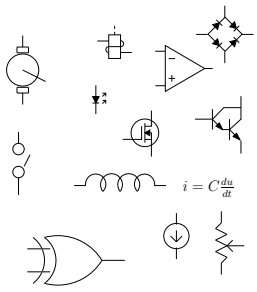
Symulator Mikrosystemów Dero v4

Metody i algorytmy obliczeniowe, modelowanie behawioralne, przykłady

Paweł Plaskura



ISBN 978-83-937245-1-2

Recenzent: dr inż. Andrzej Góralski

Dokument złożono systemem składu LaTeX

Wykorzystano system operacyjny Linux (Debian) 

Wersja dokumentu: 1.6

Symulator Mikrosystemów Dero v4
Metody i algorytmy obliczeniowe, modelowanie behawioralne, przykłady

Paweł Plaskura

http://epub.aiva.pl?isbn=978-83-937245-1-2

AIVA
ul. Wojska Polskiego 135A/25
97-300 Piotrków Trybunalski
Poland
http://www.aiva.pl
biuro@aiva.pl

ISBN 978-83-937245-1-2

Recenzja

W pracy przedstawiono symulator mikrosystemów *Dero* . Większość metod obliczeniowych została opracowana na potrzeby symulacji układów elektronicznych. W symulatorze wykorzystano metodę analogii, co zostało szczegółowo opisane. Wiedza zawarta w publikacji obejmuje swym zakresem zarówno zarys podstaw symulacji, metod i algorytmów obliczeniowych oraz zagadnienia modelowania elementów i przykłady użycia programu. Pełne przedstawienie zagadnień związanych z symulacją czyni niniejszą książkę unikalną na rynku polskim w postaci pozycji literaturowej nieodzownej dla osób zajmujących się analizą i symulacją układów.

Zakres wiedzy teoretycznej związany z symulacją przedstawiono w sposób umożliwiający nawet nieprzygotowanemu czytelnikowi dogłębne zrozumienie dosyć skomplikowanych zagadnień analizy i symulacji. Prawidłowe modelowanie nie jest możliwe bez właściwej implementacji i wykorzystania modeli elementów. Mocną stroną symulatora jest oryginalny język opisu modeli elementów. Opracowany i zaimplementowany algorytm obliczania pochodnych w modelach umożliwia uproszczenie opisu matematycznego modeli. W programie zaimplementowano oryginalny system śledzenia błędów programu, wykrywania błędów algorytmów oraz śledzenia poprawności wykonywanych obliczeń. Są to unikalne cechy opisywanego symulatora. Wspomniane zagadnienia zostały przedstawione w sposób pełny, bez niedomówień i pominięć. Jest to szczególnie cenne. Umożliwia to samodzielne tworzenie własnych poprawnych modeli elementów oraz układów. W pracy zamieszczono kompletne przykłady użycia symulatora.

Bardzo ważną cechą oprogramowania jest jego elastyczność. *Dero* może pracować jako program wywoływany z linii komendy lub usługa serwera. Ważną cechą jest możliwość integracji z platformą e-learningową *Quela* , co umożliwia wykorzystanie symulatora poprzez stronę WWW. Tworzone jest w ten sposób sieciowe środowisko symulacji, które można wykorzystać z każdego urządzenia (także mobilnego) wyposażonego w przeglądarkę i maszynę wirtualną *Java*. W pracy zagadnienia tez zostały omówione pobieżnie, gdyż nie są tematem wiodącym. Autor odsyła do odpowiedniej strony WWW projektu.

Zawarta w pracy wiedza została przedstawiona w sposób jasny i czytelny dzięki wieloletniemu interdyscyplinarnemu doświadczeniu autora.

dr inż. Andrzej Góralski

Spis treści

I Dokumentacja użytkowa **9**

1 Wstęp **11**
1.1 Dystrybucje i instalacja programu 13
1.2 Uruchamianie programu 14

2 Opis układu **17**
2.1 Stosowane oznaczenia i formaty zapisu danych 18
2.2 Linia elementu 20
2.3 Linia modelu 20
2.4 Linia podukładu 21
2.5 Definicja podukładu 21
2.6 Wprowadzanie plików 22
2.7 Przykład 23

3 Modelowanie w języku MDL **25**
3.1 .TYPE - definicja typu danych 25
3.1.1 Typ całkowity 26
3.1.2 Typ rzeczywisty 26
3.1.3 Typ wyliczeniowy 26
3.1.4 Przykłady definicji typów 27
3.2 .CONST - definicja stałej wbudowanej 27
3.3 .UNIT - definicja jednostki 28
3.4 .VARIABLE - definicja zmiennej sieciowej 28
3.5 .BUS - definicja szyny 29
3.6 .INPUT - definicja sterowania 29
3.7 .OUTPUT - definicja wyjścia modelu 30
3.8 .FUNCTION - definicja funkcji 31
3.9 .MODEL - definicja modelu 32
3.10 Instrukcje języka MDL 33
3.10.1 Operatory przypisania 33
3.10.2 Instrukcje warunkowe 33
3.10.3 Pętle 34
3.10.4 Operatory 34
3.10.5 Funkcje wbudowane i stałe 35
3.10.6 Zmienne i funkcje łączności z jądrem symulatora 36

4 Komendy **39**
4.1 Zlecenia analiz . . . 39
4.1.1 .OP - analiza punktu pracy . . . 40
4.1.2 .DC - analiza charakterystyk stałoprądowych . . . 40
4.1.3 .TR - analiza czasowa stanu nieustalonego . . . 41
4.1.4 .EDITA - analiza czasowa kierowana zdarzeniami . . . 41
4.1.5 .STDS - analiza czasowa stanu ustalonego . . . 42
4.1.6 .AC - małosygnałowa analiza częstotliwościowa . . . 43
4.1.7 .ODOS - wyznaczanie obszarów sprawności . . . 43
4.2 Dyrektywy warunków analiz . . . 44
4.2.1 .TBRP - generacja pułapek czasowych . . . 44
4.2.2 .IC - zadanie warunków początkowych . . . 44
4.2.3 .GROUP - grupowanie węzłów sieci . . . 45
4.2.4 .ALT - zmiana wartości parametru układu . . . 45
4.3 Wyprowadzanie wyników analiz . . . 46
4.3.1 .PRINT - wyprowadzanie wyników w postaci tekstowej . . . 46
4.3.2 .PROBE - formatowane wyprowadzanie wyników analiz . . . 47
4.3.3 Formaty plików danych . . . 47
4.4 Automatyczne projektowanie . . . 48
4.4.1 .SP - specyfikacje projektowe . . . 48
4.4.2 .VAR - zmienne optymalizacji . . . 50
4.4.3 .CENT - centrowanie deterministyczne . . . 52
4.4.4 .ERRC - statystyczne błędy optymalizacji . . . 52
4.4.5 .SVER - weryfikacja specyfikacji projektowych . . . 52
4.4.6 Przykład projektowania . . . 52
4.5 Opcje programu i analiz . . . 53
4.5.1 .OPTI - ustawianie opcji programu . . . 53

5 Integracja z oprogramowaniem zewnętrznym **59**
5.1 Integracja z platformą e-learningową *Quela* . . . 59
5.2 Integracja z edytorem Vim . . . 60

II Metody i algorytmy obliczeniowe **63**

6 Metody analizy i optymalizacji **65**
6.1 Równania sieci . . . 67
6.2 Algorytm Newtona-Raphsona . . . 68
6.2.1 Problemy numeryczne w algorytmie Newtona-Raphsona . . . 70
6.3 Schemat różnicowy Gear'a oparty na różnicach skalowanych . . . 71
6.3.1 Przewidywanie wartości startowej . . . 73
6.3.2 Lokalny błąd obcięcia . . . 74
6.3.3 Dobór kroku . . . 74
6.3.4 Dobór rzędu . . . 76
6.4 Schemat różnicowy trapezowy . . . 76
6.4.1 Przewidywanie wartości startowej . . . 76
6.5 Analiza czasowa . . . 79
6.6 Analiza czasowa stanu ustalonego . . . 82
6.7 Analiza czasowa kierowana zdarzeniami . . . 88

6.8 Małosygnałowa analiza częstotliwościowa . . . 92
6.8.1 Równania sieci . . . 92
6.9 Optymalizacja deterministyczna . . . 94
6.9.1 Wybrane elementy algorytmu . . . 96
6.10 Wyznaczanie obszarów sprawności . . . 100
6.10.1 Oznaczenia . . . 100
6.10.2 Podstawy teoretyczne . . . 100
6.10.3 Biliniowa funkcja układowa . . . 100
6.10.4 Analiza biliniowej funkcji układowej . . . 103

7 Równania sieci w różnych dziedzinach 111
7.1 Równania sieci w dziedzinie elektrycznej . . . 114
7.1.1 Gałąź typu prądowego . . . 114
7.1.2 Gałąź typu napięciowego . . . 115
7.1.3 Gałąź opisana równaniem ładunkowym . . . 116
7.1.4 Gałąź opisana równaniem strumieniowym . . . 116
7.1.5 Szablony równań dla elementów liniowych . . . 116
7.1.6 Szablony równań dla elementów nieliniowych . . . 120
7.2 Równania sieci w dziedzinie mechaniki . . . 122

8 Różniczkowanie symboliczne 125
8.1 Drzewo parsowania . . . 125
8.2 Różniczkowanie symboliczne . . . 126
8.2.1 Podstawowe przekształcenia drzewa . . . 126
8.2.2 Redukcja drzewa . . . 127
8.2.3 Generacja postaci symbolicznej . . . 129

9 Metody rozwiązywania układów równań liniowych 133
9.1 Zaawansowane metody klasyczne . . . 133
9.1.1 Klasyfikacja elementów niezerowych . . . 135
9.1.2 Wstępne przygotowanie macierzy . . . 136
9.1.3 Implementacja rozkładu LU . . . 137
9.1.4 Kontrola dokładności dekompozycji . . . 137

III Modele elementów 145

10 Modele biblioteczne 147
10.1 Predefiniowane typy . . . 148
10.2 Predefiniowane jednostki . . . 149
10.3 Predefiniowane stałe . . . 153
10.4 Zmienne sieciowe . . . 155
10.5 Definicje wejść modeli . . . 156
10.6 Definicje wyjść modeli . . . 157
10.7 Model rezystancji . . . 159
10.8 Model pojemności . . . 161
10.9 Model indukcyjności . . . 163
10.10 Modele źródeł . . . 165
10.10.1 Model źródła napięciowego . . . 165
10.10.2 Model źródła prądowego . . . 168

10.11Model diody 171
10.12Model diody tunelowej 175
10.13Model tranzystora BJT 176
10.14Model tranzystora MOS 186

IV Przykłady użycia programu 209

11 Układy liniowe 211
11.1 Dwójnik RLC 211
11.2 Sterowany ładunek 212
11.3 Sterowany strumień magnetyczny 213

12 Symulacja układów nieliniowych 215
12.1 Wzmacniacz z tranzystorem pracującym w układzie wspólnego emitera . 215
12.2 Wzmacniacz mocy 216
12.3 Wzmacniacz różnicowy 218
12.4 Idealny wzmacniacz operacyjny 219
12.5 Wzmacniacz operacyjny ua741 220

13 Symulacja układów cyfrowych 223
13.1 Bramki AND, NAND, OR, NOR 223
13.2 Inwerter MOS 225
13.3 Trzy inwertery MOS 226

14 Optymalizacja 229
14.1 Optymalizacja układu RR 229

15 Analiza czasowa kierowana zdarzeniami 231
15.1 Układ RC1 231
15.2 Układ RC2 232

16 Testy modeli bibliotecznych 235
16.1 Niezależne źródło napięciowe 235
16.2 Dioda półprzewodnikowa - DS 235
16.3 Tranzystor bipolarny - BJTL2 235
16.4 Tranzystor MOS 238

Bibliografia 241

Spis tablic 245

Spis rysunków 247

Oznaczenia 251

Słownik 253

Część I

Dokumentacja użytkowa

Rozdział 1

Wstęp

Od wielu lat symulacja mikrosystemów[1] stanowi szybko rozwijającą się dziedzinę. Na rynku można spotkać szereg tego typu układów np. mechaniczno-elektrycznych w samochodach, układach sterowania w przemyśle i w praktycznie każdej dziedzinie. Symulacja i optymalizacja mikrosystemów - układów pochodzących z różnych środowisk przez dedykowane oprogramowanie umożliwia podniesienie jakości i niezawodności projektowanych urządzeń. Szczególnie trudna jest symulacji układów, których części należą od różnych środowisk. Wymagane są różne algorytmy obliczeniowe i różne poziomy dokładności obliczeń dla poszczególnych środowisk. Wymaga to stosowania specjalistycznego oprogramowania.

Symulator Dero jest symulatorem układowym, pracującym pod kontrolą systemu operacyjnego Linux. Program został zaprojektowany i zaimplementowany w technice obiektowej w języku C++. Implementacja zgodnie ze standardami umożliwia stosunkowo łatwą przenośność na inne platformy sprzętowo-programowe. Dero jest samodzielnym programem, przystosowanym do uruchamiania z linii komendy[2].

Program umożliwia analizę układów liniowych i nieliniowych w dziedzinach sygnałów stałych, częstotliwości i czasu o mieszanym typie sygnałów (analogowo-cyfrowych)[3]. Może zostać wykorzystany jako narzędzie w procesie automatycznego projektowania w oparciu o wbudowane procedury optymalizacji deterministycznej. Optymalizacja polega na doborze takich parametrów układu, aby układ zmodyfikowany spełniał narzucone wymagania projektowe we wszystkich dziedzinach.

Akceptowanym opisem jest lista połączeń elementów wchodzących w skład układu - jest to tzw. symulator układowy. Istnieje możliwość zdefiniowania tzw. podukładów i ich wielokrotne wprowadzanie do układu głównego[4].

Analizowany układ może zawierać elementy opisane dowolnymi modelami. Dero posiada dedykowany język opisu modelu (MDL - *ang. Model Definition Language*) umożliwiający użytkownikowi tworzenie własnych modeli, np. uwzględniających symulowanie zależności temperaturowych. Modele mogą zostać umieszczone w opisie zadania[5]. Każdy

[1] Mikrosystemem może być np. układ mechaniczno-elektryczny, w którym znajdują się elementy pochodzące z różnych środowisk, np. siłownik ze środowiska mechanicznego i elektroniczny sterownik ze środowiska elektrycznego.

[2] Istnieje wiele wersji programu różniących się funkcjonalnością. Istnieje wersja symulatora *derod* (*ang. daemon*), która może pracować jako usługa serwera dostępna w sieci.

[3] Nie wszystkie wersje programu.

[4] Niektóre wersje programu dopuszczają definiowanie podukładów zagnieżdżonych.

[5] Niektóre modele elementów mogą zostać udostępnione w formie wbudowanej w program.

element dołączony jest do układu za pomocą węzłów. Węzły podłączone są do wejść i wyjść modeli elementów. Informację o zmianach wartości zmiennych sieciowych (sygnałów) przekazywane są do modeli przez wejścia (sterowania). Wyjścia reprezentują wartości odpowiednich funkcji układowych zależnych od wejść, pochodnych czasowych wejść i czasu (rozdział 6.1). MDL może zostać wykorzystany do modelowania mikrosystemów, gdyż umożliwia w szczególności definiowanie:

- typów danych (skalarnych i wyliczeniowych) z określeniem zakresu i tolerancji wartości,
- stałych wbudowanych,
- mnożników i jednostek,
- funkcji,
- zmiennych sieciowych wraz z poziomami dokładności analizy,
- sterowań modeli,
- wyjść modeli,
- modeli elementów.

Symulator Dero umożliwia wykonanie szeregu analiz układu wejściowego (nazwy analiz dla analizator układów elektronicznych - środowisko elektryczne):

OP analiza punktu pracy (*ang. Operating Point*),

DC charakterystyki zmian punktu pracy układu względem zmian jakiegokolwiek parametru (*ang. Direct Current*),

AC małosygnałowa analiza częstotliwościowa (AC *ang. Alternate Current*),

TR analiza czasowa stanu nieustalonego (*ang. Transient Analysis*),

ED-ITA iteracyjna analiza czasowa stanu nieustalonego kierowana zdarzeniami (*ang. Event-Driven Iterated Timing Analysis*) układów mieszanych analogowo-cyfrowych[6],

STDS analiza czasowa stanu ustalonego (*ang. Steady-State Analysis*).

Umożliwia także wykonanie optymalizacji deterministycznej oraz zmianę parametrów układu przez użytkownika. Posiada możliwość przekierowania strumienia wejściowego na standardowe wejście i tym samym interakcję z użytkownikiem.

[6] Tylko wybrane wersje programu

1.1 Dystrybucje i instalacja programu

Podstawową platformą pracy symulatora jest system Linux. W razie potrzeby można w prosty sposób stworzyć dowolną wersję na inne platformy. Program rozprowadzany jest w postaci pakietu *deb* dla dystrybucji Debian[7]. W pakiecie znajdują się biblioteki modeli elementów, przykłady oraz manual użytkownika w formacie *pdf* oraz *manpages*. Pakiet *deb* umożliwia prostą instalację oprogramowania przez administratora systemu (serwera) za pomocą standardowego programu do zarządzania pakietami oprogramowania *dpkg*[C+]. Instalacja wymaga uprawnień administratora systemu (root) na terminalu znakowym. Instalacja za pomocą systemu zarządzania pakietami umożliwia polecenia:

```
dpkg -i dero-4.00.deb
```

Program zabezpieczony jest przed nieautoryzowanym użyciem przy pomocy odpowiedniego certyfikatu. Po zainstalowaniu pakietu użytkownik musi wygenerować plik danych do wygenerowania certyfikatu[8] przy pomocy komendy:

```
dero -wcd dane_certyfikatu.txt
```

Jest to plik tekstowy, który powinien zostać wygenerowany przez program, a nie napisany za pomocą edytora tekstów. Przykład pliku *dane_certyfikatu.txt* otwartego w edytorze tekstów przedstawiono poniżej: Plik *dane_certyfikatu.txt* należy umieścić w systemie generowania certyfikatów na stronie WWW:

http://dero.aiva.pl

Wygenerowany plik certyfikatu zostanie umieszczony w serwisie WWW pobierania plików dla danego użytkownika (może także zostać przysłany pocztą elektroniczną). Plik ten należy pobrać z serwisu WWW i umieścić w katalogu domowym [9] użytkownika przy pomocy komendy:

```
cp certyfikat_uzytkownika.crt ~/.dero.crt
```

Po pomyślnym zainstalowaniu certyfikatu program jest gotowy do pracy. Plik certyfikatu *certyfikat_użytkownika.crt* jest plikiem binarnym zapisanym w formacie ASN.1 [LNC00] i zaszyfrowanym i nie można go zmieniać (zapisywać przy pomocy innych programów).

W przypadku generowania certyfikatu dla serwera, odpowiedni plik certyfikatu przeznaczonego dla serwera administrator powinien umieścić w katalogu konfiguracyjnym programu: */etc/dero/*:

```
cp certyfikat_serwera.crt /etc/dero/dero.crt
```

Należy nadać odpowiednie prawa odczytu pliku certyfikatu dla użytkowników programu.

```
chmod 444  /etc/dero/dero.crt
lub
chmod uga+r,uga-wx /etc/dero/dero.crt
```

[7] Można w łatwy sposób stworzyć inne formaty pakietów dystrybucyjnych, np. *rpm* czy *tgz*.

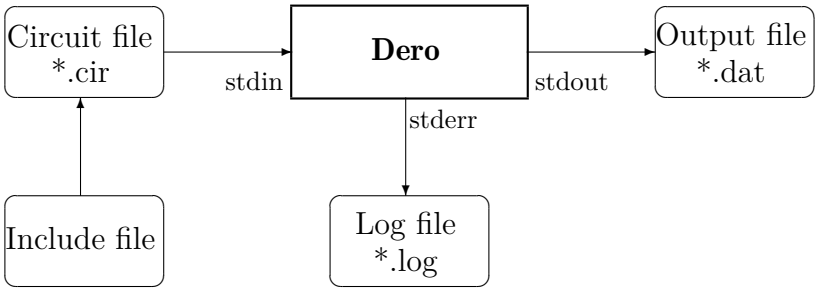
[8] Certyfikat może zostać wystawiony na określonego użytkownika, grupę użytkowników lub cały serwer (dowolna grupa i dowolny użytkownik). Prawidłowy certyfikat umożliwia uruchomienie programu.

[9] Program umożliwia załadowanie certyfikatu z innej lokalizacji z linii komendy (rozdział 1.2, opcja *-c*).

Program będzie mógł uruchomić użytkownik serwera mający prawo do odczytu pliku certyfikatu (tutaj wszyscy): */etc/dero/dero.crt*. Certyfikaty serwera mogą być rozprowadzane w postaci oddzielnych pakietów instalacyjnych w formacie *deb*.

1.2 Uruchamianie programu

Dero jest samodzielnym programem, przystosowanym do uruchamiania z linii komendy lub jako usługi serwera. Możliwa jest łatwa integracja z innymi programami. Możliwe jest również praca w trybie interaktywnym (tryb pomocny w projektowaniu) - dane wejściowe czytane są z konsoli, po zakończeniu możliwa jest kontynuacja odczytu danych z pliku wejściowego. Uruchomienie programu możliwe jest tylko po prawidłowym wczytaniu pliku certyfikatu[10]. Format linii wywołania programu wraz ze wszystkimi



Rys. 1.1: Wejście wyjście w programie

opcjami przedstawiono poniżej.

```
dero [-v] [-h] [-c crt_file.crt] [-rcd crt_data_file.txt]
     [-wcd crt_data_file.txt] [-gc crt_file.crt] [-rsa] [-m language_file]
     [-l -library_path] [-i input_file.cir] [-o data_file.dat]
     [-e error_file.txt]
```

Nawiasy [ ] oznaczają, że wystąpienie danego argumentu jest opcjonalne. Poszczególne opcje oznaczają odpowiednio:

-h, -?, –help - pomoc programu.

-v, –version - wersja programu.

-c, –certificate-file - czytaj plik certyfikatu *crt_file.crt*.

-rcd, –read-certificate-data-file - czytaj dane certyfikatu z pliku *crt_input_file.txt*.

-wcd, –write-certificate-data-file a - zapisz dane certyfikatu w pliku *crt_output_file.txt*.

-gc, –generate-certificate-file - wygeneruj certyfikat i zapisz w pliku *crt_output_file.crt*.

-sc, –show-certificate - pokaż certyfikat.

-rsa, –print-rsa-keys - drukuj klucze RSA.

[10] Certyfikat związany z użytkownikiem i sprzętem, na którym zainstalowano oprogramowanie.

-m, –**message-language** - czytaj plik zawierające komunikaty w językach narodowych *language_file*. Pliki języka znajdują się w katalogu: /usr/share/dero/lang. Przykładowo plik zawierający komunikaty w języku polskim ma nazwę *pl_PL*, w języku angielskim *en_EN*.

-mp –**print-masseges** - wypisz komunikaty programu.

-l, –**library** - ustaw ścieżkę dla bibliotek programu:*library_path*

-i, –**input-file** - czytaj dane wejściowe z pliku *input_file*.

-o, –**output-file** - zapisz plik wyjściowy: *data_file.dat*

-e, –**error-file** - zapisz błędy w pliku: *error_file*

W linii komendy można pominąć skojarzenia odpowiednich strumieni z plikami. W takim przypadku zostaną otwarte strumienie domyślne:

- wejście programu (in) zostanie skojarzone ze standardowym wejściem *stdin*,
- wyjście danych (dat) zostanie skojarzone ze standardowym wyjściem *stdout*,
- wyjście logu (log) zostanie skojarzone ze standardowym wyjściem błędów *stderr*,
- wyjście błędów programu (err) zostanie skojarzone ze standardowym wyjściem błędów *stderr*.

Rozdział 2

Opis układu

Program umożliwia przetwarzanie kilku układów (zadań) umieszczonych kolejno w strumieniu wejściowym (rys. 2.1).

Rys. 2.1: Struktura zadania

Pojedyncze zadanie rozpoczyna się dyrektywą *.TASK* i kończy się słowem kluczowym *.END*. Pojedyncze zadanie podzielone jest na dwie części:

1. opis układu, który może zawierać:
   - definicje:
     - typów

  - jednostek
  - zmiennych
  - funkcji
  - modeli
  - podobwodów
- deklaracje :
  - linii modeli przechowującej parametry modelu,
  - elementów z parametrami elementu,
  - linii podobwodów.

2. blok komend analiz rozpoczynający się od miejsca wystąpienia dyrektywy *.CMD*.

Podstawową jednostką opisu układu jest element dołączony do układu za pomocą węzłów. Z element skojarzony jest model matematyczny. Każdy element musi jawnie powoływać się na model za pośrednictwem linii modelu[1]. Wiele elementów może powoływać się na tą samą linię modelu[2]. Nazwy modeli, funkcji, zmiennych, podobwodów, muszą być unikalne w ramach obwodu głównego. Linie modeli oraz nazwy elementów muszą być unikalne w ramach obwodu głównego lub w bloku definicji podobwodu. Deklaracje węzłów, linii modeli, elementów i linii podobwodów muszą być unikalne w danym bloku. Deklaracje elementów, linii modeli i podobwodów umieszczane są w osobnych liniach. W programie mogą być dostępne dwa rodzaje modeli elementów:

- modele wewnętrzne wbudowane w program[3], które mogą być zoptymalizowane pod kątem lepszego wykorzystania algorytmów obliczeniowych.
- modele zewnętrzne definiowane przez użytkownika (rozdział 3.9).

2.1 Stosowane oznaczenia i formaty zapisu danych

Format zapisu dyrektyw Stosowane są następujące ogólne zasady opisu dyrektyw programu:

- Składnia dyrektyw programu jest umieszczona w ramce. Przykłady użycia dyrektyw najczęściej są podpisywane i także umieszczone w ramkach. Z lewej strony ramki mogą występować numery linii wydruku.
- Znakami rozdzielającymi poszczególne człony parametrów lub argumentów programu mogą być znaki ',' (przecinek) lub ' ' (spacja) w zależności od kontekstu,
- [ ] nawiasy kwadratowe oznaczają człony opcjonalne, które nie muszą wystąpić.

[1] Program nie dopuszcza domyślnego definiowania modelu elementu, tak jak np. w symulatorze Spice [Vla94] na podstawie pierwszej litery nazwy elementu.

[2] Zmiany parametrów modelu (w linii modelu) są widziane przez wszystkie elementy powołujące się na daną linię modelu. Wbudowany język opisu modelu (MDL) umożliwia definiowanie parametrów ustawianych w linii modelu (parametry modelu) i w linii elementu (parametry indywidualne elementu).

