2016

KASKADA

INSTRUKCJA OBSŁUGI

1.1 Opis programu

Program komputerowy umożliwia przeprowadzenie kaskady wyłączeń na kilku wybranych z biblioteki MATPOWER systemach testowych. System elektroenergetyczny zwizualizowano za pomocą grafów skierowanych. Wiąże się to z ograniczeniem połączeń (liczby gałęzi) między parą węzłów do dwóch, gdyż tylko w takim przypadku możliwe jest zbudowanie jednoznacznej macierzy połączeń (macierz incydencji) grafu. Rozmieszczenie węzłów w dwuwymiarowej przestrzeni wymagało optymalizacji pozycji węzłów systemu na płaszczyźnie, tak aby uzyskać jak największą czytelność prezentacji danych. W tym celu wykorzystano algorytm:

- *Force-directed placement*, dla grafów o maksymalnej liczbie wierzchołków równej 100

  Rodzina algorytmów opartych na oddziaływaniu siłowym – zakłada istnienie przyciągającej siły między połączonymi węzłami oraz siły odpychającej pomiędzy węzłami niepołączonymi żadną krawędzią. Są to algorytmy typu iteracyjnego. Przykładowy algorytm z tej rodziny przedstawiono w pracy [26].

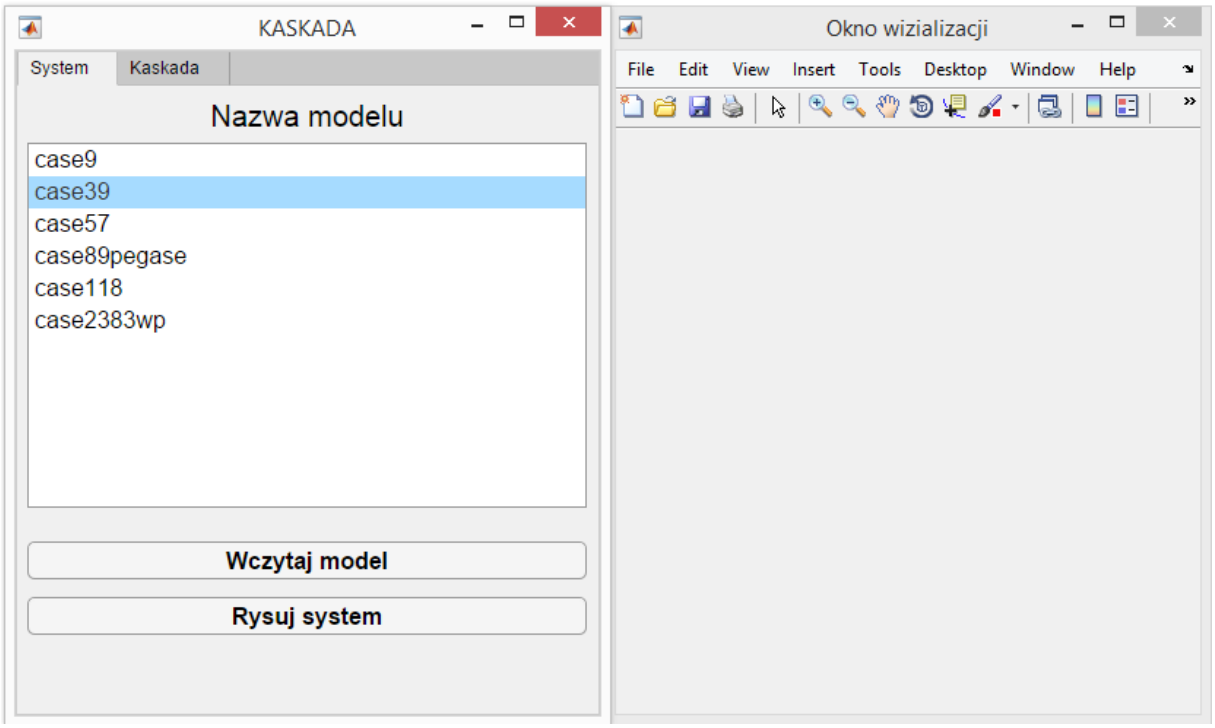
- *Subspace* (podprzestrzeń)

  Jest to domyślny algorytm wizualizujący dla grafów o liczbie wierzchołków powyżej 100. Uzyskiwana prezentacja danych jest mniej czytelna niż algorytm oparty na oddziaływaniach siłowych, jednakże jego przewagę w tym przypadku stanowi szybkość działania. Algorytm ten polega na rozłożeniu wierzchołków w wielowymiarowej przestrzeni, a następnie zrzutowanie pozycji wierzchołków z powrotem na przestrzeń dwuwymiarową.

1.2 Opis interfejsu

Celem tego punktu, było stworzenie przemyślanego, przejrzystego oraz jak najbardziej intuicyjnego interfejsu wymiany informacji pomiędzy programem KASKADA, a użytkownikiem. W tym celu przygotowano prosty interfejs graficzny w postaci okienkowej, przedstawiony na rys. 2.7. Składa się z dwóch okien, wśród których występuje:

- Dwuzakładkowe okno główne o nazwie „**Kaskada**”. Jest to główna i podstawowa część całego programu.
- Okno wizualizacji kaskady wyłączeń o nazwie „ **Okno wizualizacj**i”.



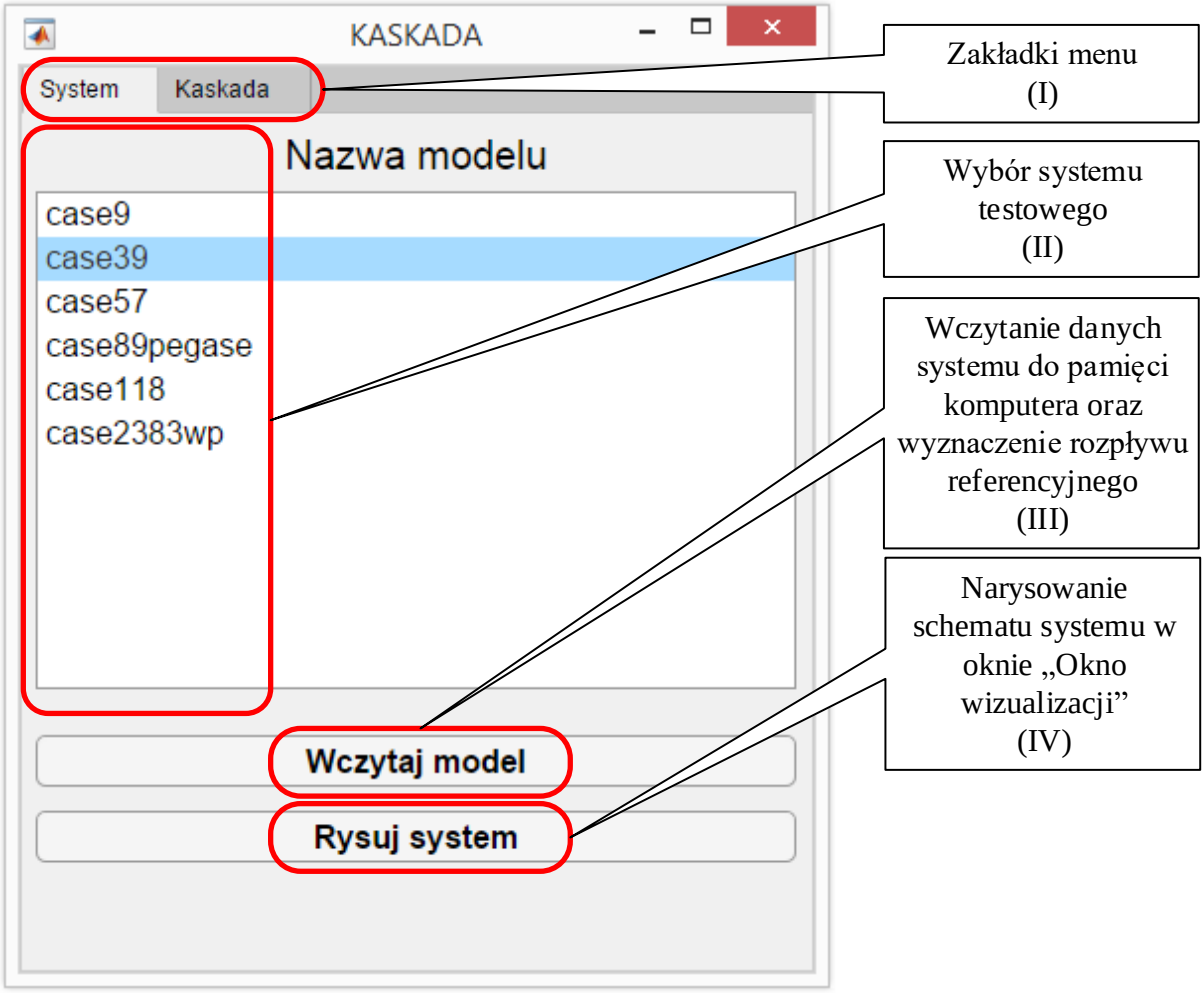
Rys. 0.1. Okna interfejsu graficznego.

1.2.1 Opis zakładki „System” w ramach okna „KASKADA”

Rolą **zakładki „System”** (rys. 2.8) jest zainicjowanie działania programu. Użytkownik powinien w tym punkcie wskazać system, dla którego chce wyznaczyć kaskadę wyłączeń **(II)**. Po dokonaniu wskazania, należy nacisnąć przycisk „**Wczytaj model**” **(III)**, który zapisuje wybrany model do pamięci komputera. Dane wszystkich modeli przechowywane są w plikach tekstowych. Wczytanie modelu wiąże się z wywołaniem następujących funkcji (ich opis zamieszczono w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.** oraz Z.2):

- *check_graph*
- *reNum*
- *mk_default*
- *solvePF*
- *plotGrid_o*

Za pomocą przycisku **„Rysuj system” (IV)** wykreśla się wybrany schemat u w module **„Okno wizualizacji”.** Naciśnięcie wspomnianego przycisku wywołuje funkcję *rePlot* **(**rozdział **Błąd! Nie można odnaleźć źródła odwołania.** oraz Z.2**).**



