyznacz_Y()

```
std::complex< double > L::wyznacz_Y ( )  [virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementuje element.

4.5.3.7 wyznacz_Z()

```
std::complex< double > L::wyznacz_Z (
            double & wartosc,
            double & freq )  [virtual]
```

Funkcja wyznaczajaca impedancje cewki.

Parametry

wartosc	Przewodnosc
freq	Czestotliwosc pracy zrodel

Zwraca

impedancja

Implementuje element.

Dokumentacja dla tej klasy została wygenerowana z plików:

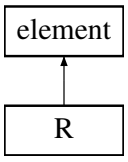
- klasy.h
- klasy.cpp

4.6 Dokumentacja klasy R

Klasa reprezentuje rezystory, jest klasa pochodna elementu.

```
#include <klasy.h>
```

Diagram dziedziczenia dla R



Metody publiczne

- R (char typ, std::pair< int, int > miejsce, double wartosc)

  Konstruktor tworzacy rezystor.
- std::complex< double > wyznacz_Z (double &wartosc, double &freq)

  Funkcja wyznaczajaca impedancje rezystora.
- std::complex< double > wyznacz_Y ()

  Funkcja wyznaczajaca admitancje elementu.
- void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca prad na galezi elementu.
- void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- void wyznacz_moce ()

  Funkcja wyznaczajaca moce dla kazdego elementu.
- double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- double czestotliwosc ()

  Funkcja zwracajaca skladowa freq dla kazdego elementu.
- ~R ()

  Destruktor elementu.

- virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0

  Funkcja wyznaczajaca impedancje elementu.
- virtual std::complex< double > wyznacz_Y ()=0

  Funkcja wyznaczajaca admitancje elementu.
- virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca prad na galezi elementu.
- virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0

  Funkcja wyznaczajaca napiecie na kazdym elemencie.
- virtual void wyznacz_moce ()=0

  Funkcja wyznaczajaca moce dla kazdego elementu.
- virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0

  Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.
- virtual double czestotliwosc ()=0

  Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dodatkowe Dziedziczone Składowe

Atrybuty publiczne dziedziczone z element

- char typ
- std::pair< int, int > umiejscowienie
- double wartosc
- double moc_czynna
- double moc_bierna
- double czest_rez
- std::complex< double > impedancja
- std::complex< double > admitancja
- std::complex< double > napiecie
- std::complex< double > prad

4.6.1 Opis szczegółowy

Klasa reprezentuje rezystory, jest klasa pochodna elementu.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 R()

```
R::R (
            char typ,
            std::pair< int, int > miejsce,
            double wartosc )
```

Konstruktor tworzacy rezystor.

Parametry

typ	Typ elementu
miejsce	Umiejscowienie elementu (wezel poczatkowy-koncowy)
wartosc	Wartosc elementu w Ohmach

4.6.2.2 ∼R()

```
R::∼R ( )
```

Destruktor elementu.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 czestotliwosc()

```
double R::czestotliwosc ( )  [virtual]
```

Funkcja zwracajaca skladowa freq dla kazdego elementu.

Dla rezystora zwroci NULL

Parametry

brak	parametrow

Zwraca

freq Czestotliwosc zrodel

Implementuje element.

4.6.3.2 wyznacz_czest_rez()

```
double R::wyznacz_czest_rez (
            const std::vector< std::shared_ptr< element > > & elementy ) [virtual]
```

Funkcja wyznaczajaca czestotliwosc rezonansowa dla kazdego elementu.

Rezonans nie zachodzi dla rezystorow

Parametry

elementy	Wektor wszystkich elementow obwodu

Zwraca

freq Czestotliwosc rezonansu kondensatora lub cewki

Implementuje element.

4.6.3.3 wyznacz_I()

```
void R::wyznacz_I (
            const std::vector< std::shared_ptr< element > > & elementy,
            std::unordered_map< int, std::complex< double > > & potencjaly ) [virtual]
```

Funkcja wyznaczajaca prad na galezi elementu.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu i admitancje

Parametry

elementy	Wektor wszystkich elementow z obwodu
potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.6.3.4 wyznacz_moce()