[3] Wybrane wersje programu.

Format zapisu liczb Program rozróżnia liczby całkowite i rzeczywiste (zmiennoprzecinkowe). Umożliwia także definiowanie jednostek i mnożników. Zapis liczb może przyjmować jedną z form:

- dla liczb całkowitych: 2, 100
- dla liczb zmiennoprzecinkowych: 1.23, $2e-15$, $2.4e7$
- w przypadku zdefiniowania mnożników (rozdział 3.3) istnieje możliwość zapisu złożonego: $1mA$ (zdefiniowano wcześniej mnożnik mA), $1.23e2kOhm$ (zdefiniowano wcześniej mnożnik $kOhm$).

Ciągi znaków Wszystkie ciągi znaków muszą zostać umieszczone w podwójnym cudzysłowie, np.:"Ciąg znaków". W ciągu znaków dopuszczalne są wszystkie znaki oprócz znaku cudzysłowu.

Zmienne sieciowe W programie zmienne sieciowe oznaczane są cyframi lub ciągiem znaków, który nie zawiera spacji. Zmienne węzłowe należące do podukładów lub modeli (zmienne wewnętrzne) poprzedzane są nazwą podukładu lub modelu oddzieloną separatorem: dwukropkiem dla podukładu, kropką dla modelu.

```
1 B − zmienna węzłowa B sieci w~obwodzie głównym
2 T1.C − zmienna wewnętrzna C (węzłowa lub gałęziowa) modelu
3                   elementu T1 w~obwodzie głównym
4 S1:B − zmienna węzłowa B sieci w~podobwodzie S1
5 S1:T1.C − zmienna wewnętrzna C (węzłowa lub gałęziowa) modelu
6                   elementu T1 w~podobwodzie S1
7 string<bus> − dołączenie elementu do zmiennych predefiniowanych jako szyna.
8                   Nazwy węzłów tworzone są przez dodanie do string zawartości definicji
9                   szyny (bus).
```

Dostęp do parametrów i opcji Dostęp do parametrów modelu lub elementu wymagany jest w bloku komend przy analizie parametrycznej. Można zmieniać wartości parametrów elementów i modeli w układzie głównym i w podukładach. Dostęp do parametrów jest realizowany zgodnie z zamieszczonym poniżej formatem.

```
1 [NazwaPodukładu:]NazwaElementu.NazwaParametruElementu
2 [NazwaPodukładu:]NazwaLiniiModelu..NazwaParametruModelu
```

Listy parametrów Listy parametrów wykorzystywane są w wielu komendach. Poszczególne parametry lub opcje w liście mogą zostać rozdzielone spacją lub opcjonalnym przecinkiem. Cała lista może zostać umieszczona w nawiasach zwykłych.

```
1 [(] par1=value [,] par2=value [,] ... [,] parN=value [)]
```

Dyrektywy programu Wszystkie dyrektywy w języku wejściowym programu zaczynają się od kropki. Najczęściej powinny zaczynać się od początku nowej linii. Dyrektywę lub komendę kończy znak nowej linii lub średnik lub w przypadku języka MDL znak średnika ";".

```
1 .DYREKTYWA [(] par1=value [,] ... [,] parN=value [)] [;]
```

Uwaga! W jednej linii nie może wystąpić wiecej niż jedna dyrektywa.

Komentarze Ciągi znaków rozpoczynające się znakiem '#' i kończące się znakiem nowej linii są komentarzami. Linie komentarza są pomijane przez program. Mogą zawierać dowolne znaki. Przykład linii komentarza przedstawiono poniżej.

```
1 # Linia komentarza
```

2.2 Linia elementu

Linia elementu umożliwia wprowadzenie elementu do układu (deklaracja elementu). Każdy element powołuje się na model poprzez linię modelu (rozdział 2.3). Linia modelu musi zostać umieszczona przed powołaniem się na nią, czyli przed linią elementu. Format linii elementu przedstawiono poniżej.

```
1 NazwaLiniiModelu.NazwaElementu ListaWęzłów [ListaParametrówElementu]
```

Nazwa linii modelu oddzielona jest kropką '.' od nazwy elementu. Lista węzłów zawiera nazwy węzłów zewnętrznych, którymi element dołączony jest do układu. Kolejność wystąpienia węzłów na liście jest zgodna z kolejnością wystąpienia w definicji modelu (dyrektywa *.PIN* - rozdział 3.9). W linii elementu można umieścić opcjonalną listę wartości parametrów elementu (parametry elementu) zdefiniowanych w modelu dyrektywą *.PARAM*. Kolejność parametrów na liście jest dowolna. Parametry, którym nie przypisano wartości w linii elementu otrzymują wartości domyślne (przypisane w modelu). Na wydruku 2.1 pokazano przykładowe deklaracje elementów.

Wydruk 2.1: Linie elementów - przykład

```
1 R.R1 1 2 R=10e3;
2 C.C2 2 3 C=1e-9;
3 O.Oper INP INM OUT 0
4 R.RS N<BUS> R=10e3;
```

W linii 1 zadeklarowano element o nazwie *R1* powołujący się na linię modelu o nazwie *R*. Element jest podłączony do układu za pomocą dwóch węzłów o nazwach *1* i *2*. Parametrowi elementu o nazwie *R* przypisano wartość $10e3$. W linii 2 zadeklarowano element o nazwie *C2* powołujący się na linię modelu *C*, dołączony do węzłów sieci *2* i *3*. Parametrowi elementu *C* przypisano wartość $1e-9$.

W linii 3 zadeklarowano element o nazwie *Oper* powołujący się na linie modelu o nazwie *O*. Element podłączony jest do układu za pomocą czterech węzłów o nazwach: *INP*, *INM*, *OUT*, *0*.

W linii 4 zadeklarowano element o nazwie *RS* podłączony do węzłów szyny (BUS). Każdy węzeł szyny otrzymuje przedrostek *N*. Sumaryczna liczba węzłów elementu musi być zgodna z definicją w modelu elementu.

2.3 Linia modelu

Linia modelu przechowuje wartości domyślne parametrów modelu zdefiniowanych w modelu jako *.COMMON*. Zmiana parametrów w linii modelu widoczna jest dla wszystkich elementów powołujących się na tą linię modelu. Zasięg widoczności linii modelu ograniczony jest do układy, tzn. w podukładach muszą zostać umieszczone linie modeli, na które będą powoływały się elementy. Format linii modelu przedstawiono poniżej.

```
1 .MODEL NazwaLiniiModelu NazwaModelu [ListaParametrówModelu]
```

Nazwa linii modelu musi być unikalna w obwodzie głównym lub w definicji podobwodu. Linie modelu zadeklarowane w obwodzie głównym i w definicji podobwodu są niezależne i mogą posiadać te same nazwy. Nazwa modelu jest nazwą modelu z linii definicji modelu *.MODEL* (rozdział 3.9). Definicja modelu musi zostać wcześniej wczytana, gdyż musi być widoczna w trakcie przetwarzania linii modelu. Na wydruku 2.2 przedstawiono przykład linii modeli.

Wydruk 2.2: Linie modeli - przykład

```
1 .MODEL R myR;
2 .MODEL TRANZYSTOR BJT2_E IS=1e-14;
```

W linii 1 zadeklarowano linię modelu o nazwie *R* powołującą się na model o nazwie *myR*. W linii 2 zadeklarowano linię parametrów wspólnych o nazwie *TRANZYSTOR* powołującą się na model o nazwie *BJT2_E*. Dodatkowo w linii parametrowi modelu *IS* przypisano wartość $1e-14$.

2.4 Linia podukładu

Linia podukładu umożliwia wprowadzenie do obwodu głównego zdefiniowanego wcześniej bloku układu (deklaracja podukładu, zobacz 2.5), który można wielokrotnie wprowadzać jako podukład. Format linii podukładu przedstawiono poniżej.

```
1 NazwaDefinicjiPodukładu:NazwaPodukładu ListaWęzłów [ListaParametrów]
```

Podukład dołączony jest do układu głównego przez węzły zewnętrzne (ListaWęzłów), których kolejność jest taka sama jak w definicji. Typy węzłów podłączenia muszą być zgodne. Węzły wewnętrzne podukładu dołączane są do listy węzłów obwodu głównego. Nazwa węzła wewnętrznego poprzedzona zostaje nazwą podukładu z dwukropkiem na końcu. Definicja podukładu musi wystąpić przed powołaniem się na nią.

Wydruk 2.3: Linia podobwodu - przykład

```
1 O741:O1 1 2 3 0;
```

Na wydruku 2.3 przedstawiono deklarację linii podukładu o nazwie *O1* powołującą się na definicję o nazwie *O741* (wydruk 2.4). Nazwa podukładu musi być unikalna. Podobwód dołączony jest do układu głównego za pomocą węzłów: *1*, *2*, *3*, *0*.

2.5 Definicja podukładu

Program umożliwia tworzenie definicji podukładów, które można włączać wielokrotnie do układu za pomocą linii podukładu (rozdział 2.4). Podukład dołączony jest za pomocą *WęzłówZewnętrznych*. Strukturę definicji podukładu przedstawiono poniżej.

```
1 .SUBCKT NazwaDefinicjiPodukładu ListaWęzłówZewnętrznych
2 LinieModeli
3 LinieElementów
4 .END
```

W definicji podukładu mogą zostać umieszczone linie modeli oraz linie elementów[4]. Węzły sieci, które nie zostały zdefiniowane jako węzły zewnętrzne (przyłączeniowe),

[4]W niektórych wersjach programu, definicje podukładów nie mogą być zagnieżdżone. Oznacza to, że w definicji podukładu nie można umieszczać linii innych podukładów.

stają się węzłami wewnętrznymi. W podukładach nie jest definiowany węzeł odniesienia (w układzie głównym oznaczany jako *0*). Jeśli elementy w podukładzie mają zostać dołączone do węzła odniesienia, to węzeł odniesienia musi znaleźć się na liście węzłów zewnętrznych. W linii deklaracji podukładu musi on zostać skojarzony z węzłem odniesienia obwodu głównego (węzłem *0*). Na wydruku 2.4 przedstawiono przykład definicji podobwodu wzmacniacza operacyjnego *O741* zbudowanego z elementów dyskretnych. Węzły *1*, *2*, *24*, *zero* w linii 1 to węzły zewnętrzne, które umożliwiają podłączenie podukładu (występują w linii deklaracji podukładu).

Wydruk 2.4: Definicja podobwodu opisującego wzmacniacz operacyjny *ua741*

```
1  .SUBCKT O741 1 2 24 zero
2  .MODEL R R;
3  .MODEL QNL BJT2_E (TYPE= 1,RC=1,RE=1,BF=80,RB=1,TF=1e-9,TR=20e-9,
4                     CJE=3e-12,CJC=2e-12,VAF=50,VAR=200,CJS=2e-12);
5  .MODEL QPL BJT2_E (TYPE=-1,RC=1,RE=1,BF=80,RB=1,TF=1e-9,TR=20e-9,
6                     CJE=3e-12,CJC=2e-12,VAF=50,VAR=210,CJS=2e-12);
7  .MODEL V SRCV_E (TD=20E-9,TR=9E-9,PW=30E-9,TF=12E-9,FREQ=0)
8  # tranzystory
9  QNL.Q1 3 2 4 4 ;
10 QNL.Q2 3 1 5 5 ;
11 QPL.Q3 7 6 4 4 ;
12 QPL.Q4 8 6 5 5 ;
13 QNL.Q5 7 9 10 10 ;
14 QNL.Q6 8 9 11 11 ;
15 QNL.Q7 VCC 7 9 9 ;
16 QNL.Q8 6 15 12 12 ;
17 QNL.Q9 15 15 VEE VEE ;
18 QPL.Q10 3 3 VCC VCC ;
19 QPL.Q11 6 3 VCC VCC ;
20 QPL.Q12 17 17 VCC VCC ;
21 QPL.Q14 22 17 VCC VCC ;
22 QNL.Q15 22 22 21 21 ;
23 QNL.Q16 22 21 20 20 ;
24 QNL.Q17 13 13 VEE VEE ;
25 QNL.Q18 VCC 8 14 14 ;
26 QNL.Q19 20 14 18 18 ;
27 QNL.Q20 22 23 24 24 ;
28 QPL.Q21 13 25 24 24 ;
29 QNL.Q22 VCC 22 23 23 ;
30 QPL.Q23 VEE 20 25 25 ;
31 # rezystancje
32 R.R1 10 VEE R=1e3 ;
33 R.R2 9 VEE R=50e3 ;
34 R.R3 11 VEE R=1e3 ;
35 R.R4 12 VEE R=3e3 ;
36 R.R5 15 17 R=39e3 ;
37 R.R6 21 20 R=40e3 ;
38 R.R7 14 VEE R=50e3 ;
39 R.R8 18 VEE R=50 ;
40 R.R9 24 25 R=25 ;
41 R.R10 23 24 R=50 ;
42 R.R11 13 VEE R=50e3 ;
43 # zasilanie
44 V.VCC VCC zero DC=15 ;
45 V.VEE VEE zero DC=-15 ;
46 .END
```

W liniach 2..7 umieszczono linie modeli (dyrektywy *.MODEL*), na które powołują się elementy. Modele elementów, do których odwołują się linie modeli (np. BJT2_E) muszą zostać wcześniej wczytane (przed blokiem definicji podobwodu). Linie 9..45 zawierają deklaracje elementów. Każdy element powołuje się na linię modelu. W każdej liniach 32..42 ustawiono wartość parametru *R* elementów *R1..11*. Każdy element opisany jest modelem *R* poprzez linię elementu o nazwie *R* (linia 2).

2.6 Wprowadzanie plików

Do strumienia wejściowego programu można wprowadzać zawartość innego pliku. Służą do tego dyrektywy *.LIB* lub *.INC*, które można umieścić w dowolnym miejscu w strumieniu wejściowym (w bloku opisu układu lub w bloku komend). Format dyrektyw zamieszczono poniżej.

```
1 .INC [CMD] "sciezka_do_pliku"
2 .LIB [CMD] "sciezka_do_pliku"
```

gdzie: CMD - jest opcjonalnym argumentem:

IFEXIST - plik jest wprowadzany tylko wtedy kiedy istnieje i ustawione są prawa dostępu do pliku.

Wprowadzanie plików realizowane jest na dwa sposoby:

- dyrektywą *.INC* można wprowadzać jakikolwiek plik. Jeśli nie podano pełnej ścieżki dostępu do pliku, to plik poszukiwany jest w katalogu roboczym
- dyrektywą *.LIB* można wprowadzać pliki biblioteczne umieszczone w katalogu bibliotek. Jeśli podano ścieżkę dostępu, to ścieżka ta traktowana jest jako ścieżka względna, względem katalogu bibliotek. Położenie katalogu bibliotek zapisane jest w konfiguracji programu lub może zostać podane w linii wywołania programu (rozdział 1.2).

Po zakończeniu czytania wprowadzanego pliku, strumień wejściowy jest przełączany na strumień, w którym wystąpiła dyrektywa *.LIB* lub *.INC*. Odczyt kontynuowany jest od następnej linii. Na wydruku 2.5 przedstawiono przykład dyrektyw wprowadzania plików.

Wydruk 2.5: Wprowadzanie zbiorów - przykład

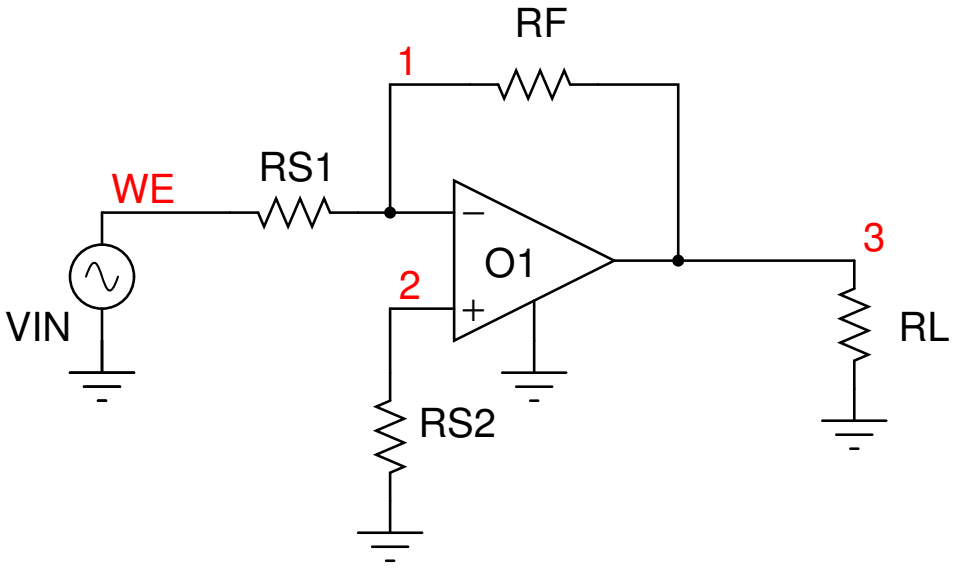
```
1 .INC "file.mdl"
2 .INC "/home/kowalski/bib/file.mdl"
3 .INC IFEXIST "/home/kowalski/bib/file.mdl"
4 .LIB "libfile.mdl"
5 .LIB IFEXIST "libfile.mdl"
```

Dyrektywa *.INC* w linii pierwszej powoduje wczytanie pliku *file.mdl* znajdującego się w katalogu roboczym. W drugiej linii zostanie wczytana zawartość pliku identyfikowanego przez podanie pełnej ścieżki dostępu *"/home/kowalski/bib/file.mdl"*. W trzeciej i w piątej linii plik wczytywany jest tylko wtedy, gdy istnieje i ustawione są odpowiednie prawa dostępu do pliku. W linii czwartej wczytywany jest plik *libfile.mdl* z katalogu bibliotecznego. Położenie katalogu bibliotecznego zapisane jest w konfiguracji programu lub można podać w linii wywołania programu.

2.7 Przykład

Na wydruku 2.6 przedstawiono plik wejściowy opisujący pojedyncze zadanie - wzmacniacz operacyjny pracujący w układzie odwracającym, którego schemat przedstawiono na rysunku 2.2. Tytuł zadania umieszczono w linii 2. Modele typów, zmiennych, funkcji, elementów zostały wprowadzone dyrektywami *.LIB* z katalogu bibliotek (linie 3..13). Definicje podukładu o nazwie *O741* opisującą wzmacniacz operacyjny (zobacz 2.4) zapisano w pliku o nazwie *ua741.sub* w katalogu roboczym. Zawartość pliku wprowadzono dyrektywą *.INC* (linia 15). W liniach 13..19 umieszczono linie modeli R, C i V powołujące się na odpowiednie modele (R, C i VT). W liniach 21..23 oraz 26..29 umieszczono deklaracje elementów (źródło napięciowe *VIN*, rezystory: *RS1*, *RS2*, *RF*, *RL*) oraz ustawiono odpowiednie wartości parametrów elementów (*DC* dla *VIN*, *R* dla *RS1*, *RS2*, *RF*, *RL*). Do obwodu głównego wprowadzono wzmacniacz operacyjny o nazwie *XO1* opisany przez podobwód *O741* (linia 26) - dołączony do węzłów: *1*, *2*, *WY*, *0* w obwodzie głównym. Plik definicji podobwodu wzmacniacza *O741* przedstawiono na wydruku 2.4.

Wydruk 2.6: Przykładowy plik (cir/nonlin-oper/ua741x1.cir)



Rys. 2.2: Przykładowy schemat analizowanego układu z dziedziny elektrycznej

```
1  # Revision : 1.1 (C) AIVA 2003-2010
2  .TASK "Operational Amplifiers ua741"
3  .LIB "types.mdl"
4  .LIB "units.mdl"
5  .LIB "vars.mdl"
6  .LIB "bjt/pnjlim.mdl"
7  .LIB "math/one.mdl"
8  .LIB "math/pulse.mdl"
9
10 .LIB "src/v.mdl"
11 .LIB "rlc/r.mdl"
12 .LIB "rlc/c.mdl"
13 .LIB "bjt/bjt2.mdl"
14
15 .INC "ua741.sub"
16
17 .MODEL R R;
18 .MODEL C C;
19 .MODEL V VT V1=0 V2=5,TD=5mS,TR=1mS,PW=7mS,TF=1mS,FREQ=0;
20
21 V.VIN IN 0 DC=-1;
22 R.RS1 IN 1 R =1kOhm;
23 R.RS2 2 0 R =0.5kOhm;
24
25 # - + out 0
26 O741:XO1 1 2 OUT 0;
27 C.CL OUT 1 C=0.1uF;
28 R.RF 1 OUT R=1kOhm;
29 R.RL OUT 0 R=100kOhm;
30 #
31 .CMD
32 .PRINT ADD TR("IN") TR("OUT") PAR(HN) PAR(RLTE) PAR(ORDER);
33 .PRINT ADD OP("IN") OP("OUT") PAR(NR);
34 .OP TITLE="OP - ua741";
35 .TRAN STEP=1mS TMIN=0mS TMAX=20mS HMAX=10mS TITLE="TRAN - ua741";
36 .END
```

Blok opisu układu kończy się w miejscu wystąpienia dyrektywy *.CMD* (linia 31) i rozpoczyna się blok komend. W linii 32 umieszczono dyrektywę wyprowadzania wyników analizy czasowej TR (zobacz 4.3.1). W linii 34 umieszczono dyrektywę analizy punktu pracy OP (zobacz 4.1.1). W linii 35 umieszczono dyrektywę wykonania analizy czasowej stanu nieustalonego TR (zobacz 4.1.3). Dyrektywa *.END* (linia 36) kończy zadanie.

Rozdział 3

Modelowanie w języku MDL

Symulator posiada dedykowany język opisu modelu (MDL - *ang. Model Definition Language*) umożliwiający definiowanie:

- typów danych (skalarnych i wyliczeniowych) z określeniem zakresu i tolerancji wartości,
- stałych wbudowanych,
- mnożników i jednostek,
- funkcji,
- zmiennych sieciowych wraz z poziomami dokładności analizy,
- sterowań modeli,
- wyjść modeli,
- modeli elementów.

Z poziomu języka MDL można model elementu może wykorzystywać zmienne wewnętrzne symulatora. Można także sterować procesem symulacji układu. Ze względu na wykorzystywany w programie algorytm Newtona-Raphsona (rozdział 6.2), istnieje konieczność obliczania pochodnych funkcji w modelach elementów. Jest to robione automatycznie. Odpowiedni kod obliczający pochodne jest generowany w trakcie przetwarzania modeli elementów - zobacz rozdział 8.

3.1 .TYPE - definicja typu danych

W programie wprowadzono możliwość definiowania nowych typów opartych na bazowych typach wewnętrznych symulatora:

- całkowitego,
- wyliczeniowego,
- rzeczywistego.

3.1.1 Typ całkowity

Zakres reprezentowanych liczb zależny jest od systemu. Struktura definicji typu całkowitego przyjmuje postać:

```
1 .TYPE NAME
2   LEFT =liczba_całkowita; # lewe ograniczenie
3   RIGHT=liczba_całkowita; # prawe ograniczenie
4   [INFO="ciąg_znaków";] # opis
5 .END
```

gdzie:

NAME jest nazwą (identyfikatorem) definiowanego typu,

LEFT jest dolnym ograniczeniem dopuszczalnych wartości,

RIGHT jest górnym ograniczeniem dopuszczalnych wartości,

INFO jest opcjonalnym opisem tekstowym.

3.1.2 Typ rzeczywisty

Zakres reprezentowanych liczb zależny jest od systemu. Struktura definicji dla typu rzeczywistego przedstawiona została poniżej:

```
1 .TYPE NAME
2   LEFT =liczba_rzeczywista; # lewe ograniczenie
3   RIGHT=liczba_rzeczywista; # prawe ograniczenie
4   RES =liczba_rzeczywista; # minimalna rozróżnialna wartość
5   [INFO="ciąg_znaków";] # opis
6 .END
```

gdzie:

NAME jest nazwą (identyfikatorem) definiowanego typu,

LEFT jest dolnym ograniczeniem dopuszczalnych wartości,

RIGHT jest górnym ograniczeniem dopuszczalnych wartości,

RES jest najmniejszą rozróżnialną wartością,

INFO jest opcjonalnym opisem tekstowym.

3.1.3 Typ wyliczeniowy

Wartości zmiennej typu wyliczeniowego są kolejnymi wartościami całkowitymi, jeżeli nie podano wartości dla poszczególnych pól. Wewnętrznie symulator przechowuje indeksy do pól typu wyliczeniowego - rozpoczynające się od 0. Definicja dla typu wyliczeniowego przedstawiona została poniżej:

```
1 .TYPE NAME
2   ENUM NAZWA1 [ = wartość_całkowita | wartość_rzeczywista];
3   ...
4   ENUM NAZWAN [ = wartość_całkowita | wartość_rzeczywista];
5   [INFO="ciąg_znaków";] # opis
6 .END
```

gdzie:

NAME jest nazwą (identyfikatorem) definiowanego typu,

ENUM definiuje listę elementów typu wyliczeniowego (NAZWA1 ... NAZWAN),

INFO jest opcjonalnym opisem tekstowym.

3.1.4 Przykłady definicji typów

Na wydruku 3.1 przedstawiono przykładowe definicje nowych typów danych.

Wydruk 3.1: Przykładowe definicje typów (lib/types.mdl)

```
1  # Revision : 1.1 (C) AIVA 2010
2  #
3  # BASIC TYPES
4  #
5
6  .TYPE NATURAL
7     LEFT = 0;
8     RIGHT = 1000000;
9  .END
10 #INFO = "liczby naturalne"
11
12 .TYPE INT
13    LEFT = -1000000;
14    RIGHT = 1000000;
15 .END
16 #INFO = "liczby całkowite"
17
18 # .TYPE LOGIC
19 # ENUM ZERO = 0;
20 # ENUM LOW = 0.2;
21 # ENUM HIGH = 3.8;
22 # ENUM ONE = 5.0;
23 # INFO = "typ wyliczeniowy"
24 # .END
25
26 .TYPE REAL
27    LEFT = -1E38;
28    RIGHT = 1E38;
29    TOLERANCE = 1.4E-12;
30 .END
31 # INFO = "liczby rzeczywiste"
```

W liniach 3..7 zdefiniowano typ całkowity o nazwie *NATURAL* o zakresie dopuszczalnych wartości $< 0, 1000000 >$.

W liniach 9..13 zdefiniowano typ całkowity o nazwie *INT* o zakresie dopuszczalnych wartości $< -1000000, 1000000 >$.

W liniach 15..21 zdefiniowano typ wyliczeniowy o nazwie *LOGIC*. Dopuszczalne wartości dla zmiennej typu to: ZERO, LOW, HIGH, ONE. Odpowiadają im odpowiednie poziomy sygnałów. Wartości typu wyliczeniowego posiadają identyfikatory od 0..3.

