

Politechnika Śląska  
Wydział Informatyki, Elektroniki i Informatyki

# Programowanie Komputerów

**ALOPP**



---

**autor**  
**prowadzący**  
**rok akademicki**  
**kierunek**  
**rodzaj studiów**  
**semestr**  
**termin laboratorium**  
**sekcja**  
**termin oddania sprawozdania**

**Filip Kudła**  
**dr. inż. Wojciech Łabaj**  
**2022/2023**  
**informatyka**  
**SSI**  
**2**  
**środa, 13:30 - 15:00**  
**62**  
**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.

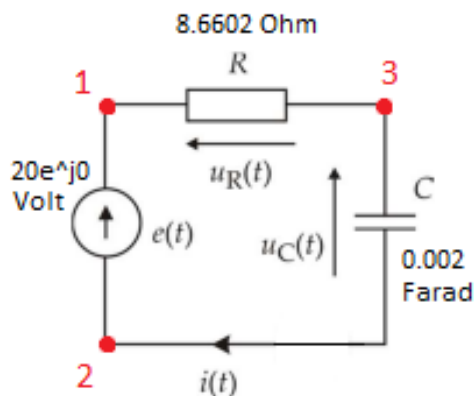
## 2.2 Algorytmy

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 zakończy 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 różną 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



<b>1 Indeks hierarchiczny</b>	<b>1</b>
1.1 Hierarchia klas	1
<b>2 Indeks klas</b>	<b>3</b>
2.1 Lista klas	3
<b>3 Indeks plików</b>	<b>5</b>
3.1 Lista plików	5
<b>4 Dokumentacja klas</b>	<b>7</b>
4.1 Dokumentacja klasy C	7
4.1.1 Opis szczegółowy	8
4.1.2 Dokumentacja konstruktora i destruktor	8
4.1.2.1 C()	8
4.1.2.2 ~C()	9
4.1.3 Dokumentacja funkcji składowych	9
4.1.3.1 czestotliwosc()	9
4.1.3.2 wyznaczc_czest_rez()	9
4.1.3.3 wyznaczc_l()	10
4.1.3.4 wyznaczc_moce()	10
4.1.3.5 wyznaczc_V()	11
4.1.3.6 wyznaczc_Y()	11
4.1.3.7 wyznaczc_Z()	11
4.2 Dokumentacja klasy E	12
4.2.1 Opis szczegółowy	13
4.2.2 Dokumentacja konstruktora i destruktor	13
4.2.2.1 E()	13
4.2.2.2 ~E()	14
4.2.3 Dokumentacja funkcji składowych	14
4.2.3.1 czestotliwosc()	14
4.2.3.2 wyznaczc_czest_rez()	14
4.2.3.3 wyznaczc_l()	15
4.2.3.4 wyznaczc_moce()	15
4.2.3.5 wyznaczc_V()	16
4.2.3.6 wyznaczc_Y()	16
4.2.3.7 wyznaczc_Z()	16
4.2.4 Dokumentacja atrybutów składowych	17
4.2.4.1 fi	17
4.2.4.2 freq	17
4.3 Dokumentacja klasy element	17
4.3.1 Opis szczegółowy	18
4.3.2 Dokumentacja funkcji składowych	18
4.3.2.1 czestotliwosc()	18

4.3.2.2 wyznacz_czest_rez()	19
4.3.2.3 wyznacz_I()	19
4.3.2.4 wyznacz_moce()	19
4.3.2.5 wyznacz_V()	20
4.3.2.6 wyznacz_Y()	20
4.3.2.7 wyznacz_Z()	20
4.3.3 Dokumentacja atrybutów składowych	21
4.3.3.1 admitancja	21
4.3.3.2 czest_rez	21
4.3.3.3 impedancja	21
4.3.3.4 moc_bierna	21
4.3.3.5 moc_czynna	21
4.3.3.6 napiecie	22
4.3.3.7 prad	22
4.3.3.8 typ	22
4.3.3.9 umiejscowienie	22
4.3.3.10 wartosc	22
4.4 Dokumentacja klasy I	22
4.4.1 Opis szczegółowy	24
4.4.2 Dokumentacja konstruktora i destruktora	24
4.4.2.1 I()	24
4.4.2.2 ~I()	24
4.4.3 Dokumentacja funkcji składowych	24
4.4.3.1 czestotliwosc()	24
4.4.3.2 wyznacz_czest_rez()	25
4.4.3.3 wyznacz_I()	25
4.4.3.4 wyznacz_moce()	26
4.4.3.5 wyznacz_V()	26
4.4.3.6 wyznacz_Y()	26
4.4.3.7 wyznacz_Z()	27
4.4.4 Dokumentacja atrybutów składowych	27
4.4.4.1 fi	27
4.4.4.2 freq	27
4.5 Dokumentacja klasy L	28
4.5.1 Opis szczegółowy	29
4.5.2 Dokumentacja konstruktora i destruktora	29
4.5.2.1 L()	29
4.5.2.2 ~L()	29
4.5.3 Dokumentacja funkcji składowych	30
4.5.3.1 czestotliwosc()	30
4.5.3.2 wyznacz_czest_rez()	30
4.5.3.3 wyznacz_I()	30