```
void R::wyznacz_moce ( ) [virtual]
```

Funkcja wyznaczajaca moce dla kazdego elementu.

Oblicza moc czynna i bierna dla kazdego elementu. Jesli moc jest dodatnia, to znaczy, ze element pobiera energie. Jesli ujemna, to znaczy, ze ja oddaje.

Parametry

brak	parametrow

Implementuje element.

4.6.3.5 wyznacz_V()

```
void R::wyznacz_V (
            std::unordered_map< int, std::complex< double > > & potencjaly )  [virtual]
```

Funkcja wyznaczajaca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

Parametry

potencjaly	Mapa potencjalow w wezlach

Implementuje element.

4.6.3.6 wyznacz_Y()

```
std::complex< double > R::wyznacz_Y ( )  [virtual]
```

Funkcja wyznaczajaca admitancje elementu.

Parametry

brak	parametrow

Zwraca

admitancja Zwraca admitancje elementu w postaci zespolonej

Implementuje element.

4.6.3.7 wyznacz_Z()

```
std::complex< double > R::wyznacz_Z (
            double & wartosc,
            double & freq ) [virtual]
```

Funkcja wyznaczajaca impedancje rezystora.

Parametry

wartosc	Opornosc rezystora
freq	Czestotliwosc pracy zrodel

Zwraca

impedancja

Implementuje element.

Dokumentacja dla tej klasy została wygenerowana z plików:

- klasy.h
- klasy.cpp

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku funkcje.cpp

```
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
#include <vector>
#include <map>
#include <unordered_map>
#include <set>
#include <cmath>
#include <memory>
#include <complex>
#include <iomanip>
#include "klasy.h"
#include "funkcje.h"
```

Definicje

- #define _USE_MATH_DEFINES

Funkcje

- std::vector< std::shared_ptr< element > > odczyt_wejscia (const std::string &nazwa_pliku)

  Funkcja odczytuje elementy z pliku tekstowego.
- std::set< int > wektor_wezlow (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja tworzy kontener unikalnych wezlow z obwodu.
- std::vector< std::shared_ptr< element > > dodaj_rez_obok_sem (std::vector< std::shared_ptr< element > > &elementy, const std::set< int > &wezly)

  Funkcja dodaje wirtualny rezystor na galaz obok SEM.
- std::pair< macierz, std::unordered_map< int, int > > coltri (const std::vector< std::shared_ptr< element > > &elementy, const std::set< int > &wezly)

  Funkcja tworzy macierz (uklad rownan) za pomoca metody Coltriego.
- void zapis_wyjscia (const std::string &nazwa_pliku, const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wypisuje do pliku wyjscia charakterystyke kazdego elementu.

5.1.1 Dokumentacja definicji

5.1.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

5.1.2 Dokumentacja funkcji

5.1.2.1 coltri()

```
std::pair< macierz, std::unordered_map< int, int > > coltri (
            const std::vector< std::shared_ptr< element > > & elementy,
            const std::set< int > & wezly )
```

Funkcja tworzy macierz (uklad rownan) za pomoca metody Coltriego.

Funkcja wprowadza do macierzy admitancje kazdego potencjalu uzywajac metody potencjalow wezlowych. Napotkane galezie ze zrodlem elektromotorycznym traktuje jako galezie z impedancja {1,0}om. Funkcja tworzy takze mape wezlow na podstawie kontenera tworzonego w wektor_wezlow(), tak aby zaczynaly sie od 0 do n.

Zwraca

Para macierzy (wektora wektorow wskaznikow) i mapy stare2nowe

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)
in	*wezly*	Kontener wezlow z obwodu

5.1.2.2 dodaj_rez_obok_sem()

```
std::vector< std::shared_ptr< element > > dodaj_rez_obok_sem (
            std::vector< std::shared_ptr< element > > & elementy,
            const std::set< int > & wezly )
```

Funkcja dodaje wirtualny rezystor na galaz obok SEM.

Funkcja dodaje rezystor o wartosci -1 obok galezi ze zrodlem elektromotorycznym, aby pozniej traktowac galaz z SEM jako galaz z opornoscia 1 om.

Zwraca

Wektor elementow (wskaznikow na element) z nowymi ujemnymi rezystorami

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)
in	*Kontener*	wezlow obwodu

5.1.2.3 odczyt_wejscia()