W liniach 23..28 zdefiniowano typ rzeczywisty o nazwie *REAL* o zakresie dopuszczalnych wartości $< -1e38, 1e38 >$ i najmniejszej rozróżnialnej wartości $1.4e - 12$.

3.2 .CONST - definicja stałej wbudowanej

Dyrektywa umożliwia definiowanie nowych stałych, które mogą zostać wykorzystane w modelach elementów. Domyślnie przyjmuje się, że wszystkie zdefiniowane stałe są typu rzeczywistego. Struktura definicji stałej przedstawiono poniżej:

```
1 .CONST NAME
2    VAL = wartość_całkowita | wartość_rzeczywista; # wartość stałej
3    [INFO= "ciąg znaków opisu";] # opis
4 .END
```

gdzie:

NAME jest nazwą definiowanej stałej,

VAL umożliwia ustawienie wartości stałej,

INFO definiuje ciąg znaków skojarzony ze definiowaną stałą (dodatkowa informacja tekstowa).

Na wydruku 3.2 przedstawiono przykładową definicję stałej π.

Wydruk 3.2: Przykładowa definicja stałej π

```
1 .CONST PI
2   VAL=3.14159;
3   INFO="Liczba PI";
4 .END
```

3.3 .UNIT - definicja jednostki

Program umożliwia definiowanie nowych jednostek używanych jako mnożniki. Strukturę definicji jednostek przedstawiono poniżej:

```
1 .UNIT
2   NAZWA1 = wartość_całkowita | wartość_rzeczywista;
3   NAZWA2 = wartość_całkowita | wartość_rzeczywista;
4   ...
5   NAZWAN = wartość_całkowita | wartość_rzeczywista;
6 .END
```

gdzie:

NAZWA1..N są unikalnymi nazwami jednostek, którym przypisano wartości.

Wartości jednostek mogą zostać jawnie zadeklarowane lub mogą zostać wyliczone na podstawie wcześniejszej definicji. Na wydruku 3.3 przedstawiono przykład definicji jednostek i mnożników. W liniach 3 i 5 wartości mnożników zostały wyliczone na podstawie wcześniej zdefiniowanych mnożników ($nV = 1e - 3uV$).

Wydruk 3.3: Przykładowa definicja jednostki

```
1 .UNIT
2         V = 1; # 1 volt - jednostka bazowa
3         mV = 1e-3V; # 1 mili Volt
4         uV = 1e-6; # 1 mikro Volt
5         nV = 1e-3uV; # 1 nano Volt
6 .END
```

3.4 .VARIABLE - definicja zmiennej sieciowej

Dyrektywa umożliwia zdefiniowania nowej zmiennej sieciowej. Możliwość definiowania nowych zmiennych sieciowych jest bardzo ważną cechą symulatora z punktu widzenia symulacji mikrosystemów. Program umożliwia zdefiniowanie typu wartości zmiennej sieciowej oraz poziomu dokładności. Strukturę definicji zmiennej sieciowej przedstawiono poniżej:

```
1 .VARIABLE NAME
2   TYPE = nazwa_typu; # typ wartości
3   TOLERANCE = wartość; # dokładność bezwzględna
4 .END
```

gdzie:

NAME jest identyfikatorem zmiennej,

TYPE definiuje typ zmiennej przechowującej wartość (patrz 3.1),

TOLERANCE definiuje dokładność bezwzględną zmiennej.

Zmiennymi sieciowymi mogą być np.: napięcia, prądy gałęzi, ładunki, strumienie. Na wydruku 3.4 przedstawiono przykładowe definicje zmiennych sieciowych.

Wydruk 3.4: Przykładowe definicje zmiennych sieciowych

```
.VARIABLE VV
   TYPE=REAL;
   TOLERANCE=1uV;
.END

.VARIABLE II
   TYPE=REAL;
   TOLERANCE=100nA;
.END

.VARIABLE FF
   TYPE=REAL;
   TOLERANCE=1u;
.END

.VARIABLE QQ
   TYPE=REAL;
   TOLERANCE=1p;
.END
```

3.5 .BUS - definicja szyny

Zmienne sieciowe mogą być grupowane w szyny (*ang. bus*) i używane w modelach i deklaracjach elementów. Strukturę definicji zmiennej sieciowej przedstawiono poniżej:

```
.BUS NAME
   NAME
   ...
   NAME
.END
```

gdzie:

NAME jest dowolną nazwą,

Definicja jest kolekcją nazwa, którym w trakcie procesu kompilacji układu nadaje się odpowiednie nazwy (także umieszczone w cudzysłowach), np.:

```
.BUS "N1"
   "A"
   "B"
.END

.BUS N2
   C
   D
.END
```

W opisie przykładowego układu nazwy zostaną rozwinięte, np.:

```
R.R X<N1> R=1
R.R X.A X.B R=1
```

Nazwa szyny oraz nazwy zmiennych szyny mogą zostać umieszczone w cudzysłowach.

3.6 .INPUT - definicja sterowania

Sterowanie wykorzystywane są w modelach elementów (rozdział 3.8). Muszą zostać zdefiniowane przed powołaniem się na nie. Struktury definicji sterowań została przedstawiona poniżej:

```
.INPUT NAME
   PLUS varid;
   [MINUS varid;] # sterowanie rożnicą zmiennych
   [INFO="ciąg_znaków";] # opis
.END
```

gdzie:

NAME jest nazwą sterowania, która powinna (ale nie musi) odpowiadać typowi zmiennej sterującej (rozdział 3.4),

PLUS jest identyfikator zmiennej sieciowej typu *varid* opisującej wartość dodatnią sygnału,

MINUS jest identyfikatorem zmiennej sieciowej typu *varid* opisującej wartość ujemną sygnału.

INFO definiuje ciąg znaków skojarzony z definiowanym sterowaniem (dodatkowa informacja tekstowa).

Wartość zmiennej sterującej jest różnicą dwóch sygnałów : $PLUS - MINUS$. Typy wartości dla zmiennych (*varid*) powinny była takie same (zobacz 3.4)[1]

Na wydruku 3.5 przedstawiono przykładowe definicje sterowań w dziedzinie elektrycznej.

Wydruk 3.5: Przykładowe definicje sterowań (plik biblioteczny inputs-e.mdl)

```
1  # Revision : 1.1 (C) AIVA 2010
2  #
3  # Prdefinded electrical inputs
4  #
5
6  # Electrical voltage input
7  .INPUT VV
8    PLUS => V;
9    MINUS => V;
10 .END
11
12 # Electrical current input
13 .INPUT II
14   PLUS => I;
15 .END
16
17 # Electrical flux input
18 .INPUT FF
19   PLUS => F;
20 .END
21
22 # Electrical charge input
23 .INPUT QQ
24   PLUS => Q;
25 .END
```

3.7 .OUTPUT - definicja wyjścia modelu

W języku MDL można definiować nowe typy gałęzi sieci (wyjść modelu)[2]. Wyjścia używane są do automatycznego układania równań sieci (rozdział 7). Wcześniej zdefiniowane wyjścia mogą zostać użyte w definicji modelu elementu (zobacz 3.9):

```
1 .OUTPUT NAME
2         TYPE = base_output_template;
3         ACROSS PLUS varid;
4         [ACROSS MINUS varid;]
5         [THROUGH varid;
6         [THROUGH varid;]]
7 .END
```

gdzie:

TYPE określa typ bazowego szablonu wyjśćia,

[1]. Uwaga! Odpowiedzialność za użycie odpowiednich typów zmiennych sieciowych spada na użytkownika. Na podstawie definicji sterowania ustawiane są typy zmiennych sieciowych w układzie. Niezgodność typów powoduje generację błędu.

[2]Nie wszystkie wersje programu umożliwiają definiowanie własnych typów wyjść.

ACROSS identyfikatory zmiennych sieciowych podłączenia węzłow wyjścia (PLUS i MINUS) gałęzi (zobacz 3.4),

THROUGH identyfikatory zmiennych gałęziowych (przepływy) skojarzonych z gałęzią - (zobacz 3.4),

Definicja wyjścia bazuje na wbudowanych w program definicjach wyjść, które umożliwiają automatyczne układania równań sieci za pomocą szablonów (stemplowanie macierzy) (zobacz rozdział 7) [3].

3.8 .FUNCTION - definicja funkcji

W języku MDL można definiować własne funkcje. Nazwy funkcji muszą być unikalne. Nie dopuszcza się zastąpienia jednej definicji funkcji inną. Parametry przekazywane są do funkcji przez wartość. Strukturę definicji funkcji przedstawiono poniżej:

Wydruk 3.6: Składnia funkcji MDL

```
1 .FUNCTION (TypeId ZmiennaZwracana1[,TypeId ZmiennaZwracana2, ...])
2 NazwaFunkcji ([TypeId Argument1, TypeId Argument1,...])
3 [TypeId var1, ... , TypeId varK;] # zmienne lokalne
4 .BEGIN
5
6 # function body
7
8 .END
```

gdzie:

NazwaFunkcji jest unikalną nazwą funkcji,

[TypeId] jest identyfikatorem typu argumentu lub zmiennej (3.1),

[Argunent1..N] są parametrami funkcji,

var1 ... varN są deklaracjami zmiennych lokalnych, przy czym każda może być innego typu. Zmienne są zawsze zerowane przy wywołaniu funkcji.

W ciele funkcji dostępne są instrukcje języka MDL (zobacz 3.10). Funkcje mogą zwracać wartości typu rzeczywistego za pomocą wbudowanej funkcji *RETURN()*. Na wydruku 3.7 przedstawiono przykładową definicje funkcji.

Wydruk 3.7: Przykładowa definicja funkcji exp1 - aproksymacja funkcji exp

```
1  .FUNCTION (REAL RET) EXP1(REAL X)
2  REAL X1;
3  .BEGIN
4      X1=X;
5      IF(X < -200){
6        X1=-200;
7      }
8      IF(X1 < 40) {
9          RET =EXP(X1);
10     }ELSE{
11         RET =EXP(40)*(X1-39);
12     }
13 .END
```

[3] Uwaga! Odpowiedzialność za użycie odpowiednich typów zmiennych spada na użytkownika. Niezgodność typów zmiennych podłączenia wyjścia powoduje generację błędu.

3.9 .MODEL - definicja modelu

Dyrektywa *.MODEL* umożliwia definiowanie modelu elementu. Definicja modelu (wydruk 3.8) składa się z dwóch części: nagłówka i ciała modelu. W nagłówku definiowane są parametry elementu i modelu, zmienne, sterowania i wyjścia modelu. Element dołączony jest do układu poprzez węzły zewnętrzne definiowane dyrektywą *.EXTERNAL*. Węzły nie będące węzłami podłączenia (zewnętrznymi), staja się węzłami wewnętrznymi elementu. Zmienne gałęziowe wymagane przez niektóre wyjścia muszą zostać zdefiniowane (dyrektywa *.FLOW*). W ciele modelu obliczane są wartości zmiennych wyjściowych oraz ich pochodnych cząstkowych względem sterowań.

Wydruk 3.8: Składnia modelu MDL

```
1  .MODEL Nazwa
2  # opcje programu
3  .OPTI nazwa_opcji=wartość[, ...];
4  # zmienne sieciowe – węzłowe i gałęziowe
5  .EXTERNAL ZmienneZewnętrzne; # węzły zewnętrzne
6  .INTERNAL ZmienneWewnętrzne; # węzły wewnętrzne
7  .GROUP groupid: zmienna[, ...]; # grupowanie węzłów
8  .FLOW przepływy [, ...]; # zmienne gałęziowe
9  # sterowania i wyjścia modelu
10 .INPUT TypeId id(WęzełLubPrzepływ [, ...]): Zmienna;
11 .OUTPUT TypeId id(WęzełPlus [WęzełLubPrzepływ [, ...]]): Zmienna;
12 # parametry elementu i modelu oraz zmienne
13 .PARAM TypeId par1 [, Par2, ...]; # parametry indywidualne
14 .COMMON TypeId par1 [, Par2, ...]; # parametry wspólne
15 .VAR TypeId var1 [, Var2, ...]; # zmienne lokalne modelu (zerowane)
16 .MEM TypeId var1 [, Var2, ...]; # zmienne lokalne modelu (zapamiętywane)
17 .BEGIN
18 # ciało modelu
19 .END
```

Znaczenie poszczególnych dyrektyw w definicji modelu jest następujące:

Nazwa jest unikalną nazwą modelu,

.OPTI ustawianie opcji modelu. Dostępne są następujące opcje:

LIMU ograniczenie zmian wartości sterowań w iteracjach NR:

BOTH ograniczanie przyrostów dodatnich i ujemnych,

LEFT ograniczanie przyrostów ujemnych,

RIGHT ograniczanie przyrostów dodatnich,

OFF brak ograniczenia.

CLLB domyślna wartość dolnego ograniczenia kroku NR,

CLUB domyślna wartość górnego ograniczenia kroku NR,

.EXTERNAL deklaracja zmiennych zewnętrznych(węzłów) do podłączenia elementu,

.INTERNAL deklaracja zmiennych wewnętrznych (węzłów) modelu.

.FLOW deklaracja zmiennych sieciowych nie będących węzłami sieci (np. zmienne ładunkowe, prądy gałęziowe),

.PARAM lista parametrów elementu,

.COMMON lista parametrów modelu (parametry ustawiane są w linii modelu - zobacz 2.3),

.VAR lista zmiennych modelu, które są każdorazowo kasowane [4] przed wywołaniem kodu modelu.

.MEM lista zmiennych modelu, które są zapamiętywane pomiędzy wywołaniami kodu modelu. Pierwsze użycie wymaga inicjalizacji.

TypeId jest identyfikatorem typu danych (rozdział 3.1).

Parametry modelu i elementu posiadają zdefiniowaną flagę (ISSET) ustawienia wartości parametru jak przedstawiono poniżej:

```
1 NazwaParametru.ISSET
```

Flaga ISSET przyjmuje wartość różną od 0 jeśli ustawiono wartość parametru w linii modelu lub elementu. Może zostać użyta w ciele modelu do sprawdzania czy parametr został zadany przez użytkownika.

Na wydruku 10.18 przedstawiono przykład definicji modelu w języku MDL.

3.10 Instrukcje języka MDL

W języku MDL zdefiniowano szereg instrukcji, które mogą zostać wykorzystane w definicji funkcji lub modelu. Należą do nich instrukcje przypisania, warunkowe, pętle. Zdefiniowano także szereg operatorów, stałych oraz funkcji wbudowanych.

3.10.1 Operatory przypisania

Zdefiniowano dwa operatory przypisania zgodnie z formatem przedstawionym poniżej.

```
1 var = expr
2 out = expr
3 out := expr
```

Pierwsza postać (linia 1) to zwykłe przypisanie wartości wyrażenia *expr* zmiennej *var*. Druga postać (linia 2) stosowana jest do przypisywanie wartości zmiennym skojarzonym z wyjściami modelu elementu, a postać trzecia (linia 3) wyłącznie do przypisywania wartości zmiennym wyjściowym.

3.10.2 Instrukcje warunkowe

Dostępne są dwie postacie instrukcji warunkowej przedstawione poniżej:

```
1 if(wyrażenie_logiczne){
2    blok_instrukcji_1
3 }else{
4    blok_instrukcji_2
5 }
6
7 if(wyrażenie_logiczne){
8    blok_instrukcji_1
9 }
```

Działanie instrukcji warunkowej jest takie samo jak w innych językach programowania. Jeżeli wyrażenie logiczne jest prawdziwe (wartość różna od 0), to wykonywany jest pierwszy blok instrukcji (*blok_instrukcji_1*). W przeciwnym przypadku wykonywany jest drugi blok (*blok_instrukcji_2*). Druga postać instrukcji posiada tylko pierwszy blok instrukcji.

[4] Oznacza to, że zmienna wymaga przypisania wartości każdorazowo przed użyciem. Niezainicjowane jawnie zmienne są źródłem błędów obliczeniowych. Użycia niezainicjowanych zmiennych jest wykrywane przez program.

3.10.3 Pętle

W programie dostępna jest jedna pętla *while*, której składnię przedstawiono poniżej.

```
while(wyrażenie_logiczne) {
  blok_instrukcji
}
```

Blok instrukcji wykonywany jest dopóki wyrażenie logiczne jest prawdziwe (przyjmuje wartość różną od 0).

3.10.4 Operatory

MDL oferuje szereg operatorów, podzielonych na dwie grupy:

- operatory porównania (wydruk 3.10.4), które są używane do porównywania wartości wyrażeń i zmiennych:

```
expr < expr # mniejszy
expr <= expr # mniejszy równy
expr > expr # większy
expr >= expr # większy równy
expr != expr # różny
expr == expr # równy
```

- operatory logiczne (wydruk 3.10.4) wykorzystywane do budowania wyrażeń logiczne. Wynikiem wykonania jest wartość logiczna $0 - fasz$ lub $1 - prawda$.

```
log_expr == log_expr # równy
log_expr != log_expr # różny
log_expr && log_expr # and (i)
log_expr || log_expr # or (lub)
```

W tabeli 3.1 zestawiono wszystkie dostępne operatory. Podano także priorytet i sposób wiązania operatorów.

Tab. 3.1: Operatory języka MDL

Operator	wiązanie	Argumenty	Znaczenie	Priorytet
^	prawe	2	potęgowanie	8
−	lewe	1	zmiana znaku	7
!	lewe	1	negacja	7
*	lewe	2	mnożenie	6
/	lewe	2	dzielenie	6
+	lewe	2	dodawanie	5
−	lewe	2	odejmowanie	5
operatory relacji				
==	lewe	2	równe	4
!=	lewe	2	nierówne	4
<=	lewe	2	mniejsze lub równe	4
>=	lewe	2	większe lub równe	4
<	lewe	2	mniejsze	4
>	lewe	2	większe	4
kontynuacja na następnej stronie				

Tab. 3.1 - kontynuacja z poprzedniej strony				
Operator	wiązanie	Argumenty	Znaczenie	Priorytet
operatory logiczne				
&&	lewe	2	iloczyn logiczny	3
||	lewe	2	suma logiczna	2
operatory przypisania				
=	lewe	2	przypisanie	1
:=	lewe	2	przypisanie	1

3.10.5 Funkcje wbudowane i stałe

Język MDL posiada szereg wbudowanych funkcji i stałych, dostępnych z poziomu modelu lub funkcji. Dodatkowo w programie zaszyto najczęściej używane stałe[5], dostępne z poziomu modelu lub funkcji. Nazwy stałych i funkcji odpowiadają przyjętej ogólnie konwencji.

Tab. 3.2: Predefiniowane funkcje wbudowane

Nazwa	Znaczenie
funkcje matematyczne	
ABS (x)	wartość bezwzględna
ACOS (x)	arcus cosinus
ASIN (x)	arcus sininus
ATAN (x)	arcus tangens
CEIL (x)	najmniejsza liczba całkowita $\geq x$
COS (x)	cosinus
COSH (x)	cosinus hiperboliczny
EXP (x)	funkcja wykładnicza
FLOOR (x)	największa liczba całkowita $\leq x$
INT (x)	część całkowita x (odrzucenie części dziesiętnej)
LOG (x)	logarytm naturalny
LOG10 (x)	logarytm dziesiętny
MAX(x,y)	większa wartość x lub y
MIN(x,y)	wartość mniejsza x lub y
SIGN (x)	znak x
SIN (x)	sinus
SINH (x)	sinus hiperboliczny
SQRT (x)	pierwiastek
TAN (x)	tangens
TANH (x)	tangens hiperboliczny
funkcje sterujące procesem obliczeń	
RETURN(x)	zakończenie wykonywania kodu mdl funkcji/modelu (wykonanie programu jest kontynuowane) - wartość zwracana x: $x = 0$ - zakończenie prawidłowe, $x <> 0$ - kod błędu x.
	kontynuacja na następnej stronie

[5]Nie wszystkie wersje programu. Stałe można zdefiniować dyrektywą .CONST - rozdział 3.2

Tab. 3.2 - kontynuacja z poprzedniej strony	
Nazwa	Znaczenie
EXIT (x)	natychmiastowe zakończenie obliczeń z kodem błędu x i przerwanie wykonania programu
ASSERT(x)	sprawdzanie niezmiennika wykorzystywane jest do kontroli poprawności wykonania kodu funkcji/modelu mdl. Wartość $x = 0$ oznacza wykonanie poprawne. Wartość $x <> 0$ powoduje wygenerowanie błędu i przerwanie wykonania programu
funkcje wejścia/wyjścia	
PRINT(string||expr,...	wydruk ciągu znaków *string*, wyrażenia *expr*

Tab. 3.3: Stałe wbudowane w program *Dero*

Nazwa	Znaczenie
C2K	273.17 addytywny przelicznik stopni C na K $[K]$
EPS0	8.854215 bezwzględna przenikalność elektryczna próżni $[F/m]$
EU	2.718281 podstawa logarytmu naturalnego
EUCONST	2.718281 stała Eulera
KBOLTZ	1.380623E-23 stała Boltzmana $[J/K]$
MI0	1.25663E-6 stała magnetyczna $[H/m]$
NISI0	1.450000E10 koncentracja nośników samoistnych w krzemie w temperaturze 273.15K $[1/m^3]$
PI	3.141592653 liczba π
QELE	1.602189E-19 ładunek elementarny $[C]$
RAD2DEG	57.295779 multiplikatywny przelicznik radianów na stopnie $[o/rad]$

3.10.6 Zmienne i funkcje łączności z jądrem symulatora

Z poziomu modelu możliwy dostęp do zmiennych ustawianych przez jądro programu oraz do szeregu funkcji sterujących procesem analizy. Wartości zmiennych przekazywanych przez jądro nie mogą być modyfikowane. Do pobrania wartości zmiennej analizatora wykorzystywana jest funkcja wbudowana *KERNEL()*. Wartości ustawiane są przez symulator w trakcie analizy. Listę dostępnych zmiennych symulatora przedstawiono w tab. 3.10.6.

Tab. 3.4: Predefiniowane zmienne symulatora dostępne z poziomu modelu.

Nazwa zmiennej	**Opis**
OP_ANALYSIS	flaga analizy OP (punktu pracy)
AC_ANALYSIS	flaga analizy AC (częstotliwościowej małosygnałowej)
TR_ANALYSIS	flaga analizy TR (czasowej)
EDITA_ANALYSIS	flaga analizy ED ITA
TIME	obecny punkt czasowy analizy
TIME_STEP	krok czasowy h w obecnym punkcie t
	kontynuacja na następnej stronie

Tab. 3.10.6 - kontynuacja z poprzedniej strony	
Nazwa zmiennej	Opis
TIME_BRKP	następna pułapka czasowa
SR_ORDER	rząd schematu różnicowego
FREQUENCY	częstotliwość (analiza AC)
GS	numer iteracji Gaussa-Seidela (ED-ITA)
NR	numer iteracji Newtona-Raphsona

Z poziomu modelu możliwa jest komunikacja z symulatorem poprzez zestaw udostępnianych funkcji[6]. Nazwy i znaczenie funkcji przedstawiono w tab. 3.5.

Tab. 3.5: Funkcje do komunikacji z kernelem z poziomu modelu

Nazwa	**Opis**
Sterowanie krokiem klasycznej analizy TR	
BOUND_TIME_STEP(step)	ograniczenie kroku symulacji do wartości *step*
TIME_BRKP(t)	generacja pułapki czasowej w punkcie t
Funkcje do współpracy z analizatorem czasowym kierowanym zdarzeniami EDITA	
SHEDULE_EVENT(analog_node, t)	generacja zdarzenia - skierowanie węzła *analog_node* do analizy w punkcie czasowym t. Wartość zwracana: 0 - wykonanie powiodło się lub kod błędu w przeciwnym przypadku.
SHEDULE_EVENT(digital_node, t, val)	wstawienie węzła *cyfrowego dig_node* na listę zdarzeń analizatora kierowanego zdarzeniami (EDITA) w punkcie czasowym t i ustawienie wartości sygnału w punkcie t na *val*. Wartość zwracana: 0 - wykonanie powiodło się lub kod błędu w przeciwnym przypadku.
EVENT_ON(digital_node)	sprawdzenie czy zdarzenie dotyczy węzła cyfrowego *digital_node*. Wartość zwracana $! = 0$ jeśli tak lub 0 w przeciwnym przypadku.

[6]Nie wszystkie wersje programu.

Rozdział 4

Komendy

Komendy mogą być umieszczane w różnych miejscach pliku wejściowego, jednak większość z nich może być umieszczana tylko w bloku komend (rysunek 2.1). Komendy można podzielić na grupy:

- komendy analiz i projektowania deterministycznego,
- komendy sterujące,
- dyrektywy wyprowadzania wyników analiz,
- dyrektywy ustawiania opcji programu i śledzenia wykonania.

4.1 Zlecenia analiz

Program umożliwia wykonanie następujących rodzajów analiz[1]:

- analiza punktu pracy (OP *ang. Operating Point*),
- małosygnałowa analiza częstotliwościowa (AC *ang. Alternate Current*),
- analiza charakterystyk stałoprądowych (DC *ang. Direct Current*),
- analiza czasowa stanów przejściowych (TR *ang. Transient Analysis*),
- analiza czasowa stanu ustalonego (STDS *ang. Steady State Analysis*),
- analiza czasowa kierowana zdarzeniami (EDITA *ang. Event-Driven Iterated Timing Analysis*).
- wyznaczanie aproksymacji odcinkowo-liniowej przestrzeni sprawności 3D dla układów nieliniowych - rozszerzona metoda ODOS (*ang. One-Dimensional Ortogonal Search*).

W jednym zadaniu można wykonać dowolnie wiele analiz dowolnego typu.

[1] Niektóre rodzaje analiz mogą być niedostępne.

4.1.1 .OP - analiza punktu pracy

Dyrektywa umożliwia wykonanie analizy punktu pracy układu. Format linii dyrektywy przedstawiono poniżej:

```
1 .OP [TITLE="ciąg znaków"][;]
```

gdzie:

TITLE jest tytułem analizy identyfikującym sekcję danych w pliku wyjściowym.

W analizie wykorzystywany jest algorytm Newtona-Raphsona (rozdział 6.2). Trudności numeryczne z analizą sieci można przezwyciężyć przez wybór algorytmu analizy punktu pracy opcją NRME (tab. 4.1). Przykład dyrektyw OP przedstawiono na wydruku 4.1.