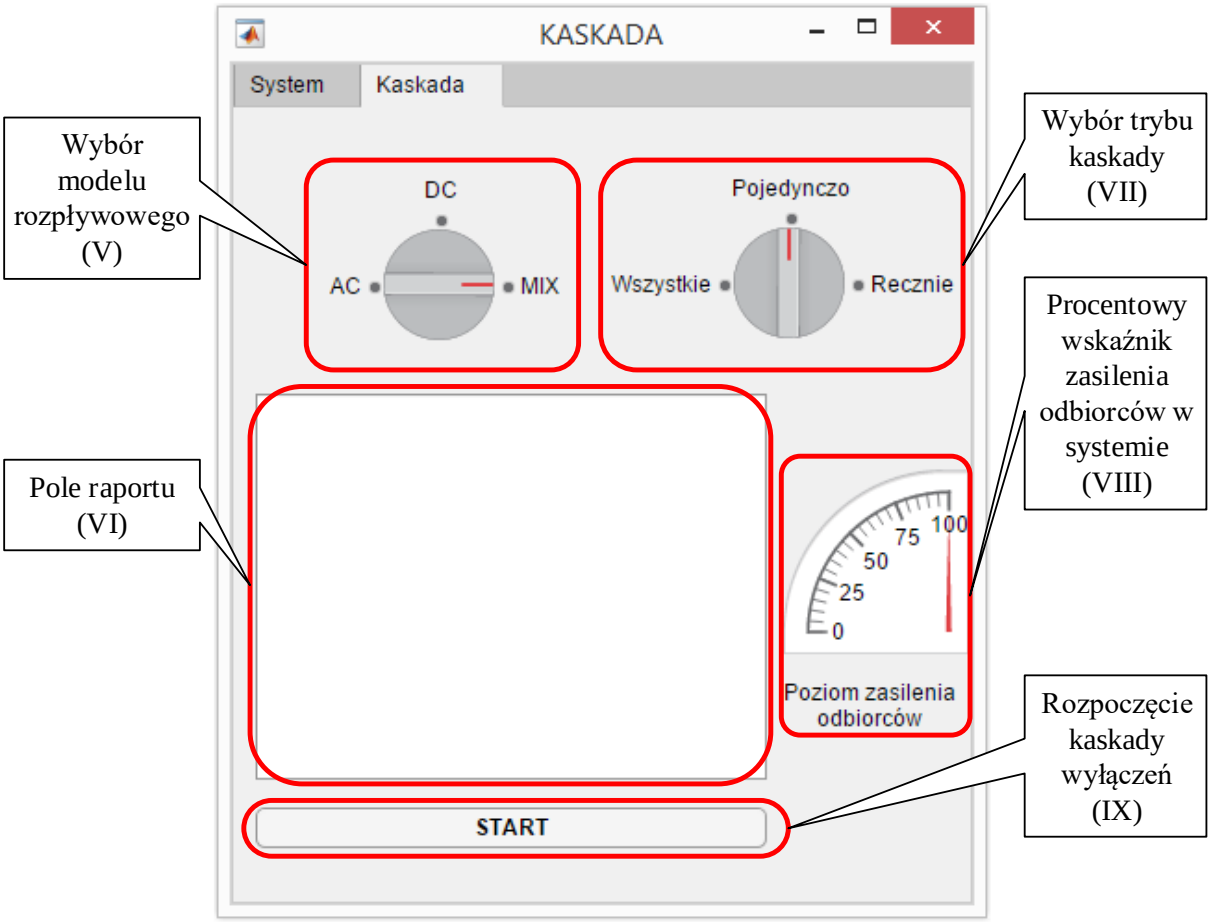
Rys. 0.2. Zakładka "System".

1.2.2 Opis zakładki „Kaskada” w ramach okna „KASKADA”

Na rys. 2.9 przedstawiono opis elementów zakładki **„Kaskada”**. Jej przeznaczeniem jest sterowanie procesem wyłączeń.

Zaczynając od góry, pierwszy **przełącznik oznaczony cyfrą rzymską V** „odpowiada” za wybór modelu, w którym będzie wyznaczony nowy rozpływ po dokonaniu zmiany w konfiguracji sieci. Dostępne są 3 opcje:

- AC – tradycyjny rozpływ liczony metodą Newtona-Raphsona opisany w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.**,
- DC – model stałoprądowy opisany w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.**,
- MIX – rozpływ jest domyślnie wyznaczany przy pomocy modelu AC, jednakże, jeżeli metoda okaże się rozbieżna , to program przechodzi na model DC.



Rys. 0.3. Zakładka Kaskada.

Pole tekstowe VI, zwraca użytkownikowi raport w którym prezentowane są informacje takie jak identyfikator wyłączanego elementu, struktura wydzielonych obszarów, itd.

Za pomocą **przełącznika VII** dokonuje się wyboru trybu, w jakim ma być przeprowadzana kaskada wyłączeń. Dostępne są 3 opcje:

- Wszystkie – wybór tej opcji oznacza, że w przypadku każdej iteracji, wyłączane są wszystkie połączenia, na których wystąpiły przekroczenia limitów. Opcja ta najbardziej odzwierciedla rzeczywistość, jednakże jej zastosowanie sprawia, że cały proces jest trudno obserwowalny i nie umożliwia wykonania wielokrokowej sekwencji ukazującej efekt domina, gdyż zazwyczaj cała sekwencja kończy się w pierwszych iteracjach tego procesu.
- Pojedyncze – jest to opcja przygotowana do ukazania efektu domina wyłączeń kaskadowych. W tym trybie program zawsze „odcina” połączenia, w którym nastąpiło największe przeciążenie (procentowe). Proces kaskady jest wobec tego dłuższy. Tryb ten, w bardzo uproszczony sposób odzwierciedla przyjęte założenie, iż czas likwidacji przeciążenia powinien być uzależniony od wartości tego przeciążenia. Warto zauwa-

żyć, że w sieci przesyłowej najważniejszym ograniczeniem definiującym maksymalny przepływ mocy na linii, są zwisy przewodów. Zwiększenie przepływu prądu, powoduje silniejsze nagrzewanie się linii, co w konsekwencji doprowadza do zwiększania się zwisu, co stanowi poważne zagrożenie (m.in. rośnie ryzyko wystąpienia zwarcia). Z tego powodu nie dopuszcza się do długotrwałej pracy linii z przeciążeniem.

- Ręcznie – program nie wyłącza automatycznie linii, na każdym etapie to użytkownik wybiera elementy systemu przeznaczone do eliminacji.

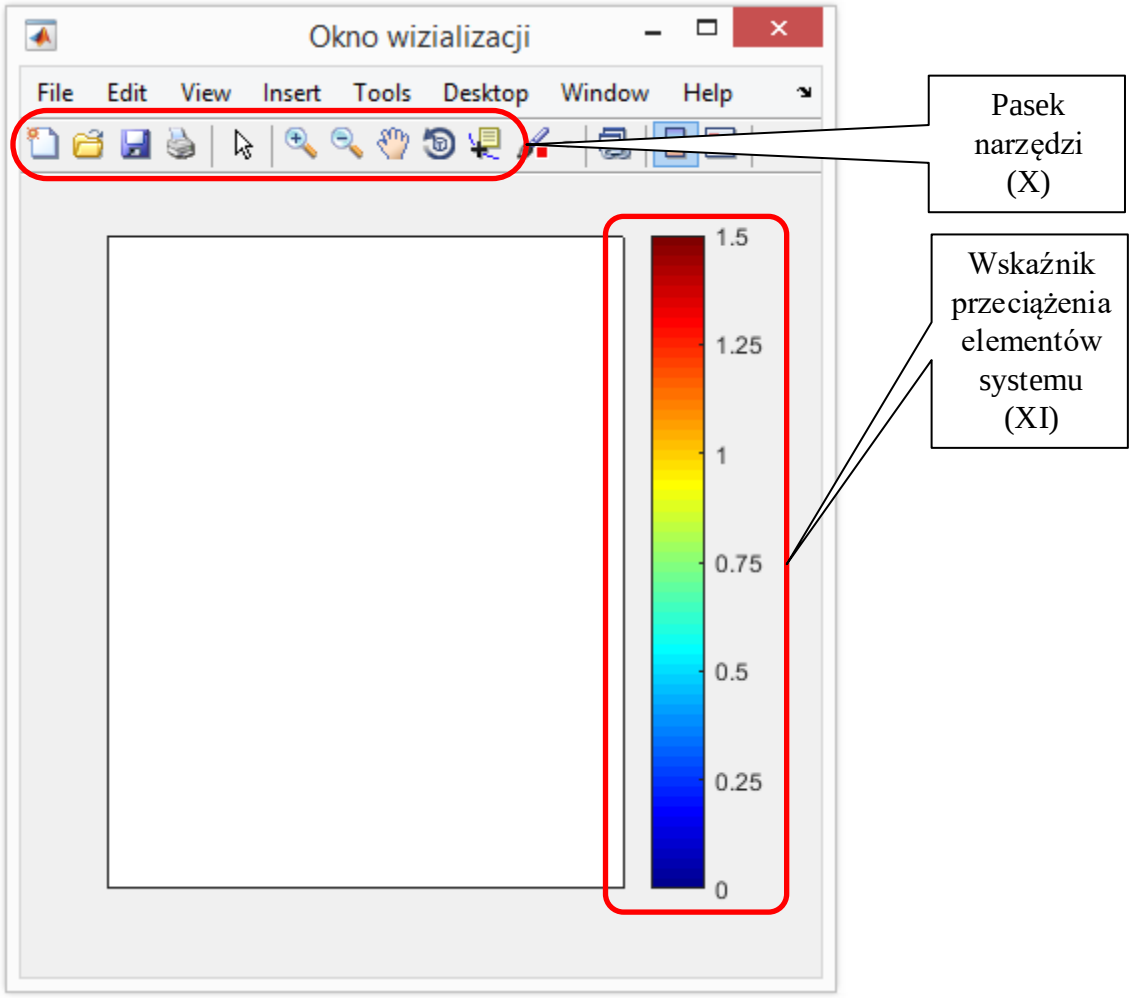
Warto dodać, że w każdym z powyższych trybów, jeżeli nie będzie żadnego przekroczenia w systemie, to użytkownik będzie poproszony o zainicjowanie kaskady poprzez wskazanie pierwszego elementu (lub elementów) przeznaczonych do wyłączenia.

Procentowy **wskaźnik zasilenia odbiorców w systemie (VIII)**, jak sama nazwa wskazuje, odzwierciedla w jakiej części system zachował integralność. Wskazanie jest wyliczone przez zsumowanie mocy czynnej odbieranej w istniejących (niewyizolowanych) węzłach systemu po każdej iteracji a następnie podzielone przez sumaryczną moc czynną odbieraną w całym systemie w stanie bazowym.

Ostatnim elementem jest **przycisk „START" (IX)** uruchamiający główną metodę klasy: ***StartCascade*** w ramach której wywoływane są**:**

- *calcFlowRatio,*
- *chooseOutage,*
- *rePlot,*
- *solvePF*.