4.5.3.4 wyznacz_moce()	31
4.5.3.5 wyznacz_V()	31
4.5.3.6 wyznacz_Y()	31
4.5.3.7 wyznacz_Z()	32
4.6 Dokumentacja klasy R	32
4.6.1 Opis szczegółowy	34
4.6.2 Dokumentacja konstruktora i destruktora	34
4.6.2.1 R()	34
4.6.2.2 $\sim R()$	34
4.6.3 Dokumentacja funkcji składowych	34
4.6.3.1 czestotliwosc()	34
4.6.3.2 wyznacz_czest_rez()	35
4.6.3.3 wyznacz_I()	35
4.6.3.4 wyznacz_moce()	35
4.6.3.5 wyznacz_V()	36
4.6.3.6 wyznacz_Y()	36
4.6.3.7 wyznacz_Z()	37
<b>5 Dokumentacja plików</b>	<b>39</b>
5.1 Dokumentacja pliku funkcje.cpp	39
5.1.1 Dokumentacja definicji	40
5.1.1.1 _USE_MATH_DEFINES	40
5.1.2 Dokumentacja funkcji	40
5.1.2.1 coltri()	40
5.1.2.2 dodaj_rez_obok_sem()	40
5.1.2.3 odczyt_wejscia()	41
5.1.2.4 wektor_wezlow()	41
5.1.2.5 zapis_wyjscia()	41
5.2 Dokumentacja pliku funkcje.h	42
5.2.1 Dokumentacja definicji	43
5.2.1.1 FUNKCJE_H	43
5.2.2 Dokumentacja funkcji	43
5.2.2.1 coltri()	43
5.2.2.2 dodaj_rez_obok_sem()	43
5.2.2.3 gauss()	44
5.2.2.4 main()	44
5.2.2.5 odczyt_wejscia()	45
5.2.2.6 wektor_wezlow()	45
5.2.2.7 zapis_wyjscia()	45
5.3 funkcje.h	47
5.4 Dokumentacja pliku gauss.cpp	47
5.4.1 Dokumentacja funkcji	48

---

5.4.1.1 gauss()	48
5.5 Dokumentacja pliku klasy.cpp	48
5.5.1 Dokumentacja definicji	48
5.5.1.1 _USE_MATH_DEFINES	49
5.5.2 Dokumentacja zmiennych	49
5.5.2.1 jeden	49
5.6 Dokumentacja pliku klasy.h	49
5.6.1 Dokumentacja definicji	50
5.6.1.1 KLASY_H	50
5.6.2 Dokumentacja definicji typów	50
5.6.2.1 macierz	50
5.6.3 Dokumentacja zmiennych	50
5.6.3.1 e	50
5.7 klasy.h	50
5.8 Dokumentacja pliku main.cpp	52
5.8.1 Dokumentacja funkcji	52
5.8.1.1 main()	52
<b>Skorowidz</b>	<b>53</b>

# Rozdział 1

## Indeks hierarchiczny

### 1.1 Hierarchia klas

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

element . . . . .	17
C . . . . .	7
E . . . . .	12
I . . . . .	22
L . . . . .	28
R . . . . .	32



## Rozdział 2

# Indeks klas

### 2.1 Lista klas

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

<a href="#">C</a>	Klasa reprezentuje kondensatory, jest klasa pochodna elementu . . . . .	<a href="#">7</a>
<a href="#">E</a>	Klasa reprezentuje źródła elektromotoryczne (SEM), jest klasa pochodna elementu . . . . .	<a href="#">12</a>
<a href="#">element</a>	Wirtualna klasa bazowa reprezentująca element i jego wartości . . . . .	<a href="#">17</a>
<a href="#">I</a>	Klasa reprezentuje źródła prądomotoryczne (SPM), jest klasa pochodna elementu . . . . .	<a href="#">22</a>
<a href="#">L</a>	Klasa reprezentuje cewki, jest klasa pochodna elementu . . . . .	<a href="#">28</a>
<a href="#">R</a>	Klasa reprezentuje rezystory, jest klasa pochodna elementu . . . . .	<a href="#">32</a>



## Rozdział 3

# Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">funkcje.cpp</a>	39
<a href="#">funkcje.h</a>	42
<a href="#">gauss.cpp</a>	47
<a href="#">klasy.cpp</a>	48
<a href="#">klasy.h</a>	49
<a href="#">main.cpp</a>	52





## Rozdział 4

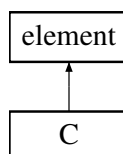
# Dokumentacja klas

### 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 tworzący kondensator.*
- std::complex< double > **wyznacz\_Z** (double &**wartosc**, double &freq)  
*Funkcja wyznaczająca impedancję kondensatora.*
- std::complex< double > **wyznacz\_Y** ()  
*Funkcja wyznaczająca admitancję elementu.*
- void **wyznacz\_I** (const std::vector< std::shared\_ptr< **element** > > &elementy, std::unordered\_map< int, std::complex< double > > &potencjaly)  
*Funkcja wyznaczająca prąd na galezi elementu.*
- void **wyznacz\_V** (std::unordered\_map< int, std::complex< double > > &potencjaly)  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- void **wyznacz\_moce** ()  
*Funkcja wyznaczająca moc dla każdego elementu.*
- double **wyznacz\_czest\_rez** (const std::vector< std::shared\_ptr< **element** > > &elementy)  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- double **czestotliwosc** ()  
*Funkcja zwracająca składową freq dla każdego 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 skadowa 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 destruktoru

#### 4.1.2.1 C()

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

Konstruktor tworzący kondensator.

## Parametry

<i>typ</i>	Typ elementu
<i>miejsce</i>	Umieszczenie elementu (wezel poczatkowy-koncowy)
<i>wartosc</i>	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

<i>brak</i>	parametrow
-------------	------------

## Zwraca

freq Czystotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow obwodu
-----------------	------------------------------------

## Zwraca

freq Czystotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow z obwodu
<i>potencjaly</i>	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

<i>brak</i>	parametrow
-------------	------------

Implementuje [element](#).