```
std::vector< std::shared_ptr< element > > odczyt_wejscia (
            const std::string & nazwa_pliku )
```

Funkcja odczytuje elementy z pliku tekstowego.

Odczytuje na podstawie wprowadzonych danych w formacie <typ>,<wezel poczatkowy>,<wezel koncowy>,<wartosc>,<fi>,<fre

Parametry

in	*nazwa_pliku*	Nazwa pliku wejsciowego

Zwraca

elementy Wektor elementow (wskaznikow na element)

5.1.2.4 wektor_wezlow()

```
std::set< int > wektor_wezlow (
            const std::vector< std::shared_ptr< element > > & elementy )
```

Funkcja tworzy kontener unikalnych wezlow z obwodu.

wpisujac kazdy wezel elementu w sposob posortowany od najmniejszych do najwiekszych wartosci.

Zwraca

Kontener wezlow obwodu

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)

5.1.2.5 zapis_wyjscia()

```
void zapis_wyjscia (
```

```
const std::string & nazwa_pliku,
const std::vector< std::shared_ptr< element > > & elementy )
```

Funkcja wypisuje do pliku wyjscia charakterystyke kazdego elementu.

Funkcja wypisuje wszystkie elementy obwodu, ich parametry i bilans obwodu do pliku tekstowego. Prad oraz napiecie funkcja podaje w postaci modulu liczby zespolonej oraz wzoru Eulera na kat przesuniecia fazowego dla kazdego elementu Funkcja oblicza bilans mocy na podstawie mocy oddanej lub pobranej dla kazdego elementu.

Parametry

in	*nazwa_pliku*	Nazwa pliku wyjsciowego
in	*elementy*	Wektor elementow

5.2 Dokumentacja pliku funkcje.h

```
#include <string>
#include <vector>
#include <set>
#include <unordered_map>
#include <memory>
#include "klasy.h"
```

Definicje

- #define FUNKCJE_H

Funkcje

- int main (int liczba_param, char ∗param[ ])

  Funkcja glowna programu.
- std::vector< std::shared_ptr< element > > odczyt_wejscia (const std::string &nazwa_pliku)

  Funkcja odczytuje elementy z pliku tekstowego.
- std::set< int > wektor_wezlow (const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja tworzy kontener unikalnych wezlow z obwodu.
- std::vector< std::shared_ptr< element > > dodaj_rez_obok_sem (std::vector< std::shared_ptr< element > > &elementy, const std::set< int > &wezly)

  Funkcja dodaje wirtualny rezystor na galaz obok SEM.
- std::pair< macierz, std::unordered_map< int, int > > coltri (const std::vector< std::shared_ptr< element > > &elementy, const std::set< int > &wezly)

  Funkcja tworzy macierz (uklad rownan) za pomoca metody Coltriego.
- std::unordered_map< int, std::complex< double > > gauss (const std::pair< macierz, std::unordered_map< int, int > > &uklad_rownan_i_mapa, const std::set< int > &wezly)

  Funkcja oblicza macierz za pomoca metody eliminacji Gaussa-Jordana-Crouta.
- void zapis_wyjscia (const std::string &nazwa_pliku, const std::vector< std::shared_ptr< element > > &elementy)

  Funkcja wypisuje do pliku wyjscia charakterystyke kazdego elementu.

5.2.1 Dokumentacja definicji

5.2.1.1 FUNKCJE_H

```
#define FUNKCJE_H
```

5.2.2 Dokumentacja funkcji

5.2.2.1 coltri()

```
std::pair< macierz, std::unordered_map< int, int > > coltri (
            const std::vector< std::shared_ptr< element > > & elementy,
            const std::set< int > & wezly )
```

Funkcja tworzy macierz (uklad rownan) za pomoca metody Coltriego.

Funkcja wprowadza do macierzy admitancje kazdego potencjalu uzywajac metody potencjalow wezlowych. Napotkane galezie ze zrodlem elektromotorycznym traktuje jako galezie z impedancja {1,0}om. Funkcja tworzy takze mape wezlow na podstawie kontenera tworzonego w wektor_wezlow(), tak aby zaczynaly sie od 0 do n.

Zwraca

Para macierzy (wektora wektorow wskaznikow) i mapy stare2nowe

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)
in	*wezly*	Kontener wezlow z obwodu

5.2.2.2 dodaj_rez_obok_sem()