1.2.3 Opis okna „Okno wizualizacji”



Rys. 0.4. Okno wizualizacji kaskady

Okno wizualizacji służy do graficznej prezentacji wyników. **Pasek narzędzi (X)** składa się z następujących opcji – poczynając od lewej:

- utworzenie nowego okna,
- otwarcie obrazu konfiguracji systemu z pliku,
- zapis aktualnego obrazu konfiguracji systemu do pliku,
- drukowanie,
- edycja obszaru okna,
- przybliżenie obrazu,
- oddalenie obrazu,
- przesuwanie (chwytak),
- obrót w przestrzeni trójwymiarowej,

- utworzenie znacznika danych,

Powyższe narzędzia pochodzą z domyślnej biblioteki środowiska MATLAB. Zaznaczone ramką funkcje, są najbardziej przydatne do pracy z programem KASKADA.

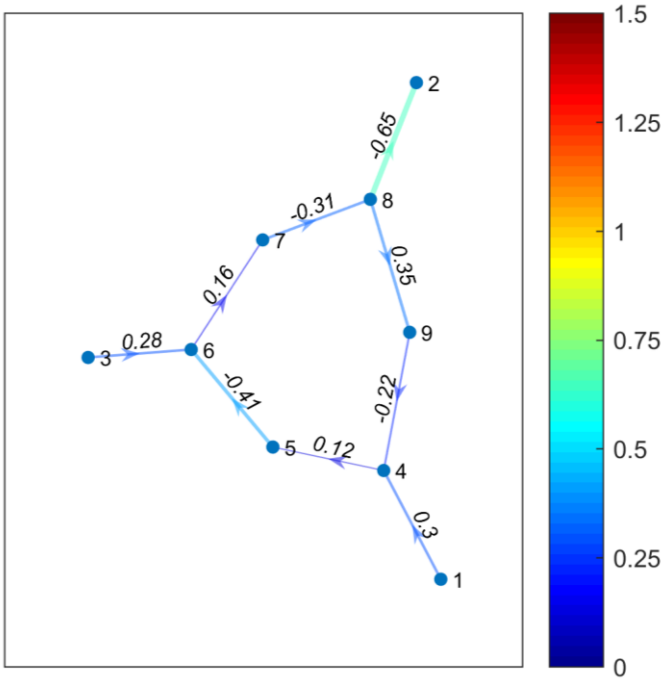
Wskaźnik przeciążenia elementów systemu (XI) jest legendą, względem której kolorowane są wszystkie połączenia w analizowanym systemie. Kolor danej linii, lub transformatora jest zależny od stopnia przeciążenia. Skala tego wskaźnika odnosi się do jednostek względnych, gdzie jedynka oznacza pełne obciążenie danego elementu.

1.3 Prezentacja wizualizacji zaimplementowanych systemów

W tym punkcie zostaną pokazane efekty wygenerowania wszystkich systemów testowych zaimplementowanych w programie KASKADA, tj.:

- case9,
- case39,
- case57,
- case89pegase,
- case118,
- case2383.

1.3.1 System *case9* (rys. 2.11)



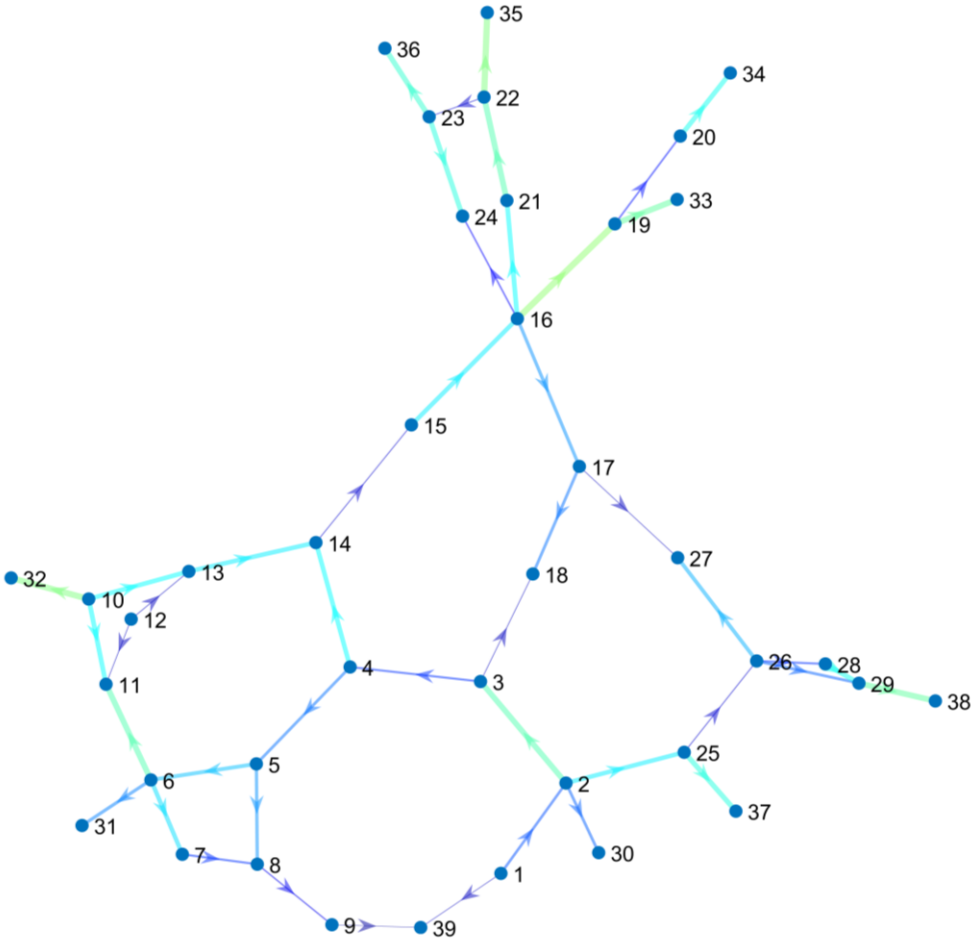
Rys. 0.5. System: *case9.*

Uzyskany efekt wizualizacji w przypadku systemu 9-węzłowego (najważniejsze fakty na jego temat zawarto w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.**) jest najbardziej zadowalający. Rozmieszczenie węzłów uzyskanego grafu jest równomierne, dzięki czemu uzyskuje się bardzo czytelną strukturę systemu, w której nie wystepują przecięcia krawędzi. Wadą tego systemu jest niewielka liczba możliwych do uzyskania wyłączeń. W praktyce niemożliwe jest przeprowadzenie „zadowalającej kaskady" z uwagi na fakt, iż sieć ta tworzy tylko jedno oczko, co bardzo szybko doprowadza do powstania wyizolowanych względem siebie wysp.

Aby uzyskać jak najlepszą czytelność schematów, usunięto etykiety przepływów oraz legendę kolorystyczną dla systemów bardziej rozbudowanych niż *case9*

1.3.2 System *case39*

Uzyskany efekt wizualizacji w przypadku systemu 39-węzłowego (najważniejsze fakty na jego temat również zawarto w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.**) jest również zadowalający. Rozmieszczenie węzłów uzyskanego grafu jest równomierne, chociaż zdarzają się węzły rozmieszczone w zbyt małej odległości względem siebie. Niemniej również w tym przypadku uzyskuje się czytelną strukturę systemu, w której nie występują przecięcia krawędzi. System ten umożliwia uzyskanie „dość dużej kaskady", z uwagi na to, iż sieć ta jest wielokrotnie zamknięta. Z uwagi na wysoką czytelność oraz zadowalającą liczbę oczek w sieci, wybrano ten system do dalszych rozważań w rozdziale 3.



Rys. 0.6. System: *case39*.

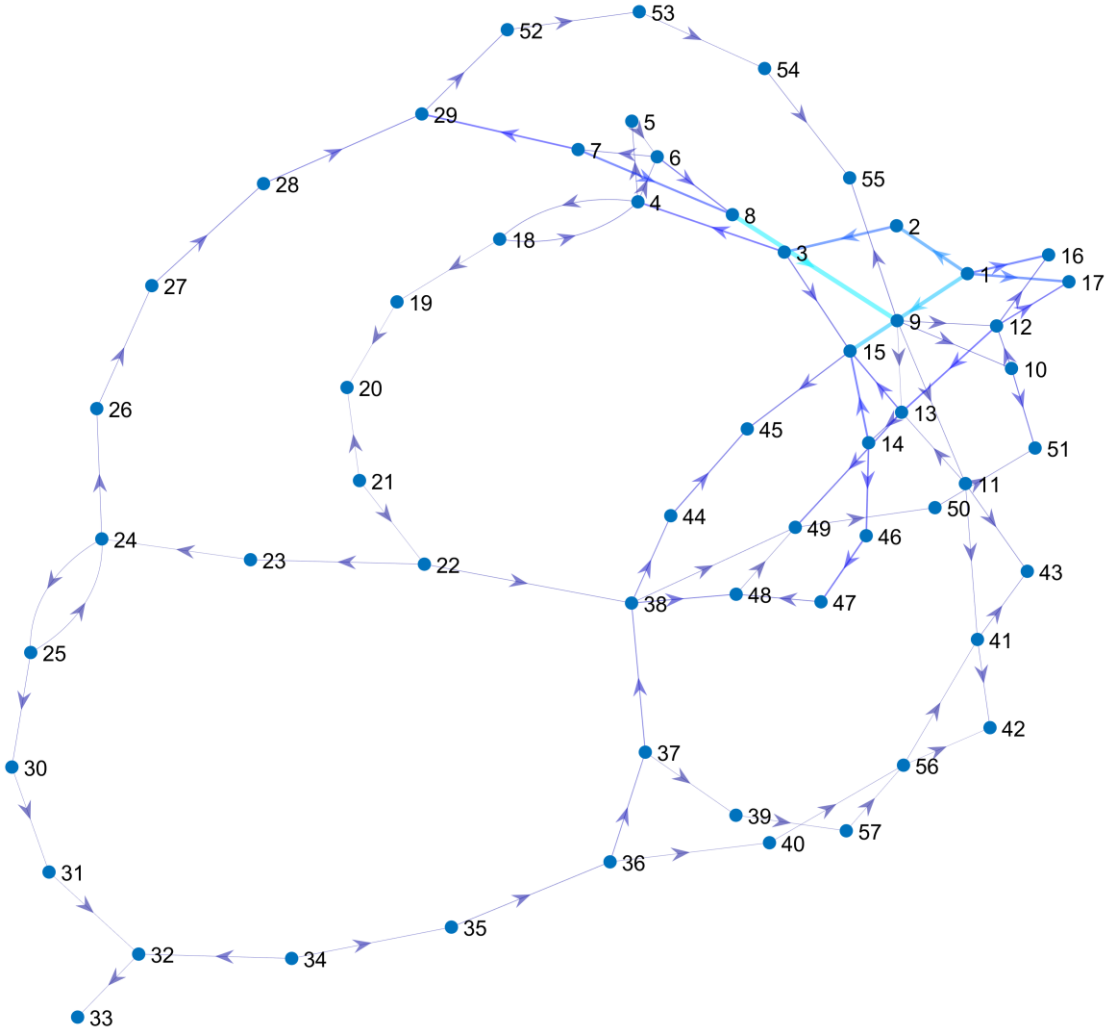
1.3.3 System *case57* (rys. 2.13)

Najważniejsze cechy:

- 57 węzłów, 7 generatorów, 80 linii lub transformatorów,
- System IEEE

Brak napięć znamionowych oraz limitów na elementach systemu. Ustawiono wartości domyślne przy użyciu funkcji *mk_default* (opis zamieszczono w rozdz. **Błąd! Nie można odnaleźć źródła odwołania.**).