Wydruk 4.1: Przykład dyrektywy analizy OP

```
1 .OP ;
2 .OP TITLE="analiza stałoprądowa";
```

4.1.2 .DC - analiza charakterystyk stałoprądowych

Zlecenie umożliwia wykonanie zagnieżdżonej analizy stałoprądowej sieci. Dla każdej wartości parametru VAR1 wykonywana jest analiza w pełnym zakresie zmienności VAR2.

```
1 .DC VAR1=par START1=val STOP1=val STEP1=val
2 [VAR2=par START2=val STOP2=val STEP2=val]
3 [TITLE="tytuł"][;]
```

Znaczenie parametrów linii komendy jest następujące ($x = 1, 2$):

VARx identyfikator zmiennej niezależnej,

STARTx wartość początkowa zakresu zmienności VARx,

STOPx wartość końcowa zakresu zmienności VARx,

STEPx przyrostem zmiennej niezależnej w kolejnych punktach analizy,

TITLE tytuł analizy - tytuł sekcji danych w pliku wyjściowym.

Na wydruku 4.2 przedstawiono przykłady użycia komendy analizy DC.

Wydruk 4.2: Przykład dyrektyw analizy DC

```
1 .DC VAR1=V1.V START1=3 STOP1=5 STEP1=0.2 TITLE="analiza DC";
2
3 .DC VAR1=V1.V START1=2 STOP1=5 STEP1=0.2
4     VAR2=V2.V START2=1 STOP2=2 STEP2=0.1
5     TITLE="zagnieżdżona analiza DC";
```

Linia pierwsza zawiera zlecenie analizy charakterystyk w funkcji $V1.V$ - parametru V elementu $V1$. Parametr V1.V zmienia się w zakresie od 3 do 5 z krokiem nie większym niż 0.2. W linii 3,4 i 5 przedstawiono zlecenia obliczenia rodziny charakterystyk w funkcji parametru $V1.V$ dla kolejno zmienianych wartości parametru $V2.V$. Sterowanie przebiegiem i dokładnością analizy umożliwiają opcje analizatora OP - rozdział 4.1.1.

4.1.3 .TR - analiza czasowa stanu nieustalonego

Dyrektywa umożliwia wykonanie analizy czasowej stanu przejściowego (rozdział 6.5). Format linii komendy przedstawiono poniżej:

```
1 .TR [T0=val] [TMIN=val] TMAX=val [TBEG=val] [HINI=val] STEP=val [UIC]
2     [TITLE="tytuł"];
```

gdzie:

T0 przesunięcie punktu początkowego czasu analizy (wartość domyślna 0),

TMIN czas początkowy wyprowadzania wyników analizy (wartość domyślna 0),

TMAX czas końca analizy,

TBEG czas początkowy, od którego analizator rozpoczyna analizę (wartość domyślna 0),

HINI początkowy krok analizy,

STEP krok wyprowadzania wyników analizy w zakresie od TMIN do TMAX. Wartość argumentu STEP jest ograniczona: $STEP < TMAX$,

UIC wymuszenie załadowania warunków początkowych zadanych przez użytkownika,

TITLE tytuł analizy identyfikujący sekcję danych w pliku wyjściowym. Tytuł analizy po znaku = musi zostać umieszczony w cudzysłowie.

TFMT format wyprowadzania argumentu analizy (czasu): REAL (0 - wartość domyślna) lub DATE (1) - data zgodna z normą ISO 8601: $YYYY-MM-DDhh:mm:ss$.

W zleceniu muszą zostać podane przynajmniej argumenty: STEP oraz TMAX. Przykład dyrektyw analizy TR przedstawiono na wydruku 4.3.

Wydruk 4.3: Przykład dyrektyw analizy TR

```
1 .TR STEP=0.01N TMAX=1N;
2 .TR STEP=0.01N TMAX=1N TMIN=0.5N HINI=0.02N UIC TITLE="Analiza TR";
```

4.1.4 .EDITA - analiza czasowa kierowana zdarzeniami

Analiza czasowa stanu nieustalonego kierowana zdarzeniami (*ang. Event-Driven Iterated Timing Analysis*)[2] przeznaczona jest do analizy układów cyfrowych i/lub analogowych o ustalonym kierunku przepływu sygnału. Zaletą analizy jest możliwość znacznego skrócenia czasu analizy w szczególności układów cyfrowych. Format linii komendy przedstawiono poniżej:

```
1 .EDITA [GROUP=OFF|USR|DYN] TMAX=val DT=val [UIC] [TITLE="tytuł"]
```

gdzie:

GROUP flaga sterująca procesem dynamicznego grupowania węzłów sieci: OFF - grupowanie wyłączone, USR - grupowanie statyczne wykonane przez użytkownika, DYN - grupowanie dynamiczne wykonywane przez program,

[2] Analiza dostępna w niektórych wersjach programu.

TMAX czas końca analizy,

DT minimalny odstęp czasowy pomiędzy zdarzeniami,

UIC - wymuszenie załadowania warunków początkowych zadanych przez użytkownika,

TITLE tytuł analizy identyfikujący sekcję danych w pliku wyjściowym.

W zleceniu musi zostać podany przynajmniej jeden argument: TMAX. Wartość argumentu DT musi spełniać zależność $DT < TMAX$.

Analiza kierowana zdarzeniami może wymagać dodatkowych danych przyśpieszających działanie analizatora (dyrektywa '.GROUP', rozdział 4.2.3). Wyniki analizy wyprowadzane są na bieżąco we wszystkich punktach analizy i tylko dla tych zmiennych, które były analizowane. Przykład dyrektywy analizy EDITA przedstawiono na wydruku 4.4.

Wydruk 4.4: Przykład dyrektyw analizy TR

```
1 .EDITA TMAX=1N DT=0.01N UIC TITLE="Analiza EDITA";
```

4.1.5 .STDS - analiza czasowa stanu ustalonego

dyrektywa umożliwia wykonanie analizy czasowej okresowego stanu ustalonego (rozdział 6.6). format linii komendy przedstawiono poniżej:

```
1 .STDS [TMIN=val] TMAX=val [TBEG=val] [HINI=val] STEP=val
2       [UIC] [AUT] PER=val [TITLE="tytuł"];
```

gdzie argumenty zlecenia oznaczają odpowiednio:

TMIN czas początkowy wyprowadzania wyników analizy (wartość domyślna 0),

TMAX czas końca analizy,

TBEG czas początkowy, od którego analizator rozpoczyna analizę (wartość domyślna 0),

HINI początkowy krok analizy,

STEP krok wyprowadzania wyników analizy w zakresie tmin..tmax,

UIC wymuszenie załadowania warunków początkowych zadanych przez użytkownika,

AUT oznaczenie, że analizowany układ jest układem autonomicznym,

PER okres wymuszenia dla układów nieautonomicznych i przybliżona wartość okresu dla układów autonomicznych,

TITLE tytuł analizy identyfikujący sekcję danych w pliku wyjściowym.

dyrektywa wymaga podania przynajmniej argumentów: step, tmax i per. Wartość argumentu step jest ograniczona: $step < tmax$. Przykład dyrektyw analizy stds przedstawiono na wydruku 4.5.

Wydruk 4.5: przykład dyrektyw analizy stds

```
1 .STDS STEP=0.01n TMAX=1n PER=2n;
2 .STDS STEP=0.01n TMAX=1n TMIN=0.5n HINI=0.02n UIC PER=3n TITLE="stds";
```

4.1.6 .AC - małosygnałowa analiza częstotliwościowa

Dyrektywa umożliwia wykonanie małosygnałowej analizy częstotliwościowej przy wymuszeniach sinusoidalnych (rozdział 6.8). Format linii komendy przedstawiono poniżej:

```
.AC FMIN=val FMAX=val FPTS=val SCALE=LIN|DEC [TITLE="tytuł"];
```

gdzie:

FMIN częstotliwość początkowa analizy,

FMAX częstotliwość końcowa analizy,

FPTS liczba punktów analizy,

SCALE rodzaj skali: liniowa (LIN) lub logarytmiczna (DEC),

TITLE opcjonalny tytuł analizy identyfikujący sekcję danych w pliku wyjściowym. Tytuł analizy po znaku '=' musi zostać umieszczony w cudzysłowie.

Przykład dyrektyw analizy AC przedstawiono na wydruku 4.6.

Wydruk 4.6: Przykład dyrektyw analizy AC

```
.AC SCALE=LIN FPTS=10 FMIN=10 FMAX=1000 TITLE="Analiza AC";
.AC SCALE=DEC FPTS=10 FMIN=10 FMAX=1000 TITLE="Analiza AC";
```

4.1.7 .ODOS - wyznaczanie obszarów sprawności

Dyrektywa umożliwia obliczenie rodzin odcinków sprawności, przy pomocy których aproksymuje się przestrzeń sprawności (3D). Szczegółowy opis metody znajduje się w rozdziale 6.10. Format linii komendy przedstawiono poniżej:

```
.ODOS VP_I:OUT_I VP_J:OUT_J VP_K:OUT_K;
```

gdzie:

VP_I jest i-tym parametrem uzmiennionym dyrektywą .VAR - zobacz 4.4.2,

OUT_I jest nazwą i-tej zmiennej skojarzonej z wyjściem elementu.

Przykład dyrektywy ODOS przedstawiono na wydruku 4.7.

Wydruk 4.7: Przykład dyrektyw ODOS

```
...
# Elementy ukladu
R.R1 1 2 R=1kOhm;
R.R2 2 3 R=2kOhm;
R.R3 3 0 R=3kOhm;
....
# Dyrektywa ODOS
.ODOS R1.R:O R2.R:O R3.R:O;
```

4.2 Dyrektywy warunków analiz

4.2.1 .TBRP - generacja pułapek czasowych

Dyrektywa umożliwia generację punktów pułapek czasowych, w których krok analizatora czasowego jest ustawiany na krok minimalny (patrz opcja HMIN, tab. 4.1). Pułapki czasowe mogą być generowane okresowo, jeśli podano argument 'PER'. Format linii dyrektywy przedstawiono poniżej:

```
1 .TBRP TIME=val;
2 .TBRP TIME=val PER=val;
3 .TBRP CMD;
```

gdzie CMD jest jedną z komend:

CLEAR usunięcie wszystkich zdefiniowanych pułapek czasowych,

PRINT wydruk zdefiniowanych pułapek czasowych.

Na wydruku 4.8 zamieszczono przykład użycia dyrektyw '.TBRP'.

Wydruk 4.8: Przykład użycia komendy '.TBRP'

```
1 .TBRP TIME=1e-3;
2 .TBRP TIME=1e-3 PER=1E-4;
3 .TBRP PRINT;
4 .TBRP CLEAR;
```

W linii 1 ustawiono pojedynczą pułapkę czasową dla $t = 1e-3S$. W linii 2 ustawiono pułapkę czasową w punkcie $t = 1e - 3S$ i dalej okresowo co $1e - 4S$, aż do końca zakresu analizy. W linii 3 wydrukowano liste zdefiniowanych pułapek czasowych. W linii 4 usunięto wcześniej zdefiniowane pułapki czasowe.

4.2.2 .IC - zadanie warunków początkowych

W programie istnieje możliwość zadania warunków początkowych (*ang. initial condition*) zmiennych sieciowych. Warunki początkowe wykorzystywane są np. w analizie czasowej tran.

```
1 .IC nazwazmiennej = val [, nazwazmiennej = val, ldots ];
2 .IC cmd;
```

gdzie:

nazwazmiennej jest nazwą zmiennej sieciowej (nazwą węzła sieci lub nazwa zmiennej gałęziowej),

val jest wartością początkową zmiennej,

cmd jest jedną z komend:

CLEAR - zerowanie warunków początkowych,

SAVE nazwa_pliku - zapisanie warunków początkowych w pliku tekstowym o nazwie *nazwa_pliku* w postaci ciągu komend:
.IC nazwazmiennej = val ;
.IC nazwazmiennej = val ;

Zapisane wartości można powtórnie włączyć do opisu układu dyrektywą '.inc' (zobacz rozdział 2.6). Warunki początkowe mogą zostać użyte jako warunki startowe dla:

- analizy czasowej (tran) jeśli w zleceniu analizy podano argument uic,
- analizy op/dc dla iteracji w algorytmie nr, jeśli ustawiono opcję nrme=2.

Przykład użycia dyrektywy przedstawiono na wydruku 4.9.

Wydruk 4.9: przykład użycia komendy .ic

```
1 .IC l1=1e-3;
2 .IC SAVE FILE="init.cnd";
3 .IC CLEAR;
```

W pierwszej linii ustawiona zostaje wartość początkowa (1e-3) zmiennej o nazwie l1. W drugiej linii ustawione warunki początkowe zmiennych zostają zapisane w pliku o nazwie 'init.cnd' w katalogu roboczym. W linii trzeciej następuje wyzerowanie wcześniej ustawionych warunków początkowych.

Wyniki działania komendy '.ic' kumulują się. Komenda nie zmienia wcześniej ustawionych wartości zmiennych jeśli nie wystąpiły one w linii komendy. Po znaku kończącym opis układu wszystkie wartości początkowe zmiennych sieciowych są wyzerowane.

4.2.3 .GROUP - grupowanie węzłów sieci

Komenda ma zastosowanie w kierowanej zdarzeniami analizie czasowej EDITA (rozdział 4.1.4). Umożliwia tworzenie statycznych bloków węzłów silnie, dwukierunkowo sprzężonych. Bloki muszą być analizowane techniką klasyczną (rozwiązywane są układy równań z wykorzystaniem techniki macierzy rzadkich). Przyśpieszenie działania analizatora uzyskuje się przez redukcję nakładów obliczeniowych na znalezienie takich węzłów i zgrupowanie ich. Pomiędzy blokami stosowany jest algorytm kierowany zdarzeniami (rozdział 6.7).

```
1 .GROUP group_id węzeł_1 [,] ... [,] węzeł_n;
2 .GROUP NEW node1 ... noden;
3 .GROUP;
```

4.2.4 .ALT - zmiana wartości parametru układu

Dyrektywa umożliwia zmianę parametru układu i wykonanie analiz parametrycznych.

```
1 .ALT par=val;
```

gdzie:

par jest nazwą parametru układu (zobacz rozdział 2.1 - parametry i opcje),

val jest wartością przypisywaną parametrowi[3].

Przykład użycia dyrektywy przedstawiono na wydruku 4.10.

Wydruk 4.10: przykład dyrektywy zmiany parametru układu

```
1 .ALT r1.r=10;
2 .ALT x1:vcc.dc=10;
3 .ALT x1:src..v1=-1;
```

W linii pierwszej przypisano wartość 10 parametrowi r elementu $r1$. W linii drugiej ustawiono wartość parametru dc elementu o nazwie vcc włączonego do obwodu głównego poprzez linię podobwodu o nazwie $x1$. W linii trzeciej ustawiono wartość parametru wspólnego $v1$ w linii modelu src w obwodzie $x1$.

[3] uwaga! w trakcie ustawiania nowej wartości nie jest sprawdzana poprawność wartości parametru. oznacza to, że można ustawić np. ujemną wartość pojemności.

4.3 Wyprowadzanie wyników analiz

4.3.1 .PRINT - wyprowadzanie wyników w postaci tekstowej

Dyrektywa '.PRINT' wyprowadza do pliku logu wyniki analiz w postaci tekstowej. Format komendy przedstawiono poniżej:

```
1 .PRINT cmd id( szablon_nazwy_zmiennej ) [ ... ];
```

gdzie:

cmd jest identyfikatorem komendy:

ADD - dodaje zmienne do listy wydruku,

DEL - usuwa zmienne z listy wydruku,

id jest identyfikatorem analizy, dla której będą wyprowadzane wartości zmiennych lub parametrów:

OP w iteracjach NR w analizie OP,

AC dla punktów f w małosygnałowej analizie częstotliwościowej,

DC w analizie charakterystyk,

TR w punktach czasowych analizy czasowej TR,

PAR wydruk wartości zmiennych wewnętrznych analizatorów:

LTE lokalny błąd obcięcia analizy czasowej (TR),

ORDE rząd schematu różnicowego w analizie czasowej (TR),

NR liczba iteracji Newtona-Raphsona w każdym punkcie czasowym analizy TR/STDS/EDITA,

STEP krok czasowy analizatora TR,

REL całkowita liczba iteracji Gaussa-Seidela w danym punkcie czasowym w analizie czasowej kierowanej zdarzeniami EDITA,

szablon_nazwy_zmiennej jest ciągiem znaków, który określa zmienną sieciową. Szablon musi zostać umieszczony w podwójnym cudzysłowie ", jeśli zawiera białe znaki:

***** zastępuje jakikolwiek ciąg znaków złożony z liter lub cyfr,

? zastępuje pojedynczy znak.

Na wydruku 4.11 przedstawiono przykład użycia dyrektywy '.PRINT'.

Wydruk 4.11: Przykład użycia komendy .PRINT

```
1 .PRINT ADD TR("*") OP("W*3?4") AC("V?3") PAR(LTE);
2 .PRINT DEL TR("*") OP("W*3?4") AC("V?3") PAR(LTE);
```

W nazwach użyto szablonów nazw. W linii 1 drukowane są wartości wszystkich zmiennych (TR("*")) w punktach czasowych dla analizy czasowej stanu nieustalonego. Wyprowadzane są także wyniki w iteracjach NR w analizie OP dla wszystkich zmiennych (OP("W*3?4")) o nazwach rozpoczynających się literą 'W', po której może wystąpić jakikolwiek ciąg znaków zakończony znakami: '3', jakimkolwiek znakiem i '4'. Wyprowadzane są także wartości parametru LTE analizatora w analizie czasowej. W linii 3 zostają usunięte z listy wydruku wszystkie zmienne zdefiniowane w linii 2.

4.3.2 .PROBE - formatowane wyprowadzanie wyników analiz

Dyrektywa zapisuje wyniki analizy w pliku wyjściowym (*.dat) w formacie ustawionym przy pomocy opcji 'DAT'(tab. 4.1). Format komendy przedstawiono poniżej. Znaczenie argumentów jest takie same jak dla dyrektywy '.PRINT' (rozdział 4.3.1).

```
.PROBE cmd id( szablon_nazwy_zmiennej ) [ ... ];
```

Na wydruku 4.12 przedstawiono przykład użycia dyrektywy '.PROBE'. W nazwach zmiennych użyto szablonów analogicznie do przykładu z rozdziału 4.3.1.

Wydruk 4.12: Przykład użycia komendy '.PROBE'

```
.PROBE ADD TR("*") OP("W*3?4") AC("V?3") PAR(LTE);
.PROBE DEL TR("*") OP("W*3?4") AC("V?3") PAR(LTE);
```

4.3.3 Formaty plików danych

Dostępne są następujące formaty[4]:

- CAZM - format tekstowy kompatybilny z CAZM [D+90],
- SPICE3 - format tekstowy kompatybilny z symulatorem SPICE [L.W75, A+81].

CAZM

Dane zapisane są w postaci tekstowej. W pliku można umieścić wiele bloków danych. Format pojedynczego bloku przedstawiono na wydruku 4.13. Bliższe informacje można znaleźć w [D+90].

Wydruk 4.13: Format pojedynczego bloku danych w formacie CAZM

```
TITLE DATE
Title:
Flags:
No. Variables: value
No. Points: value
Command
Variables:
List of printed variables separated by newline of format
index name
.
.
.
Analysis type
Space separated printed variables names corresponding to column \\
Space separated values of printed variables

value11 value12 ... value1N
. . ... .
. . ... .
. . ... .
valueM1 valueM2 ... valueMN
```

SPICE3

Format SPICE3 jest formatem RAW (zobacz dokumentację programu [Vla94]). Plik jest akceptowany np. przez posprocessor graficzny *nutmeg* dystrybuowany z programem SPICE3 lub gwave [Tel11]. Format pliku przedstawiono na wydruku 4.14.

[4] W zależności od wersji programu mogą być dostępne także inne formaty zapisu danych wyjściowych.

Wydruk 4.14: Format pliku danych SPICE3

```
Title: Title from the title line of a circuit
Date: Sun May 2 18:17:03 1999
Plotname: TR analysis
Flags: real
No. Variables: 4
No. Points: 19
Command:
Variables:
Number name type
0 time time
1 TR1.x unknown
2 V(2) voltage
3 V(1) voltage
Values:
In format:
Point number value of independent variable
       signal values - each in separated line
1 2.000000000000000e-11
       0.000000000000000000000000e+00
       1.648459140369595371703 4e-19
       1.000000000000000000000000e+01
2
      9.999999999990000e-03
      0.000000000000000000000000e+00
      2.155220906966650292757 1e-20
      1.000000000000000000000000e+01
3 ...
      ...
      ...
      ...
.
.

NASTĘPNY BLOK DANYCH
```

4.4 Automatyczne projektowanie

W programie dostępne są moduły umożliwiające automatyczne projektowanie. Dla potrzeb automatycznego projektowania można definiować specyfikacji projektowe i warunki ich działania. Specyfikacje można włączać i wyłączać oraz usuwać. Ponadto można uzmienniać i ustawiać parametry projektowe. Proces automatycznego projektowania obejmuje następujące etapy:

1. Zadanie wymagań na odpowiedzi układu - specyfikacje projektowe. Mogą one dotyczyć np.: punktu pracy, charakterystyk statycznych, odpowiedzi czasowej, charakterystyk częstotliwościowych.

2. Uzmiennienie parametrów układu celem dopasowania odpowiedzi układu do zadanych specyfikacji projektowych.

3. Wykonanie optymalizacji dającej w efekcie poprawę tzw. funkcji celu (rozdział 6.9). Optymalizacja może wymagać wielokrotnego wykonania analiz tego samego układu dla różnych zestawów uzmiennionych parametrów. Parametry mogą być skalowane. Specyfikacjom projektowym można przypisywać wagi.

4.4.1 .SP - specyfikacje projektowe

Specyfikacje projektowe to ograniczenia nałożone np. na charakterystyki układu. Program umożliwia definiowanie dwóch rodzajów specyfikacji projektowych: specyfikacji

bezwzględnej lub względnej. Dyrektywa '.SP' umożliwia wykonywanie różnych operacji na specyfikacjach: definiowanie/usuwanie, włączanie/wyłączanie, drukowanie. Format dyrektywy przedstawiono poniżej:

```
1 .SP cmd cmd_arg [typ [zmiena=val] [S1=val [W1=val [S2=val [W2=val]]]]]
```

gdzie:

cmd jest identyfikatorem operacji wykonywanej na specyfikacjach projektowych:

ADD - dodanie nowej specyfikacji (*).

DEL - usunięcie specyfikacji (skasowanie specyfikacji) (**).

ACT - włączenie nieaktywnej specyfikacji (**).

DIS - wyłączenie aktywnej specyfikacji (**).

PRI - wydruk specyfikacji.

(*) Wymagane jest podanie nazwy specyfikacji oraz wartości argumentu, którego dotyczy specyfikacja oraz przynajmniej jednego ograniczenia s1

(**) Wymagane jest podanie nazwy specyfikacji i wartości argumentu, którego dotyczy specyfikacja.

cmd_arg argument komendy *cmd* określa rodzaj specyfikacji:

REL - specyfikacja względna,

ABS - specyfikacja bezwzględna.

lub w przypadku komendy *PRI* steruje wydrukiem specyfikacji w pliku logu.

ACT wydruk aktywnych specyfikacji,

DIS wydruk nieaktywnych specyfikacji,

typ określa dziedzinę, w której określono wartość specyfikacji dla sygnału lub zmiennej niezależnej:

OP - ograniczenie nałożone na punkt pracy,

TR - ograniczenie nałożone na odpowiedź czasową,

MAGN - ograniczenie nałożone na amplitudę odpowiedzi częstotliwościowej (analiza AC),

PHAS - ograniczenie nałożone na fazę odpowiedzi częstotliwościowej (analiza AC),

DC - ograniczenie nałożone na charakterystykę stałoprądową (analiza DC).

zmienna jest identyfikatorem zmiennej, na którą nałożono ograniczenie. Nazwa może przyjmować jedną z postaci:

TIME - zmienna niezależna w analizie czasowej,

FREQ - zmienna niezależna w analizie częstotliwościowej (analiza AC),

parametr - ograniczenie nałożone na parametr[5] w analizie charakterystyk stałoprądowych (analiza DC).

[5]Nazwa parametru może być nazwą argumentu VAR1 lub VAR2 podanych w linii komendy analizy DC - rozdział 4.1.2

S1, S2 określają wartości ograniczeń (specyfikacji) projektowych,

W1, W2 określają wagi specyfikacji odpowiednio dla S1 i S2.

Zdefiniowane specyfikacje od razu stają się aktywne. Specyfikacje wykorzystywane są do określenia funkcji celu (rozdział 6.9). Jeśli podano specyfikację S2 to rzeczywista wartość specyfikacji obliczana jest ze zależności:

$$s = \frac{s_1\ w_1 + s_2\ w_2}{w_1 + w_2}$$

Rzeczywista waga danej specyfikacji w funkcji celu dana jest zależnością:

$$w = \frac{w_1 + w_2}{2}$$

Pojedyncza dyrektywa specyfikacji określa pojedynczą składową funkcji celu postaci:

- dla specyfikacji bezwzględnej (ABS)

$$w\ (f(x) - s)$$

- dla specyfikacji względnej (REL)

$$w\ \frac{f(x) - s}{s}$$

$f(x)$ oznacza rzeczywistą (wyliczoną) wartość 'sygnału', na który nałożono ograniczenia.

Wydruk 4.15: Przykład definiowania specyfikacji projektowych

```
1 .SP ADD REL TR TIME=1NS I1 S1=1.3 W1=0.9 S2=3 W2=0.1;
2 .SP ADD ABS OP OP V1 S1=1.3 W1=0.2 S2=3 W2=0.2;
3 .SP ADD REL DC V1.V=1V I2 S1=1.3 W1=0.3 S2=3 W2=0.3;
4 .SP ADD ABS MAGN FREQ=1kHz 22 S1=1.3 W1=0.4 S2=3 W2=0.4;
5 .SP ADD REL PHAS FREQ=1kHz W4 S1=1.3 W1=0.5 S2=3 W2=0.7;
```

Na wydruku 4.15 przedstawiono przykłady użycia dyrektywy '.SP'. W linii 1 zdefiniowano specyfikację bezwzględną w analizie czasowej dla $t = 1nS$ dla zmiennej I1. Zdefiniowano specyfikacje dolną S1 i górną S2, obie z tymi samymi wagami W1, W1. W linii 2 zdefiniowano specyfikację względną w analizie punktu pracy dla zmiennej V1. Zdefiniowano specyfikacje dolną S1, górną S2 oraz wagi W1, W1. W linii 3 zdefiniowano specyfikację bezwzględną w analizie charakterystyk (DC). Dla wartości zmiennej $V1.V = 1V$ nałożono ograniczenia na wartość sygnału I2. Zdefiniowano specyfikacje dolną S1, górną S2 oraz wagi W1, W1. W linii 4 zdefiniowano specyfikację względną dla modułu sygnału o nazwie 22 dla częstotliwości $f = 1kHz$. Zdefiniowano specyfikacje dolną S1, górną S2 oraz wagi W1, W1.