#### 4.1.3.5 wyznacz\_V()

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

Funkcja wyznaczająca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

##### Parametry

<i>potencjaly</i>	Mapa potencjalow w wezlach
-------------------	----------------------------

Implementuje [element](#).

#### 4.1.3.6 wyznacz\_Y()

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

Funkcja wyznaczająca admitancje elementu.

##### Parametry

<i>brak</i>	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 wyznaczająca impedancje kondensatora.

##### Parametry

<i>wartosc</i>	Kapacytancja
<i>freq</i>	Czestotliwosc pracy zrodel

Zwraca

impedancja

Implementuje [element](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

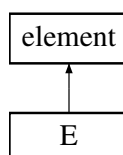
- [klasy.h](#)
- [klasy.cpp](#)

## 4.2 Dokumentacja klasy E

Klasa reprezentuje źródła 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 tworzący źródło SPM.*
- std::complex< double > [wyznacz\\_Z](#) (double &[wartosc](#), double &[freq](#))  
*Funkcja wyznaczająca impedancję źródła SEM.*
- std::complex< double > [wyznacz\\_Y](#) ()  
*Funkcja wyznaczająca admitancję elementu.*
- void [wyznacz\\_I](#) (const std::vector< std::shared\_ptr< [element](#) > > &[elementy](#), std::unordered\_map< int, std::complex< double > > &[potencjaly](#))  
*Funkcja wyznaczająca prąd na galezi elementu.*
- void [wyznacz\\_V](#) (std::unordered\_map< int, std::complex< double > > &[potencjaly](#))  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- void [wyznacz\\_moce](#) ()  
*Funkcja wyznaczająca moc dla każdego elementu.*
- double [wyznacz\\_czest\\_rez](#) (const std::vector< std::shared\_ptr< [element](#) > > &[elementy](#))  
*Funkcja wyznaczająca częstotliwość rezonansową dla każdego elementu.*
- double [czestotliwosc](#) ()  
*Funkcja zwracająca składową freq dla każdego elementu.*
- [~E](#) ()  
*Destruktor elementu.*
- virtual std::complex< double > [wyznacz\\_Z](#) (double &[wartosc](#), double &[freq](#))=0  
*Funkcja wyznaczająca impedancję elementu.*

- virtual std::complex< double > `wyznacz_Y` ()=0  
*Funkcja wyznaczająca admitancję elementu.*
- virtual void `wyznacz_I` (const std::vector< std::shared\_ptr< `element` > > &elementy, std::unordered\_map< int, std::complex< double > > &potencjaly)=0  
*Funkcja wyznaczająca prąd na galezi elementu.*
- virtual void `wyznacz_V` (std::unordered\_map< int, std::complex< double > > &potencjaly)=0  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- virtual void `wyznacz_moce` ()=0  
*Funkcja wyznaczająca moce dla każdego elementu.*
- virtual double `wyznacz_czest_rez` (const std::vector< std::shared\_ptr< `element` > > &elementy)=0  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- virtual double `czestotliwosc` ()=0  
*Funkcja zwracająca składową freq dla każdego 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 > `napięcie`
- std::complex< double > `prad`

### 4.2.1 Opis szczegółowy

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

Posiada dodatkowo zmienne double `fi` oznaczająca przesunięcie fazowe źródła oraz double `freq` oznaczająca częstotliwość źródła.

### 4.2.2 Dokumentacja konstruktora i destruktor

#### 4.2.2.1 E()

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

Konstruktor tworzący źródło SPM.

## Parametry

<i>typ</i>	Typ elementu
<i>miejsce</i>	Umieszczenie elementu (wezel poczatkowy-koncowy)
<i>wartosc</i>	Wartosc skuteczna elementu w Voltach
<i>fi</i>	Przesuniecie fazowe zrodla
<i>freq</i>	Czestotliwosc pracy zrodla