Uzyskane efekty wizualizacji w przypadku systemu 57-węzłowego są gorsze względem systemu 39-węzłowego, jednakże możliwa do uzyskania liczba wariantów jest względem niego większa.



Rys. 0.7. System *case57*.

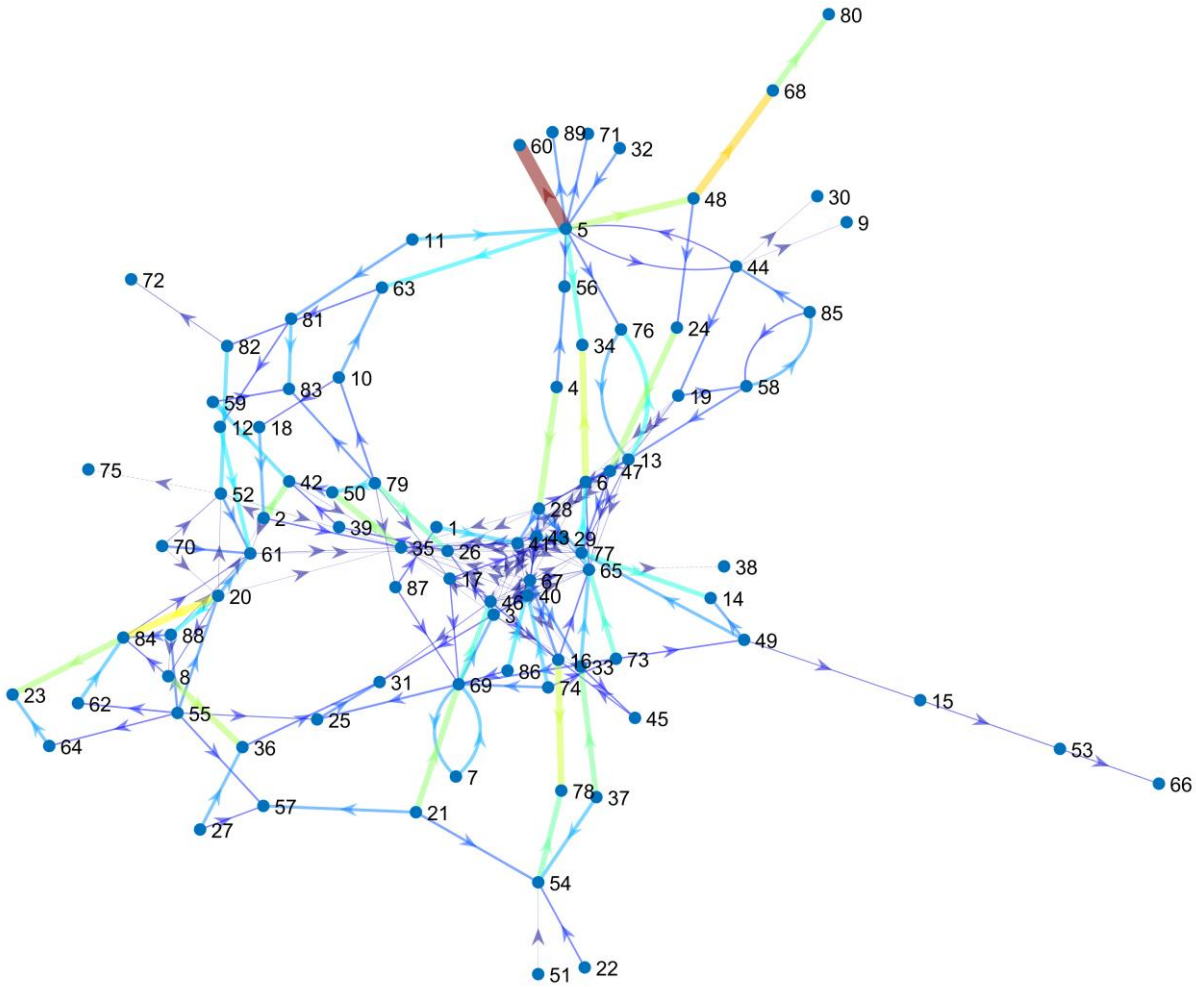
1.3.4 System *case89pegase* (rys. 2.14)

Najważniejsze cechy:

- 89 węzłów, 12 generatorów, 210 linii lub transformatorów,
- System reprezentuje wycinek europejskiej sieci przesyłowej
- Napięcia znamionowe: 150, 220, 380 kV

Brak limitów na niektórych elementach systemu. Ustawiono tam wartości domyślne przy użyciu funkcji *mk_default* (rozdz. **Błąd! Nie można odnaleźć źródła odwołania.**).

Uzyskane efekty wizualne są niezadowalające. Zaobserwowano zdecydowanie zbyt duże zagęszczenie przecięć krawedzi w centralnym obszarze, co uniemożliwia jakiekolwiek wykorzystanie tego systemu w praktyce. Dodatkowo występuje wiele krawędzi, które się przecinają. Podsumowując, metody wizualizacji grafów zastosowane w programie KASKADA oraz występujące w środowisku MATLAB nie są zbyt dobrze dobrane dla systemów tej klasy.



Rys. 0.8. System *case89pegase*.

1.3.5 System *case118* (rys. 2.15)

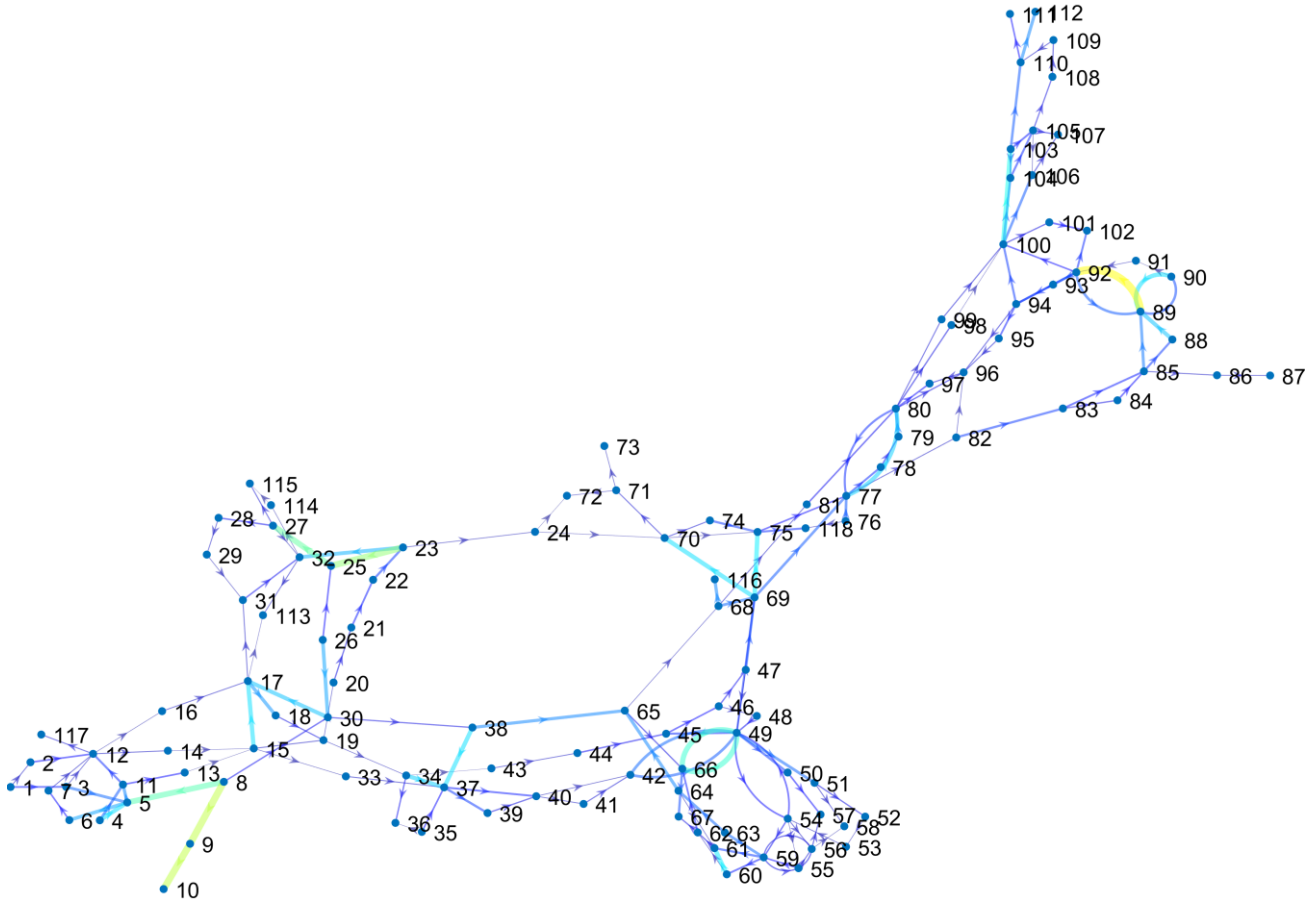
Podstawowe informacje na temat tego systemu podano w rozdziale **Błąd! Nie można odnaleźć źródła odwołania.**

Uzyskany efekt wizualizacji w przypadku systemu 118-węzłowego jest w takim samym stopniu zadowalający, jak w przypadku systemu 39-węzłowego. Fakt ten, jak i duża liczba potencjalnych wariantów wyłączeń, pozwalają wskazać również ten system, jako rekomendowany do procesów dydaktycznych. Warto zaznaczyć, że również w tym przypadku brakowało części informacji na temat limitów na połączeniach. Z tego względu przyjęto wartości domyślne przy wykorzystaniu funkcji *mk_default* (rozdz. **Błąd! Nie można odnaleźć źródła odwołania.**).

