9.12.2021r.

Mgr Inż. Maurycy Mański

Symulator obwodów elektrycznych I

1. Wstęp i cel pracy

Celem pracy było wykonanie symulatora obliczającego liniowe obwody elektryczne. Zadanie te miało rozwinąć umiejętności programistyczne w języku Python(3) oraz przypomnieć metody rozwiązywania układów elektrycznych (dostępne jest darmowe oprogramowanie posiadające zbliżoną funkcjonalność jak np. „Ltspice”). Na ten moment elementy obwodowe zostały ograniczone do idealnych źródeł napięciowych oraz rezystancji. Skrypt wyznacza spadki napięcia na rezystancjach gałęziowych oraz prądy gałęziowe w analizie stałoprądowej. Problematyką projektu jest wprowadzenie danych obwodu do macierzy korzystając z praw teorii obwodów, rozwiązanie równania macierzowego oraz zbadanie dla jakich danych wejściowych mogłyby pojawić się błędy w programie.

2. Narzędzia

Do projektu wykorzystano następujące narzędzia:

- interpreter języka Python 3.8.10,
- edytor tekstu „Vim”,
- Linux Mint 20.

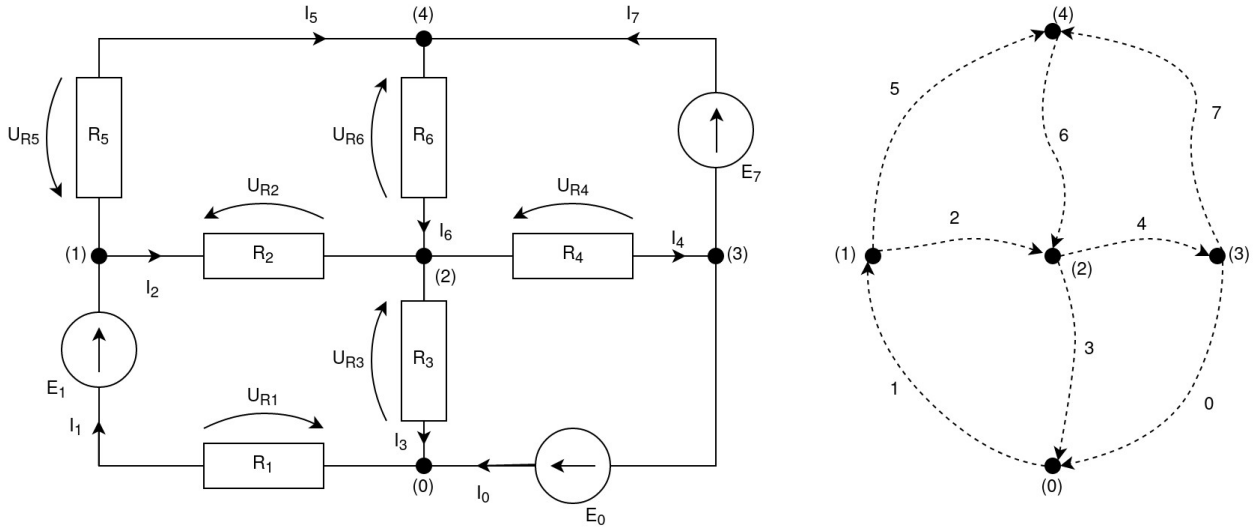
3. Podstawy teoretyczne

Do zrozumienia opisu działania programu konieczna będzie znajomość kilku pojęć z tematyki układów elektrycznych:

- **Obwód** jest to układ elektryczny posiadający jedno lub wiele oczek, który można opisać za pomocą schematu elektrycznego.
- **Oczko** jest zbiorem gałęzi stanowiącym zamkniętą drogę dla przepływu prądu.
- **Gałąź** jest szeregowym połączeniem elementów pasywnych (rezystancja, pojemność, indukcyjność) i aktywnych (idealne źródło napięciowe/prądowe) stanowiącym dwójnik (element dwukońcówkowy). Posiada ona 2 węzły na brzegach, które są łączone (współdzielone)

z pozostałymi gałęziami. Brak połączenia gałęzi po obu stronach powoduje brak możliwości przepływu prądu (prąd gałęziowy zerowy oraz spadki napięcia na elementach pasywnych również zerowe), co jest równoważne z odrzuceniem gałęzi podczas obliczeń.

- **Węzeł** jest punktem połączenia gałęzi.
- **Graf prądowy** jest uproszczeniem schematu obwodu, w którym zostały usunięte elementy pasywne i aktywne – pozostają tylko połączenia z oznaczonym kierunkiem przepływu prądu oraz węzły. Grafy prądowe są przydatne przy tworzeniu komputerowych symulacji obwodów.



- W obwodzie występuje liczba **w** węzłów, **g** gałęzi oraz **o = g - (w-1)** liniowo niezależnych oczek.

Lewa strona obrazka przedstawia schemat przykładowego obwodu a prawa - jego graf prądowy. Węzły są oznaczone jako kropki i zostały ponumerowane z użyciem nawiasów. Każda gałąź posiada swój własny prąd (plus ewentualnie rezystancję, idealne źródło napięciowe) z indeksem równym numerowi gałęzi. Obwód posiada 8 gałęzi (g=8), 5 węzłów (w=5) oraz 4 liniowo niezależne oczka (o=4). Wyznaczenie spadków napięcia na rezystancjach i prądów gałęziowych jest równoważne z obliczeniem przedstawionego obwodu.

Macierze w programie są uzupełniane na podstawie następujących praw teorii obwodów:

- **Prądowe prawo Kirchhofa (PPK)** – mówi, że suma prądów wpływających do węzła jest równa sumie prądów z niego wypływających.
- **Napięciowe prawo Kirchhofa (NPK)** – mówi, że w danym oczku suma spadków napięć na elementach obwodowych jest równa sumie napięć sił elektromotorycznych. Rozważając konkretne oczko należy przyjąć kierunek (zgodnie ze wskazówkami zegara lub przeciwnie) dla źródeł napięciowych oraz przeciwny do niego dla spadków napięcia. Jeżeli strzałka napięcia elementu obwodowego zgadza się z przyjętym kierunkiem to wartość wprowadzana jest do równania ze znakiem + a jeżeli nie to -.
- **Prawo Ohma (równanie definicyjne dla rezystancji)** – prawo wprowadza pojęcie rezystancji czyli elementu obwodowego, na którym zawsze napięcie jest liniowo proporcjonalne do prądu zgodnie z jej wartością:

  $R = U / I$

R – rezystancja [Ω]

U – napięcie (spadek napięcia) na rezystancji [U]

I – prąd przepływający przez rezystancję [A]

Aby zachować powyższą równość, kierunek strzałki napięcia U na rezystancji R jest zawsze przeciwny do kierunku przepływu prądu I.