**4.2.2.2  $\sim E()$** 

```
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

<i>brak</i>	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

<i>elementy</i>	Wektor wszystkich elementow obwodu
-----------------	------------------------------------

## Zwraca

freq Czystotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow z obwodu
<i>potencjaly</i>	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

<i>brak</i>	parametrow
-------------	------------

Implementuje [element](#).

#### 4.2.3.5 wyznac\_V()

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

Funkcja wyznaczająca napięcie na każdym elemencie.

Wykorzystuje do tego obliczone wcześniej potencjały w każdym węźle obwodu.

##### Parametry

<i>potencjaly</i>	Mapa potencjałów w węzłach
-------------------	----------------------------

Implementuje [element](#).

#### 4.2.3.6 wyznac\_Y()

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

Funkcja wyznaczająca admitancję elementu.

##### Parametry

<i>brak</i>	parametrow
-------------	------------

##### Zwraca

admitancja Zwraca admitancję elementu w postaci zespolonej

Implementuje [element](#).

#### 4.2.3.7 wyznac\_Z()

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

Funkcja wyznaczająca impedancję źródła SEM.

Wykorzystuje do tego przesunięcie fazowe źródła (na płaszczyźnie kartezjańskiej)

##### Parametry

<i>wartosc</i>	Wartość skuteczna źródła
<i>freq</i>	Częstotliwość pracy źródeł

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:

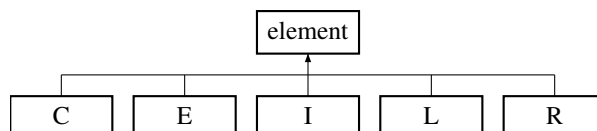
- [klasy.h](#)
- [klasy.cpp](#)

## 4.3 Dokumentacja klasy element

Wirtualna klasa bazowa reprezentująca element i jego wartości.

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

Diagram dziedziczenia dla element



### Metody publiczne

- virtual std::complex< double > [wyznacz\\_Z](#) (double &[wartosc](#), double &freq)=0  
*Funkcja wyznaczająca impedancję elementu.*
- virtual std::complex< double > [wyznacz\\_Y](#) ()=0  
*Funkcja wyznaczająca admitancję elementu.*
- virtual void [wyznacz\\_I](#) (const std::vector< std::shared\_ptr< [element](#) > > &elementy, std::unordered\_map< int, std::complex< double > > &potencjaly)=0  
*Funkcja wyznaczająca prąd na galezi elementu.*
- virtual void [wyznacz\\_V](#) (std::unordered\_map< int, std::complex< double > > &potencjaly)=0  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- virtual void [wyznacz\\_moce](#) ()=0  
*Funkcja wyznaczająca moce dla każdego elementu.*
- virtual double [wyznacz\\_czest\\_rez](#) (const std::vector< std::shared\_ptr< [element](#) > > &elementy)=0  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- virtual double [czestotliwosc](#) ()=0  
*Funkcja zwracająca składową freq dla każdego 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 reprezentująca element i jego wartości.

Klasa przechowuje informacje o elemencie układu:

- typ (rezystor, źródło SEM, źródło SPM, kondensator, cewka)
- umiejscowienie (węzeł początkowy, węzeł końcowy)
- wartosc
- odłożone napięcie
- prad przepływający przez element
- moc wydzielona Posiada funkcje wirtualne obliczające impedancje, admitancje, prad, napięcie, czestotliwosc rezonansowa elementu.

### 4.3.2 Dokumentacja funkcji składowych

#### 4.3.2.1 czestotliwosc()

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

Funkcja zwracająca składowa freq dla każdego elementu.

#### Parametry

<i>brak</i>	parametrow
-------------	------------

#### Zwraca

freq Czystotliwosc 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 wyznaczająca czestotliwosc rezonansowa dla kazdego elementu.

##### Parametry

<i>elementy</i>	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 wyznaczająca prąd na galezi elementu.

Wykorzystuje do tego obliczone wcześniej potencjaly w kazdym wezle obwodu.

##### Parametry

<i>elementy</i>	Wektor wszystkich elementow z obwodu
<i>potencjaly</i>	Mapa potencjalow w wezлах

Implementowany w [E](#), [I](#), [R](#), [C](#) i [L](#).

#### 4.3.2.4 wyznacz\_moce()

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

Funkcja wyznaczająca 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

<i>brak</i>	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 wyznaczająca napięcie na każdym elemencie.

Wykorzystuje do tego obliczone wcześniej potencjały w każdym węzle obwodu.

## Parametry

<i>potencjaly</i>	Mapa potencjałów w węzłach
-------------------	----------------------------

Implementowany w [E](#), [I](#), [R](#), [C](#) i [L](#).

**4.3.2.6 wyznacz\_Y()**

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

Funkcja wyznaczająca admitancję elementu.

## Parametry

<i>brak</i>	parametrow
-------------	------------

## Zwraca

admitancja Zwraca admitancję 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 wyznaczająca impedancję elementu.

## Parametry

<i>wartosc</i>	Wartosc parametru podana z pliku tekstowego
<i>freq</i>	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 prąd

```
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:

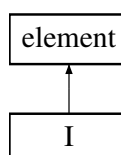
- [klasy.h](#)

## 4.4 Dokumentacja klasy I

Klasa reprezentuje źródła prądomotoryczne (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 tworzący źródło SEM.*
- std::complex< double > `wyznacz_Z` (double &`wartosc`, double &`freq`)  
*Funkcja wyznaczająca impedancję źródła SPM.*
- std::complex< double > `wyznacz_Y` ()  
*Funkcja wyznaczająca admitancję elementu.*
- void `wyznacz_I` (const std::vector< std::shared\_ptr< `element` > > &`elementy`, std::unordered\_map< int, std::complex< double > > &`potencjaly`)  
*Funkcja wyznaczająca prąd na galezi elementu.*
- void `wyznacz_V` (std::unordered\_map< int, std::complex< double > > &`potencjaly`)  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- void `wyznacz_moce` ()  
*Funkcja wyznaczająca moce dla każdego elementu.*
- double `wyznacz_czest_rez` (const std::vector< std::shared\_ptr< `element` > > &`elementy`)  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- double `czestotliwosc` ()  
*Funkcja zwracająca składową freq dla każdego elementu.*
- `~I` ()  
*Destruktor elementu.*
- virtual std::complex< double > `wyznacz_Z` (double &`wartosc`, double &`freq`)=0  
*Funkcja wyznaczająca impedancję elementu.*
- virtual std::complex< double > `wyznacz_Y` ()=0  
*Funkcja wyznaczająca admitancję elementu.*
- virtual void `wyznacz_I` (const std::vector< std::shared\_ptr< `element` > > &`elementy`, std::unordered\_map< int, std::complex< double > > &`potencjaly`)=0  
*Funkcja wyznaczająca prąd na galezi elementu.*
- virtual void `wyznacz_V` (std::unordered\_map< int, std::complex< double > > &`potencjaly`)=0  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- virtual void `wyznacz_moce` ()=0  
*Funkcja wyznaczająca moce dla każdego elementu.*
- virtual double `wyznacz_czest_rez` (const std::vector< std::shared\_ptr< `element` > > &`elementy`)=0  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- virtual double `czestotliwosc` ()=0  
*Funkcja zwracająca składową freq dla każdego 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 > `napięcie`
- std::complex< double > `prad`

### 4.4.1 Opis szczegółowy

Klasa reprezentuje źródła prądomotoryczne (SPM), jest klasa pochodna elementu.