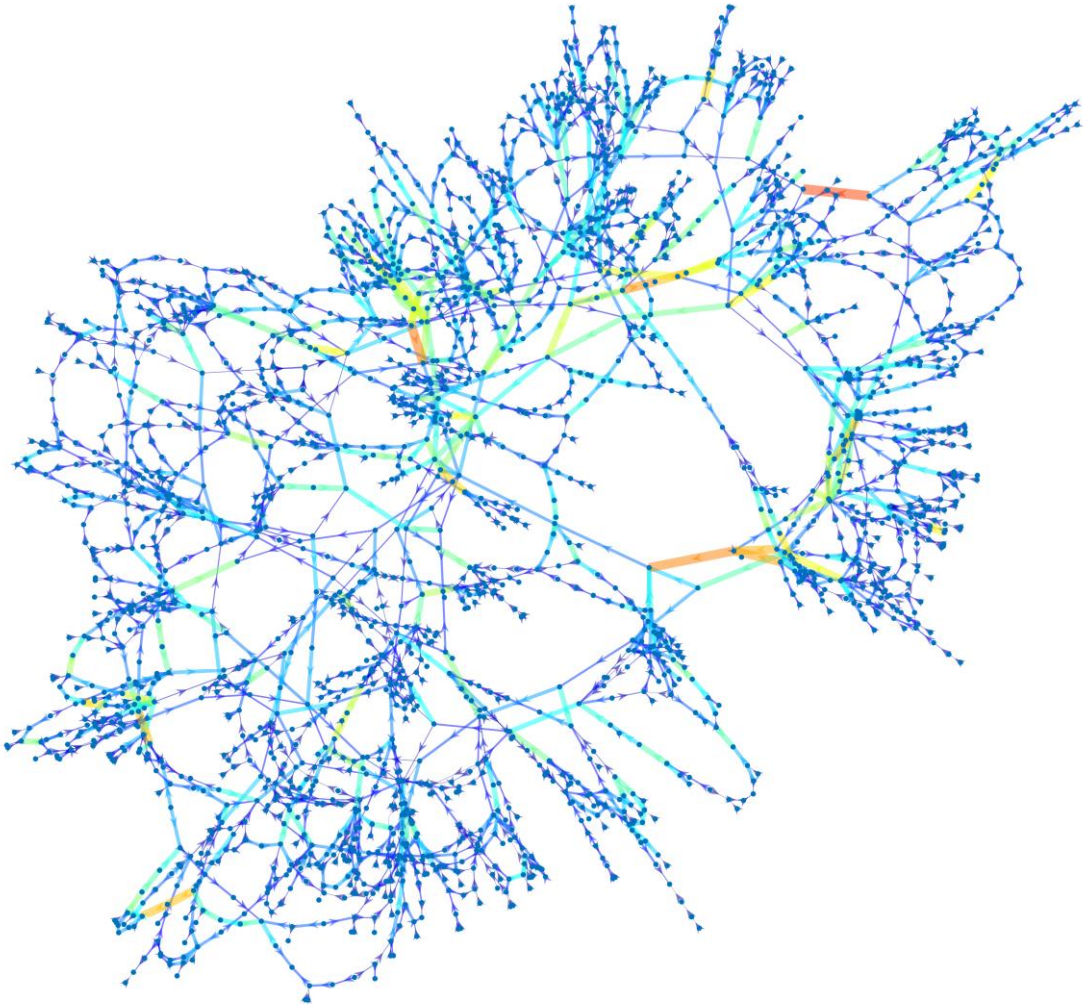
1.3.6 System *case2383wp* (rys. 2.16)

System case 2383wp został zaimplementowany do programu głównie w celach sprawdzenia, jak algorytmy wizualizacyjne oraz rozpływowe będą się „sprawdzać” dla naprawdę dużych systemów elektroenergetycznych. Jest to model sezonowy polskiego KSE w trakcie szczytu

zimowego na przełomie lat 1999/2000. System ten operuje na napięciach 110, 220 oraz 400 kV. Generacja ponad 2000 węzłów oraz niemal 3000 połączeń między nimi, przerasta możliwości przeciętnego komputera klasy PC. Obliczenia rozpływowe, a przede wszystkim wizualizacja trwają stanowczo zbyt długo, aby móc zastosować systemy o takim rozmiarze w programie KASKADA na przeciętnym sprzęcie komputerowym. Sam efekt wizualizacyjny jest akceptowalny, nie występuje takie zagęszczenie przecięć jak w systemie 89-węzłowym, a ponadto węzły są równo rozmieszczone na płaszczyźnie, dzięki czemu uzyskuje się bardzo dobrą przejrzystość schematu – mając na uwadze liczbę jego elementów. W tym systemie, w celach prezentacyjnych usunięto również etykiety węzłów z rysunku.



Rys. 0.9 System *case118*.



Rys. 0.10. System *case2383wp*.

Z.1. Klasa KASKADA (program główny)