- Do obliczenia obwodu należy wyprowadzić **w-1 równań PPK, o równań NPK oraz g równań definicyjnych** (prawa Ohma). Łącznie musi powstać 2g równań, które przy założeniu ich liniowej niezależności (która jest konieczna do otrzymania pojedynczego, poprawnego wyniku) umożliwiają obliczenie 2g niewiadomych - wartość prądu i spadku napięcia na rezystancji dla każdej gałęzi.
- Zachowanie powyższej liczby równań dla ich poszczególnych typów oraz wybór liniowo niezależnych o oczek do równań NPK gwarantuje liniową niezależność układu równań. Jeżeli wybrane oczka są zależne liniowo to przynajmniej jedno oczko jest kombinacją liniową pozostałych, przez co w rezultacie otrzymamy (błędnie) nieskończoną liczbę rozwiązań. Liniowo niezależne oczka można odczytać ze schematu wybierając możliwie najmniejsze oczka.

Uwagi dodatkowe:

- Prąd na wszystkich elementach w gałęzi jest jednakowy a ich kolejność nie wpływa na rozwiązanie obwodu (połączenie szeregowe).
- Można założyć dowolny kierunek prądu gałęzi (jego odwrócenie spowoduje otrzymanie wyniku o wartości przeciwnej co jest sytuacją równoznaczną). Należy pamiętać zawsze o przeciwnym strzałkowaniu spadków napięcia na elementach pasywnych do założonego kierunku prądu!

Przykładowy układ równań obwodu z rysunku:

Typ równania	Numer równania	Równanie
PPK	1.	$I_0 + I_3 – I_1 = 0$
	2.	$I_1 - I_2 – I_5 = 0$
	3.	$I_2 + I_6 - I_3 – I_4 = 0$
	4.	$I_4 - I_0 – I_7 = 0$
NPK	5.	$U_{R1} + U_{R2} + U_{R3} = E_1$
	6.	$-U_{R3} + U_{R4} = E_0$
	7.	$-U_{R2} +U_{R5} + U_{R6} = 0$
	8.	$U_{R4} + U_{R6} = E_7$
Równania definicyjne	9.	$-I_0 * R_0 + U_{R0} = 0$
	10.	$-I_1 * R_1 + U_{R1} = 0$
	11.	$-I_2 * R_2 + U_{R2} = 0$
	12.	$-I_3 * R_3 + U_{R3} = 0$
	13.	$-I_4 * R_4 + U_{R4} = 0$
	14.	$-I_5 * R_5 + U_{R5} = 0$
	15.	$-I_6 * R_6 + U_{R6} = 0$
	16.	$-I_7 * R_7 + U_{R7} = 0$

Równania zostały zapisane w taki sposób, aby po lewej stronie znajdowały się niewiadome (prądy gałęziowe, spadki napięcia na rezystancjach) a po prawej jedynie wartości. W PPK prądy wyjściowe zostały przeniesione na lewą stronę z przeciwnym znakiem a równania definicyjne przekształcono następująco R = U / I => -I * R + U = 0. Możliwe jest otrzymanie wartości niezerowej jedynie w równaniach NPK. Przedstawiona forma układu równań umożliwia łatwe przekształcenie do postaci równania macierzowego.

$$A=\begin{pmatrix} PPK_{1,1} & PPK_{1,2} & \cdots & PPK_{1,2g} \\ \vdots & \vdots & \ddots & \vdots \\ PPK_{w-1,1} & PPK_{w-1,2} & \cdots & PPK_{w-1,2g} \\ NPK_{w,1} & NPK_{w,2} & \cdots & NPK_{w,2g} \\ \vdots & \vdots & \ddots & \vdots \\ NPK_{g,1} & NPK_{g,2} & \cdots & NPK_{g,2g} \\ D_{g+1,1} & D_{g+1,2} & \cdots & D_{g+1,2g} \\ \vdots & \vdots & \ddots & \vdots \\ D_{2g,1} & D_{2g,2} & \cdots & D_{2g,2g} \end{pmatrix} \quad X=\begin{pmatrix} I_1 \\ \vdots \\ I_g \\ U_{g+1} \\ \vdots \\ U_{2g} \end{pmatrix} \quad B=\begin{pmatrix} PPK'_1=0 \\ \vdots \\ PPK'_{w-1}=0 \\ NPK'_w \\ \vdots \\ NPK'_g \\ D'_{g+1}=0 \\ \vdots \\ D'_{2g}=0 \end{pmatrix}$$

$$A*X=B$$

Pierwsza połowa wektora X to poukładane w kolejności (od 1 do g) nieznane prądy gałęziowe a drugą - odpowiadające im spadki napięcia na rezystancjach. Każdemu równaniu odpowiada wiersz o tym samym indeksie macierzy A i B. Współczynniki przy niewiadomych w lewej części równania zostają zapisane jako współczynniki macierzy A a wartość z prawej strony jest wprowadzana do wektora B. Uzupełniona macierz A i B powyższymi równaniami:

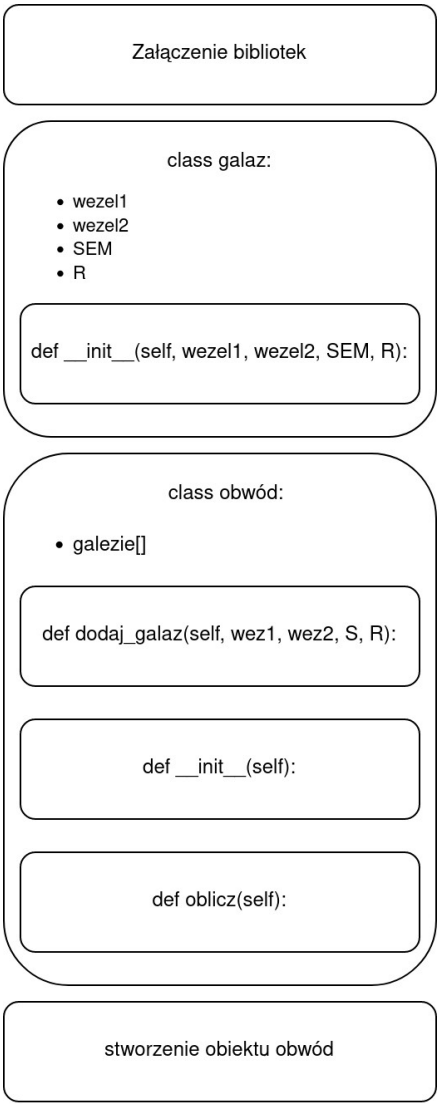
$$A=\begin{pmatrix} 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ -R_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -R_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -R_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -R_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -R_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -R_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -R_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad B=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ E_1 \\ E_0 \\ 0 \\ E_7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Rozwiązaniem równania macierzowego A * X = B z powyższymi macierzami jest wektor X, czyli wartości wszystkich prądów gałęziowych oraz spadków napięcia na rezystancjach obwodu z rysunku.