Posiada dodatkowo zmienne double *fi* oznaczająca przesunięcie fazowe źródła oraz double *freq* oznaczająca częstotliwość źródła.

### 4.4.2 Dokumentacja konstruktora i destruktor

#### 4.4.2.1 I()

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

Konstruktor tworzący źródło SEM.

#### Parametry

<i>typ</i>	Typ elementu
<i>miejsce</i>	Umieszczenie elementu (węzeł początkowy-koncowy)
<i>wartosc</i>	Wartość skuteczna elementu w Amperach
<i>fi</i>	Przesunięcie fazowe źródła
<i>freq</i>	Częstotliwość pracy źródła

#### 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 zwracająca składową *freq* dla każdego elementu.

## Parametry

<i>brak</i>	parametrow
-------------	------------

## Zwraca

freq Czystotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow obwodu
-----------------	------------------------------------

## Zwraca

freq Czystotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow z obwodu
<i>potencjaly</i>	Mapa potencjalow w wezlach

Implementuje [element](#).

#### 4.4.3.4 wyznacz\_moce()

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

Funkcja wyznaczająca moce dla każdego elementu.

Oblicza moc czynna i bierna dla każdego elementu. Jeśli moc jest dodatnia, to znaczy, że element pobiera energię. Jeśli ujemna, to znaczy, że ją oddaje.

##### Parametry

<i>brak</i>	parametrow
-------------	------------

Implementuje [element](#).

#### 4.4.3.5 wyznacz\_V()

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

Funkcja wyznaczająca napięcie na każdym elemencie.

Wykorzystuje do tego obliczone wcześniej potencjały w każdym węzle obwodu.

##### Parametry

<i>potencjaly</i>	Mapa potencjałów w węzłach
-------------------	----------------------------

Implementuje [element](#).

#### 4.4.3.6 wyznacz\_Y()

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

Funkcja wyznaczająca admitancję elementu.

##### Parametry

<i>brak</i>	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 wyznaczająca impedancje zrodla SPM.

Wykorzystuje do tego przesuniecie fazowe zrodla (na plaszczyznie kartezjanskiej)

**Parametry**

<i>wartosc</i>	Wartosc skuteczna zrodla
<i>freq</i>	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:

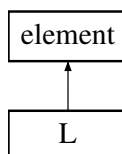
- [klasy.h](#)
- [klasy.cpp](#)

## 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 tworzący cewkę.*
- `std::complex< double > wyznacz_Z (double &wartosc, double &freq)`  
*Funkcja wyznaczająca impedancję cewki.*
- `std::complex< double > wyznacz_Y ()`  
*Funkcja wyznaczająca admitancję elementu.*
- `void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)`  
*Funkcja wyznaczająca prąd na galezi elementu.*
- `void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)`  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- `void wyznacz_moce ()`  
*Funkcja wyznaczająca moce dla każdego elementu.*
- `double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)`  
*Funkcja wyznaczająca częstotliwość rezonansową dla każdego elementu.*
- `double czestotliwosc ()`  
*Funkcja zwracająca składową freq dla każdego elementu.*
- `~L ()`  
*Destruktor elementu.*
- `virtual std::complex< double > wyznacz_Z (double &wartosc, double &freq)=0`  
*Funkcja wyznaczająca impedancję elementu.*
- `virtual std::complex< double > wyznacz_Y ()=0`  
*Funkcja wyznaczająca admitancję elementu.*
- `virtual void wyznacz_I (const std::vector< std::shared_ptr< element > > &elementy, std::unordered_map< int, std::complex< double > > &potencjaly)=0`  
*Funkcja wyznaczająca prąd na galezi elementu.*
- `virtual void wyznacz_V (std::unordered_map< int, std::complex< double > > &potencjaly)=0`  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- `virtual void wyznacz_moce ()=0`  
*Funkcja wyznaczająca moce dla każdego elementu.*
- `virtual double wyznacz_czest_rez (const std::vector< std::shared_ptr< element > > &elementy)=0`  
*Funkcja wyznaczająca częstotliwość rezonansową dla każdego elementu.*
- `virtual double czestotliwosc ()=0`  
*Funkcja zwracająca składową freq dla każdego 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 tworzący cewkę.

Parametry

<i>typ</i>	Typ elementu
<i>miejsce</i>	Umiejscowienie elementu (węzeł początkowy-koncowy)
<i>wartosc</i>	Wartość 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 zwracająca składowa freq dla każdego elementu.

Dla cewki zwróci NULL

##### Parametry

<i>brak</i>	parametrow
-------------	------------

##### Zwraca

freq Czystotliwosc 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 wyznaczająca czestotliwosc rezonansowa dla każdego elementu.

rezonans -> 1/sqrt(LC)

##### Parametry

<i>elementy</i>	Wektor wszystkich elementow obwodu
-----------------	------------------------------------

##### Zwraca

freq Czystotliwosc 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 wyznaczająca prąd na gałęzi elementu.

Wykorzystuje do tego obliczone wcześniej potencjały w każdym węzle obwodu i admitancje

Parametry

<i>elementy</i>	Wektor wszystkich elementów z obwodu
<i>potencjaly</i>	Mapa potencjałów w węzłach

Implementuje [element](#).