Tablica Z. 1. Klasa KASKADA

```
classdef CascadeGUI < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        MainWindow        matlab.ui.Figure                    % KASKADA
        TabGroup          matlab.ui.container.TabGroup        % System, Kas...
        TabBox            matlab.ui.container.Tab             % System
        LoadButton        matlab.ui.control.Button            % Wczytaj model
        DrawButton        matlab.ui.control.Button            % Rysuj system
        ListBox           matlab.ui.control.ListBox           % case9, case...
        Label             matlab.ui.control.Label             % Nazwa modelu
        Tab               matlab.ui.container.Tab             % Kaskada
        TypeKnob          matlab.ui.control.DiscreteKnob      % Wszystkie, ...
        StartButton       matlab.ui.control.Button            % START
        TextArea          matlab.ui.control.TextArea
        NinetyDegreeGauge matlab.ui.control.NinetyDegreeGauge % [0 100]
        Label2            matlab.ui.control.Label             % Poziom zasi...
        Label3            matlab.ui.control.Label             % odbiorców
        ModelKnob         matlab.ui.control.DiscreteKnob      % AC, DC, MIX
    end


    properties (Access = private)
        mpr % mpr
        mpc
        hPlot
        hGraph
        hBar
        hFigure
        XData
        YData
        NodeName
        result
    end

    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.ListBox
            app.MainWindow.Position = getPosition('menu', 400, 400);
            app.TabGroup.Position(3) = app.MainWindow.Position(3);
            app.TabGroup.Position(4) = app.MainWindow.Position(4);
            app.hFigure = figure;
            app.hFigure.Position = getPosition('graph', 400, 400)
            app.hFigure.IntegerHandle = 'off';
            app.hFigure.Name = 'Okno wizializacji';

        end

        % LoadButton button pushed function
        function load(app)
            app.mpc = loadcase(app.ListBox.Value);
```

```
            app.mpc = check_graph(app.mpc);
            if size(app.mpc.bus,1)~= max(app.mpc.bus(:,1));
                app.mpc = reNum(app.mpc);
            end
            %Wypełnienie pustych limitów danymi domyślnymi, jeśli potrzeba
            app.mpc = mk_default(app.mpc,0.900);
            app.mpr = solvePF(app.mpc);
            [app.hPlot,app.hGraph,app.hBar] = plotGrid_o(app.mpr);
            app.XData = app.hPlot.XData;
            app.YData = app.hPlot.YData;
            app.NodeName = app.hPlot.NodeLabel;
            app.TextArea.Value ='';
            cla

            res2excel(app.mpr, 'load.xlsx')
            msgbox('Pomyslnie zaladowano model do pamieci')


        end

        % DrawButton button pushed function
        function draw(app)
            rePlot(app.mpr,app.hGraph,app.XData, app.YData, app.NodeName)

        end

        % StartButton button pushed function
        function StartCascade(app)
            define_constants
            pause('on');
            flagMode = strjoin(cellstr(app.ModelKnob.Value));
            flagType = strjoin(cellstr(app.TypeKnob.Value));
            noBr = size(app.mpr.branch,1);
            ExistingDemand0 = sum(app.mpr.bus(app.mpr.bus(:, BUS_TYPE) ~= 4, PD));
            %             ExistingGeneration0 = sum(app.mpr.gen(:, PG));
            tempMpr = app.mpr;
            licz = 0;
            newTextCount = 1;


            while noBr>2 && tempMpr.success == 1 && ~isempty(tempMpr.gen(:, GEN_STATUS)>0)
                licz = licz +1;
                app.mpr = tempMpr;
                [lf, noBr] = calcFlowRatio(tempMpr);
                flag = any(lf>1);
                [app.hPlot,app.hBar] = rePlot(app.mpr, app.hGraph,app.XData, app.YData, ...
                app.NodeName);
                %
                app.hPlot.EdgeLabel={}
                if flag
                    overLoads = zeros(noBr,1);
                    overLoads(lf>1) = lf(lf>1);
                    switch flagType
                        case 'Wszystkie'

                            pause(2)
                            app.mpr.branch(lf>1, BR_STATUS) = 0;
                            tempReport = app.mpr.branch(lf>1, :);
```

```
                        case 'Pojedynczo'

                            pause(2)
                            [maxValue,maxId] = max(overLoads);
                            app.mpr.branch(maxId, BR_STATUS) = 0;
                            tempReport = app.mpr.branch(maxId, :);

                        case 'Recznie'

                            [app.mpr, tempReport] = chooseOutage(app.mpr, app.hPlot, ...
                            app.hGraph);
                    end
                else
                    [app.mpr, tempReport] = chooseOutage(app.mpr, app.hPlot, app.hGraph);
                end

                switch flagMode
                    case 'AC'
                        [tempMpr, isolData] = solvePF(app.mpr, 'AC');
                    case 'DC'
                        [tempMpr, isolData] = solvePF(app.mpr, 'DC');
                    case 'MIX'
                        [tempMpr, isolData] = solvePF(app.mpr, 'AC');
                        res2excel(tempMpr, sprintf(['licz_AC_' sprintf('%d',licz) '.xls']))
                        if (tempMpr.success == 0)
                            waitfor(msgbox('Przechodze na model DC'));
                            tempMpr = solvePF(app.mpr, 'DC');
                            res2excel(tempMpr, sprintf(['licz_DC_' sprintf('%d',licz)
                            '.xls']))
                        end
                end

                report(licz).outages = tempReport;
                report(licz).outages(:,18) =
                sqrt(tempReport(:,PF).^2+tempReport(:,QF).^2)./tempReport(:,RATE_A);
                report(licz).success = tempMpr.success;
                report(licz).ExistingDemand = sum(tempMpr.bus(tempMpr.bus(:,BUS_TYPE) ~= 4,...
                PD));
                report(licz).ExistingGeneration = sum(tempMpr.gen(:, PG));
                app.NinetyDegreeGauge.Value = report(licz).ExistingDemand /
                ExistingDemand0*100;

                for idResult = 1: size(report(licz).outages,1)
                    app.TextArea.Value{newTextCount} = sprintf('%d: Wylaczenie linii %d-%d o
                    przeciazeniu %.2f.', ...
                    licz, report(licz).outages(idResult,F_BUS),...
                    report(licz).outages(idResult,T_BUS),...
                    report(licz).outages(idResult,18));
                    newTextCount =  newTextCount+1;
                end
                if ~isempty(isolData.bus)
                    app.TextArea.Value{newTextCount} = sprintf('Brak mozliwosci pracy
                    wyspowej, wyizolowano nastepujace wezly:');
                    newTextCount =  newTextCount+1;
                    app.TextArea.Value{newTextCount} = sprintf('%d (Pd= %d,Pg= %d MW), ',...
                    isolData.bus');
                    newTextCount =  newTextCount+1;
```

```
                app.TextArea.Value{newTextCount} = sprintf('Sumarycznie wyizolowano %d
                odbiorcow o Pd = %d MW', ...
                size(isolData.bus,1),sum(isolData.bus(:,2)));
                newTextCount =  newTextCount+1;
                app.TextArea.Value{newTextCount} = sprintf('Sumarycznie wyizolowano %d
                generatorow o Pg = %d MW', ...
                size(isolData.gen,1),sum(isolData.gen(:,2)));
                newTextCount =  newTextCount+1;
            end
        end
        if tempMpr.success == 1
            [app.hPlot,app.hBar] = rePlot(tempMpr,app.hGraph,app.XData, app.YData,...
            app.NodeName );
        else
            [app.hPlot,app.hBar] = rePlot(app.mpr,app.hGraph,app.XData, app.YData, ...
            app.NodeName );
            report(licz).ExistingDemand = 0;
            app.NinetyDegreeGauge.Value = report(licz).ExistingDemand /
            ExistingDemand0*100;
        end
        app.TextArea.Value{newTextCount} = sprintf('KONIEC KASKADY');

        app.result = report;
        clear isolData

    end
end

% App initialization and construction
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create MainWindow
        app.MainWindow = uifigure;
        app.MainWindow.Position = [100 100 400 400];
        app.MainWindow.Name = 'KASKADA';
        setAutoResize(app, app.MainWindow, true)

        % Create TabGroup
        app.TabGroup = uitabgroup(app.MainWindow);
        app.TabGroup.Units = 'pixels';
        app.TabGroup.Position = [0 0 400 400];

        % Create TabBox
        app.TabBox = uitab(app.TabGroup);
        app.TabBox.Units = 'pixels';
        app.TabBox.Title = 'System';

        % Create LoadButton
        app.LoadButton = uibutton(app.TabBox, 'push');
        app.LoadButton.ButtonPushedFcn = createCallbackFcn(app, @load);
        app.LoadButton.FontSize = 16;
        app.LoadButton.FontWeight = 'bold';
        app.LoadButton.Position = [7 93 383 26];
        app.LoadButton.Text = 'Wczytaj model';
```

```
            % Create DrawButton
            app.DrawButton = uibutton(app.TabBox, 'push');
            app.DrawButton.ButtonPushedFcn = createCallbackFcn(app, @draw);
            app.DrawButton.FontSize = 16;
            app.DrawButton.FontWeight = 'bold';
            app.DrawButton.Position = [7 55 383 26];
            app.DrawButton.Text = 'Rysuj system';

            % Create ListBox
            app.ListBox = uilistbox(app.TabBox);
            app.ListBox.Items = {'case9', 'case39', 'case57', 'case89pegase', 'case118',...
            'case2383wp'};
            app.ListBox.FontSize = 16;
            app.ListBox.Position = [7 142 383 195];
            app.ListBox.Value = 'case9';

            % Create Label
            app.Label = uilabel(app.TabBox);
            app.Label.HorizontalAlignment = 'center';
            app.Label.VerticalAlignment = 'center';
            app.Label.FontSize = 20;
            app.Label.Position = [132 343 133 26];
            app.Label.Text = 'Nazwa modelu';

            % Create Tab
            app.Tab = uitab(app.TabGroup);
            app.Tab.Units = 'pixels';
            app.Tab.Title = 'Kaskada';

            % Create TypeKnob
            app.TypeKnob = uiknob(app.Tab, 'discrete');
            app.TypeKnob.Items = {'Wszystkie', 'Pojedynczo', 'Recznie'};
            app.TypeKnob.Position = [270 265 60 60];
            app.TypeKnob.Value = 'Pojedynczo';

            % Create StartButton
            app.StartButton = uibutton(app.Tab, 'push');
            app.StartButton.ButtonPushedFcn = createCallbackFcn(app, @StartCascade);
            app.StartButton.FontWeight = 'bold';
            app.StartButton.Position = [12 30 276 22];
            app.StartButton.Text = 'START';

            % Create TextArea
            app.TextArea = uitextarea(app.Tab);
            app.TextArea.FontSize = 10;
            app.TextArea.Position = [12 67 276 168];

            % Create NinetyDegreeGauge
            app.NinetyDegreeGauge = uigauge(app.Tab, 'ninetydegree');
            app.NinetyDegreeGauge.Position = [297 135 100 100];
            app.NinetyDegreeGauge.Value = 100;

            % Create Label2
            app.Label2 = uilabel(app.Tab);
            app.Label2.HorizontalAlignment = 'center';
            app.Label2.VerticalAlignment = 'center';
            app.Label2.Position = [296.5 107 93 15];
            app.Label2.Text = 'Poziom zasilenia';
```

```
            % Create Label3
            app.Label3 = uilabel(app.Tab);
            app.Label3.HorizontalAlignment = 'center';
            app.Label3.VerticalAlignment = 'center';
            app.Label3.Position = [315 93 57 15];
            app.Label3.Text = 'odbiorców';

            % Create ModelKnob
            app.ModelKnob = uiknob(app.Tab, 'discrete');
            app.ModelKnob.Items = {'AC', 'DC', 'MIX'};
            app.ModelKnob.Position = [81 264 60 60];
            app.ModelKnob.Value = 'MIX';
        end
    end

    methods (Access = public)

        % Construct app
        function app = CascadeGUI()

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.MainWindow)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.MainWindow)
        end
    end
end
```

Z.2. Funkcje zewnętrzne programu KASKADA

Tablica Z. 2. Funkcja calcFlowRatio

```
function [ flowRatio, noBr, fb, tb, pf, qf, sf, flow_limit] = calcFlowRatio( mpr )

define_constants
    noBr = size(mpr.branch,1);
    fb = mpr.branch(:,F_BUS);
    tb = mpr.branch(:,T_BUS);
    pf = mpr.branch(:,PF);
    qf = mpr.branch(:,QF);
    sf = sqrt(pf.^2+qf.^2);
    flow_limit = mpr.branch(:,RATE_A);
    % Obciazenie danej linii
    flowRatio = sf./flow_limit;
end
```

Tablica Z. 3. Funkcja check_graph

```
function mpc = check_graph( mpc )

    define_constants
    noMpcBr = size(mpc.branch,1);
    ftb=[mpc.branch(:,F_BUS) mpc.branch(:,T_BUS)];
    [~,id,~] = unique(ftb, 'rows', 'stable');
    temp_From = zeros(noMpcBr,1);
    repId = ones(noMpcBr,1);
    repId(id) = 0;
    repId = logical(repId);
    temp_From(repId) = mpc.branch(repId, F_BUS);
    mpc.branch(repId, F_BUS) = mpc.branch(repId, T_BUS);
    mpc.branch(repId, T_BUS) = temp_From(repId);
    %Posortowanie linii
    mpc.branch = sortrows(mpc.branch,[F_BUS T_BUS])
    clear repId temp_From;

end
```

Tablica Z. 4. Funkcja chooseOutage

```
function [ mpr, report ] = chooseOutage(mpr, hPlot, hGraph)

%   chooseOutage Interaktywny wybor linii do wylaczenia.
%   Funkcja wyszukuje minimalną odległość między środkiem każdej linii
%   grafu, a współrzędną pobraną od użytkownika. Na tej podstawie
%   wyszukiwany jest numer linii do wyłączenia (BR_STATUS = 0)
%   LPM - wybór linii do wyłączenia
%   PPM - anulowanie ostatniego wyboru
%   SPACE - wyjście z funkcji

    define_constants

    ftb=[mpr.branch(:,F_BUS) mpr.branch(:,T_BUS)];
    [~,ig,ib] = intersect(hGraph.Edges.EndNodes,ftb,'rows','stable');

    %Zamiana zmiennych tekstowych w parametrze "NodeLabel" na zmienne typu num
    tempG = table2array(hGraph.Edges);
    nodeId = cellfun(@str2num, hPlot.NodeLabel);

    %pętla w której uzyskuje się współrzędne x,y węzła fromBus i toBus
    sizeG = size(tempG,1);
    %prelokacja zmiennych
    xCenter = zeros(1,sizeG);
    yCenter = zeros(1,sizeG);

    for i = 1:sizeG
    xFrom = hPlot.XData(nodeId == tempG(i,1));
    yFrom = hPlot.YData(nodeId == tempG(i,1));
    xTo = hPlot.XData(nodeId == tempG(i,2));
    yTo = hPlot.YData(nodeId == tempG(i,2));
    xCenter(i) = (xFrom+xTo)/2;
    yCenter(i) = (yFrom+yTo)/2;
    end

    %zmienne dla przypadku: PPM
    copyBrStatus = mpr.branch(:, BR_STATUS);
    copyLineStyle = hPlot.LineStyle;

    %nadanie zmiennej wartosci paczatkowej
    clicButton = 1;

    while clicButton ~= 32      % Spacja zeby wyjsc z petli
        [clicX, clicY, clicButton] = ginput(1);

        switch clicButton
        case 1     % LPM
           %Obliczneie długości euklidesowej i wyszukanie minimum
           length = sqrt((clicX- xCenter).^2+(clicY- yCenter).^2);
           %poprawic moga byc bledy-->
           [minValB,~] = min(length(ig));
           minIdB = find(length(ig)==minValB);
           minIdG = find(length==minValB);
           mpr.branch(minIdB, BR_STATUS) = 0;



    highlight(hPlot, hGraph.Edges.EndNodes(minIdG,1),...
```

```
                hGraph.Edges.EndNodes(minIdG,2),...
                'LineStyle',':')

        case 3     % PPM
            mpr.branch(:, BR_STATUS) = copyBrStatus;
            hPlot.LineStyle = copyLineStyle;
        otherwise
            ...
        end
    end
    %Na potrzeby raportu
    report = mpr.branch(xor(mpr.branch(:,BR_STATUS), copyBrStatus),:);

end
```