```
std::vector< std::shared_ptr< element > > dodaj_rez_obok_sem (
            std::vector< std::shared_ptr< element > > & elementy,
            const std::set< int > & wezly )
```

Funkcja dodaje wirtualny rezystor na galaz obok SEM.

Funkcja dodaje rezystor o wartosci -1 obok galezi ze zrodlem elektromotorycznym, aby pozniej traktowac galaz z SEM jako galaz z opornoscia 1 om.

Zwraca

Wektor elementow (wskaznikow na element) z nowymi ujemnymi rezystorami

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)
in	*Kontener*	wezlow obwodu

5.2.2.3 gauss()

```
std::unordered_map< int, std::complex< double > > gauss (
            const std::pair< macierz, std::unordered_map< int, int > > & uklad_rownan_i_↩
mapa,
            const std::set< int > & wezly )
```

Funkcja oblicza macierz za pomoca metody eliminacji Gaussa-Jordana-Crouta.

Funkcja sprowadza macierz do postaci trojkatnej (uzyskuje zero nad i pod przekatna macierzy). Zamienia wiersze (rownanie) z innym wierszem, w ktorym wystepuje wiekszy wspolczynnik w kolumnie. Metoda ta sprowadza macierz rozszerzona ukladu rownan do postaci bazowej (macierzy jednostkowej). Z tej postaci mozna wprost odczytac potencjaly w wezlach.

Zwraca

Mapa potencjalow w wezlach

Parametry

in	*uklad_rownan_i_mapa*	Para macierzy i mapy stare2nowe
in	*wezly*	Kontenera wezlow obwodu

5.2.2.4 main()

```
int main (
            int liczba_param,
            char * param[] )
```

Funkcja glowna programu.

Zwraca

int

Parametry

in	*liczba_param*	Liczba parametrow
in	*param[]*	Tablica parametrow

5.2.2.5 odczyt_wejscia()

```
std::vector< std::shared_ptr< element > > odczyt_wejscia (
            const std::string & nazwa_pliku )
```

Funkcja odczytuje elementy z pliku tekstowego.

Odczytuje na podstawie wprowadzonych danych w formacie <typ>,<wezel poczatkowy>,<wezel koncowy>,<wartosc>,<fi>,<fre

Parametry

in	*nazwa_pliku*	Nazwa pliku wejsciowego

Zwraca

elementy Wektor elementow (wskaznikow na element)

5.2.2.6 wektor_wezlow()

```
std::set< int > wektor_wezlow (
            const std::vector< std::shared_ptr< element > > & elementy )
```

Funkcja tworzy kontener unikalnych wezlow z obwodu.

wpisujac kazdy wezel elementu w sposob posortowany od najmniejszych do najwiekszych wartosci.

Zwraca

Kontener wezlow obwodu

Parametry

in	*elementy*	Wektor elementow (wskaznikow na element)

5.2.2.7 zapis_wyjscia()

```
void zapis_wyjscia (
            const std::string & nazwa_pliku,
            const std::vector< std::shared_ptr< element > > & elementy )
```

Funkcja wypisuje do pliku wyjscia charakterystyke kazdego elementu.

Funkcja wypisuje wszystkie elementy obwodu, ich parametry i bilans obwodu do pliku tekstowego. Prad oraz napiecie funkcja podaje w postaci modulu liczby zespolonej oraz wzoru Eulera na kat przesuniecia fazowego dla kazdego elementu Funkcja oblicza bilans mocy na podstawie mocy oddanej lub pobranej dla kazdego elementu.

Parametry

in	*nazwa_pliku*	Nazwa pliku wyjsciowego
in	*elementy*	Wektor elementow

5.3 funkcje.h

Idź do dokumentacji tego pliku.