4.4.2 .VAR - zmienne optymalizacji

Do obsługi zmiennych optymalizacji służy dyrektywa '.VAR' (rozdział 6.9). Format linii dyrektywy przedstawiono poniżej:

```
1 .VAR cmd par [SCALE=val] [MIN=val] [MAX=val];
```

gdzie:

cmd - jest identyfikatorem operacji wykonywanej na zmiennych:

ADD - dodanie nowej zmiennej optymalizacji.Wymagane jest podania przynajmniej wartości parametru SCALE oraz opcjonalnie wartości parametrów MIN i MAX.

DEL - usuwanie zmiennych optymalizacji. Wszystkie podane w argumencie zmienne optymalizacji stają się nieaktywne i zostają usunięte.

ACT - aktywacja zdefiniowanych wcześniej zmiennych optymalizacji,

DIS - wyłączenie zdefiniowanych wcześniej zmiennych optymalizacji,

UPD - zmiana parametrów uzmiennienia niektórych parametrów optymalizacji,

FIX - ustalenie wartości zdefiniowanego wcześniej parametru,

VAR - uzmiennienie wartości zdefiniowanego wcześniej parametru,

PRI - wydruk zdefiniowanych zmiennych optymalizacji.

PAR - jest zdefiniowanym parametrem układu[6] lub dla komendy PRI jednym z przełączników:

VARP - wydruk parametrów układu, którym nadano status potencjalnej zmienności,

FIXP - wydruk parametrów układu, które nie mają statusu potencjalnej zmienności,

VAR - wydruk parametrów układu, które są aktywnymi zmiennymi optymalizacji,

FIX - wydruk parametrów układu, które nie są aktywnymi zmiennymi optymalizacji.

SCALE - jest współczynnikiem skalowania,

MIN - jest dolnym ograniczeniem zmian wartości argumentu,

MAX - jest górnym ograniczeniem zmian wartości argumentu.

Na wydruku 4.16 przedstawiono przykładowe użycie dyrektywy.

Wydruk 4.16: Przykład dyrektyw warunków działania specyfikacji

```
1 .VAR ADD V1.PAR1 SCALE=1 MIN=1 MAX=20;
2 .VAR DIS V1.PAR1;
3 .VAR ACT V1.PAR1;
4 .VAR VAR V1.PAR1;
5 .VAR FIX V1.PAR1;
6 .VAR UPD V1.PAR1 SCALE=2 MIN=2 MAX=10;
7 .VAR PRI VAR;
8 .VAR PRI ACT;
9 .VAR DEL V1.PAR1;
```

W linii 1 dodano zmienną optymalizacji o nazwie “V1.PAR”, ustawiono współczynnik skalowania “SCALE”, wartości minimalną i maksymalną zakresu zmienności. W linii 2 wyłączono zmienną optymalizacji “V1.PAR1”. W linii 3 aktywowano zmienną. W linii 4 uzmienniono, a w linii 5 ustalono wartość zdefiniowanego parametru. W linii 6 zmieniono parametry uzmiennienia. W linii 7 i 8 wydrukowano zmienne optymalizacji. W linii 9 usunięto zdefiniowaną zmienną optymalizacji.

[6]Uwaga! Jeśli parametr jest parametrem wspólnym, to zmiana jest widoczna dla wszystkich elementów powołujących się na daną linię modelu. Parametry wspólne związane są z linią modelu (rozdział 2.3)

4.4.3 .CENT - centrowanie deterministyczne

Dyrektywa powoduje wykonanie serii analiz układu w celu znalezienia takich wartości aktywnych zmiennych optymalizacji, które zagwarantują minimalizację funkcji celu, tzn. spełnienie założeń co do wartości zdefiniowanych specyfikacji projektowych (zobacz dyrektywa SP - rozdział 4.4.1). Format dyrektywy przedstawiono poniżej.

```
1 .CENT;
```

4.4.4 .ERRC - statystyczne błędy optymalizacji

Dyrektywa umożliwia obliczenie wrażliwości funkcji celu na wewnętrzne zmienne optymalizacji. Format dyrektywy przedstawiono poniżej.

```
1 .ERRC;
```

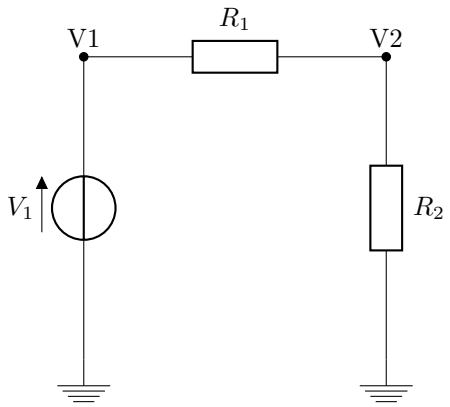
4.4.5 .SVER - weryfikacja specyfikacji projektowych

Dyrektywa umożliwia wykonanie serii analiza dla zdefiniowanych zmiennych projektowych, celem ich weryfikacji. Format dyrektywy przedstawiono poniżej.

```
1 .SVER;
```

4.4.6 Przykład projektowania

Na rysunku 4.1 przedstawiono schemat dwójnika rezystancyjnego, którego wartości parametrów elementów R_1 i R_2 będą dobierane w procesie projektowania. Na wydruku 4.17 przedstawiono plik wejściowy dla symulatora.



Rys. 4.1: Optymalizowany dwójnik RR

W przykładzie zdefiniowano

Wydruk 4.17: Przykład procesu optymalizacji dla dwójnika RR (cir/test-ana/cent2r.cir)

```
1 .TASK "DWOJNIK R-R"
2 # -- pliki biblioteczne ------------------------------
3 .LIB "types.mdl"
4 .LIB "units.mdl"
```

```
5  .LIB "vars.mdl"
6  .LIB "math/pulse.mdl"
7  .LIB "math/temp3.mdl"
8  .LIB "src/v.mdl"
9  .LIB "rlc/r.mdl"
10 .LIB "rlc/l.mdl"
11 .LIB "rlc/c.mdl"
12 # -- linie modeli ----------------------------------------
13 .MODEL R R
14 .MODEL C C
15 .MODEL V V (V1=0, V2=0)
16 # -- lista połączeń --------------------------------------
17 V.V1 V1 0 DC=2V,DEBUG=0
18 R.R1 V1 V2 R=1k
19 R.R2 V2 0 R=1k
20 # -- dyrektywy analiz -------------------------------------
21 .CMD
22 .OPTI REXG=1,PIV=1,ITL4=100,ITL1=40,POST=1,HMIN=1E-8
23 # -- specyfikacje projektowe ------------------------------
24 .SP ADD OP 0 REL V2 S1=0.5 W1=2 S2=0.7 W2=1
25 .SP ADD OP 0 REL V1 S1=2 W1=1 S2=2.5 W2=1
26 # -- zmienne projektowe -----------------------------------
27 .VAR ADD R1.R SCALE=10 MIN=100 MAX=1000
28 .VAR ADD R2.R SCALE=1000 MIN=100 MAX=6000
29 # -- opcje ------------------------------------------------
30 .OPTI WOPT=AF,ECHO=ON
31 .OPTI EOPT=1e-3,STIN=0.1,STMX=0.5,MXAN=20
32 # -- centrowanie deterministyczne -------------------------
33 .CENT
34 .ERRC
35 .SVER
36 # -- wydruki specyfikacji i zmiennych projektowych ---
37 .SP PRI ACT
38 .VAR PRI VAR
39 .END
```

W liniach 24..25 zdefiniowano specyfikacje projektowe dla węzłów $V1$ i $V2$. W liniach 27..28 zdefiniowano zmienne projektowe. Wykonano centrowanie deterministyczne w linii 33 oraz użyto dyrektyw projektowania w liniach 34..35. W liniach 37..38 wydrukowano listę specyfikacji projektowych i zmiennych optymalizacji.

4.5 Opcje programu i analiz

4.5.1 .OPTI - ustawianie opcji programu

Dyrektywa umożliwia ustawianie lub zmianę wartości opcji programu.

```
1 .OPTI opt=val, ...;
```

gdzie: opt jest nazwą opcji zawartą w tab. 4.1, natomiast val jest wartością liczbową lub ciągiem znaków. Opcje podzielone zostały na grupy. Przykład użycia dyrektywy przedstawiono na wydruku 4.18.

Wydruk 4.18: Przykład użycia dyrektywy .OPTI

```
1 .OPTI FIXS=0, ALLP=1, HMIN=0.001;
2 .OPTI OPTS;
```

W linii 1 ustawiono wartości odpowiednich opcji: FIXS, ALLP, HMIN. W linii 2 wydrukowano listę opcji wraz z wartościami ustawionymi i domyślnymi.

Każdorazowe wystąpienie dyrektywy .OPTI zmienia wartość tylko tych opcji, które wystąpiły w linii zlecenia.

Tab. 4.1: Lista opcji

Nazwa	Wartość	Opis
OPTS	-	wydruk opcji
PAUSE	OFF	(ON,OFF) zatrzymanie programu w przypadku błędu i oczekiwanie na reakcję operatora

kontynuacja na następnej stronie

Tab. 4.1 - kontynuacja z poprzedniej strony		
Nazwa	Wartość	Opis
STOP	OFF	(ON,OFF) zatrzymanie programu w wybranych punktach i oczekiwanie na reakcję operatora
ECHO	OFF	(ON,OFF) włączenie wydruku zbioru wejściowego w pliku logu
Opcje analiz OP/DC/TR		
RELT	1e-5	względna dokładność zmiennych sieciowych w algorytmie NR. Wartości bezwzględne ustawiane są indywidualnie dla każdego typu zmiennej.
ERTR	0.001	względna dokładność analizy czasowej stanu nieustalonego TR
EPTR	0.001	względna dokładność analizy czasowej stanu nieustalonego TR wykorzystywana do sprawdzania dokładności po analizie w punkcie t_n jeśli włączona jest opcja POST
ERST	0.01 (0.001)	względna dokładność analizy czasowej stanu ustalonego STDS
SR	GEAR	wybór schematu różnicowego w analizie TR: GEAR lub TRAP (schemat różnicowy trapezowy)
ORDE	0	maksymalny rząd schematu różnicowego Gear'a w analizie TR. Wartość domyślna 0 oznacza rząd zmienny.
MAXO	6	maksymalny rząd SR w trakcie analizy.
HMIN	> 0	minimalny krok czasowy analizatora TR lub zmiennej niezależnej w analizie DC. Wartość domyślna TR: $1e-11 \cdot t_{max}$, DC: $1e-6 \cdot (v_{max} - v_{min})$
LTEA	1	mnożnik błędu względnego w analizach TR/STDS/DC $(0 \ldots 1)$. Im mniejsza wartość, tym kryterium doboru kroku jest łagodniejsze
STEP	> 0	krok początkowy w analizach TR/DC/OP (tylko w metodzie kontynuacji). Wartość domyślna $STEP = HMIN$
SSMIN	1e-16	najmniejsza rozróżnialna wartość w analizatorze TR
FIXS	0	wymuszenie stałego kroku analizy DC i TR
POST	OFF	kontrola lokalnego błędu obcięcia (LTE - rozdział 6.22) po analizie w danym punkcie czasowym. Wartości dopuszczalne: OFF - wyłączona, RLTE - algorytm wykorzystujący mnożnik kroku r_{LTE}, DELT - algorytm sprawdzający błąd LTE działający na przyrostach
ITL1	100	maksymalna liczba iteracji NR w analizie punktu pracy (OP)
ITL4	50	maksymalna liczba iteracji NR na punkt czasowy w analizie TR
ITL5	40000	maksymalna całkowita liczba iteracji NR w trakcie analizy TR
ITL7	50	maksymalna liczba iteracji STDS w analizie stanu ustalonego
		kontynuacja na następnej stronie

Tab. 4.1 - kontynuacja z poprzedniej strony		
Nazwa	Wartość	Opis
NBPS	0	włączanie omijania w iteracjach NR
NRME	AUTO	wybór algorytmu analizy punktu pracy: AUTO - algorytm Newtona-Raphsona z automatycznym punktem startowym, UIC - algorytm Newtona-Raphsona z warunkami początkowymi zadanymi przez użytkownika (.IC), CONT - metoda kontynuacji.
NRST	MOD	algorytm liczenia testu stopu iteracji NR: MOD - test stopu dla sterowań i wyjść modeli nieliniowych, VAR - algorytm kontrolujący przyrosty wszystkich zmiennych sieciowych, BOTH - obydwa algorytmy włączone.
Analiza czasowa kierowana zdarzeniami - ED ITA		
ALPH	1e-1	współczynnik grupowania
REL1	100	maksymalna liczba iteracji GS na krok
REL2	40000	całkowita liczba iteracji GS
RELG	10	dopuszczalna liczba iteracji po wykonaniu grupowania
RELS	0	rodzaj testu stopu iteracji Gaussa-Seidela: 0 - klasyczny, 1 - modyfikowany
RDEL	1e-7	bezwzględna dokładność obliczenia zmiennych
REPS	1e-7	względna dokładność obliczenia zmiennych
ESYN	0	synchronizacja węzłów zewnętrznych przez węzły zależne (0-off, 1-on)
IDLT	0.5	mnożnik maksymalnej zmiany sygnału (DELTA=IDLT*Vmax) w każdym kroku ($IDLT \in < 0, 1 >$)
Optymalizacja		
MXAN	100	maksymalna liczba obliczeń funkcji celu
EOPT	0.01	względna dokładność optymalizacji
STIN	0.005	krok początkowy w algorytmie centrowania deterministycznego
STMX	0.1	krok maksymalny w algorytmie centrowania deterministycznego
WOPT	1	wydruki kontrolne optymalizacji: OFF - brak wydruków, FIN - wydruki końcowe, DU - dodatkowe wydruki w trakcie analiz, AF - wydruki końcowe analiz, DA - dodatkowe wydruki w trakcie analiz, FULL - maksymalna ilość informacji na wydrukach.
Rozwiązywanie układów równań (LU)		
REXG	ZERO	wybór sposobu przestawień wierszy w macierzy układu przed rozwiązaniem: OFF - wyłączone przestawienia, ZERO - eliminacja zer z przekątnej, MIN - minimalizacja liczby elementów.

kontynuacja na następnej stronie

Tab. 4.1 - kontynuacja z poprzedniej strony

Nazwa	Wartość	Opis
PIV	SCALE	sposób wyboru elementów podstawowych (*pivotów*): BASIC - minimalizacja liczby wypełnień tylko w pierwszej iteracji, SCALE - skalowanie elementów w każdej iteracji, REORD - przestawienia elementów w każdej iteracji.
PIVR	1e-13	względna wartość *pivota*
PIVT	1e-3	bezwzględna wartość *pivota*
PIVM	0.1	minimalna wartość *pivota*
Modele		
LIMU	BOTH	ograniczenie zmian wartości sterowań w iteracjach NR: BOTH - ograniczanie przyrostów dodatnich i ujemnych, LHS - ograniczanie przyrostów ujemnych, RHS - ograniczanie przyrostów dodatnich, OFF - brak ograniczenia.
CLLB	2	domyślna wartość dolnego ograniczenia kroku NR
CLUB	12	domyślna wartość górnego ograniczenia kroku NR
Sterowanie wydrukiem		
DAT	CAZM	format zapisu zbioru (*.dat) dla postprocesora graficznego: TXT, CAZM, SPICE3
ALLP	0	0 - wydruk wyników tylko w punktach rastra wydruku, 1 - wydruk wyników analiz we wszystkich analizowanych punktach
WTBR	1	wypisanie informacji o pułapkach czasowych (*ang. breakpoints*): 0 - brak wydruków, 1 - pułapki użytkownika, 2 - pułapki programu, 3 - pułapki użytkownika i programu.
WNRI	0	poziom wydruku NR: 0 - brak wydruków.
WDCT	1	poziom wydruku analiz OP,DC,TR,STDS: 0 - brak wydruków, 1 - wydruki końcowe, 2 - 1 + wyniki iteracji STDS, 3 - 2 + wydruki wyników pośrednich, 4 - 3 + wydruk struktury macierzy w trakcie analizy,
WAC	1	poziom wydruku analizy AC: 0 - brak wydruków, 1 - wydruki końcowe, 2 - 1+informacje dodatkowe, 3 - 2+wydruk struktury macierzy przed analizą, 4 - 3+wydruk macierzy w trakcie analizy, 5 - 4+struktura macierzy w trakcie analizy.
DIME	0	informacje statystyczne o układzie: 0 - tylko statystyka, 1 - lista nazw: modeli, funkcji, typów, natur, środowisk, 2 - 1 + informacje szczegółowe, 3 - lista nazw układów, 4 - 3+lista elementów w układzie, 5 - wydruki szczegółowe bazy danych.
Śledzenie pracy programu		
DEBU_OP	0	śledzenie analizatora OP: 0 - brak wydruków, 1 - informacje o parametrach i opcjach analizatora, śledzenie wywołań usług i wewnętrzny identyfikator macierzy układu, 2 - 1+numer iteracji, 3 - 2 + mnożnik kroku i przyrost kroku w metodzie kontynuacji.

kontynuacja na następnej stronie

Tab. 4.1 - kontynuacja z poprzedniej strony

Nazwa	Wartość	Opis
DEBU_AC	0	śledzenie analizatora AC: 1 - informacje o parametrach i opcjach analizatora, śledzenie wywołań usług, 2 - 1 + częstotliwość analizy i identyfikator analizowanego punktu, 3 - 2 + wewnętrzne informacje o utworzonej macierzy układu.
DEBU_TR	0	śledzenie analizatora TR: 0 - brak wydruków, 1 - informacje o parametrach i opcjach analizatora, śledzenie wywołań usług, 2 - 1 + wektor pułapek czasowych, mnożnik wyjść, współczynnik dyskretyzacji (γ), informacje o skróceniu kroku, 3 - 2 + wektor przeszłości, 4 - 3 + wewnętrzne wektory SR, 5 - 4 + zatrzymanie po każdym punkcie czasowym.
DEBU_STDS	0	śledzenie analizatora STDS: 0 - brak wydruków, 1 - informacje o parametrach i opcjach analizatora, śledzenie wywołań usług, identyfikator analizowanego okresu, 2 - 1 + informacje o analizowanym punkcie czasowym, 3 - 2 + wektor wyników analizy, 4 - 3 + wyznaczone nowe warunki początkowe, mnożnik okresu, 5 - 4 + wydruk macierzy, 6 - 5 + wydruk wewnętrznych wektorów i macierzy analizatora przy wyznaczaniu nowych warunków początkowych.
DEBU_DF	0	śledzenie schematu różnicowego Gear'a: 0 - brak wydruków, 1 - informacje o wywołaniach usług SR, opcje i parametry, 2 - 1 + predykcja, czas analizy, 3 - 2 + moduły wartości maksymalnych zmiennych, nowy krok, wyznaczony i przyjęty maksymalny mnożnik kroku r_{LTE} i rząd SR, informacje o skróceniu kroku, 4 - 3 + wydruk wektorów wewnętrznych, dobór kroku, mnożniki r_{LTE}, dobór rzędu SR, 5 - 4 + informacje o aktualizacji wektorów wewnętrznych,
DEBU_MTX	0	śledzenie pakietu macierzy rzadkich (MTX): 0 - brak wydruków, 1 - informacje o wywołaniach usług, 2 - 1 + wydruk macierzy po rozkładzie LU i wektorów rozwiązań po podstawieniu w przód.
DEBU_TB	0	śledzenie modułu pułapek czasowych: 0 - brak wydruków, 1 - informacje o wywołaniach usług.
DEBU_OUT	0	śledzenie modułu wyprowadzania danych wyjściowych: 0 - brak wydruków, 1 - informacje o wywołaniach usług i wartościach stemplowanych w macierzy.
DEBU_MOD	0	śledzenie modeli elementów: 0 - brak wydruków, 1 - informacje o wywołaniach usług.

kontynuacja na następnej stronie

Tab. 4.1 - kontynuacja z poprzedniej strony

Nazwa	Wartość	Opis
DEBU_CIR	0	śledzenie opisu układu: 0 - brak wydruków, 1 - informacje o wywołaniach usług.
DEBU_MDL	0	śledzenie wykonania kodu MDL: 0 - brak wydruków, 1 - informacje o wywołaniach usług, 2 - 1 + śledzenie wykonywania kodu MDL, 3 - 2 + większy poziom szczegółowości śledzenia wykonania kodu MDL.
DEBU_PAR	0	śledzenie operacji na parametrach i opcjach: 0 - brak wydruków, 1 - informacje o wywołaniach usług.
DEBU_LEX	0	śledzenie pracy parsera wejściowego
DEBU_ELE	0	śledzenie operacji na elementach: 0 - brak wydruków, 1 - informacje o wywołaniach usług.

Rozdział 5

Integracja z oprogramowaniem zewnętrznym

Na potrzeby obsługi symulatora Dero został stworzony system symulacji przez internet DeroWWW. DeroWWW zostało następnie zintegrowane z platformą e-learningową Quela [Pla12] - rozdział 5.1. Do sprawnej edycji plików wejściowych dla systemu Dero został stworzone pliki składni przeznaczone dla edytora *vim* i jego mutacji - rozdział 5.2. Jest to edytor wywodzący się od doskonale znanego w środowisku systemu Unix edytora *vi*.

5.1 Integracja z platformą e-learningową *Quela*

Platforma Quela (wersja 2) została zintegrowana z systemem symulacji DeroWWW. Interfejs użytkownika platformy jest intuicyjny. Nie ma potrzeby uczenia się obsługi platformy. Wszystkie możliwe akcje na zasobach platformy wyświetlane są wtedy, gdy są potrzebne i zalogowany użytkownik posiada odpowiednie prawa dostępu. System wyposażony został w dokumentację: w formacie *pdf*, system pomocy *manpages*, a także dokumentację typu *howto*. W dokumentacji zamieszczono szereg przykładów. Na rys. 5.1 przedstawiono główne okno systemu DeroWWW jako kontekst pracy platformy Quela dla użytkownika na prawach administratora.

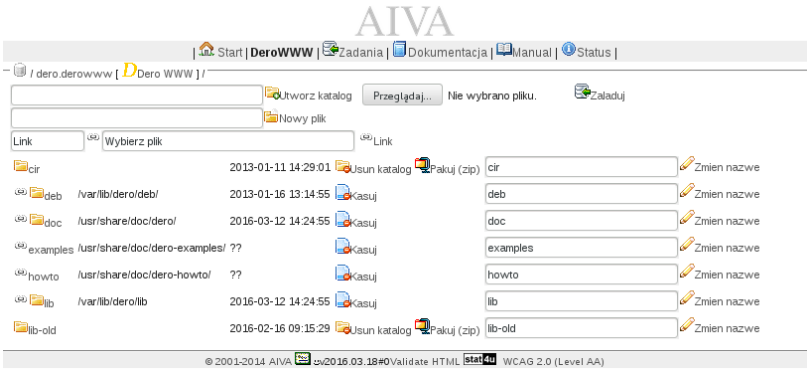
Na rys. 5.2 przedstawiono okno z listą zadań dla symulatora Dero. W katalogu tym znajdują się także pliki błędów (**.err*), pliki wyjściowe (**.out*) oraz pliki w formacie (**.plt*) dla apletu do wizualizacji przebiegów.

Na rys. 5.3 przedstawiono okno z zawartością katalogu dokumentacji systemu.

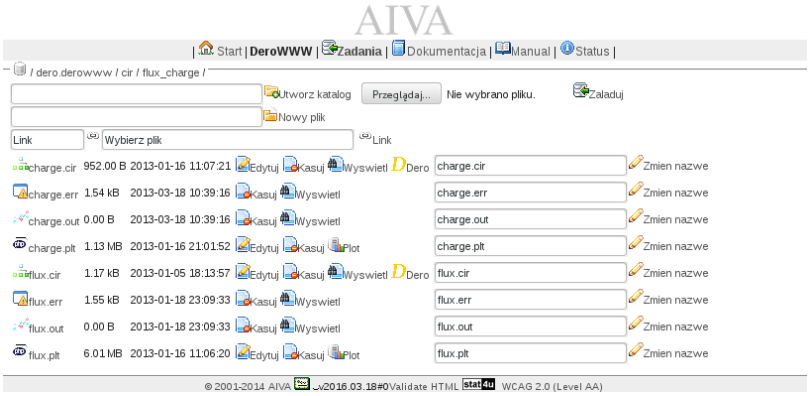
Na rys. 5.4 przedstawiono początek dokumentacji z systemu pomocy *manpages* symulatora Dero.

Na rys. 5.5 przedstawiono wygląd graficznego postprocesora z wynikami analizy przykładowego układu. Przedstawiony aplet został napisany w języku Java. Umożliwia skalowanie wykresów oraz ich wydruk. Do prawidłowego działania apletu wymagana jest instalacja maszyny wirtualnej Java.

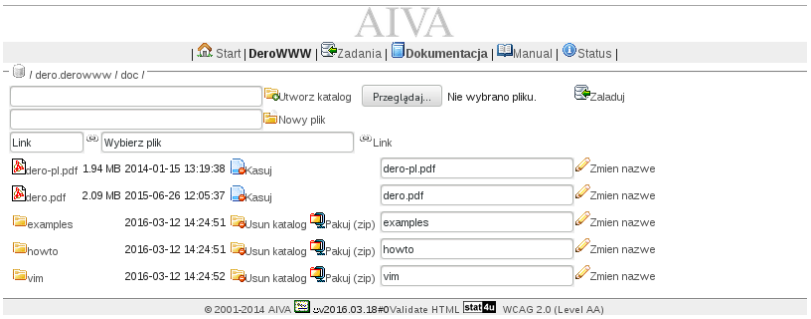
Możliwość uruchomienia apletu została przedstawiona w kontekście platformy Quela. W kontekście DeroWWW interfejs wygląda tak samo, posiada tylko wymienione manu na menu DeroWWW (poprzednie rysunki).



Rys. 5.1: Interfejs systemu DeroWWW na platformie Quela.