4. Opis programu

Wstęp:

Symulator został napisany w ten sposób, aby uruchamiając go jako skrypt (samodzielnie uruchomiony plik) użytkownik prowadził interakcje z programem przy pomocy terminala - wprowadzał i wyświetlał dane. W momencie załączenia pliku jako moduł, dane obwodowe będą wprowadzane z użyciem metody a obliczone parametry będą zwracane z użyciem innej metody. Taki zabieg ma ułatwić stworzenie w przyszłości interfejsu graficznego, który mógłby być rozwijany w osobnym pliku. Program umieszcza dane wejściowe do zdefiniowanej struktury, przetwarza dane w celu wyprowadzenia równań PPK, NPK i definicyjnych, wprowadza je do macierzy a następnie rozwiązuje równanie macierzowe. Należy zaznaczyć, że w programie macierze będą numerowane od 0 a nie od 1 jak zostało to zaprezentowane w części teoretycznej. Struktura pliku została zaprezentowana poniżej.



Biblioteki:

Na początku pliku zostaje załączona biblioteka „NumPy”, która wprowadza nowy typ zmiennych – macierze oraz funkcje wykonujące operacje na tym typie danych. NumPy udostępnia również funkcje

realizujące algorytmy takie jak np. rozwiązywanie równania macierzowego. Skorzystano również z biblioteki „SymPy”, która umożliwia sprawdzanie niezależności liniowej wierszy macierzy. Podczas pracy nad symulatorem posłużono się biblioteką „time” do pomiaru czasu wykonywania metody oblicz(), dążąc do skrócenia czasu jej wykonywania.

Klasa galaz, struktura danych obwodowych:

Strukturą do której wprowadzane będą dane obwodowe to klasa „galaz”. Poszczególna gałąź obwodu zostanie odzwierciedlona obiektem tej klasy. Polami tej klasy są: wezel1, wezel2, SEM oraz R. Pierwsze 2 pola określają z jakimi węzłami połączona jest gałąź, następne wprowadza wartość siły elektromotorycznej a ostatnie rezystancję gałęzi. **Założenia strzałkowania w modelowanej gałęzi:**

1. prąd płynie od węzła1 do węzła2
2. idealne źródło napięciowe wskazuje na węzeł2
3. strzałka spadku napięcia na rezystancji wskazuje węzeł1

W przypadku gdy chcemy założyć przeciwny kierunek przepływu prądu musimy zamienić ze sobą wartości 2 pierwszych pól obiektu. Operacja ta wiąże się ze zmianą kierunku siły elektromotorycznej oraz spadku napięcia na rezystancji – spadek napięcia ma zawsze kierunek przeciwny do założonego przepływu prądu w gałęzi. Jeżeli chcemy wprowadzić źródło napięciowe w kierunku przeciwnym do założonego przepływu prądu należy przypisać sile elektromotorycznej wartość ujemną. Klasa posiada standardowy konstruktor przypisujący do pól obiektu odpowiadające im parametry przekazane w trakcie jego tworzenia.

Klasa obwod, metoda dodaj_galaz(), konstruktor:

Największą część kodu programu zajmuje opis klasy „obwod”. Wewnątrz niej znajduje się pojedyncze pole „galezie” będące listą gałęzi (obiektów galaz), konstruktor oraz 2 metody. Po stworzeniu obiektu obwod lista gałęzi jest domyślnie pusta, jednak w trakcie wykonywania programu zostaje uzupełniona poprzez metodę „dodaj_galaz()”. Wewnątrz tej funkcji tworzony jest nowy obiekt galaz z polami o wartościach równych przesyłanym parametrom a następnie dodawany jest on do listy. Konstruktor klasy obwod nie przyjmuje żadnych parametrów a instrukcje wewnątrz niego wykonywane są tylko, jeżeli omawiany plik został uruchomiony samodzielnie (jako skrypt). Konstruktor klasy obwod odpowiada za pozyskanie przez program informacji o poszczególnych gałęziach obwodu. Wewnątrz niego możliwe jest wprowadzanie gałęzi „na sztywno” zmieniając znaczniki komentarzy, co jest wygodnym sposobem szczególnie przy rozbudowanych obwodach i testowaniu poprawności działania programu. **Wprowadzając gałęzie bezpośrednio w skrypcie należy przestrzegać następujących reguł:**

1. Gałęzie muszą tworzyć pojedynczy obwód (jedno lub wielooczkowy); dozwolony jest pojedynczy otwarty obwód.
2. W każdym oczku musi wystąpić przynajmniej jedna (niezerowa) rezystancja.
3. Dozwolone jest łączenie szeregowe i równoległe gałęzi.
4. Dozwolone są zwarcia (gałęzie z zerową rezystancją i zerową siłą elektromotoryczną).
5. Dozwolone jest wprowadzanie gałęzi „wiszących” – połączonych tylko z jednej strony.
6. Dozwolone jest połączenie gałęzi do samej siebie (oba węzły gałęzi tej samej wartości).

7. Węzły gałęzi nie mogą przyjmować wartości ujemnych.
8. Wartość rezystancji nie może być ujemna.

Domyślnie zaimplementowano w skrypcie ręczne wprowadzanie danych obwodowych poprzez terminal, program sprawdza poprawność wprowadzonych danych, informuje użytkownika o ewentualnych problemach i ponawia próbę pozyskania danej informacji. Użytkownik musi jedynie samodzielnie zapewnić punkt 2, który nie jest sprawdzany. Na końcu konstruktora wywoływana jest metoda self.oblicz(), która oblicza obwód na podstawie wprowadzonych gałęzi. Jeżeli omawiany plik zostanie dołączony jako moduł do innego pliku .py to instrukcja warunkowa sprawdzająca zawartość zmiennej środowiskowej „__name__” zwróci False i program pominie polecenia konstruktora. W tej sytuacji gałęzie należy dodać wywołując metodę <obiekt_obwod>.dodaj_galaz() w głównym, osobnym pliku .py, w którym powstał obiekt obwod.