#### 4.5.3.4 wyznacz\_moce()

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

Funkcja wyznaczająca moc dla każdego elementu.

Oblicza moc czynną i bierną dla każdego elementu. Jeśli moc jest dodatnia, to znaczy, że element pobiera energię. Jeśli ujemna, to znaczy, że ją oddaje.

Parametry

<i>brak</i>	parametrow
-------------	------------

Implementuje [element](#).

#### 4.5.3.5 wyznacz\_V()

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

Funkcja wyznaczająca napięcie na każdym elemencie.

Wykorzystuje do tego obliczone wcześniej potencjały w każdym węzle obwodu.

Parametry

<i>potencjaly</i>	Mapa potencjałów w węzłach
-------------------	----------------------------

Implementuje [element](#).

#### 4.5.3.6 wyznacz\_Y()

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

Funkcja wyznaczająca admitancje elementu.

#### Parametry

<i>brak</i>	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 wyznaczająca impedancje cewki.

#### Parametry

<i>wartosc</i>	Przewodnosc
<i>freq</i>	Czestotliwosc pracy zrodel

#### Zwraca

impedancja

Implementuje [element](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

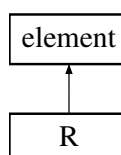
- [klasy.h](#)
- [klasy.cpp](#)

## 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 tworzący rezystor.*
- std::complex< double > `wyznacz_Z` (double &`wartosc`, double &`freq`)  
*Funkcja wyznaczająca impedancję rezystora.*
- std::complex< double > `wyznacz_Y` ()  
*Funkcja wyznaczająca admitancję elementu.*
- void `wyznacz_I` (const std::vector< std::shared\_ptr< `element` > > &`elementy`, std::unordered\_map< int, std::complex< double > > &`potencjaly`)  
*Funkcja wyznaczająca prąd na galezi elementu.*
- void `wyznacz_V` (std::unordered\_map< int, std::complex< double > > &`potencjaly`)  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- void `wyznacz_moce` ()  
*Funkcja wyznaczająca moc dla każdego elementu.*
- double `wyznacz_czest_rez` (const std::vector< std::shared\_ptr< `element` > > &`elementy`)  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- double `czestotliwosc` ()  
*Funkcja zwracająca składową freq dla każdego elementu.*
- `~R` ()  
*Destruktor elementu.*
- virtual std::complex< double > `wyznacz_Z` (double &`wartosc`, double &`freq`)=0  
*Funkcja wyznaczająca impedancję elementu.*
- virtual std::complex< double > `wyznacz_Y` ()=0  
*Funkcja wyznaczająca admitancję elementu.*
- virtual void `wyznacz_I` (const std::vector< std::shared\_ptr< `element` > > &`elementy`, std::unordered\_map< int, std::complex< double > > &`potencjaly`)=0  
*Funkcja wyznaczająca prąd na galezi elementu.*
- virtual void `wyznacz_V` (std::unordered\_map< int, std::complex< double > > &`potencjaly`)=0  
*Funkcja wyznaczająca napięcie na każdym elemencie.*
- virtual void `wyznacz_moce` ()=0  
*Funkcja wyznaczająca moc dla każdego elementu.*
- virtual double `wyznacz_czest_rez` (const std::vector< std::shared\_ptr< `element` > > &`elementy`)=0  
*Funkcja wyznaczająca częstotliwość rezonansowa dla każdego elementu.*
- virtual double `czestotliwosc` ()=0  
*Funkcja zwracająca składową freq dla każdego 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 > `napięcie`
- 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 tworzący rezystor.

##### Parametry

<i>typ</i>	Typ elementu
<i>miejsce</i>	Umieszczenie elementu (węzeł początkowy-koncowy)
<i>wartosc</i>	Wartość 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 zwracająca składową freq dla każdego elementu.

Dla rezystora zwróci NULL

##### Parametry

<i>brak</i>	parametrow
-------------	------------

Zwraca

freq Czesotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow obwodu
-----------------	------------------------------------

Zwraca

freq Czesotliwosc 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

<i>elementy</i>	Wektor wszystkich elementow z obwodu
<i>potencjaly</i>	Mapa potencjalow w wezlach

Implementuje [element](#).

#### 4.6.3.4 wyznacz\_moce()

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

Funkcja wyznaczająca 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

<i>brak</i>	parametrow
-------------	------------

Implementuje [element](#).

#### 4.6.3.5 wyznacz\_V()

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

Funkcja wyznaczająca napiecie na kazdym elemencie.

Wykorzystuje do tego obliczone wczesniej potencjaly w kazdym wezle obwodu.

#### Parametry

<i>potencjaly</i>	Mapa potencjalow w wezlach
-------------------	----------------------------

Implementuje [element](#).

#### 4.6.3.6 wyznacz\_Y()

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

Funkcja wyznaczająca admitancje elementu.

#### Parametry

<i>brak</i>	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 wyznaczająca impedancje rezystora.

##### Parametry

<i>wartosc</i>	Opornosc rezystora
<i>freq</i>	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 (układ równań) za pomocą metody Coltrięgo.

Funkcja wprowadza do macierzy admitancje każdego potencjału używając metody potencjałów węzłowych. Napotkane galezie ze źródłem elektromotorycznym traktuje jako galezie z impedancją {1,0}Ω. Funkcja tworzy także mapę węzłów na podstawie kontenera utworzonego w [wektor\\_wezlow\(\)](#), tak aby zaczynały się od 0 do n.

#### Zwraca

Para macierzy (wektora wektorów wskaźników) i mapy stare2nowe

#### Parametry

in	<i>elementy</i>	Wektor elementów (wskaźników na element)
in	<i>wezly</i>	Kontener węzłów 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 gałęzi obok SEM.