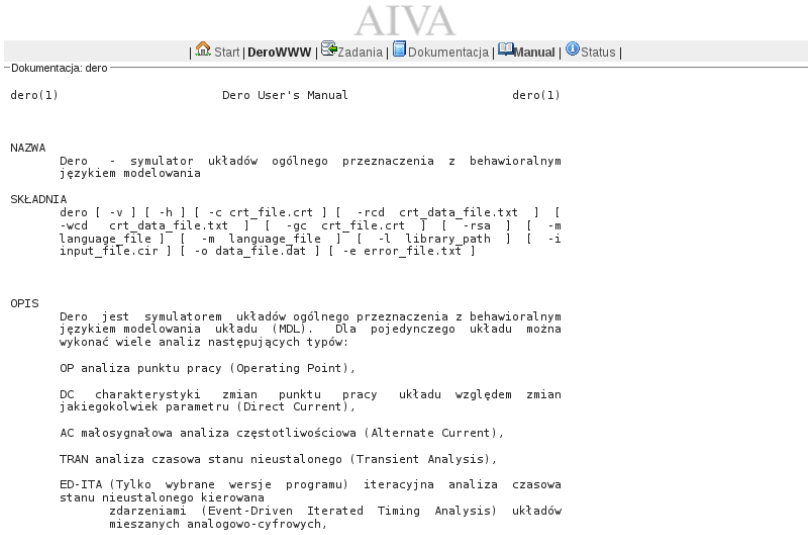
Rys. 5.2: Katalog zadań i wyników dla symulatora Dero.



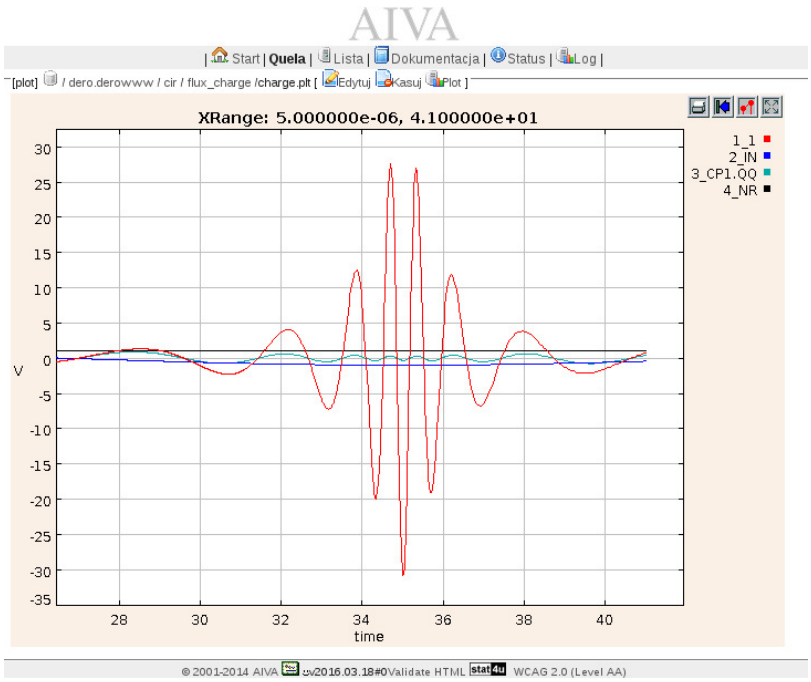
Rys. 5.3: Katalog dokumentacji.

5.2 Integracja z edytorem Vim

Na potrzeby efektywnej obróbki plików wejściowych symulatora (plików modeli **.mdl* i plików układów **.cir*) opracowano pliki składni dla edytora vim [pag] W celu inte-



Rys. 5.4: Okna systemu pomocy *manpages*.



Rys. 5.5: Aplet wyników analizy przykładowego układu.

gracji z edytorem należy stworzyć plik składni (*dero.vim*) oraz wprowadzić modyfikacje do pliku typów programu vim. Odpowiednie pliki dostarczane są w postaci pakietu instalacyjnego - *dero4-vim.deb*.

Plik składni dla edytora Vim przedstawiono na wydruku 5.1.

Wydruk 5.1: Plik do formatowania składni edytora vim

```
" Vim syntax file for Dero4 symulator
" Revision : 1.3
" Language:     Dero
" Maintainer: Plasky <plasky@aiva.pl> (C) AIVA 2009
" Last change:  Date : 2016/12/0613 : 47 : 50

" Remove any old syntax stuff hanging around
syn clear

syn case ignore

syn keyword deroStatement PREPROCESSED TIMEBREAKPOINTS REPORT
syn keyword deroLabel TRAN DC OP
syn keyword deroConditional IF ELSE THEN END
syn keyword deroRepeat DO FOR WHILE

syn region deroString start=+"+  end=+"+
syn region deroCommand start="\.[a-zA-Z]" end="[a-zA-Z0-9_]*"

syn match deroDelimiter "[;}{]"
syn match deroMatrixDelimiter "[][]"

syn match deroNumber "-\=\<[0-9]\+\.[0-9]\+[eE]-\=[0-9]\+\>"
syn region deroComment start="#" end="
"

syn keyword deroType OPTI INTERNAL EXTERNAL FLOW OUTPUT INPUT GROUP
syn keyword deroType PARAM COMMON VAR MEM

syn keyword deroFunction PRINT_ID PRINT SET_TBRK
syn keyword deroFunction RETURN EXIT

syn sync lines=250

if !exists("did_dero_syntax_inits")
  let did_dero_syntax_inits = 1
  hi link deroComment Comment
  hi link deroCommand Statement
  hi link deroConditional Conditional
  hi link deroDelimiter Identifier
  hi link deroFunction Function
  hi link deroLabel Type
  hi link deroMatrixDelimiter Identifier
  hi link deroNumber Number
  hi link deroOperator  Operator
  hi link deroRepeat  Repeat
  hi link deroStatement  Statement
  hi link deroString String
  hi link deroType Type
endif

let b:current_syntax = "dero4"

" vim: ts=8
```

Część II

Metody i algorytmy obliczeniowe

Rozdział 6

Metody analizy i optymalizacji

Oznaczenia

A- macierz,

Y- macierz admitancyjna układu,

D- macierz elementów diagonalnych macierzy A,

L- macierz elementów pod przekątną macierzy A, indukcyjność,

U- macierz elementów nad przekątną macierzy A,

x- niewiadoma, wektor niewiadomych, uogólniona zmienna sieciowa

B- wektor prawych stron, wektor wymuszeń,

b- wektor prawych stron, wektor wymuszeń,

C- macierz charakterystyczna, pojemność,

$\rho(C)$ - promień spektralny macierzy C,

G- konduktancja,

R- rezystancja,

S- wektor sterowań modelu,

v- napięcie,

i- prąd,

q- ładunek,

$x_{i,n}^{k,p}$ - zmienna sieciowa,

^{k} - indeks górny numeru iteracji Gaussa-Seidela,

^{p} - indeks górny numeru iteracji Newtona-Raphsona,

$_{i}$ - indeks dolny numeru zmiennej,

$_{n}$ - indeks dolny chwili czasowej,

$\hat{x}$ - zmienna sieciowa poddana transformacji (ograniczeniu wartości),

N - liczba równań, wymiar układu, liczba węzłów,

n - indeks chwili czasowej,

h - krok czasowy,

t - czas,

t_n - n-ty punkt czasowy,

$_n$ - indeks dolny n - tego punktu czasowego,

$\dot{x}$ - pochodna czasowa zmiennej, tzn. $\dot{x} = \frac{dx}{dt}$,

γ - współczynnik w schemacie różnicowym zależny od rodzaju schematu różnicowego $\dot{x} = \gamma\ x + d_x$,

d_x - wektor przeszłości zmiennej x w schemacie różnicowym,

LTE - lokalny błąd obcięcia (*ang. LTE*),

l - rząd schematu różnicowego,

K, K_1 - mnożniki kroku całkowania numerycznego,

ϵ, α, δ - współczynniki skalowania kroku,

δ - błąd bezwzględny,

δ_x - błąd bezwzględny zmiennej typu x,

ϵ - błąd względny,

ϵ_x - błąd względny zmiennej typu x,

h - podwektor wektora w,

$_{s,d}$ - indeksy dolne podobwodów,

$f(x, \dot{x}, t)$ - funkcja układowa,

$vecSt$ - wektor wymuszeń, których wartość zmienia się w czasie,

p- parametr układu (np. dla zadania optymalizacji).

6.1 Równania sieci

Nieliniową sieć można opisać za pomocą układu równań postaci (6.1).

$$f(x, \dot{x}, t) = 0 \tag{6.1}$$

gdzie: x jest wektorem zmiennych sieciowych, $\dot{x}$ jest wektorem pochodnych zmiennych x względem czasu, t jest czasem.

Rozwiązywanie tego typu równań jest kilkuetapowe. Równanie (6.1) należy zdyskretyzować celem eliminacji pochodnej $\dot{x}$ za pomocą schematu różnicowego [J.O95] postaci (6.12)

$$\dot{x}_n = \gamma\ x_n - d_{xn} \tag{6.2}$$

gdzie: d_x jest wektorem przeszłości, γ współczynnikiem zależnym od rodzaju schematu różnicowego. Otrzymany w ten sposób układ nieliniowych równań algebraicznych (6.3).

$$f(x_n, \gamma\ x_n - d_{xn}, t_n) = 0 \tag{6.3}$$

Można go rozwiązać np. metodą Newtona-Raphsona (zobacz 6.2) przez linearyzację równań - rozwinięcie w szereg Taylora [J.O95] do wyrazów rzędu pierwszego. Otrzymamy wtedy układ równań liniowych.

$$\underbrace{\left[\frac{\delta f(x_n^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}{\delta x_{t_n}} + \gamma \frac{\delta f(x_{t_n}^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}{\delta \dot{x}_{t_n}}\right]}_{\frac{df^{p-1}}{dx_n}} (x_n^p - x_n^{p-1}) =$$

$$= -\underbrace{f(x_n^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}_{f^{p-1}} \tag{6.4}$$

Po pogrupowaniu czynników i wprowadzeniu oznaczeń przyjmuje on postać (6.5).

$$\underbrace{\frac{df^{p-1}}{dx_n}}_{Y} \cdot {x_n}^p = \underbrace{-f^{p-1} + \frac{df^{p-1}}{dx_n} \cdot x_n^{p-1}}_{B} \tag{6.5}$$

Pochodne po lewej stronie równania wchodzą do macierzy Y, a wartości po prawej stronie do wektora prawych stron B. Układ algebraicznych równań liniowych można rozwiązać jedną z metod [J.O95] (zobacz 9.1) np.:

- metodą eliminacji Gaussa
- metodą rozkładu LU,
- metodami iteracyjnymi.

6.2 Algorytm Newtona-Raphsona

Metoda Newtona-Raphsona [J.O95] jest klasyczną metodą rozwiązywania nieliniowych równań algebraicznych[1]. Załóżmy, że rozpatrujemy funkcję $f(x)$, która jest różniczkowalna w punkcie x_0. Funkcję f można rozwinąć rozwinąć w szereg Taylora do wyrazów rzędu drugiego w otoczeniu punktu x_0.

$$f(x) = f(x_0) + \frac{\delta f(x_0)}{\delta x}(x - x_0)$$

Równanie można rozwiązać ($f(x) = 0$) iteracyjnie.

$$\frac{\delta f(x_0)}{\delta x}(x - x_0) = -f(x_0)$$

Ciąg kolejnych przybliżeń rozwiązań jest zbieżny, jeśli punkt startowy leży w obszarze zbieżności (nie jest zbytnio oddalony od rozwiązania). Odpowiednie wartości x i x_0 będą wartościami otrzymywanymi w kolejnych iteracjach NR. Wprowadzając indeks iteracji p otrzymamy: $x^{(p)} = x$, $x^{(p-1)} = x_0$. Odpowiednie przyrosty wartości sygnałów w p-tej iteracji oznaczmy jako $\Delta x^{(p)}$.

$$\Delta x^{(p)} = x^{(p)} - x^{(p-1)}$$

Po wprowadzeniu oznaczeń i pogrupowaniu otrzymamy równanie iteracji NR.

$$\frac{\delta f(x^{(p-1)})}{\delta x}\Delta x^{(p)} = -f(x^{(p-1)})$$

Algorytm operuje na przyrostach zmiennych Δx. Z tego względu nazywany jest algorytmem przyrostowym NR[2].

Prawa strona równania zależy tylko od wartości zmiennej sterującej z poprzedniej iteracji $x^{(p-1)}$. Pochodne po lewej stronie wchodzą do macierzy Y, a wartości po prawej stronie do macierzy B (patrz rozdział 9.1).

Iteracyjny proces rozwiązania kończy się z chwilą osiągnięcia zadanej dokładności, tzn. po spełnieniu kryterium zakończenia iteracji NR:

$$| \Delta x^{(p)} | < \epsilon \ \max(| x^{(p)} |, | x^{(p-1)} |) + \delta$$

gdzie: ϵ - wartość błędu bezwzględnego, a δ - wartość błędu względnego zmiennej x.

W praktyce stosowana jest inna wersja algorytmu NR, gdzie niewiadomymi są wartości bezwzględne x a nie Δx. Równanie przyjmuje wtedy postać (6.6).

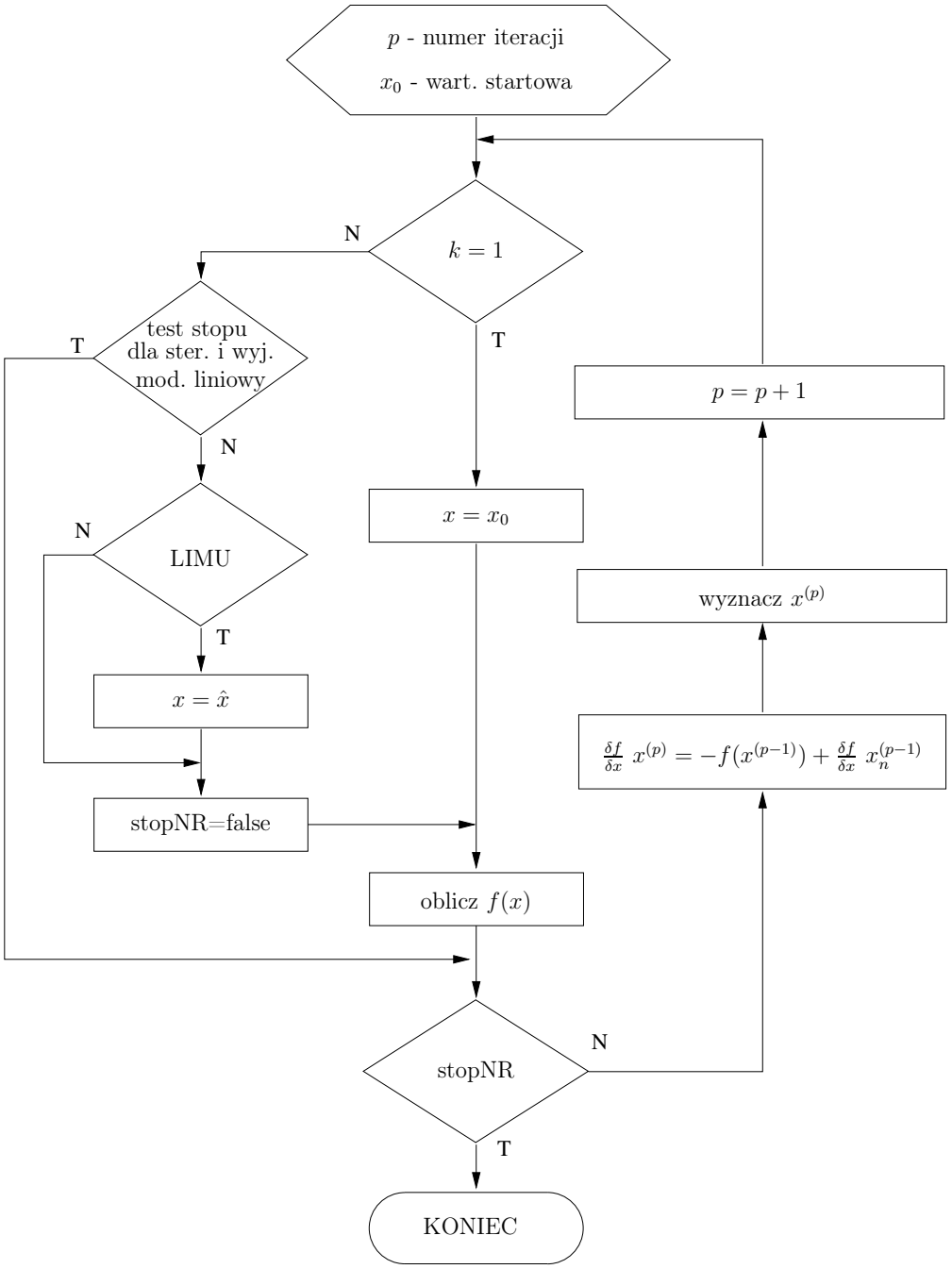
$$\frac{\delta f(x^{(p-1)})}{\delta x}\ x^{(p)} = -f(x^{(p-1)}) + \frac{\delta f(x^{(p-1)})}{\delta x}\ x_n^{(p-1)} \tag{6.6}$$

Proces iteracyjnego rozwiązywania równania (6.6) można zapisać w postaci algorytmu 1.

[1] W nowoczesnych symulatorach mikrosystemów wykorzystywane są także inne metody np. metoda Powell'a [P+07]

[2] Zaletą algorytmu przyrostowego jest możliwość wykonywania obliczeń na maszynach dysponujących typami danych o stosunkowo małym zakresie. W praktyce stosuje się częściej algorytm działający na zmiennych x, a nie na przyrostach Δx

Algorytm 1 Algorytm Newtona-Raphsona



6.2.1 Problemy numeryczne w algorytmie Newtona-Raphsona

Do podstawowych problemów numerycznych w algorytmie NR należą:

- przekroczenie zakresu reprezentowanych liczb ze względu na często spotykane w praktycznych układach funkcje wykładnicze. Rozwiązaniem jest tutaj zastosowanie odpowiedniej aproksymacji funkcji wykładniczych oraz ich pochodnych.
- silne oscylacje algorytmu połączone z powolną zbieżnością. Rozwiązaniem problemu jest ograniczenie szybkości zmian zmiennych sterujących x.

Aproksymacja funkcji wykładniczych Aproksymacja funkcji wykładniczych umożliwia zmniejszenie wartości pochodnych funkcji f poza zakresem użytecznym, co ogranicza niebezpieczeństwo wyjścia poza zakres reprezentowanych liczb.

$$expo(x) = \begin{cases} exp(40)(x-39) & x > 40 \\ exp(x) & 0 \le x \le 40 \\ \frac{1}{1-x^2} & x < 0 \end{cases} \tag{6.7}$$

$$dxpo(x) = \begin{cases} exp(40) & x > 40 \\ exp(x) & 0 \le x \le 40 \\ \begin{cases} 1 & \text{wersja liniowa} \\ \frac{1}{1-x^2} & \text{wersja hiperboliczna} \end{cases} & x < 0 \end{cases} \tag{6.8}$$

$$cpx1am(x,m) = \begin{cases} (1-x)^{-m} & x \le 0.9 \\ 10^m[m(10x-9)+1] & x > 0.9 \end{cases} \tag{6.9}$$

Transformacje zmiennych sterujących Tłumienie przyrostów zmiennych sterujących x jest bardzo skuteczną metodą przyśpieszania zbieżności algorytmu NR. Zmienne sterujące podlegają transformacji:

$$\hat{x}^{(p)} = \hat{x}^{(p-1)} + \frac{1}{k} sgn(x^{(p)} - \hat{x}^{(p-1)}) ln(1 + k\|x^{(p)} - \hat{x}^{(p-1)}\|) \tag{6.10}$$

gdzie: $\hat{x}$ - wartość ograniczona, $k = max(CLLB, CLUB - p)$ decyduje o sile ograniczenia, CLUB i CLLB są odpowiednimi parametrami decydującymi o sile ograniczania, p - numer iteracji NR.

Algorytm NR (6.6) w wersji tłumionej (6.11) został zastosowany w programie.

$$\frac{\delta f(\hat{x}^{(p-1)})}{\delta x}\ \hat{x}^{(p)} = -f(\hat{x}^{(p-1)}) + \frac{\delta f(\hat{x}^{(p-1)})}{\delta x}\ \hat{x}_n^{(p-1)} \tag{6.11}$$

6.3 Schemat różnicowy Gear'a oparty na różnicach skalowanych

Schemat różnicowy umożliwia obliczenie pochodnej czasowej zmiennej sieciowej ($\dot{x}$) występującej w równaniu sieci (6.1). W programie zastosowano schemat różnicowy Gear'a oparty na różnicach skalowanych ze zmiennym rzędem ($l = 1..6$). Dobór rzędu odbywa się automatycznie. Krok algorytmu zmienny, dobierany na podstawie lokalnego błędu obcięcia (*ang. LTE*). Ogólna postać schematu różnicowego dla n-tego punktu czasowego dana jest (6.12).

$$\dot{x}_n = \gamma\ x_n + d_{xn} \tag{6.12}$$

gdzie: γ - współczynnik dyskretyzacji SR, d_{xn} - wektor przeszłości.

Ogólna postać schematu różnicowego opartego na różnicach skalowanych wyrażona jest pewnym wielomianem interpolacyjnym w_n stopnia l (rząd schematu różnicowego) w $l+1$ różnych kolejno uporządkowanych punktach osi t (krotność SR). Wielomian ten powstaje przez l-krotne dzielenie wielomianu z resztami przez kolejne dwumiany $(t_n - t_{n-k})$, dla $k = 0, l$. Po przekształceniach ([J.O95, wzory (8.27)-(8.39), str.248]) otrzymamy postać SR rzędu l opartego na różnicach skalowanych (6.13).

$$\underbrace{\dot{x}_n}_{\dot{x}_n} = \underbrace{a_0\ x_n}_{\gamma\ x_n} \underbrace{- \sum_{k=0}^{l-1} (c_k\ a_k\ d_{n-1}^k\ x_{n-1})}_{d_{xn}} \tag{6.13}$$

Współczynnikami schematu różnicowego są wielkości a_k (6.14) i c_k(6.15). Wartości współczynników przechowuje się w tablicach AK i CK o długościach równych rzędowi SR (l).

$$a_k = \sum_{j=k+1}^{l} \frac{1}{t_n - t_{n-j}} = \sum_{j=k+1}^{l} \frac{1}{\underbrace{t_n - t_{n-1}}_{h_n} + \underbrace{t_{n-1} - t_{n-j}}_{hs(j-1)}} = \sum_{j=k+1}^{l} \frac{1}{h_n + hs_{n-1}(j-1)} \tag{6.14}$$

$$c_k = \begin{cases} 1 & \text{dla } k = 0 \\ \prod_{j=1}^{k} \frac{t_n - t_{n-j}}{t_{n-1} - t_{n-1-j}} = \prod_{j=1}^{k} \frac{\overbrace{t_n - t_{n-1}}^{h_n} + \overbrace{t_{n-1} - t_{n-j}}^{hs_{n-1}(j-1)}}{\underbrace{t_{n-1} - t_{n-1-j}}_{hs_{n-1}(j)}} = \prod_{j=1}^{k} \frac{h_n + hs_{n-1}(j-1)}{hs_{n-1}(j)} & \text{dla } k = 1, \ldots, l \end{cases} \tag{6.15}$$

Współczynniki c_k i a_k w punkcie t_n można zapisać wykorzystując tablicę sum kroków hs_n. Ze względu na możliwość odrzucenia kroku h_n przewidzianego w punkcie czasowym t_{n-1} dobrze jest wykorzystać tablicę hs_{n-1}[3] niezaktualizowaną z punktu t_{n-1}. Ograniczy to nakłady obliczeniowe. Do obliczenia współczynników c_k można wykorzystać tablicę niezaktualizowaną hs_{n-1} odpowiednio przekształcając wzory.

Obliczanie różnic chwil czasowych - tablica hs Różnice $t_n - t_{n-j}$, np. w (6.15), można efektywnie obliczać przechowując odpowiednie sumy kroków czasowych pomiędzy punktami t_{n-j} a t_n. Służy do tego tablica sum kroków h_s o długości równej rzędowi SR. Wprowadźmy oznaczenie poszczególnych kroków czasowych

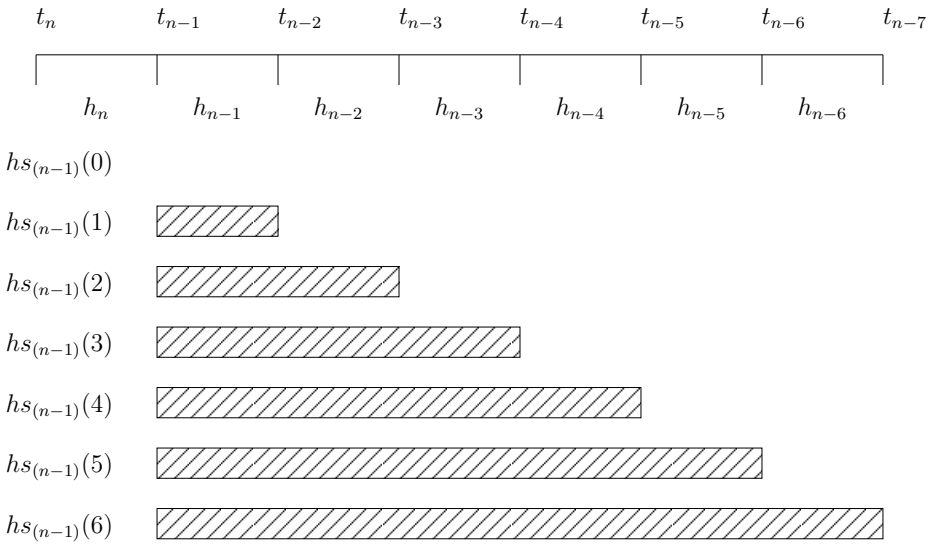
[3] Tablice hs_{n-1} aktualizuje się do hs_n dopiero po przejściu do punktu t_{n+1}.

$$h_n = t_n - t_{n-1}$$

w n-tym kroku w punkcie t_n. W punkcie t_{n-1} zawartość poszczególnych komórek tablicy (6.16) będzie sumą odpowiednich kroków czasowych h_n (6.16).

$$\begin{aligned}
hs_{n-1}[0] &= t_{n-1} - t_{n-1-0} = 0 \\
hs_{n-1}[1] &= t_{n-1} - t_{n-1-1} = h_{n-1} \\
hs_{n-1}[2] &= t_{n-1} - t_{n-1-2} = t_{n-1} - t_{n-2} + t_{n-2} - t_{n-3} = h_{n-1} + h_{n-2} \\
hs_{n-1}[3] &= t_{n-1} - t_{n-1-3} = t_{n-1} - t_{n-2} + t_{n-2} - t_{n-3} + t_{n-3} - t_{n-4} = h_{n-1} + h_{n-2} + h_{n-3} \\
&\dots \\
hs_{n-1}[l] &= t_{n-1} - t_{n-1-l} = h_{n-1} + \dots + h_{n-l}
\end{aligned} \tag{6.16}$$

Zawartość tablicy w t_{n-1} przedstawiono obrazowo na rys. 6.1.