Metoda oblicz():

Metoda oblicz() służy do obliczenia i zwrócenia użytkownikowi niewiadomych zawartych w wektorze niewiadomych X – prądów gałęziowych i spadków napięć na rezystancjach. W pierwszej kolejności wyznaczana jest ilość gałęzi w polu galezie oraz najwyższy numer węzła powiększony o 1. Wartości te potrzebne są do zainicjowania macierzy wykorzystywanych w dalszej części metody:

- macierz A o wymiarach 2 * liczba gałęzi x 2 * liczba gałęzi,
- wektor B o wymiarach 2 * g,
- macierz połączeniowa P o wymiarach najwyższy numer węzła powiększony + 1 x liczba gałęzi,
- wektor niewiadomych X („wynik”) o wymiarach 2 * liczba gałęzi.

Wymiary macierzy A, B, wynik są zgodne z opisem teoretycznym w rozdziale 3. Macierz połączeniowa będzie pełniła funkcję grafu prądowego obwodu. Liczba wierszy w tej macierzy może być nadmiarowa w przypadku kiedy węzły w obwodzie nie będą numerowane od 0 lub kiedy będą występowały przerwy w numeracji, tak jak ma to miejsce w przykładzie 2 w części 5. Przykłady. Następnie uzupełniana jest macierz P w ten sposób, że wiersze reprezentują węzły a kolumny gałęzie. Jeżeli z danego węzła wypływa prąd gałęziowy to wprowadzana jest wartość -1, a jeśli wpływa to 1. Po uzupełnieniu macierzy w każdej kolumnie występuje jedna 1 i -1 oraz możliwe są zerowe wiersze, jeżeli nie występuje w obwodzie węzeł o danej wartości. Wyjątkiem jest sytuacja kiedy gałąź jest połączona do samej siebie, wtedy prąd wpływa do węzła i z niego wypływa więc sumarycznie gałąź nie oddziałuje na resztę obwodu i pozostawiana jest wartość 0. Zerowe wiersze będą powodować zużycie pamięci operacyjnej, jednak w ten sposób nie będzie potrzeby dodatkowego indeksowania węzłów. Założono, że w przypadku powstania w przyszłości interfejsu graficznego, będą tam węzły optymalnie numerowane.

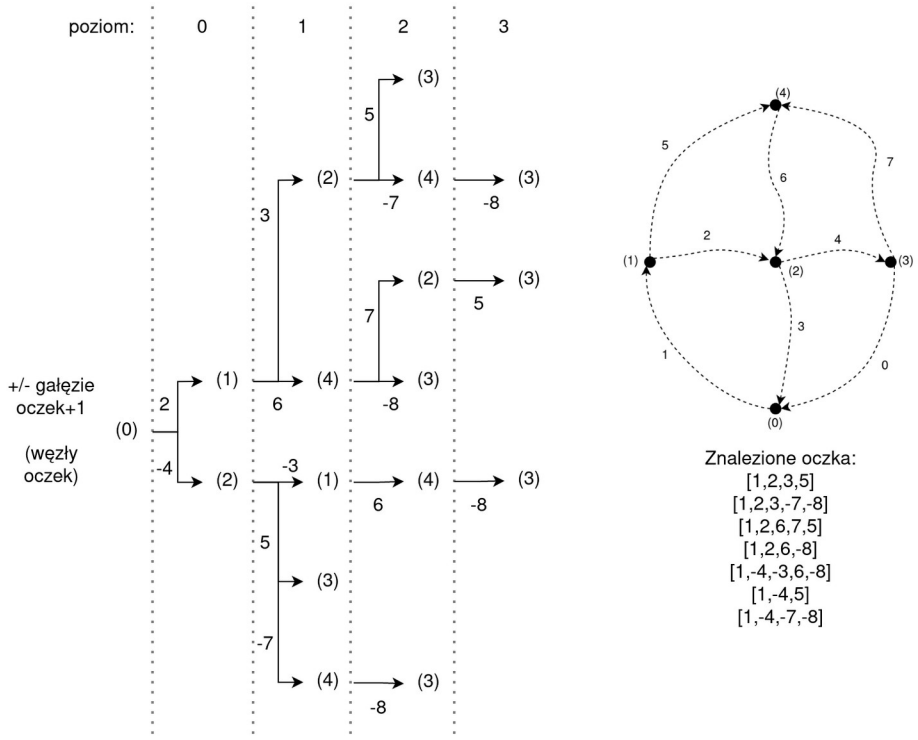
Następną czynnością w programie będzie stworzenie listy wszystkich możliwych oczek w obwodzie, które będą zależne liniowo (z powodu ich ilości), ale nie będą się powtarzać. Tworzone są listy:

- „oczka”, do której wprowadzane będą wszystkie znalezione oczka,
- „pozycja”, która będzie wskaźnikiem aktualnie badanej gałęzi (kolumny) dla poszczególnego poziomu,

- „tym_wezly”, w której będą przechowywane numery węzłów, które zostały napotkane na drodze w trakcie szukania oczka,
- „tym_galezie”, w której znajdować się będą zwiększone (lub zmniejszone) o 1 numery gałęzi, które napotkano na drodze w trakcie szukania oczka; jeżeli kierunek prądu napotkanej gałęzi podczas trasowania oczka okaże się przeciwny do kierunku prądu pierwszej gałęzi zainicjowanej w liście to będzie ona dodawana ze znakiem przeciwnym i zmniejszana o 1; zmiana numeracji ma na celu zachowanie informacji o zgodności kierunku prądu dla gałęzi 0.