```
00001 #pragma once
00006 // DECLARATIONS (HEADERS) OF FUNCTIONS
00007
00008 #ifndef FUNKCJE_H
00009 #define FUNKCJE_H
00010
00011 #include <string>
00012 #include <vector>
00013 #include <set>
00014 #include <unordered_map>
00015 #include <memory>
00016
00017 #include "klasy.h"
00018
00025 int main(int liczba_param, char* param[]);
00026
00033 std::vector<std::shared_ptr<element» odczyt_wejscia(const std::string& nazwa_pliku);
00034
00035 //void wypisz_elementy(const std::vector<std::shared_ptr<element»& elementy);
00036
00043 std::set<int> wektor_wezlow(const std::vector<std::shared_ptr<element»& elementy);
00044
00052 std::vector<std::shared_ptr<element» dodaj_rez_obok_sem(std::vector<std::shared_ptr<element»&
      elementy, const std::set<int>& wezly);
00053
00063 std::pair<macierz, std::unordered_map<int, int» coltri(const std::vector<std::shared_ptr<element»&
      elementy, const std::set<int>& wezly);
00064
00065 //void wypisz(const macierz& potencjaly);
00066
00077 std::unordered_map<int, std::complex<double» gauss(const std::pair<macierz, std::unordered_map<int,
      int»& uklad_rownan_i_mapa, const std::set<int>& wezly);
00078
00087 void zapis_wyjscia(const std::string& nazwa_pliku, const std::vector<std::shared_ptr<element»&
      elementy);
00088
00089 #endif
```

5.4 Dokumentacja pliku gauss.cpp

```
#include <iostream>
#include <unordered_map>
#include <set>
#include <cmath>
#include "klasy.h"
#include "funkcje.h"
```

Funkcje

- std::unordered_map< int, std::complex< double > > gauss (const std::pair< macierz, std::unordered_map< int, int > > &uklad_rownan_i_mapa, const std::set< int > &wezly)

  Funkcja oblicza macierz za pomoca metody eliminacji Gaussa-Jordana-Crouta.

5.4.1 Dokumentacja funkcji

5.4.1.1 gauss()

```
std::unordered_map< int, std::complex< double > > gauss (
            const std::pair< macierz, std::unordered_map< int, int > > & uklad_rownan_i_↩
mapa,
            const std::set< int > & wezly )
```

Funkcja oblicza macierz za pomoca metody eliminacji Gaussa-Jordana-Crouta.

Funkcja sprowadza macierz do postaci trojkatnej (uzyskuje zero nad i pod przekatna macierzy). Zamienia wiersze (rownanie) z innym wierszem, w ktorym wystepuje wiekszy wspolczynnik w kolumnie. Metoda ta sprowadza macierz rozszerzona ukladu rownan do postaci bazowej (macierzy jednostkowej). Z tej postaci mozna wprost odczytac potencjaly w wezlach.

Zwraca

Mapa potencjalow w wezlach

Parametry

in	*uklad_rownan_i_mapa*	Para macierzy i mapy stare2nowe
in	*wezly*	Kontenera wezlow obwodu

5.5 Dokumentacja pliku klasy.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include "klasy.h"
#include "funkcje.h"
```

Definicje

- #define _USE_MATH_DEFINES

Zmienne

- std::complex< double > jeden {1, 0}

5.5.1 Dokumentacja definicji

5.5.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

5.5.2 Dokumentacja zmiennych

5.5.2.1 jeden

```
std::complex<double> jeden {1, 0}
```

5.6 Dokumentacja pliku klasy.h

```
#include <map>
#include <unordered_map>
#include <vector>
#include <string>
#include <complex>
#include <cmath>
```

Komponenty

- class element

  Wirtualna klasa bazowa reprezentujaca element i jego wartosci.
- class E

  Klasa reprezentuje zrodla elektromotoryczne (SEM), jest klasa pochodna elementu.
- class I

  Klasa reprezentuje zrodla pradomotoryczne (SPM), jest klasa pochodna elementu.
- class R

  Klasa reprezentuje rezystory, jest klasa pochodna elementu.
- class C

  Klasa reprezentuje kondensatory, jest klasa pochodna elementu.
- class L

  Klasa reprezentuje cewki, jest klasa pochodna elementu.

Definicje

- #define KLASY_H

Definicje typów

- typedef std::vector< std::vector< std::complex< double > > > macierz

Zmienne

- const double e = std::numeric_limits<double>::min()

5.6.1 Dokumentacja definicji

5.6.1.1 KLASY_H

```
#define KLASY_H
```

5.6.2 Dokumentacja definicji typów

5.6.2.1 macierz

```
typedef std::vector<std::vector<std::complex<double> > > macierz
```