Funkcja dodaje rezystor o wartości -1 obok gałęzi ze źródłem elektromotorycznym, aby później traktować gałęzi z SEM jako gałęzi z opornością 1 Ω.

#### Zwraca

Wektor elementów (wskaźników na element) z nowymi ujemnymi rezystorami

## Parametry

in	<i>elementy</i>	Wektor elementow (wskaznikow na element)
in	<i>Kontener</i>	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	<i>nazwa_pliku</i>	Nazwa pliku wejscowego
----	--------------------	------------------------

## 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	<i>elementy</i>	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 wyjścia charakterystykę każdego elementu.

Funkcja wypisuje wszystkie elementy obwodu, ich parametry i bilans obwodu do pliku tekstowego. Prąd oraz napięcie funkcja podaje w postaci modułu liczby zespolonej oraz wzoru Eulera na kąt przesunięcia fazowego dla każdego elementu. Funkcja oblicza bilans mocy na podstawie mocy oddanej lub pobranej dla każdego elementu.

#### Parametry

in	<i><code>nazwa_pliku</code></i>	Nazwa pliku wyjściowego
in	<i><code>elementy</code></i>	Wektor elementów

## 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 główna 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 węzłów 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 (układ równan) za pomocą metody Coltrięgo.*
- 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 pomocą 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 wyjścia charakterystykę każdego 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 (układ 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	<i>elementy</i>	Wektor elementow (wskaznikow na element)
in	<i>wezly</i>	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	<i>elementy</i>	Wektor elementow (wskaznikow na element)
in	<i>Kontener</i>	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	<i>uklad_rownan_i_mapa</i>	Para macierzy i mapy stare2nowe
in	<i>wezly</i>	Kontenera wezlow obwodu

## 5.2.2.4 main()

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

Funkcja glowna programu.

## Zwraca

int

## Parametry

in	<i>liczba_param</i>	Liczba parametrow
in	<i>param[ ]</i>	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	<i>nazwa_pliku</i>	Nazwa pliku wejscowego
----	--------------------	------------------------

#### 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 największych wartosci.

#### Zwraca

Kontener wezlow obwodu

#### Parametry

in	<i>elementy</i>	Wektor elementow (wskaznikow na element)
----	-----------------	--

### 5.2.2.7 zapis\_wyjścia()

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

Funkcja wypisuje do pliku wyjścia charakterystykę każdego elementu.

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



## Parametry

in	<i>nazwa_pliku</i>	Nazwa pliku wyjściowego
in	<i>elementy</i>	Wektor elementów

## 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>>&
00053 elementy, const std::set<int>& wezly);
00063 std::pair<macierz, std::unordered_map<int, int>> coltri(const std::vector<std::shared_ptr<element>>&
00064 elementy, const std::set<int>& wezly);
00065 //void wypisz(const macierz& potencjaly);
00066
00077 std::unordered_map<int, std::complex<double>> gauss(const std::pair<macierz, std::unordered_map<int,
00078 int>& uklad_rownan_i_mapa, const std::set<int>& wezly);
00087 void zapis_wyjscia(const std::string& nazwa_pliku, const std::vector<std::shared_ptr<element>>&
00088 elementy);
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	<b>uklad_rownan_i_mapa</b>	Para macierzy i mapy stare2nowe
in	<b>wezly</b>	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 reprezentująca element i jego wartości.*
- class [E](#)  
*Klasa reprezentuje źródła elektromotoryczne (SEM), jest klasą pochodną elementu.*
- class [I](#)  
*Klasa reprezentuje źródła prądomotoryczne (SPM), jest klasą pochodną elementu.*
- class [R](#)  
*Klasa reprezentuje rezystory, jest klasą pochodną elementu.*
- class [C](#)  
*Klasa reprezentuje kondensatory, jest klasą pochodną elementu.*
- class [L](#)  
*Klasa reprezentuje cewki, jest klasą pochodną 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	<i>liczba_param</i>	Liczba parametrow
in	<i>param</i> []	Tablica parametrow

# Skorowidz

- `_USE_MATH_DEFINES`
    - `funkcje.cpp`, [40](#)
    - `klasy.cpp`, [48](#)
  - `~C`
    - `C`, [9](#)
  - `~E`
    - `E`, [14](#)
  - `~I`
    - `I`, [24](#)
  - `~L`
    - `L`, [29](#)
  - `~R`
    - `R`, [34](#)
- `admitancja`
  - `element`, [21](#)
- `C`, [7](#)
  - `~C`, [9](#)
  - `C`, [8](#)
  - `czestotliwosc`, [9](#)
  - `wyznacz_czest_rez`, [9](#)
  - `wyznacz_I`, [10](#)
  - `wyznacz_moce`, [10](#)
  - `wyznacz_V`, [10](#)
  - `wyznacz_Y`, [11](#)
  - `wyznacz_Z`, [11](#)
- `coltri`
  - `funkcje.cpp`, [40](#)
  - `funkcje.h`, [43](#)
- `czest_rez`
  - `element`, [21](#)
- `czestotliwosc`
  - `C`, [9](#)
  - `E`, [14](#)
  - `element`, [18](#)
  - `I`, [24](#)
  - `L`, [30](#)
  - `R`, [34](#)
- `dodaj_rez_obok_sem`
  - `funkcje.cpp`, [40](#)
  - `funkcje.h`, [43](#)
- `E`, [12](#)
  - `~E`, [14](#)
  - `czestotliwosc`, [14](#)
  - `E`, [13](#)
  - `fi`, [17](#)
  - `freq`, [17](#)
- `wyznacz_czest_rez`, [14](#)
- `wyznacz_I`, [15](#)
- `wyznacz_moce`, [15](#)
- `wyznacz_V`, [15](#)
- `wyznacz_Y`, [16](#)
- `wyznacz_Z`, [16](#)

e

- `klasy.h`, [50](#)

element, [17](#)