Zmienna poziom przechowuje informację która z kolei gałąź i węzeł są poszukiwane do oczka. Jego zainicjowana wartość to 0 i będzie ona zwiększana o 1 po znalezieniu kolejnej gałęzi i węzła oraz zmniejszana o 1 po znalezieniu pełnego oczka lub przeskanowaniu wszystkich gałęzi (kolumn) w danym węźle. Na początku do listy tym_galezie i tym_wezly wprowadzana jest pojedyncza gałąź oraz węzeł2 tej gałęzi. Następnie wiersz macierzy połączeniowej o numerze zainicjowanej wartości w liście tym_wezly jest skanowany w poszukiwaniu gałęzi – elementów o niezerowych wartościach. Dla odczytanej wartości 0 w wierszu (braku połączenia z daną gałęzią) zmienna pozycja dla danego poziomu jest zwiększana, przez co w następnej iteracji sprawdzana będzie następna gałąź (sprawdzana kolejna kolumna). Jeżeli zostanie znaleziona wartość -1 to znaczy, że badana gałąź (o numerze kolumny) posiada prąd zgodny z zainicjowaną gałęzią w liście a jeżeli 1 to przeciwny. Warunkiem koniecznym aby dodać do listy nową gałąź i węzeł jest brak obecności rozważanego węzła w liście węzłów oraz gałąź nie może być ta sama co pierwsza pierwotnie zainicjowana. Po spełnieniu powyższych kryteriów nowy węzeł jest dodawany do listy, gałąź z indeksem zwiększonym o 1 jest dodawana ze znakiem + do listy jeżeli znaleziono w wierszu macierzy -1 lub – dla wartości 1 oraz zmienna poziom jest zwiększana o 1. Po dodaniu nowej gałęzi i węzła do list przeszukiwany jest wiersz macierzy połączeniowej o numerze równym ostatniemu elementowi w liście węzłów. W momencie wprowadzenia do listy węzłów węzła1 pierwszej, zainicjowanej gałęzi, kopia tym_galezie jest wprowadzana do listy oczka, pod warunkiem że nie występuje już ono w liście ani jej odpowiednik z przeciwnymi wartościami. Po jego dodaniu usuwany jest ostatni element z list gałęzi i węzłów, redukowany jest poziom o 1 i pozycja dla tego poziomu zostaje zwiększona o 1. Jeśli zmienna pozycja dla danego poziomu wyjdzie poza liczbę kolumn (zostaną przeskanowane wszystkie gałęzie dla danego węzła i nie zostanie znalezione połączenie) to usuwany jest ostatni element w listach tym_galezie i tym_wezly, zerowana jest pozycja dla obecnego poziomu i zmniejszany jest poziom o jeden. Dzięki temu mechanizmowi algorytm cofnie się, jeżeli zostały znalezione gałęzie i węzły, które nie mogą finalnie połączyć się z zainicjowaną gałęzią przez aktualnie poprowadzoną trasę. Po znalezieniu wszystkich oczek wartość zmiennej poziom spadnie do 0 i kiedy pozycja dla poziomu 0 wyjdzie poza zakres przerywana jest pętla co jest równoznaczne ze znalezieniem wszystkich oczek dla zainicjowanej gałęzi. Aby mieć pewność, że wszystkie możliwe oczka znajdują się w liście, proces ten jest wykonywany dla każdej gałęzi w obwodzie. Wyjątkiem są gałęzie, które są połączone do samych siebie. Dla tych gałęzi powyższy algorytm jest pomijany i dodawane są do listy oczek jako pojedyncze, jednoelementowe oczka (zwiększone o 1, znak nie ma znaczenia). Następny obrazek wizualizuje proces szukania oczek dla gałęzi 0(1) obwodu przedstawionego w części teoretycznej. Oczka znalezione wcześniej przez algorytm znajdują się wyżej na

obrazku. Zerowe, nadmiarowe wiersze macierzy połączeniowej nie brały udziału w algorytmie i nie spowolniały wykonania skryptu.



Uzyskano listę unikalnych, wszystkich możliwych oczek w obwodzie, które są zależne liniowo.

Wprowadzanie równań do macierzy A i B:

W kolejnym kroku usuwane są zerowe wiersze z macierzy połączeniowej i aktualizowana jest realna wartość węzłów w obwodzie poprzez sprawdzenie liczby jej wierszy. Zredukowana macierz P jest tak naprawdę listą równań PPK, dlatego do macierzy A wprowadzana jest ona z pominięciem ostatniego wiersza (aby zachować liniową niezależność) w miejscu kolumn 0-(g-1) odpowiedzialnych za prądy gałęziowe. Dla tych samych wierszy w wektorze B występują domyślnie zainicjowane zera. Wprowadzenie równań PPK umieszczono wewnątrz obsługi wyjątków, która będzie użyteczna kiedy w obwodzie będą występować same gałęzie połączone do siebie w pojedynczym węźle. W tej sytuacji wartość zmiennej liczby węzłów byłaby niepoprawnie zerowa, ponieważ gałęzie połączone ze sobą nie modyfikują pól w macierzy połączeniowej i pozostawiają zainicjowane zera. W każdej innej sytuacji, przy opisie pojedynczego obwodu liczba węzłów będzie poprawna.

Aby móc wprowadzić równania NPK do macierzy A trzeba najpierw wybrać spośród oczek tylko te, które są liniowo niezależne. W tym celu sprawdzana jest liczba oczek w liście i na jej podstawie inicjowane są 2 nowe macierze wypełnione zerami:

- tym_A o wymiarach liczba oczek x liczba galezi,
- tym_B o wymiarach liczba oczek.

Każdemu oczku będzie odpowiadać wiersz o tym samym numerze tym_A i tym_B. Do tym_A wprowadzana jest 1 w kolumnę o indeksie odpowiadającym zmodyfikowanej wartości elementu oczka (wartości bezwzględnej elementu - 1), jeżeli element był dodatni przed modyfikacją lub -1 jeżeli był ujemny. Do tym_B wprowadzana jest suma sił elektromotorycznych gałęzi (numer po modyfikacji) znajdujących się w danym oczku ze znakiem + przy dodatnich wartościach elementu przed modyfikacją lub – przy ujemnych. Sytuacja jest o tyle ułatwiona, że siły elektromotoryczne są zawsze przeciwnie strzałkowane do spadków napięcia w programie więc podążając tym samym kierunkiem co spadki napięcia zostanie zachowane przeciwne strzałkowanie dla źródeł napięciowych zgodnie z definicja NPK. W tym momencie trzeba wybrać spośród wierszy te, które są liniowo niezależne i zostanie do tego wykorzystana jedynie macierz tym_A, ponieważ wystarczającą informacją jest obecność i kierunek gałęzi w oczku która jest zawarta w tym_A. Do operacji sprawdzania niezależnych liniowo wierszy wykorzystano funkcję z biblioteki SymPy. Następnie usuwane są wiersze z tymczasowych macierzy, odpowiadające oczkom, które wprowadzają liniową zależność. Tak przygotowane macierze tym_A i tym_B wprowadzane są do macierzy A i B pod równaniami PPK tj. do wierszy (w - 1) – (g - 1) tak, żeby kolumny macierzy tym_A odpowiadały spadkom napięcia w macierzy A - kolumny g – (2g-1). Operacje opisane w 2 ostatnich zdaniach umieszczono wewnątrz obsługi wyjątków, aby umożliwić zwrócenie zerowych wyników w przypadku pojedynczego otwartego obwodu.