Rys. 6.1: Zawartość tablicy sum kroków hs_{n-1} w chwili t_{n-1}.

Tablicę sum kroków należy uaktualniać w każdym kroku czasowym. Zauważmy, że w punkcie t_n tablicę należy zaktualizować zgodnie z (6.17).

$$hs_n(j) = \begin{cases} hs_{n-1}[j-1] & \text{dla } j = l, 2 \\ h_n & \text{dla } j = 1 \\ 0 & \text{dla } j = 0 \end{cases} \tag{6.17}$$

Aktualizacja polega na dodaniu h_n i przepisaniu zawartości elementów wektora, np.: $hs[6] = h_n + hs[5], hs[5] = h_n + hs[4]$. W wyniku operacji zapomnieniu ulega zawartość ostatniej komórki $hs[6]$. W komórkę $hs[1]$ należy wpisać krok h_n.

Na przykład po przejściu do punktu t_3 analizowanego krokiem h_3 tablica hs_n będzię zawierała dane względem poprzedniego punktu t_2 (hs_{n-1}) w stosunku do punktu $t_{n=3}$:

$$\begin{aligned} hs_{n-1}[0] &= 0 \\ hs_{n-1}[1] &= h_2 \\ hs_{n-1}[2] &= h_2 + h_1 \\ hs_{n-1}[3] &= 0 \\ hs_{n-1}[4] &= 0 \\ hs_{n-1}[5] &= 0 \\ hs_{n-1}[6] &= 0 \end{aligned} \tag{6.18}$$

Uwaga! Ze względu na możliwość odrzucenia kroku h_n do obliczeń współczynników a_k i c_k wykorzystuje się tablicę hs_{n-1}. Aktualizacja hs wykonywana jest przed przejściem do t_{n+1}.

Aktualizacja różnic skalowanych Aktualizacja różnic skalowanych, wymagająca niewielkich nakładów obliczeniowych, może zostać wykonana zgodnie z (6.19)[4].

$$\begin{aligned} d_n^0\ x_{n-1} &= x_n \\ d_n^k\ x_n &= x_n - \sum_{j=0}^{k-1} (c_j\ d_{n-1}^j\ x_{n-1}) \\ & \quad k = 1, l \end{aligned} \tag{6.19}$$

Tablica różnic wymaga inicjalizacji przy starcie algorytmu analizy czasowej. W pierwszym punkcie czasowym $t_n = t_1$ należy przyjąć wartości początkowe $d_0^0\ x_0$, $d_0^1\ x_0$ zgodnie z (6.20) i rozpocząć schematem rzędu $l = 1$.

$$\begin{aligned} d_0^0\ x_0 &= x_0 \\ d_0^1\ x_0 &= 0 \end{aligned}$$

Po obliczeniu wartości x_1[5] można wyznaczyć z (6.19)

$$\begin{aligned} d_1^1\ x_1 &= x_1 \\ d_1^2\ x_1 = d_1^1\ x_1 &= x_1 - x_0 \end{aligned}$$

i następny krok wykonać SR rzędu 2.

6.3.1 Przewidywanie wartości startowej

Przewidywanie wartości startowej SR oparte jest tych samych punktach dyskretyzacji. SR i SR dla predykcji różnią się tylko liczbą współczynników a nie ich wartościami.

[4] Wartość $d_{n-1}^k x_{n-1}$ została oznaczona w programie jako wektor DW [J.O95, wzór (8.41) str 249].
[5] Wartość x_1 wyznacza się przez analizę układu w punkcie t_1.

Predykcja ([J.O95, wzór (8.40) str. 248]) dla SR Gear'a realizowana jest zgodnie ze schematem do predykcji rzędu l (6.20).

$$x_n^{(pred,l)} = \underbrace{\sum_{k=0}^{l} (c_k \; d_{n-1}^k \; x_{n-1})}_{dx} \tag{6.20}$$

6.3.2 Lokalny błąd obcięcia

Do kontroli lokalnego błędu obcięcia (LTE) (*ang. Local Truncation Error* - LTE) używa się zadanych przez użytkownika błędów bezwzględnego δ i względnego ϵ w każdym kroku całkowania (w każdym n-tym punkcie czasowym). Wartość lokalnego błędu obcięcia zadaną przez użytkownika można wyznaczyć z (6.21) na podstawie znajomości δ i ϵ. Wartość lokalnego błędu obcięcia oblicza się z zależności (6.22).

$$LTE_{user} = \epsilon \; \max(\mid x_n^{(pred,l)} \mid, \mid x_n \mid) + \delta \tag{6.21}$$

$$LTE = \mid x_n^{(pred,l)} - x_n \mid \tag{6.22}$$

gdzie: $x_n^{(pred,l)}$ jest wartością przewidywaną z (6.20) rzędem l, x_n jest wartością obliczoną (dokładną) w n-tym punkcie czasowym. Wartość ϵ ustawiana jest opcją *ERTR* (rozdział 4.5.1), natomiast δ jest wartością błędu bezwzględnego zmiennej danego typu.

Test dokładności analizy Analiza jest dokładna, jeśli w każdym punkcie analizy spełniona jest zależność $LTE <= LTE_{user}$, która w n-tym punkcie czasowym przyjmuje w postaci (6.23).

$$\mid x_n^{(pred,l)} - x_n \mid <= \epsilon \; \max(\mid x_n^{(pred,l)} \mid, \mid x_n \mid) + \delta \tag{6.23}$$

6.3.3 Dobór kroku

W praktycznych zastosowaniach stosowane są dwie metody doboru kroku analizy:

- na podstawie liczby iteracji Newtona-Raphsona,
- na podstawie lokalnego błędu obcięcia.

Dobór kroku na podstawie lokalnego błędu obcięcia LTE W metodzie wykorzystywany jest lokalny błąd obcięcia SR do wyznaczenia mnożnika nowego kroku r_{LTE} z (6.24).

$$r_{LTE} = \frac{\mid LTE_{user} \mid}{\mid LTE \mid} = \left[\frac{h_{n+1}}{h_n}\right]^{l+1} \tag{6.24}$$

Wartość współczynnika r_{LTE} zależy od ilorazów wartości kroków h_n i h_{n+1} oraz od rzędu schematu różnicowego l. Jest on ilorazem modułów wartości lokalnego błędu obcięcia zadanego przez użytkownika LTE_{user} i obliczonego w trakcie analizy LTE. Przewidywany nowy, optymalny nowy krok h_{n+1} (6.25) gwarantuje utrzymanie zadanego poziomu lokalnego błędu obcięcia LTE_{user}.

$$h_{n+1} = r_{LTE} \; h_n \tag{6.25}$$

Jeśli $r_{LTE} > 1$ oznacza to, że wartość lokalnego błędu obcięcia jest mniejsza niż założona i otrzymane rozwiązanie jest prawidłowe - krok można wydłużyć. W programie wprowadzono dodatkowy mnożnik wydłużenia kroku (opcja *LTEA* - rozdział 4.5.1):

$$h_{n+1} = LTEA\ r_{LTE}\ h_n$$

Ograniczanie przyrostów kroku W praktyce predykowany krok h_{n+1} podlega raptownym zmianom, co powoduje problemy ze stabilnością [R.K77]. Szybkość zmian kroku trzeba ograniczać. Stopniowe zmiany kroku można uzyskać stosując ograniczanie przyrostów kroku (6.26).

$$h_{n+1} = \begin{cases} h_n\ \max(s_l, r_{LTE}) & r_{LTE} < 1 \\ h_n & 1 \le r_{LTE} \le \alpha \\ h_n\ \min(s_u, \beta\ r_{LTE}) & r_{LTE} > \alpha \end{cases} \tag{6.26}$$

Współczynnik α umożliwia wykorzystanie tego samego kroku wielokrotnie. Współczynniki s_l, s_u są mnożnikami wydłużenia lub skrócenia kroku, β uniemożliwia wydłużenie kroku o wartość wynikającą z (6.25). Ponieważ wartość LTE jest estymowana, to czasami może się zdarzyć przypadek doboru za długiego kroku i konieczność wykonania kroku wstecz. Zapobiega temu mnożnik β. W praktyce stosuje się następujące wartości parametrów:

$$\begin{aligned} s_l &= 0.25 \\ s_u &= 2.0 \\ \alpha &= 1.2 \\ \beta &= 0.9 \end{aligned}$$

Obliczenie r_{LTE} Wartość r_{LTE} jest minimalną wartością wyznaczaną dla zmiennych sieciowych r_{LTE} (6.27) na podstawie wartości LTE i LTE_{user} wyznaczanych dla każdej zmiennej sieciowej.

$$r_{LTE} = \min_i \left| \frac{LTE_{user}}{LTE} \right|^{\frac{1}{l+1}} \tag{6.27}$$

gdzie: i - indeks zmiennej sieciowej, l - rząd SR.

Dobór kroku na podstawie liczby iteracji NR Krok dobierany jest na podstawie liczby wykonanych iteracji NR (*ang. iterative count time step control*) w następujący sposób:

$$h_{n+1} = \begin{cases} h_n / s_1 & p > p_{max} \\ h_n & p_{min} \le p \le p_{max} \\ h_n\ s_2 & p < p_{min} \end{cases} \tag{6.28}$$

Główną ideą tej metody jest utrzymanie stałej liczby iteracji NR w każdym punkcie czasowym analizy. Wartości współczynników s_1 i s_2 wyznaczane są doświadczalnie.

6.3.4 Dobór rzędu

Stosowanie SR Gear'a wyższych rzędów umożliwia stosowanie dłuższego kroku przy tym samym poziomie zadanego LTE. Ze względu na stabilność SR rząd l można zmieniać jednorazowo tylko o 1 i stosować rzędy tylko 1...6.

Zmianę rzędu wykonuje się po zakończeniu analizy w chwili t_n SR rzędu l_n. Za pomocą predykcji rzędów $l = 1, ..., l_n + 1$ oblicza się wartości przewidywane zmiennych dla wszystkich dopuszczalnych rzędów l, $x_n^{pred(l)}$. Mając rozwiązania dokładne x_n estymuje się lokalny błąd obcięcia LTE, który wystąpiłby w punkcie t_n gdyby użyto rzędu l. Estymacja ta nie jest do końca dobra, gdyż x_n liczone rzędem l_n (o jeden niższym), a nie l. Niedokładność ta mieści się jednak w błędzie estymacji LTE. Wyznacza się mnożniki kroku r_{LTE_l} dla każdego rzędu l i wybiera się rząd l_{n+1} dla największego mnożnika r_{LTE_l}.

$$l_{n+1} = \max_{r_{LTE_l}} 1, ..., (l_n + 1)$$

Wybrany rząd gwarantuje najdłuższy krok ($h_{n+1} = r_{LTE}\ h_n$) przy zadanym poziomie lokalnego błędu obcięcia LTE.

6.4 Schemat różnicowy trapezowy

W programie zaimplementowano także schemat różnicowy trapezowy (1-krotny, 2-go rzędu), ze względu na dobre właściwości numeryczne - dobrą dokładność i stabilność. Schemat różnicowy trapezowy można zapisać dla n-tego punktu czasowego w postaci (6.29).

$$x_n = \frac{1}{2} h_n \dot{x}_n + \frac{1}{2} h_n \dot{x}_{n-1} + x_{n-1} \tag{6.29}$$

Równanie (6.29) można przepisać do postaci (6.12) otrzymując (6.30).

$$\dot{x}_n = \underbrace{\frac{2}{h_n}}_{\gamma} x_n + \underbrace{\left(-\dot{x}_{n-1} - \frac{2}{h_n} x_{n-1} \right)}_{d_x} \tag{6.30}$$

Do przewidywania wartości startowej stosuje się predykcję rzędu 2 dostosowaną do wybranego SR.

6.4.1 Przewidywanie wartości startowej

Ogólna formuła predykcji l-tego rzędu dana jest równaniem (6.31).

$$x_n^0 = \sum_{k=1}^{l+1} \alpha_k^l x_{n-k} \tag{6.31}$$

Współczynniki α dane są równaniem (6.32).

$$\alpha_k^l = \prod_{i=1, i \neq k}^{l+1} \frac{s_i}{s_i - s_k} = \prod_{i=1, i \neq k}^{l+1} \frac{h s_{n-1}(i-1)}{h s_{n-1}(i-1) - h s_{n-1}(k-1)} \tag{6.32}$$

Współczynniki s są kolejnymi sumami kroków czasowych danych równaniem (6.33), które można zapisać z wykorzystaniem tablicy sum kroków[6] hs_{n-1} z poprzedniej chwili czasowej t_{n-1}.

$$s_i = t_n - t_{n-i} = \sum_{k=0}^{i} h_{n-k} = h_n + hs_{n-1}(i-1) \tag{6.33}$$

Podejście takie umożliwia łatwą implementację cofania wstecz, gdyż w s zawiera krok h_n z bieżącego punktu t_n.

Rozpisane współczynniki dla predykcji

Dla SR trapezowego stosuje się predykcję rzędu 2 (tzn. $l = 2$). Rozpisując równanie (6.31) otrzymamy:

$$x_n^0 = \alpha_1^2 x_{n-1} + \alpha_2^2 x_{n-2} + \alpha_3^2 x_{n-3}$$

Współczynniki α można zapis jawnie

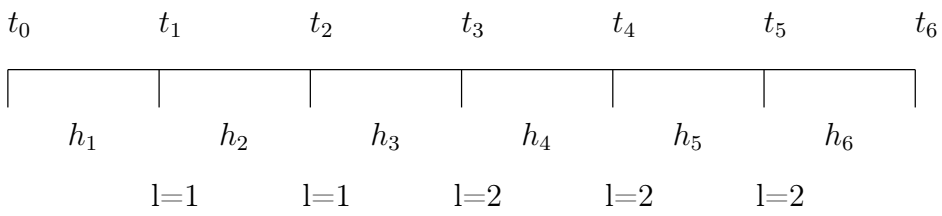
$$\begin{aligned}
\alpha_1^2 &= \frac{s_2 s_3}{(s_2 - s_1)(s_3 - s_1)} \\
\alpha_2^2 &= \frac{s_1 s_3}{(s_1 - s_2)(s_3 - s_2)} \\
\alpha_3^2 &= \frac{s_1 s_2}{(s_1 - s_3)(s_2 - s_3)}
\end{aligned}$$

Współczynniki s można jawnie zapisać jako.

$$\begin{aligned}
s_1 &= t_n - t_{n-1} = h_n = h_n + hs_{n-1}(0) \\
s_2 &= t_n - t_{n-2} = h_n + h_{n-1} = h_n + hs_{n-1}(1) \\
s_3 &= t_n - t_{n-3} = \underbrace{h_n}_{(t_n - t_{n-1})} + \underbrace{h_{n-1}}_{(t_{n-1} - t_{n-2})} + \underbrace{h_{n-2}}_{(t_{n-2} - t_{n-3})} = h_n + hs_{n-1}(2)
\end{aligned} \tag{6.34}$$

Jak widać do prawidłowego działania predykcji potrzebne są wartości sygnałów z poprzednich chwil czasowych. W przypadku startu analizy, gdy nie ma wyników analizy w odpowiedniej liczbie punktów, stosuje się predykcję niższego rzędu. W pierwszym punkcie predykcja nie jest wykonywana. Wartość początkowa brana jest z analizy punktu pracy lub może być zadana przez użytkownika. W miarę analizy kolejnych punktów rząd predykcji jest zwiększany aż do rzędu l wymaganego przez zastosowany schemat różnicowy. Ilustruje to rysunek 6.2.

[6] zobacz tablica sum kroków (6.16)



Rys. 6.2: Rząd l SR przy starcie w kolejnych punktach czasowych t_n.

6.5 Analiza czasowa

Klasyczna analiza czasowa umożliwia wyznaczenie odpowiedzi czasowej stanu nieustalonego sieci nieliniowej opisanej równaniami algebraiczno - różniczkowymi postaci

$$f(x, \dot{x}, t) = 0 \tag{6.35}$$

Proces rozwiązywania rozpoczyna się od obliczenia pochodnej $\dot{x}$, czyli tzw. dyskretyzacji za pomocą schematu różnicowego [J.O95] postaci

$$\dot{x}_n = \gamma\ x_n + d_{xn}$$

gdzie: d_x jest wektorem przeszłości, γ współczynnikiem zależnym od rodzaju schematu różnicowego. Otrzymamy w ten sposób układ nieliniowych równań algebraicznych

$$f(x_n, \gamma\ x_n + d_{xn}, t_n) = 0$$

Można go rozwiązać metodą Newtona-Raphsona przez linearyzację równań, tzn. rozwinięcie w szereg Taylora do wyrazów rzędu pierwszego.

$$\underbrace{\left[\frac{\delta f(x_n^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}{\delta x_{t_n}} + \gamma \frac{\delta f(x_{t_n}^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}{\delta \dot{x}_{t_n}}\right]}_{\frac{\delta f^{p-1}}{\delta x_n}} (x_n^p - x_n^{p-1}) =$$

$$= -\underbrace{f(x_n^{p-1}, \gamma\ x_n^{p-1} - d_{xn}, t_n)}_{f^{p-1}} \tag{6.36}$$

Po pogrupowaniu czynników i wprowadzeniu oznaczeń przyjmuje on postać równań rozwiązywanych iteracyjnie.

$$\underbrace{\frac{\delta f^{p-1}}{\delta x_n}}_{Y}\ {x_n}^p = \underbrace{-f^{p-1} + \frac{\delta f^{p-1}}{\delta x_n}\ x_n^{p-1}}_{B} \tag{6.37}$$

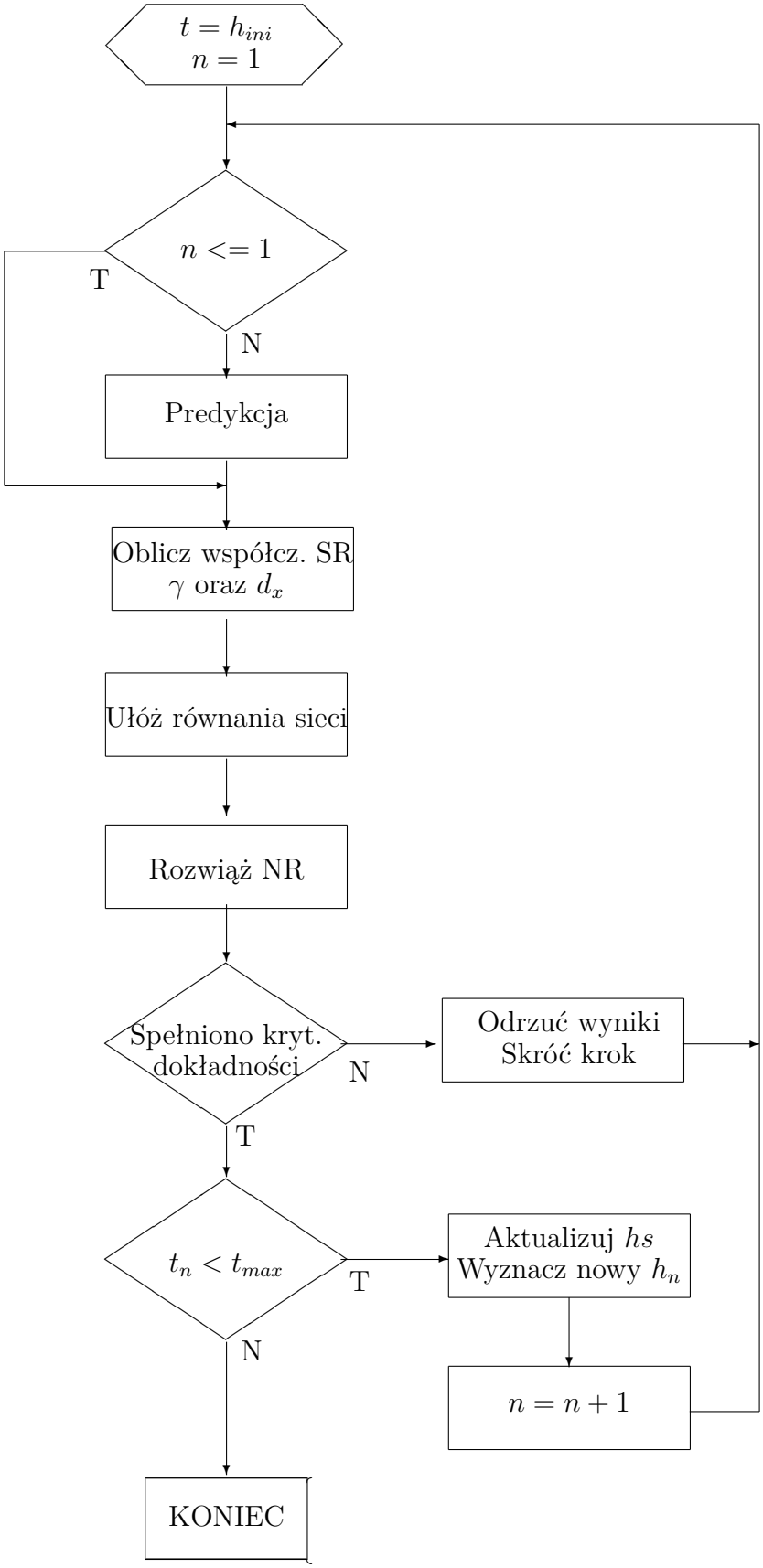
Ogólny algorytm analizy czasowej został przedstawiony w postaci graficznej na rys. 2.

Algorytm analizy czasowej 3 zastosowany w programie *Dero* jest dużo bardziej rozbudowany. Algorytm uwzględnia m.in.:

- właściwości równań sieci,
- możliwość doboru kroku czasowego całkowania,
- możliwość odrzucenia obliczeń w danym punkcie t_n, cofnięcie do poprzedniego punktu czasowego t_{n-1} i wykonanie całkowania z nowym, krótszym krokiem $h_n^{'}$,
- kontrolę dokładności rozwiązania po analizie w punkcie t_n,
- wykorzystanie schematu różnicowego Gear'a (rozdział 6.3) z doborem rzędu SR.

Na wydruku 3 przedstawiono uproszczony algorytm analizy czasowej zastosowany w symulatorze *Dero* . Szczegóły implementacyjne zostały pominięte dla większej czytelności algorytmu.

Algorytm 2 Algorytm analizy czasowej



Algorytm 3 Uproszczony algorytm analizy czasowej zastosowany w symulatorze *Dero* .

Require: x_0 ▷ wartości początkowe x
$t_n = 0$, $h_n = h_{ini}$ ▷ inicjalizacja czasu i kroku
$f(x, \dot{x}, t) = 0$ ▷ równania różniczkowe zwyczajne
while $t_n < t_{max}$ **do** ▷ pętla czasu
  if $t_n == t_{break}$ **then**
    $h_n = h_{min}$ ▷ pułapka czasowa
  end if ▷ dyskretyzacja równań - SR przed analizą w punkcie t_n
  oblicz współczynniki SR a_k i c_k na podstawie h_n i hs_{n-1}
  oblicz γ oraz d_{xn}
  $f(x_n, \gamma\, x_n + d_{xn}, t_n) = 0$ ▷ układ równań nieliniowych
  repeat ▷ pętla Newtona-Raphsona - linearyzacja równań
    ▷ układ równań liniowych
    $$\underbrace{\left(\frac{\delta f^{p-1}}{\delta x_n} + \frac{\delta f^{p-1}}{\delta \dot{x}_n}\right)}_{Y} {x_n}^p = \underbrace{-f^{p-1} + \left(\frac{\delta f^{p-1}}{\delta x_n} + \frac{\delta f^{p-1}}{\delta \dot{x}_n}\right) x_n^{p-1}}_{B}$$
    $Yx^{(p)} = B$ ▷ ułóż równania sieci np. metodą ZMPW
    $x^{(p)} = Y^{-1}B$ ▷ rozwiąż układ równań liniowych (LU)
  until zbieżność NR osiągnięta ▷ test stopu NR

  if włączona opcja POST **then** ▷ kontrola LTE po analizie
    oblicz LTE_{POST} ▷ po analizie
    if $LTE_{POST} > LTE_{USR}$ **then**
      odrzuć wyniki analizy w punkcie t_n
      wyznacz nowy krok h_n na podstawie wyznaczonego LTE_{POST}
    end if
  end if

  if wyniki analizy w punkcie t_n zaakceptowane **then**
    wyprowadź wyniki analizy w punkcie t
    ▷ SR po analizie w punkcie t_n
    aktualizuj DW_{n-1} na DW_n dla rzędu $k = min(l_{max}, l+1)$
    wyznacz iloraz $\frac{LTE}{LTE_{USR}}$ dla rzędów $1, \ldots, l$
    wyznacz $r_{LTE}(1, \ldots, l+1)$ dla każdego rzędu
    wybierz max mnożnik $r_{LTE} = max_i r_{LTE}(i)$ i nowy rząd $l_{n+1} = i$
    wyznacz nowy krok $h_{n+1} = r_{LTE} h_n$ z r_{LTE} (6.26)
    aktualizuj tablicę sum kroków hs_{n-1} na hs_n
    wyznacz iloraz $\frac{LTE}{LTE_{USR}}$ dla rzędów $1, \ldots, l$
    $t_{n+1} = t_n + h_{n+1}$ ▷ następny punkt
    $n = n + 1$ ▷ indeks następnego punktu t
    if $t_n + h_{n+1} > t_{break}$ **then**
      skróć krok aby wpaść w pulapkę czasową $h_{n+1} = t_{break} - t_n$
    end if
  end if
end while

6.6 Analiza czasowa stanu ustalonego

Analiza czasowa stanu ustalonego (*ang. steady state analysis*) ma za zadanie znalezienie takich warunków początkowych rozwiązywania równań sieci, które dają stan ustalony odpowiedzi, o ile istnieje. Załóżmy, że rozpatrujemy dynamiczną sieć nieliniową opisaną równaniami algebraiczno - różniczkowymi mającą rozwiązania przy warunkach początkowych $x_0 = x(0)$.

$$f(x, \dot{x}) = s(t)$$

Załóżmy, że w sieci istnieją pobudzenia okresowe $s(t)$ o tej samej częstotliwości f lub n wymuszeń o częstotliwościach, współmiernych $m_1 f_1 = m_2 f_2 = \ldots = m_n f_n$, co zapewnia ich superpozycje. W ogólności odpowiedź takiej sieci $x(t)$ jest nieokresowa, lecz może istnieć odpowiedź okresowa (stan ustalony), której okres T z góry można przewidzieć. Istnienie stanu ustalonego jest niepewne. Dowodzi się, że jeśli równania można sprowadzić do postaci stanowej $\dot{x} = f(x, t)$, gdzie f jest okresowa o okresie T (po prawej stronie równania istnieją okresowe wymuszenia), to istnieje stan ustalony o okresie T. Zanim stan ustalony zostanie osiągnięty w sieci mamy stan nieustalony.