- `admitancja`, [21](#)
- `czest_rez`, [21](#)
- `czestotliwosc`, [18](#)
- `impedancja`, [21](#)
- `moc_bierna`, [21](#)
- `moc_czynna`, [21](#)
- `napiecie`, [21](#)
- `prad`, [22](#)
- `typ`, [22](#)
- `umiejscowienie`, [22](#)
- `wartosc`, [22](#)
- `wyznacz_czest_rez`, [18](#)
- `wyznacz_I`, [19](#)
- `wyznacz_moce`, [19](#)
- `wyznacz_V`, [20](#)
- `wyznacz_Y`, [20](#)
- `wyznacz_Z`, [20](#)

fi

- `E`, [17](#)
- `I`, [27](#)

freq

- `E`, [17](#)
- `I`, [27](#)

funkcje.cpp, [39](#)

- `_USE_MATH_DEFINES`, [40](#)
- `coltri`, [40](#)
- `dodaj_rez_obok_sem`, [40](#)
- `odczyt_wejscia`, [41](#)
- `wektor_wezlow`, [41](#)
- `zapis_wyjscia`, [41](#)

funkcje.h, [42](#)

- `coltri`, [43](#)
- `dodaj_rez_obok_sem`, [43](#)
- `FUNKCJE_H`, [43](#)
- `gauss`, [44](#)
- `main`, [44](#)
- `odczyt_wejscia`, [45](#)
- `wektor_wezlow`, [45](#)
- `zapis_wyjscia`, [45](#)

FUNKCJE\_H

funkcje.h, [43](#)

gauss  
  funkcje.h, [44](#)  
  gauss.cpp, [48](#)

gauss.cpp, [47](#)  
  gauss, [48](#)

I, [22](#)  
   $\sim I$ , [24](#)  
  czestotliwosc, [24](#)  
  fi, [27](#)  
  freq, [27](#)  
  I, [24](#)  
  wyznacz\_czest\_rez, [25](#)  
  wyznacz\_I, [25](#)  
  wyznacz\_moce, [26](#)  
  wyznacz\_V, [26](#)  
  wyznacz\_Y, [26](#)  
  wyznacz\_Z, [27](#)

impedancja  
  element, [21](#)

jeden  
  klasy.cpp, [49](#)

klasy.cpp, [48](#)  
  \_USE\_MATH\_DEFINES, [48](#)  
  jeden, [49](#)

klasy.h, [49](#)  
  e, [50](#)  
  KLASY\_H, [50](#)  
  macierz, [50](#)

KLASY\_H  
  klasy.h, [50](#)

L, [28](#)  
   $\sim L$ , [29](#)  
  czestotliwosc, [30](#)  
  L, [29](#)  
  wyznacz\_czest\_rez, [30](#)  
  wyznacz\_I, [30](#)  
  wyznacz\_moce, [31](#)  
  wyznacz\_V, [31](#)  
  wyznacz\_Y, [31](#)  
  wyznacz\_Z, [32](#)

macierz  
  klasy.h, [50](#)

main  
  funkcje.h, [44](#)  
  main.cpp, [52](#)

main.cpp, [52](#)  
  main, [52](#)

moc\_bierna  
  element, [21](#)

moc\_czynna  
  element, [21](#)

napiecie

element, [21](#)

odczyt\_wejscia  
  funkcje.cpp, [41](#)  
  funkcje.h, [45](#)

prad  
  element, [22](#)

R, [32](#)  
   $\sim R$ , [34](#)  
  czestotliwosc, [34](#)  
  R, [34](#)  
  wyznacz\_czest\_rez, [35](#)  
  wyznacz\_I, [35](#)  
  wyznacz\_moce, [35](#)  
  wyznacz\_V, [36](#)  
  wyznacz\_Y, [36](#)  
  wyznacz\_Z, [36](#)

typ  
  element, [22](#)

umiescowienie  
  element, [22](#)

wartosc  
  element, [22](#)

wektor\_wezlow  
  funkcje.cpp, [41](#)  
  funkcje.h, [45](#)

wyznacz\_czest\_rez  
  C, [9](#)  
  E, [14](#)  
  element, [18](#)  
  I, [25](#)  
  L, [30](#)  
  R, [35](#)

wyznacz\_I  
  C, [10](#)  
  E, [15](#)  
  element, [19](#)  
  I, [25](#)  
  L, [30](#)  
  R, [35](#)

wyznacz\_moce  
  C, [10](#)  
  E, [15](#)  
  element, [19](#)  
  I, [26](#)  
  L, [31](#)  
  R, [35](#)

wyznacz\_V  
  C, [10](#)  
  E, [15](#)  
  element, [20](#)  
  I, [26](#)  
  L, [31](#)  
  R, [36](#)

wyznacz\_Y



C, [11](#)  
E, [16](#)  
element, [20](#)  
I, [26](#)  
L, [31](#)  
R, [36](#)  
wyznacz\_Z  
C, [11](#)  
E, [16](#)  
element, [20](#)  
I, [27](#)  
L, [32](#)  
R, [36](#)  
zapis\_wyjścia  
funkcje.cpp, [41](#)  
funkcje.h, [45](#)