Ostatnim typem równań wprowadzanych do macierzy A są równania definicyjne elementów obwodowych, w przypadku obecnego stanu symulatora – są wyłącznie prawem Ohma dla rezystancji. Macierz B może przyjąć wartości niezerowe tylko dla równań NPK, więc pozostawiane są domyślnie zainicjowane zera. Do macierzy A wprowadzane jest tyle równań definicyjnych ile jest gałęzi, ponieważ dla każdej gałęzi przypisana jest rezystancja gałęziowa. Poszczególne równanie definicyjne jest wierszem w macierzy A, które wiąże prądy gałęziowe ze spadkami napięcia na rezystancjach. W kolumny odpowiadające spadku napięcia na rezystancji danej gałęzi wpisywana jest 1 a w kolumny prądu tej samej gałęzi ujemna wartość rezystancji gałęziowej.

Na końcu rozwiązywane jest równanie macierzowe A * X = B, używając gotowej funkcji z biblioteki NumPy, w wyniku której zwracane są obliczone wartości niewiadomych – wektor X. Wynik jest rysowany, jeżeli plik został uruchomiony jako skrypt oraz grupowany jest w osobnych listach U i I, które zwraca metoda oblicz().

Tworzenie obieku obwod:

Jeżeli plik został uruchomiony samodzielnie to tworzony jest obiekt klasy obwod przez co rozpoczyna się proces zbierania danych obwodowych przez program, opisany w konstruktorze a następnie na ich podstawie metoda oblicz() wyświetla obliczone spadki napięcia i prądy gałęziowe.

Analiza działania programu pod kątem różnych możliwych danych wejściowych:

Kolor punktu mówi czy program miałby problem z danymi wejściowymi:

- wprowadzenie pojedynczej gałęzi do obwodu: w=2, g=1, PPK=1, NPK=0, DEF=1

- wprowadzenie 2 pojedynczych niepołączonych gałęzi (2 obwodów otwartych) do pojedynczego obiektu obwód: w=4, g=2, PPK=3, NPK=-1, DEF=2
- wprowadzenie pojedynczej gałęzi połączonej samej do siebie (oba węzły mają identyczną wartość): w=1, g=1, PPK=0, NPK=1, DEF=1
- wprowadzenie 2 pojedynczych gałęzi połączonych do samych siebie w osobnych węzłach w=2, g=2, PPK = 1, NPK=1, DEF=2
- wprowadzenie połączenia szeregowego (2 gałęzie są połączone w 1 węźle): w+=1, g+=1, PPK+=1, NPK==NPK, DEF+=1
- wprowadzenie połączenia równoległego (nie powstają nowe węzły): w==w, g+=1, PPK==PPK, NPK+=1, DEF+=1
- wprowadzenie gałęzi wiszącej zmienia parametry identycznie jak połączenie szeregowe
- dowolne oczko w obwodzie, w którym występuje tylko jedno lub wiele idealnych źródeł napięciowych bez innych typów elementów obwodowych nie ma sensu z perspektywy teorii obwodów; w stworzonym programie sytuacja ta spowodowałaby przypisanie wartości zerowych spadkom napięcia a następnie wprowadzenie tych zer do równania oczka, które jeżeli suma napięć sił elektromotorycznych się równoważy to otrzymane zostanie równanie tożsamościowe 0=0 (nieskończenie wiele rozwiązań) lub równanie sprzeczne 0=(!0).

Planowane usprawnienia:

Fakt występowania nadmiarowej ilości niewiadomych i równań, powoduje wydłużenie wykonania sprawdzenia nieliniowej niezależności wierszy macierzy tym_A oraz obliczenie równania macierzowego. W obecnym etapie rozwoju oprogramowania, nie jest to problem znaczący, jednak wraz z jego postępem przewidywany jest wzrost ilości równań i niewiadomych w obwodzie ze względu na nowe typy elementów obwodowych. Program można usprawnić poprzez:

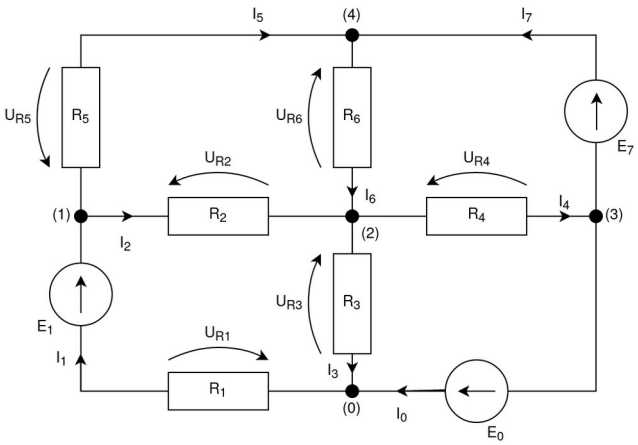
1. odrzucenie w obliczeniach spadku napięcia dla rezystancji zerowej, powodowałoby usunięcie 1 kolumny w macierzy A oraz 1 wiersza w macierzy A , B (równanie definicyjne) oraz X (szukana),
2. odrzucenie z obliczeń gałęzi wiszącej, powodowałoby usunięcie 2 zmiennych – prądu gałęzi oraz spadku napięcia na rezystancji, co skutkowałoby usunięciem 2 kolumn w macierzy A oraz 2 wierszy w macierzy A, B (równanie PPK i definicyjne) oraz X (2 szukane).

Podejście te będzie wymagało wprowadzenia indeksów oznaczających parametry obwodowe w wektorze X. Szczególnie 1 punkt jest istotny, ponieważ w praktyce często występują gałęzie jedynie z idealnym źródłem napięciowym bez rezystancji (ewentualnie z innym elementem obwodowym, który nie został jeszcze wprowadzony do programu) a użytkownicy posiadający wiedzę na temat obwodów elektrycznych nie będą wprowadzać gałęzi wiszących.

5. Przykłady

Przygotowano 4 przykłady obwodów, które zostały rozwiązane z użyciem opisywanego symulatora a następnie sprawdzono otrzymane wyniki programem „Ltspice XVII”. W programie Ltspice konieczne jest wybranie węzła, który będzie masą (odniesieniem) a gałęzie są numerowane od 1 – nie od 0 jak w stworzonym symulatorze. Podane wartości potencjałów na wszystkich węzłach obliczane są względem węzła masy. Znając obliczone spadki napięcia na rezystancjach oraz napięcia na siłach elektromotorycznych jesteśmy w stanie obliczyć potencjał węzła (względem masy) sumując napięcia na drodze od węzła masy do docelowego węzła, gdzie wybór drogi nie wpływa na wynik. Prąd obliczony przez Ltspice może mieć przeciwną wartość ze względu na inny niż założony sposób strzałkowania a gałęzie wiszące i zwarcia są pomijane.