Tablica Z. 5. Funkcja getPosition

```
function Position = getPosition(figure_type, widthF, heighF)

    screenSize = get(0,'Screensize');
    xp = screenSize(3);
    yp = screenSize(4);

        switch figure_type
        case 'menu'
                Position = [(xp-2*widthF)/2,(yp-heighF)/2,widthF,heighF];
        case 'graph'
                Position = [(xp)/2+10,(yp-heighF)/2,widthF,heighF];
        otherwise
            warning('Unexpected type')
        end
end
```

Tablica Z. 6. Funkcja mk_default

```
function mpc = mk_default(mpc,I_MAX)
    %mk_default: Funkcja nadająca domyslne wartości limitow
    %   S_lim [MVA] = sqrt(3) * 1.1 [kA] * Un_bus [kV]
    %   Jesli nie zdefiniowano napięć na węzłach to przyjmuje się 220kV

    define_constants
    id = find(~mpc.branch(:, RATE_A));
    %Napięcia węzłów
    mpc.bus(find(mpc.bus(:,BASE_KV)==0), BASE_KV) = 220;
    %Limity na liniach
    if(~isempty(id))
        noMpcBr = size(mpc.branch,1);
        which_bus(:,1) = mpc.branch(:, F_BUS);

        for i = 1 : noMpcBr
            which_bus(i,2) = mpc.bus(find(mpc.bus(:,BUS_I)==which_bus(i,1)),BASE_KV);
        end

        mpc.branch(id, RATE_A) = sqrt(3)*I_MAX*which_bus(id,2)
    end
end
```

Tablica Z. 7. Funkcja mycm

```
function mycm(choice)
    switch choice
        case 1
            c = jet;
        case 2
            c = [0 0 0
                0 0 1
                0 1 1
                1 1 0
                1 0.75 0
                1 0 0
                ];
        otherwise
            c=jet;
    end

    colormap(c);
end
```

Tablica Z. 8. Funkcja plotGrid_o

```
function [hPlot,hGraph,hBar]=plotGrid_o(mpr)

    define_constants
    mycm(1);

    noBr = size(mpr.branch,1);
    nodeLabel = mpr.bus(:,BUS_I);
    fb = mpr.branch(:,F_BUS);
    tb = mpr.branch(:,T_BUS);
    pf = mpr.branch(:,PF);
    qf = mpr.branch(:,QF);

    %Power flow sf ma kierunek taki jak moc czynna
    sf = sign(pf).*sqrt(pf.^2+qf.^2);
    flow_limit = mpr.branch(:,RATE_A);
    % Obciazenie danej linii
    lf = sf./flow_limit;

    hGraph = digraph(fb, tb, lf);

    LWidths = 5*abs(hGraph.Edges.Weight)+0.001;

    LColor = abs(hGraph.Edges.Weight);

    %Dodanie podpisów na krawędziach
    temp_string1 = num2str(round(hGraph.Edges.Weight,2));
    temp_string2 = cellstr(repmat({''},noBr,1));
    edgeLabels = strcat(temp_string1,temp_string2);

    %wyroznienie gruboscia linii oraz kolorem w funkcji obciazenia
    hPlot = plot(hGraph,'Layout','force','Iterations', 250,'LineWidth',LWidths,...
            'EdgeCData',LColor,'EdgeLabel', edgeLabels, 'NodeLabel', nodeLabel);
    % Options: 'auto' (default) | 'circle' | 'force' | 'layered' | 'subspace'

    %Wyzerowanie osi
    set(gca, 'XTick', [], 'YTick', [])
```

```
    %Wyroznienie linia przerywana linii wylaczonych
    id2 = (hGraph.Edges.Weight==0);
    highlight(hPlot,hGraph.Edges.EndNodes(id2,1),hGraph.Edges.EndNodes(id2,2),...
        'LineStyle','--')

    %dodanie belki z kolorami
    hBar = colorbar('Ticks', [0 0.25 0.5 0.75 1 1.25 1.5]);
    caxis(gca,[0 1.50]);
end
```

Tablica Z. 9. Funkcja reNum

```
function mpc = reNum(mpc)
    define_constants
    noBus = size(mpc.bus,1);
    copyMpc = mpc;
    mpc.bus(:,BUS_I) = 1:noBus;

    for i = 1 : noBus
       idBusNoF = mpc.branch(:,F_BUS)==copyMpc.bus(i,BUS_I);
       mpc.branch(idBusNoF,F_BUS) = mpc.bus(i,BUS_I);
       idBusNoT = mpc.branch(:,T_BUS)==copyMpc.bus(i,BUS_I);
       mpc.branch(idBusNoT,T_BUS) = mpc.bus(i,BUS_I);

       idGenNo = mpc.gen(:,GEN_BUS)==copyMpc.bus(i,BUS_I);
       mpc.gen(idGenNo,GEN_BUS) = mpc.bus(i,BUS_I);

    end
end
```

Tablica Z. 10. Funkcja rePlot

```
function [hPlotN,hBar]=rePlot(mpr,hGraph,XData,YData, NodeName)

    define_constants
    mycm(1);

    noG = size(hGraph.Edges,1);
    noBr = size(mpr.branch,1);

    %wyzerowanie wag
    hGraph.Edges.Weight(:) = 0;

    %
    fb = mpr.branch(:,F_BUS);
    tb = mpr.branch(:,T_BUS);
    pf = mpr.branch(:,PF);
    qf = mpr.branch(:,QF);

    %Power flow sf ma kierunek taki jak moc czynna
    sf = sign(pf).*sqrt(pf.^2+qf.^2);


    flow_limit = mpr.branch(:,RATE_A);
    % Stosunek obciazenia danej linii
    lf = sf./flow_limit;

    % Przypisanie nowych wag do grafu dla zmodyfikowanego, przez wyłaczenia
```

```
    % systemu. Porównanie macierzy - funkcja intersect
    tempB = [fb tb];
    [~,ig,ib] = intersect(hGraph.Edges.EndNodes,tempB,'rows','legacy');
    hGraph.Edges.Weight(ig) =  lf(ib);

    %Skalowanie długości linii za pomocą power flow, jeśli są jakieś
    %nienumeryczne śmieci, to grubość linii ustala się na 0,5. 1 w funkcji find
    %przyspiesza wykonanie (ogranicza liczbę wyszukiwanych niezerowych
    %elementow)

    if  isempty(find(~isfinite(hGraph.Edges.Weight),1))
        LWidths = 3*abs(hGraph.Edges.Weight)+0.1;
    else
        LWidths = 0.5;
    end

    %Dodanie podpisów na krawędziach
    temp_string1 = num2str(round(hGraph.Edges.Weight,2));
    temp_string2 = cellstr(repmat({''},noG,1));

    edgeLabels = strcat(temp_string1,temp_string2);
    LColor = abs(hGraph.Edges.Weight);

    hPlotN = plot(hGraph,'LineWidth',LWidths,'EdgeCData',LColor,...
             'XData', XData, 'YData', YData, 'NodeLabel', NodeName,...
             'EdgeLabel', edgeLabels);

    %Wyzerowanie osi
    set(gca, 'XTick', [], 'YTick', [])

    %Wyroznienie linia przerywana linii wylaczonych
    id2 = (hGraph.Edges.Weight==0);
    highlight(hPlotN,hGraph.Edges.EndNodes(id2,1),hGraph.Edges.EndNodes(id2,2),...
        'LineStyle','--')
    %Dodanie belki z kolorami
    hBar = colorbar('Ticks', [0 0.25 0.5 0.75 1 1.25 1.5]);
    caxis(gca,[0 1.50]);
end
```

Tablica Z. 11. Funkcja res2excel

```
function res2excel(res, filename)

    names = fieldnames(res);

    for i = 1: numel(names)
        if(~isstruct(res.(names{i})));
          xlswrite(filename,res.(names{i}),names{i});
        end
    end
end
```

Tablica Z. 12. Funkcja solvePF

```
function [mpr, isolData,mpck] = solvePF(mpr, solverType)
    %solvePF: Funkcja izolująca węzły pozbawione połączeń z siecią oraz
    %rozwiązująca rozpływ mocy
    %
    define_constants

    isolData.bus = [];
    isolData.branch = [];
    isolData.gen = [];

    mpck = extract_islands(mpr);

    noIslands = numel(mpck);
    if noIslands > 1

        for id=1:noIslands
            if ~isempty(find(mpck{id}.bus(:, BUS_TYPE) == 3,1))
                mpr = mpck{id};
            else
                isolData.bus = [ isolData.bus; mpck{id}.bus(:, BUS_I) ...
                                 mpck{id}.bus(:, PD)];
                isolData.branch = [ isolData.branch; mpck{id}.branch(:, F_BUS) ...
                                 mpck{id}.branch(:, T_BUS)];
                isolData.gen = [ isolData.gen; mpck{id}.gen(:, GEN_BUS) ...
                                 mpck{id}.gen(:, PG)];
            end
        end
        %06-11-2016 Dodanie generacji do struktury isolData.bus
        % isolData.bus = ['idBus' 'PD' 'PG']
        isolData.bus(:,3) = 0;
        [~,temp,~] = intersect(isolData.bus(:,1), isolData.gen(:,1),'rows','stable');
        isolData.bus(temp,3) = isolData.gen(:,2);
    else
        mpr = mpck{1};
    end

    if ~isempty(mpr.gen(:, GEN_STATUS)>0)
    switch nargin
        case 1
            mpr = runpf(mpr);
        case 2
            solverType = mpoption('model', solverType);
            mpr = runpf(mpr, solverType);
    end
    else
        mpr.success = 0;
    end
end
```

Z.3. Funkcje do analizy błędów metod stałoprądowych

Tablica Z. 13. Skrypt do przeprowadzenia zbiorczej analizy błędów

```
cases = {'case9', 'case39', 'case118'};
RESULTS = struct;

for i=1:size(cases,2)
    RESULTS.(cases{i}) = compMethods(cases{i});
    TempE = getE(RESULTS.(cases{i}).mPTDF.E);
    RESULTS.(cases{i}).hHist = plotHist(cases{i}, TempE);
    RESULTS.(cases{i}).hPlot = plotFlows(cases{i}, RESULTS.(cases{i}).mAC.n0,...
        RESULTS.(cases{i}).mPTDF.n0,RESULTS.(cases{i}).RATE, RESULTS.(cases{i}).mDC.n0);
    RESULTS.(cases{i}).D = Analysis(RESULTS,cases{i});
end
```