5.6.3 Dokumentacja zmiennych

5.6.3.1 e

```
const double e = std::numeric_limits<double>::min()
```

5.7 klasy.h

Idź do dokumentacji tego pliku.

```
00001 #pragma once
00006 #ifndef KLASY_H
00007 #define KLASY_H
00008
00009 #include <map>
00010 #include <unordered_map>
00011 #include <vector>
00012 #include <string>
00013 #include <complex>
00014 #include <cmath>
00015
00016
00017 typedef
00018 std::vector<std::vector<std::complex<double>> macierz;
00019
00020 const double e = std::numeric_limits<double>::min();
00021
00033 class element
00034 {
00035     public:
00036         char typ;
```

```
00037         std::pair<int, int> umiejscowienie;
00038         double wartosc, moc_czynna, moc_bierna, czest_rez;
00039         std::complex<double> impedancja, admitancja, napiecie, prad;
00046         virtual std::complex<double> wyznacz_Z(double& wartosc, double& freq) = 0;
00052         virtual std::complex<double> wyznacz_Y() = 0;
00059         virtual void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy,
      std::unordered_map<int, std::complex<double>& potencjaly) = 0;
00065         virtual void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly) = 0;
00072         virtual void wyznacz_moce() = 0;
00078         virtual double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy) = 0;
00084         virtual double czestotliwosc() = 0;
00085 };
00090 class E : public element
00091 {
00092     public:
00093         double fi, freq;
00102         E(char typ, std::pair<int, int> miejsce, double wartosc, double fi, double freq);
00109         std::complex<double> wyznacz_Z(double& wartosc, double& freq);
00115         std::complex<double> wyznacz_Y();
00122         void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy, std::unordered_map<int,
      std::complex<double>& potencjaly);
00128         void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly);
00134         void wyznacz_moce();
00141         double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy);
00147         double czestotliwosc();
00151         ~E();
00152 };
00157 class I : public element
00158 {
00159     public:
00160         double fi, freq;
00169         I(char typ, std::pair<int, int> miejsce, double wartosc, double fi, double freq);
00177         std::complex<double> wyznacz_Z(double& wartosc, double& freq);
00183         std::complex<double> wyznacz_Y();
00190         void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy, std::unordered_map<int,
      std::complex<double>& potencjaly);
00196         void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly);
00202         void wyznacz_moce();
00209         double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy);
00215         double czestotliwosc();
00219         ~I();
00220 };
00224 class R : public element
00225 {
00226     public:
00233         R(char typ, std::pair<int, int> miejsce, double wartosc);
00240         std::complex<double> wyznacz_Z(double& wartosc, double& freq);
00246         std::complex<double> wyznacz_Y();
00253         void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy, std::unordered_map<int,
      std::complex<double>& potencjaly);
00259         void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly);
00265         void wyznacz_moce();
00272         double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy);
00279         double czestotliwosc();
00283         ~R();
00284 };
00288 class C : public element
00289 {
00290     public:
00297         C(char typ, std::pair<int, int> miejsce, double wartosc);
00304         std::complex<double> wyznacz_Z(double& wartosc, double& freq);
00310         std::complex<double> wyznacz_Y();
00317         void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy, std::unordered_map<int,
      std::complex<double>& potencjaly);
00323         void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly);
00329         void wyznacz_moce();
00336         double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy);
00343         double czestotliwosc();
00347         ~C();
00348 };
00352 class L : public element
00353 {
00354     public:
00361         L(char typ, std::pair<int, int> miejsce, double wartosc);
00368         std::complex<double> wyznacz_Z(double& wartosc, double& freq);
00374         std::complex<double> wyznacz_Y();
00381         void wyznacz_I(const std::vector<std::shared_ptr<element>& elementy, std::unordered_map<int,
      std::complex<double>& potencjaly);
00387         void wyznacz_V(std::unordered_map<int, std::complex<double>& potencjaly);
00393         void wyznacz_moce();
00400         double wyznacz_czest_rez(const std::vector<std::shared_ptr<element>& elementy);
00407         double czestotliwosc();
00411         ~L();
00412 };
00413
00414 #endif
```

5.8 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <string>
#include <map>
#include <vector>
#include <memory>
#include "klasy.h"
#include "funkcje.h"
```

Funkcje

- int main (int liczba_param, char ∗param[ ])

  Funkcja glowna programu.

5.8.1 Dokumentacja funkcji

5.8.1.1 main()

```
int main (
            int liczba_param,
            char * param[] )
```

Funkcja glowna programu.

Zwraca

int

Parametry

in	*liczba_param*	Liczba parametrow
in	*param[]*	Tablica parametrow

Skorowidz