1. $E_0 = E_1 = E_7 = 10[V]$, $R_1 = R_2 = R_3 = R_4 = R_5 = R_6 = 10[\Omega]$, węzeł masy (0)



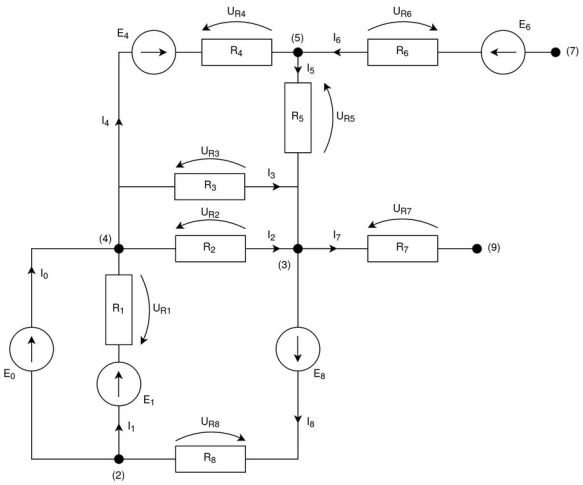

```
--- Operating Point ---

V(3):          -10             voltage
V(1):          2.72727         voltage
V(n002):       -7.27273        voltage
V(2):          -1.81818        voltage
V(4):          0               voltage
I(R7):         0.181818        device_current
I(R6):         -0.272727       device_current
I(R5):         -0.818182       device_current
I(R4):         -0.181818       device_current
I(R3):         -0.454545       device_current
I(R2):         0.727273        device_current
I(V8):         0.0909091       device_current
I(V2):         -0.727273       device_current
I(V1):         -0.909091       device_current
```

```
maurycy@Maurisjo:~/Dokumenty/kodzenie$ python3 symulator.py
---------------------------------
I0 = 0.90909090909090992 [A]
---------------------------------
I1 = 0.72727272727272273 [A]
---------------------------------
I2 = 0.4545454545454545 [A]
---------------------------------
I3 = -0.18181818181818182 [A]
---------------------------------
I4 = 0.8181818181818181 [A]
---------------------------------
I5 = 0.27272727272727265 [A]
---------------------------------
I6 = 0.18181818181818182 [A]
---------------------------------
I7 = -0.090909090909090105 [A]
---------------------------------
U0 = 0.0 [V]
---------------------------------
U1 = 7.272727272727273 [V]
---------------------------------
U2 = 4.545454545454545 [V]
---------------------------------
U3 = -1.8181818181818183 [V]
---------------------------------
U4 = 8.181818181818182 [V]
---------------------------------
U5 = 2.7272727272727266 [V]
---------------------------------
U6 = 1.8181818181818183 [V]
---------------------------------
U7 = 0.0 [V]
```

2. $E_0 = 10[V]$, $E_1 = -3.3[V]$, $E_4 = -1\ .5[V]$, $E_6 = -0.5[V]$, $E_8 = 3.3[V]$

$R_1 = R_2 = R_3 = R_4 = R_5 = R_6 = R_7 = R_8 = 10[\Omega]$, węzeł masy (2)



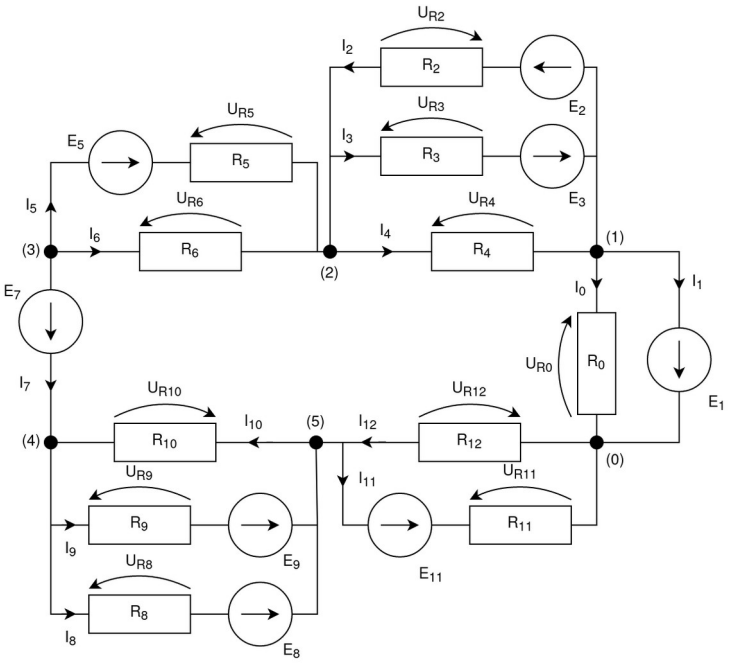

```
--- Operating Point ---

V(4):          10              voltage
V(n003):       -3.3            voltage
V(n001):       8.5             voltage
V(n002):       7.24286         voltage
V(7):          7.74286         voltage
V(n004):       9.28571         voltage
V(3):          5.98571         voltage
V(5):          7.24286         voltage
V(9):          5.98571         voltage
I(R9):         0.928571        device_current
I(R8):         0               device_current
I(R7):         0               device_current
I(R6):         0.125714        device_current
I(R5):         -0.125714       device_current
I(R4):         -0.401429       device_current
I(R3):         -0.401429       device_current
I(R2):         1.33            device_current
I(V9):         -0.928571       device_current
I(V7):         0               device_current
I(V5):         -0.125714       device_current
I(V2):         1.33            device_current
I(V1):         -2.25857        device_current
```

```
maurycy@Maurisjo:~/Dokumenty/kodzenie$ python3 symulator.py
---------------------------------
I0 = 2.2585714285714285 [A]
---------------------------------
I1 = -1.33 [A]
---------------------------------
I2 = 0.4014285714285716 [A]
---------------------------------
I3 = 0.4014285714285716 [A]
---------------------------------
I4 = 0.1257142857142858 [A]
---------------------------------
I5 = 0.1257142857142858 [A]
---------------------------------
I6 = -0.0 [A]
---------------------------------
I7 = -0.0 [A]
---------------------------------
I8 = 0.9285714285714285 [A]
---------------------------------
U0 = 0.0 [V]
---------------------------------
U1 = -13.3 [V]
---------------------------------
U2 = 4.014285714285716 [V]
---------------------------------
U3 = 4.014285714285716 [V]
---------------------------------
U4 = 1.257142857142858 [V]
---------------------------------
U5 = 1.257142857142858 [V]
---------------------------------
U6 = -0.0 [V]
---------------------------------
U7 = -0.0 [V]
---------------------------------
U8 = 9.285714285714285 [V]
```