Tablica Z. 14. Funkcja compMethods

```
    function RESULTS = compMethods(caseName)

    define_constants
    base_mpc = loadcase(caseName);
    rPTDF = n_1PTDF(base_mpc);
    rDC = n_1DC(base_mpc);
    rAC = n_1AC(base_mpc);

    % Sprawdzenie czy PTDF, jest zgodne z DC
    if (max(max(abs(rPTDF.n1.sf-rDC.n1.sf)))>10^-6)
        error('Przepływ bazowy nie zgadza sie!!!')
    end

    % Błędy
    E = rPTDF.n1.sf-rAC.n1.sf;
    RE = E./rAC.n1.sf;
    AE = abs(E);
    ARE = abs(E./rAC.n1.sf);

    TempE = getE(E);
    N = numel(TempE);
    ME = sum(TempE)/N;
    MAX_AE = max(abs(TempE));
    MAE = sum(abs(TempE))/N;
    RMSE = sqrt(transpose(TempE)*(TempE)/N);
    % Przygotowanie podpisów
    rowNamesSf = arrayfun(@(x,y) sprintf('Sf na linii %d_%d',x,y),...
        base_mpc.branch(:,F_BUS), base_mpc.branch(:,T_BUS),'UniformOutput', false)';
    colNames = arrayfun(@(x,y) sprintf('Linia%d_%d',x,y), base_mpc.branch(:,F_BUS),...
        base_mpc.branch(:,T_BUS),'UniformOutput', false)';

    % Struktura wynikowa
    RESULTS.RATE = base_mpc.branch(:,RATE_A);
    RESULTS.mDC.n0 = rDC.n0.sf;
    RESULTS.mDC.n1 = rDC.n1.sf;

    RESULTS.mAC.n0 = rAC.n0.sf;
    RESULTS.mAC.n1 = rAC.n1.sf;

    RESULTS.mPTDF.n0 = rPTDF.n0.sf;
```

```
    RESULTS.mPTDF.n1 = rPTDF.n1.sf;
    RESULTS.mPTDF.E = E;
    RESULTS.mPTDF.AE = AE;
    RESULTS.mPTDF.RE = RE;
    RESULTS.mPTDF.ARE = ARE;
    RESULTS.mPTDF.ERRORS.ME = ME;
    RESULTS.mPTDF.ERRORS.MAE = MAE;
    RESULTS.mPTDF.ERRORS.MAX_AE = MAX_AE;
    RESULTS.mPTDF.ERRORS.RMSE = RMSE;

end
```

Tablica Z. 15. Funkcja n_1PTDF

```
function  rPTDF = n_1PTDF(base_mpc)

    define_constants
    noBr = size(base_mpc.branch, 1);
    noBus = size(base_mpc.bus, 1);
    slackBus = base_mpc.bus(base_mpc.bus(:,BUS_TYPE)==3,BUS_I);
    PTDF = makePTDF(base_mpc, slackBus);
    [val, ibus, igen] = intersect(base_mpc.bus(:,BUS_I),base_mpc.gen(:,GEN_BUS));
    base_Pi = -base_mpc.bus(:,PD);
    base_Pi(ibus) = base_Pi(ibus) + base_mpc.gen(igen,PG);
    base_pf = PTDF*base_Pi;

    % Sprawdzenie czy nie ma bledu
    temp_mpr = rundcpf(base_mpc);
    temp_err = abs(temp_mpr.branch(:,PF)-base_pf);
    if (max(temp_err(:))>10^-6)
    error('Przepływ bazowy nie zgadza sie!!!')
    end
    clear temp_err temp_mpr;
    %

    pfPTDFn1 = nan(noBr);
    for LineNo = 1 : noBr;
            mpc = base_mpc;
            fromBus = mpc.branch(LineNo, F_BUS);
            toBus = mpc.branch(LineNo, T_BUS);
            baseBrFlow = base_pf(LineNo);
            fprintf('Linia %d: PTDF_bus%d = %f, PTDF_bus%d = %f\n', LineNo, fromBus, ...
            PTDF(LineNo, fromBus), toBus, PTDF(LineNo, toBus))
            syms x
            eqn = baseBrFlow + PTDF(LineNo, fromBus)*(x) + PTDF(LineNo, toBus)*(-x) == x;
            sol = solve(eqn, x);
            % odwrotna konwencja x nie jest wstrzyknięciem, a 'pobraniem' mocy:
            dPj = double(sol);
            if ~isempty(dPj)
                pj = zeros(noBus, 1);
                pj(fromBus) = dPj;
                pj(toBus) = -dPj;
                dPf_mat = PTDF*pj;
                pfPTDFn1(:, LineNo) = base_pf + dPf_mat;
                pfPTDFn1(LineNo, LineNo) = 0;
            end
    end
    rPTDF.n0.pf = base_pf;
    rPTDF.n0.qf = zeros(size(base_pf));
    rPTDF.n0.sf = sqrt(rPTDF.n0.pf.^2+rPTDF.n0.qf.^2);
```

```
    rPTDF.n1.pf = pfPTDFn1;
    rPTDF.n1.qf = zeros(size(pfPTDFn1));
    rPTDF.n1.sf = sqrt(rPTDF.n1.pf.^2+rPTDF.n1.qf.^2);
end
```

Tablica Z. 16. Funkcja n_1DC

```
function  rDC = n_1DC(base_mpc)
    define_constants
    noBr = size(base_mpc.branch, 1);
    noBus = size(base_mpc.bus, 1);
    base_mpr = rundcpf(base_mpc);
    pfDCn1 = nan(noBr);
    qfDCn1 = nan(noBr);
    for LineNo = 1 : noBr;
        mpc = base_mpc;
        mpc.branch(LineNo,BR_STATUS) = 0;
        mpr = rundcpf(mpc);
        if (mpr.success ==1)
            pfDCn1(:, LineNo) = mpr.branch(:,PF);
            qfDCn1(:, LineNo) = mpr.branch(:,QF);
        else
            pfDCn1(:, LineNo) = NaN;
            qfDCn1(:, LineNo) = NaN;
        end
    end
    rDC.n0.pf = base_mpr.branch(:,PF);
    rDC.n0.qf = base_mpr.branch(:,QF);
    rDC.n0.sf = sqrt(rDC.n0.pf.^2+rDC.n0.qf.^2);

    rDC.n1.pf = pfDCn1;
    rDC.n1.qf = qfDCn1;
    rDC.n1.sf = sqrt(rDC.n1.pf.^2+rDC.n1.qf.^2);

end
```

Tablica Z. 17. Funkcja n_1AC

```
function  rAC = n_1AC(base_mpc)
    define_constants
    noBr = size(base_mpc.branch, 1);
    noBus = size(base_mpc.bus, 1);
    base_mpr = runpf(base_mpc);
    pfACn1 = nan(noBr);
    qfACn1 = nan(noBr);
    for LineNo = 1 : noBr;
        mpc = base_mpc;
        mpc.branch(LineNo,BR_STATUS) = 0;
        mpr = runpf(mpc);
        if (mpr.success ==1)
            pfACn1(:, LineNo) = mpr.branch(:,PF);
            qfACn1(:, LineNo) = mpr.branch(:,QF);
        else
            pfACn1(:, LineNo) = NaN;
            qfACn1(:, LineNo) = NaN;
        end
    end
    rAC.n0.pf = base_mpr.branch(:,PF);
```

```
    rAC.n0.qf = base_mpr.branch(:,QF);
    rAC.n0.sf = sqrt(rAC.n0.pf.^2+rAC.n0.qf.^2);

    rAC.n1.pf = pfACn1;
    rAC.n1.qf = qfACn1;
    rAC.n1.sf = sqrt(rAC.n1.pf.^2+rAC.n1.qf.^2);
end
```

Tablica Z. 18. Funkcja getE

```
function newE = getE(E)
    IdE = and(isfinite(E),~eye(size(E)));
    newE = E(IdE);
end
```

Tablica Z. 19. Funkcja plotFlows

```
function f1 = plotFlows(caseName,sfAC, sfPTDF, RATE, sfDC)
    define_constants
    if(nargin == 5)
            % Sprawdzenie czy PTDF, jest zgodne z DC
        if (max(abs(sfPTDF-sfDC))>10^-6)
            error('Przepływ bazowy nie zgadza sie!!!')
        end
    end
    f1 = figure;
    axis
    grid on
    hold on
    p1 = plot(sfAC,'Color','black','Marker', 'x');
    p2 = plot(sfPTDF,'Color','blue');
    legend([p1 p2], 'Przepływ AC','Przepływ DC')
    if ~isempty(find(RATE,1))
        p3 = plot(RATE, 'LineStyle', '--','Color','black');
        legend([p1 p2 p3], 'Przepływ AC','Przepływ DC', 'Limit')
    end
    title(sprintf('Przepływy mocy pozornej w systemie %s', caseName));
    xlabel('Numer połączenia w systemie')
    ylabel('Przepływ mocy pozornej [MVA]')
    [M,I] = max(abs(sfPTDF-sfAC));
    xlim([-1 numel(sfAC)]+1);
    a = annotation('arrow');
    a.Color = 'red';
    a.Parent = gca;
    a.X = [I I];
    a.Y = [sfPTDF(I) sfAC(I)];
    a.HeadStyle = 'vback3'
    a.HeadLength = 5;
    a.LineWidth = 1;
    a.HeadWidth = 7;
    hold off
    ax = gca;
    ax.XTickMode = 'manual'
    ax.YTickMode = 'manual'
    savefig(gcf,[caseName 'p'],'compact')
    print([caseName 'p'],'-dpng','-r300');
end
```

Tablica Z. 20. Funkcja plotHist

```
function hHist = plotHist(caseName, E)
    figure
    hHist = histogram(E,'Normalization', 'probability', 'BinWidth', 10);
    title(sprintf('Rozkład błędów dla systemu %s', caseName));
    grid on;

    axis([-160, 60, 0, 0.8]);
    ax.XTick = [-160:20:60];
    xlabel('Przedziały wartości błędu [MVA]')
    ylabel('Częstość występowania [-]')
    ax = gca;
    ax.XTickMode = 'manual'
    ax.YTickMode = 'manual'
    savefig(gcf,[caseName 'h'],'compact')
    print([caseName 'h'],'-dpng','-r300');
end
```

Tablica Z. 21. Funkcja Analysis

```
function D = Analysis(RESULTS, caseName)

    RATE = RESULTS.(caseName).RATE;
    E = RESULTS.(caseName).mPTDF.E;
    AC = RESULTS.(caseName).mAC.n1;
    DC = RESULTS.(caseName).mPTDF.n1;

    if isempty(find(RATE,1))
        RATE = repmat(max(max(AC)), size(AC));
    else
        RATE = repmat(RATE, 1, size(AC,2));
    end

    sW = AC./RATE;
    D = zeros(10,2);
    for t = 1:10
        threshold = t/10;
        tempE = zeros(size(E));
        tempE(sW>threshold) = E(sW>threshold);
        tempE = tempE./RATE;
        distR = max(max(abs(tempE)));
        D(t,1) = t/10;
        D(t,2) = distR;
    end
end
```