3. $E_1 = E_2 = E_3 = E_5 = E_7 = E_8 = E_9 = E_{11} = 10[V]$

$R_0 = R_2 = R_3 = R_4 = R_5 = R_6 = R_8 = R_9 = R_{10} = R_{11} = R_{12} = 10[\Omega]$, węzeł masy (0)

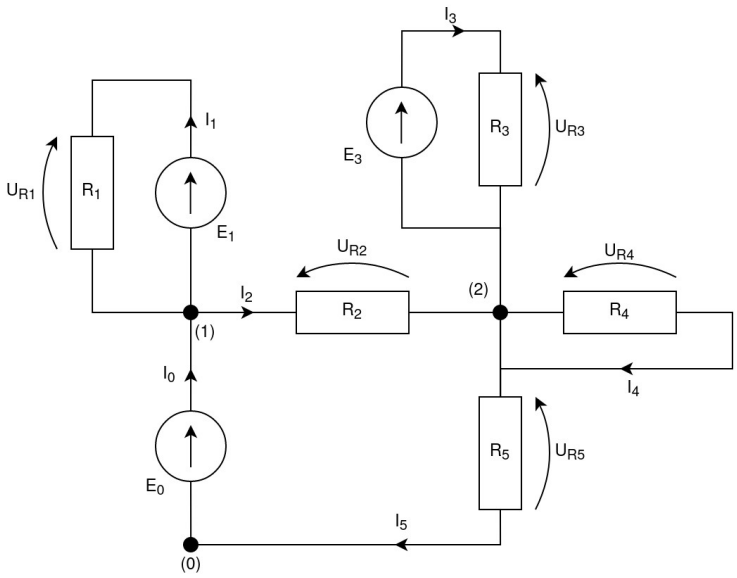



```
--- Operating Point ---

V(1):          -10             voltage
V(n001):       0               voltage
V(2):          -11.3333        voltage
V(n003):       -20             voltage
V(n002):       -8.33333        voltage
V(3):          -18.3333        voltage
V(n006):       -13             voltage
V(4):          -8.33333        voltage
V(n004):       -13             voltage
V(5):          -3              voltage
V(n005):       7               voltage
I(R13):        0.3             device_current
I(R12):        -0.7            device_current
I(R11):        0.533333        device_current
I(R10):        -0.466667       device_current
I(R9):         -0.466667       device_current
I(R7):         0.7             device_current
I(R6):         -0.3            device_current
I(R5):         0.133333        device_current
I(R4):         -0.866667       device_current
I(R3):         1.13333         device_current
I(R1):         -1              device_current
I(V12):        -0.7            device_current
I(V10):        -0.466667       device_current
I(V9):         -0.466667       device_current
I(V8):         -0.4            device_current
I(V6):         -0.3            device_current
I(V4):         -0.866667       device_current
I(V3):         -1.13333        device_current
I(V2):         -0.6            device_current
```

```
I0 = -1.0 [A]
I1 = 0.5999999999999994 [A]
I2 = 1.1333333333333335 [A]
I3 = 0.8666666666666665 [A]
I4 = -0.13333333333333347 [A]
I5 = 0.3 [A]
I6 = -0.7 [A]
I7 = 0.4 [A]
I8 = 0.4666666666666665 [A]
I9 = 0.4666666666666665 [A]
I10 = 0.5333333333333334 [A]
I11 = 0.7 [A]
I12 = 0.29999999999999993 [A]
U0 = -10.0 [V]
U1 = 0.0 [V]
U2 = 11.333333333333336 [V]
U3 = 8.666666666666664 [V]
U4 = -1.3333333333333348 [V]
U5 = 3.0 [V]
U6 = -7.0 [V]
U7 = 0.0 [V]
U8 = 4.666666666666665 [V]
U9 = 4.666666666666665 [V]
U10 = 5.333333333333335 [V]
U11 = 7.0 [V]
U12 = 2.9999999999999996 [V]
```

4. $E_0 = 10[V]$, $E_1 = -10[V]$, $E_3 = 4[V]$,
$R_1 = 2[\Omega]$, $R_2 = 2.5[\Omega]$, $R_3 = 2[\Omega]$, $R_4 = 2[\Omega]$, $R_5 = 0[\Omega]$, węzeł masy (0)




```
maurycy@Maurisjo:~/Dokumenty/kodzenie$ python3 symulator.py
I0 = 4.0 [A]
I1 = -5.0 [A]
I2 = 4.0 [A]
I3 = 2.0 [A]
I4 = -0.0 [A]
I5 = 4.0 [A]
U0 = 0.0 [V]
U1 = -10.0 [V]
U2 = 10.0 [V]
U3 = 4.0 [V]
U4 = 0.0 [V]
U5 = 0.0 [V]
```

```
--- Operating Point ---

V(1):          10              voltage
V(n001):       0               voltage
V(n002):       4               voltage
I(R4):         2               device_current
I(R3):         -4              device_current
I(R2):         -5              device_current
I(V4):         -2              device_current
I(V2):         5               device_current
I(V1):         -4              device_current
```

6. Podsumowanie

Stworzony skrypt wykonuje symulację komputerową obwodów złożonych z idealnych źródeł napięciowych i rezystancji. Zwraca on w krótkim czasie, bezbłędnie wyznaczone prądy gałęziowe oraz spadki napięcia na rezystancjach. Dane obwodowe są wprowadzane w formie opisu poszczególnej gałęzi z użyciem terminala lub wpisywane są na sztywno w skrypcie. Istnieje możliwość wykorzystania napisanego pliku jako silnik zwracający obliczone parametry obwodu. Niezawodność programu zależy od przestrzegania reguł jego korzystania, ustalonych w niniejszym raporcie. Zbadano działanie symulatora pod różnym kątem po czym wykonano 4 przykłady. Przyszłością projektu będzie:

- wprowadzenie usprawnień w obliczeniach,
- wprowadzenie nowych elementów obwodowych tj. idealne źródło prądowe, pojemność, indukcyjność,
- stworzenie interfejsu graficznego,
- udostępnienie innego typu analizy niż analiza stałoprądowa – analizy operatorowej, wskazowej.