Każdym warunkom początkowym $x_0 = x(t_0)$ w pewnej chwili t_0 odpowiada pewien przebieg rozwiązania $x(t)$, przybierający po okresie wartość $x(T)$. Wartość $x(T)$ można potraktować jako kolejne przybliżenie warunków początkowych. Odwzorowanie to nazwiemy odwzorowaniem podstawowym i oznaczymy symbolem Φ.

$$x(T) = \Phi(x_0) \tag{6.38}$$

Stan ustalony okresowy istnieje wtedy i tylko wtedy, gdy odwzorowanie podstawowe ma punkt stały x_0^*.

$$x_0^* = \Phi(x_0^*)$$

Odwzorowanie podstawowe Φ jest liniowe wtedy i tylko wtedy, gdy sieć jest liniowa. Można je wówczas wyrazić funkcją liniową:

$$x(T) = A(x_0 - x_0^*) + x_0^* \tag{6.39}$$

gdzie: A jest macierzą przejścia. Macierz przejścia przekształca różnice dwóch warunków początkowych po k-tym i $k+1$-szym okresie

$$\begin{aligned} x^{(k)}(T) &= A(x_0^{(k)} - x_0^*) + x_0^* \\ x^{(k+1)}(T) &= A(x_0^{(k+1)} - x_0^*) + x_0^* \end{aligned} \tag{6.40}$$

na różnice odpowiednich wyników po k-tym okresie:

$$x^{(k)}(T) - x^{(k+1)}(T) = A(x_0^{(k)} - x_0^{(k+1)}) \tag{6.41}$$

Po zastosowaniu liniowej aproksymacji przekształcenia podstawowego (6.38) w otoczeniu punktu stałego i zakładając ciągłość jego pochodnej otrzymamy postać podobna do (6.39).

$$\Phi(x_0^{(k)}(T) = x(T) = x_0^* + \frac{\delta\Phi(x_0^*)}{\delta x_0}(x_0^{(k)} - x_0^*) \tag{6.42}$$

Macierz pochodnych $\frac{\delta\Phi(x_0^*)}{\delta x_0}$ decyduje o sposobie dochodzenia do stanu ustalonego. Po odjęciu od obu stron równania (6.42) wektora $x_0^{(k)}$ otrzymujemy po przekształceniu zależność

$$\underbrace{\Phi(x_0^{(k)})}_{x^{(k)}(T)} - x_0^{(k)} = [\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1](x_0^{(k)} - x_0^*)$$

z której po wykonaniu oszacowania za pomocą normy spektralnej wynika, że jeżeli macierz pochodnych $\frac{\delta\Phi(x_0^*)}{\delta x_0}$ ma największa wartość własna bliska 1, to

$$||\Phi(x_0^{(k)}) - x_0^{(k)}|| << ||x_0^{(k)} - x_0^*||$$

Zatem różnica odpowiedzi na krańcach jednego okresu jest mała w porównaniu z odległością od rozwiązania ustalonego. Proces ustalania drgań będzie trwał przez wiele okresów. Warunki początkowe odpowiadające stanowi ustalonemu są zerem funkcji $f(x_0)$:

$$f(x_0) = \Phi(x_0) - x_0$$

Najprostszą metodą rozwiązania równania ($f(x_0) = 0$) jest metoda iteracji prostej, polegająca na konstruowaniu ciągu przybliżeń rozwiązania według reguły

$$x_0^{(k+1)} = \Phi(x_0^{(k)})$$

Oznacza to, że za warunki początkowe po k-tym okresie analizy sieci przyjmuje się wynik analizy po poprzednim $k-1$ okresie. Jest to zatem rozwiązywanie okres po okresie metoda rzędu pierwszego.

W praktyce stosuje się raczej rząd 2, tj. metodę Newtona-Raphsona (zobacz 6.2). Polega ona na linearyzacji funkcji $f(x_0)$ w otoczeniu przybliżenia $x_0^{(k)}$ warunków początkowych:

$$f(x_0) = \Phi(x_0^{(k)}) - x_0^{(k)} + [\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1](x_0^{(k+1)} - x_0^{(k)}) \tag{6.43}$$

Rozwiązując równanie $f(x_0) = 0$, po przekształceniach, otrzymamy:

$$[\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1]\ x_0^{(k+1)} = -(\Phi(x_0^{(k)}) - x_0^{(k)}) + [\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1]x_0^{(k)} \tag{6.44}$$

Jeżeli sieć i jej odwzorowanie podstawowe są liniowe, to krok newtonowski w sieci nieliniowej $(A-1)x_0^{(k+1)} = (A-1)x_0^*$ jest dokładnie tym samym co konstrukcja lokalnej aproksymacji liniowej sieci i jej odwzorowania podstawowego (6.39) oraz obliczenie stanu ustalonego jako punktu stałego (6.39).

W praktyce, jeśli linearyzacja nie jest zbudowana dokładnie w bieżącym punkcie, ale w różny sposób korzysta z punktów bliższych lub dalszych to mamy do czynienia z krokiem quasi-newtonowskim. Możliwe są zatem dwa rodzaje metod:

- identyfikacja wektora prawych stron i macierzy pochodnych na podstawie analizy sieci nieliniowej, co wymaga wykonania analizy wrażliwościowej zmiennych po czasie.

- identyfikacja przekształcenia podstawowego na podstawie przebiegu kilku punktów rozwiązania w odstępach okresu T.

Do drugiej z tych metod należy metoda Bukowskiego zastosowana w programie.

Metoda Bukowskiego Metoda [J.O95] polega na estymacji macierzy pochodnych $\frac{\delta\Phi(x_0^*)}{\delta x_0}$ odwzorowania podstawowego (6.42) dla każdej iteracji w sposób odpowiadający wielowymiarowej metodzie siecznych[7].

Algorytm wymaga inicjalizacji celem wstępnej identyfikacji zlinearyzowanego metodą siecznych odwzorowania podstawowego, tzn. wymaga n wektorów warunków początkowych $x_0^{(i)}$ o numerach i od $k-n$ do k oraz n wektorów wyników analizy $\Phi(x_0^{(i)})$ za okres wychodzące z wektorów $x_0^{(i)}$. Można je wyznaczyć na podstawie trzech wstępnych analiz czasowych za okres $(< 0, PER >)$.

Analizy za okres wychodzące z warunków początkowych x_0 spełniają odwzorowanie liniowe wtedy i tylko wtedy, gdy wektory wektory różnic warunków początkowych y

$$y^{(i)} = x_0^{(i)} - x_0^{(i-1)}$$

oraz wektory różnic z odpowiedzi układu po okresie wychodzących z warunków początkowych x_0

$$z^{(i)} = \Phi(x_0^{(i)}) - \Phi(x_0^{(i-1)})$$

powiązane są macierzą przejścia $A^{(k)}$, gdzie $i = k - n, k$ jest numerem analizy za okres.

$$A^{(k)} y^{(k)} = z^{(k)}$$

Dysponujemy n takimi związkami, co umożliwia ich połączenie przez wprowadzenie macierzy przyrostów $Y^{(k)}$, $Z^{(k)}$ i zapisanie układu n równań (znak $\|$ oznacza ustawienie kolumn obok siebie).

$$A^{(k)} \underbrace{[y^{(k-i)} \| \dots \| y^{(k)}]}_{Y^{(k)}} = \underbrace{[z^{(k-i)} \| \dots \| z^{(k)}]}_{Z^{(k)}}$$

Jak wspomniano wyznaczenie macierzy przejścia wymaga wyznaczenia $Y^{(k)}$ i $Z^{(k)}$. Jest to możliwe po wykonaniu $n+1$ analiz za okres. Analizy te można zastąpić trzema analizami za okres i wziąć pod uwagę tylko n końcowych punktów analizy w każdym okresie. Na tej podstawie można wyznaczyć wektory y, z i zbudować Y, Z. Liczba wziętych pod uwagę punktów n musi być równa liczbie istotnych zmiennych stanu. W pierwszych trzech okresach[8] otrzymamy:

- po pierwszym okresie - warunki początkowe $x_0^{(1)}$,

[7] Metoda siecznych to algorytm interpolacji liniowej. Polega na przyjęciu, że funkcja na dostatecznie małym odcinku $< a, b >$ w przybliżeniu zmienia się w sposób liniowy. Możemy wtedy na odcinku $< a, b >$ krzywą $y = f(x)$ zastąpić sieczną. Za przybliżoną wartość pierwiastka przyjmujemy punkt przecięcia siecznej z osią OX.

[8] Pierwsze 3 okresy analizy muszą być wykonywany przy tych samych danych wejściowych analizatora czasowego, aby nie zmieniać zachowania analizatora i nie zaburzać danych wejściowych analizatora stanu ustalonego. W tym celu należy np. zapamiętać wektor modułów wartości maksymalnych zmiennych przed pierwszą analizą i odtwarzać go przed każdą następną analizą za okres. Wektor ten wykorzystywany jest np. w teście stopu algorytmu NR i do wyznaczania kroku analizy czasowej.

- po drugim okresie - $x_0^{(1)}$, $x_0^{(2)}$, $\Phi(x_0^{(1)}) = x_0^{(2)}$, $y^{(2)} = x_0^{(2)}$ - $x_0^{(1)}$,
- po trzecim okresie - mamy: $x_0^{(2)}$, $x_0^{(3)}$, $\Phi(x_0^{(2)}) = x_0^{(3)}$, $z^{(2)} = \Phi(x_0^{(2)}) - \Phi(x_0^{(1)})$.

Po zakończeniu analiz wstępnych (po trzecim okresie) z kolejnych wektorów y, z konstruowane są przybliżone macierze Y i Z. Służą one do wyznaczenia pierwszego przybliżenia macierzy przejścia $A^{(0)}$ $(k = 0)$ odwzorowania podstawowego (6.41).

$$A^{(k)} = Z^{(k)}(Y^{(k)})^{-1}$$

Macierz A jest lokalną aproksymacją analizowanego zagadnienia. Jeżeli rozpatrywane punkty spełniają model liniowy, to macierze Y i Z są nieosobliwe i można zidentyfikować macierz przejścia odwzorowania liniowego $A^{(k)} = (Y^{(k)})^{-1}Z^{(k)}$ lokalnie aproksymując analizowane zagadnienie nieliniowe. Podstawiając macierz A do (6.43) na miejsce macierzy pochodnych $\frac{\delta\Phi(x_0^{(k)}}{\delta x_0}$, (6.43) stanie się lokalna aproksymacja liniową przechodzącą przez wprowadzone punkty. Po przekształceniach otrzymamy układ dwóch równań, który umożliwia wyznaczenie stanu ustalonego dla zidentyfikowanego liniowego przekształcenia podstawowego.

$$\begin{aligned} (Y^{(k)} - Z^{(k)})w &= x_0^{(k)} - \Phi(x_0^{(k)}) \\ x_0^{(k+1)} = x_0^{(k)} + Y^{(k)}w &= \Phi(x_0^{(k)}) + Z^{(k)}w \end{aligned} \tag{6.45}$$

Iteracje zaczynają się od wyznaczenia nowych warunków początkowych $x_0^{(k+1)}$ z (6.45). Na początku z pierwszego równania oblicza się w

$$w = (\underbrace{Y^{(k)} - Z^{(k)}}_{YZ})^{-1}(x_0^{(k)} - \Phi(x_0^{(k)}))$$

a następnie z drugiego $x_0^{(k+1)}$

$$x_0^{(k+1)} = \Phi(x_0^{(k)}) + Z^{(k)}w$$

Następnie wykonuje się jedną analizę za okres z wyznaczonych warunków początkowych $x_0^{(k+1)}$. Rozwiązanie sieci po okresie przyjmuje się za następne przybliżenie rozwiązania. Następnie sprawdza się test zakończenia iteracji przez porównanie warunków początkowych analizy z warunkami początkowymi z poprzedniej iteracji (6.46) i ewentualnie przerywa obliczenia, gdy wartość ta jest mniejsza od wartości zadanej (opcja $ERST$).

$$\frac{\|x_0^{(k)} - x_0^{(k-1)}\|}{\max_k(\|x_0\|)} < ERST \tag{6.46}$$

W przypadku kontynuacji na podstawie rozwiązania bieżącego oblicza się przyrosty

$$\begin{aligned} y^{(k+1)} &= x_0^{(k+1)} - x_0^{(k)} \\ z^{(k+1)} &= \Phi(x_0^{(k+1)}) - \Phi(x_0^{(k)}) \end{aligned}$$

i aktualizuje się tablice Y, Z przez zapomnienie najstarszych przyrostów $y^{(k-n)}$, $z^{(k-n)}$ i dopisanie nowo obliczonych przyrostów $y^{(k+1)}$, $z^{(k+1)}$. W następnym kroku wykonywana jest kolejna analiza za jeden okres. Sprawdzany jest test stopu. W przypadku kontynuacji wyznaczane są nowe warunki początkowe i cały proces iteracyjny

powtarza się. Analiza jest wykonywana do momentu uzyskania stanu ustalonego, tzn. do momentu, gdy wyznaczone warunki początkowe pokryją się z wynikami poprzedniej analizy z dokładnością względna nie gorsza niż $ERST$ (6.46).

Macierze Y i Z są aktualizowane w procesie iteracyjnym w ten sposób, że najstarsze wektory różnic y i z są zastępowane wektorami wyznaczonymi, natomiast macierz A jest wyznaczana na podstawie Y i Z.

$$A^{(k)} = Z^{(k)}(Y^{(k)})^{-1}$$

[9]

Omawiany algorytm Bukowskiego wymaga wykonania w jednej iteracji quasi-newtonowskiej jednej analizy czasowej za okres. Nakład na znalezienie nowych warunków początkowych, czyli rozwiązanie układu równań (6.45) wymaga wykonania

$$\frac{n^3}{3} - \frac{n}{3} + 2n^2$$

działań równoważnych mnożeniu i dzieleniu co odpowiada analizie w jednym dodatkowym punkcie czasowym.

Algorytm 4 Algorytm analizy czasowej stanu ustalonego.

Require: s ▷ liczba zmiennych stanu
Require: T ▷ okres analizy
Require: k=0 ▷ licznik okresów analizy stanu ustalonego (iteracji STDS)
1: **for** $k = 0 \ldots s$ **do**
2: wykonaj analizę czasową TR za okres (0...T) ▷ inicjalizacja algorytmu
3: **end for**
4: **repeat**
5: oblicz $x_0^{(k+1)}$
6: wykonaj analizę czasową TR za okres (0...T)
7: $x^{(k+1)} = \Phi(x^{(k)})$
8: oblicz test zakończenia iteracji STDS
9: **if** zakończ iteracje STDS **then**
10: $y^{(k+1)} = x_0^{(k+1)} - x_0^{(k)}$
11: $z^{(k+1)} = \Phi(x^{(k+1)}) - \Phi(x^{(k)})$
12: oblicz Y, Z
13: **end if**
14: **until** osiągnięto zbieżności pętli STDS
15: KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

[9] Dla układów generacyjnych wektor warunków początkowych jest rozszerzany o dodatkową zmienną - okres odpowiedzi T. Po każdym okresie wyznaczany jest nowy okres odpowiedzi $T^{(k)}$.

Algorytm 5 Algorytm analizy czasowej stanu ustalonego metodą Bukowskiego.

Require: MST, T_{MIN}, T_{MAX}, T ▷ liczba zmiennych stanu, zakres i okres analizy T
Require: $N_p = 400, h = T/N_p$ ▷ liczba punktów w okresie T i krok h analizy czasowej
1: **for** $k = 1 \ldots ITL7$ & $STOP_{STDS} == false$ **do** ▷ k-iteracja STDS ▷ Inicjalizacja algorytmu
2: $pe = N_p$ ▷ liczba punktów czasowych do końca okresu
3: $p = 1$, $pe = N_p$
4: **for** $p <= N_p$, $p++$, $p_e --$ **do** ▷ punkt czasowy analizy
5: **if** $== 1$ **then** ▷ Inicjalizacja algorytmu - 1. okres
6: $t_p = t_{min} + p * h$ ▷ punkt czasowy w 1 okresie analizy
7: **else**
8: $t_p = 0 + p * h$ ▷ punkt w kolejnym okresie T
9: **end if**
10: wykonaj analizę czasową TR w okresie do punktu t_p
11: **if** pe <= MST **then**
12: **if** k == 1 **then** ▷ pierwszy okres
13: zapamiętaj warunki początkowe $x_0^{(1)}$ punkcie t_p
14: **end if**
15: **if** k == 2 **then** ▷ drugi okres - zapamiętaj i oblicz w punkcie t_p
16: $x_0^{(1)}$, $x_0^{(2)}$
17: $\Phi(x_0^{(1)}) = x_0^{(2)}$
18: $y^{(2)} = x_0^{(2)}$ - $x_0^{(1)}$
19: **end if**
20: **if** k == 3 **then** ▷ trzeci okres - zapamiętaj i oblicz w punkcie t_p
21: $x_0^{(2)}$, $x_0^{(3)}$
22: $\Phi(x_0^{(2)}) = x_0^{(3)}$
23: $z^{(2)} = \Phi(x_0^{(2)}) - \Phi(x_0^{(1)})$.
24: **end if**
25: **if** k > 3 **then** ▷ następne okresy - zapamiętaj i oblicz w punkcie t_p
26: oblicz $x_0^{(k+1)}$ ▷ wzór (9.21)
27: $x^{(k+1)} = \Phi(x^{(k)})$
28: oblicz test zakończenia iteracji STDS
29: **if** zakończ iteracje STDS **then**
30: $y^{(k+1)} = x_0^{(k+1)} - x_0^{(k)}$
31: $z^{(k+1)} = \Phi(x^{(k+1)}) - \Phi(x^{(k)})$
32: oblicz Y, Z
33: **end if**
34: **end if**
35: **end if**
36: zapamiętaj wektor maksymalnych modułów zm. sieciowych $|x_{max}|$
37: **if** $\frac{|x_0^{(k)} - x_0^{(k-1)}|}{\max_k(|x_0|)} < ERST$ **then**
38: $STOP_{STDS} = true$ ▷ koniec - stan ustalony osiągnięty
39: **goto KONIEC**
40: **else** ▷ **stan ustalony nieosiągnięty - test stopu niespełniony**
41: **oblicz zmienne**
42: $y^{(k+1)} = x_0^{(k+1)} - x_0^{(k)}$
43: $z^{(k+1)} = \Phi(x_0^{(k+1)}) - \Phi(x_0^{(k)})$
44: **aktualizuj tablice Y, Z przez zapomnienie najstarszych przyrostów $y^{(k-n)}$, $z^{(k-n)}$ i dopisanie nowo obliczonych przyrostów $y^{(k+1)}$, $z^{(k+1)}$** ▷ **wyznacz nowe warunki początkowe $x_0^{(k+1)}$**
45: $w = (Y^{(k)} - Z^{(k)})^{-1}(x_0^{(k)} - \Phi(x_0^{(k)}))$
46: $x_0^{(k+1)} = \Phi(x_0^{(k)}) + Z^{(k)}w$
47: **end if**
48: **end for** ▷ **Koniec analizy w okresie**
49: **end for**
50: **KONIEC - Warunki początkowe dla stanu ustalonego wyznaczone**

6.7 Analiza czasowa kierowana zdarzeniami

Metoda analizy czasowej kierowanej zdarzeniami (*ang. Event-Driven Iterated Timing Analysis*) jest jedną z najbardziej efektywnych metod symulacji układów elektronicznych. Jest szczególnie efektywna w symulacji układów unilateralnych, tzn. o ustalonym kierunku przepływu sygnału. Idea opiera się o wykorzystanie relaksacyjnych technik rozwiązywania układów równań, gdzie każde równanie rozwiązywane jest osobno. W kolejnych relaksacjach otrzymywane jest kolejne przybliżenie rozwiązania dokładnego. Metoda została opublikowana przez A.R.Newton'a i A.L.Sangiovanni-Vincentelli w pracy [ARN83]. Metoda umożliwia analizę węzłów (równań) układu z indywidualnym krokiem czasowym i łatwe wykorzystanie techniki usypiania węzłów. W trakcie procesu analizy generowane są zdarzenia (definicja 6.7.1) opisane dwoma wielkościami [indeks węzła, czas analizy].

Definicja 6.7.1. *Zdarzeniem nazywamy proces wyznaczenia sygnału określonego węzła w określonym punkcie czasowym.*

Algorytm ED-ITA umożliwia bardzo efektywną symulację układów z elementami unilateralnymi. W programie zaimplementowano algorytm zmodyfikowany umożliwiający symulację układów analogowych z silnie sprzężonymi węzłami. Siła sprzężenia jest badana w trakcie analizy. W stosunku do algorytmu oryginalnego różni się on przede wszystkim:

- wprowadzeniem algorytmu blokowego analizy węzłów silnie sprzężonych wykorzystującego techniki bezpośrednie rozwiązywania układów równań,
- zastosowaniem grupowania węzłów silnie sprzężonych:
  - statycznego - zadanego przez użytkownika,
  - dynamicznego - wykonywanego w trakcie analizy,
- zastosowaniem dynamicznego rozgrupowywania grup węzłów silnie sprzężonych, także z podziałem na mniejsze grupy,
- wprowadzeniem algorytmu przenumerowania węzłów w trakcie analizy,
- wprowadzeniem synchronizacji czasów analiz węzłów zewnętrznych,
- wprowadzeniem synchronizacji czasów analizy węzłów - grup węzłów z synchronizacją czasu (grup T),
- zaimplementowaniem analizy modeli opisanych modelami behawioralnymi,
- wprowadzeniem mechanizmu cofania analizy w czasie,
- zastosowaniem tylko jednej listy zdarzeń i jednej podręcznej listy roboczej,
- wprowadzeniem algorytmu sukcesywnego uaktualniania listy zdarzeń,

Zdarzenia opisują zarówno propagację sygnału w układzie jak i dynamiczne zachowanie układu w trakcie analizy. Działanie oryginalnego algorytmu 6.7 opiera się na dwóch listach:

Do działania algorytmu 6 potrzebne są dwie listy, które w praktyce mogą być dynamicznymi tablicami:

- NEL - lista zdarzeń (*ang. Next Event List*),
- EQ - lista węzłów/grup węzłów, które mają zostać przeanalizowane w danej chwili czasowej.

Lista zdarzeń NEL przechowuje węzły i zapamiętuje czas ich analizy. Każdy punkt czasowy t_m odpowiada jednemu zdarzeniu. W danym punkcie czasowym pamiętane są indeksy węzłów i grup węzłów. W trakcie analizy węzły te są chronologicznie kopiowane z NEL do wektora roboczego EQ.

Do prawidłowego działania algorytmu potrzebne są:

- warunki początkowe dla analizy zadane przez użytkownika lub otrzymane w wyniku analizy punktu pracy,
- prawidłowe dane ustawione w strukturach bazy danych o układzie,
- ustawione wartości parametrów analizy.

Działanie algorytmu rozpoczyna się od zainicjalizowania tablic roboczych. Listy EQ i NEL są puste. Czas analizy zostaje ustawiony na 0. Na listę zdarzeń NEL z czasem $t = 0$ zostają wstawione węzły dołączone do niezależnych źródeł zewnętrznych.

- W linii 68 rozpoczyna się główna pętla czasowa algorytmu. Z wektora zdarzeń pobierane jest najbliższe zdarzenie o czasie t. Węzły związane z tym zdarzeniem przepisywane są do wektora roboczego EQ. Dalej posługujemy się wektorem EQ.

  Rozpoczynamy pętlę relaksacji - linia 9. Dla każdego węzła lub grupy SCC z EQ obliczamy napięcia węzłowe.

  Jeżeli węzeł nie należy do grupy SCC, to równanie dla węzła rozwiązujemy metodą relaksacyjną.

  Dla grupy węzłów rozwiązywany jest układ równań metodą bezpośrednią omówioną w rozdziale 9.1. Jeżeli nie uzyskano zbieżności, to na listę EQ wstawiane są chronologicznie węzły zależne (*ang. fanouts*). W tej samej relaksacji są one od razu analizowane. Jeżeli nie uzyska się dla nich zbieżności, to ich węzły zależne także wstawiane są na EQ. Tak więc już w pierwszej relaksacji na EQ mogą znajdować się wszystkie węzły, które odczuły zmianę sygnału. W przypadku układów bilateralnych w kolejnych relaksacjach na EQ mogą być kierowane dodatkowe węzły zależne, które odczuły zmianę sygnału na skutek istnienia sprzężeń zwrotnych. W przypadku niezbieżności choćby w jednym węźle wykonywana jest kolejna relaksacja, w której analizowane są tylko te węzły, w których nie uzyskano zbieżności lub zostały wstawione jako węzły zależne. W przypadku braku zbieżności w zadanej liczbie relaksacji k_{max} węzły, które nie uzyskały zbieżności są grupowane. Dla grupy ustawiany jest tzw. znacznik czasu życia grupy, który mówi co ile punktów czasowych analizy, grupę należy próbować podzielić na mniejsze lub próbować usunąć z niej węzły słabo sprzężone. Jest to realizowane po ułożeniu układu równań dla grupy i wyzerowaniu znacznika czasu życia grupy.

- Po zakończeniu pętli relaksacji wszystkie analizowane węzły znajdują się na liście EQ. Flaga *conv* określa, czy została osiągnięta zbieżność. Jeśli nie, to wykonywany jest krok czasowy wstecz do ostatniego punktu analizy. Krok czasowy zostaje skrócony do kroku minimalnego h_{min}. Węzły i grupy zostają wstawione na listę

Algorytm 6 Opracowanego algorytmu analizy czasowej kierowanej zdarzeniami

Require: K, α, α_1, $k_{max}, T_{gmax}, t = 0$, warunki początkowe
Inicjalizacja listy NEL i EQ
Wstaw węzły źródeł zewnętrznych na NEL z czasem $t = 0$ ▷ czas początkowy
repeat ▷ pętla zdarzeń
  pobranie z NEL zdarzenia o czasie t
  skopiowanie węzłów z NEL(t) na EQ
  dla każdego węzła w z EQ wyznaczenie wartości startowej v_0
  $k = k_{max}$, $conv = NIE$ ▷ inicjalizacja licznika relaksacji i flagi zbieżności
  $FlagaGrupowania$ = NIE
  for $k > 0 \& conv == NIE$ **do** ▷ pętla relaksacji
    $conv = NIE$ ▷ flaga kasowana przez test stopu
    for każdego węzła w z EQ **do** ▷ pętla węzłów
      if węzeł w "niezbieżny" **then**
        if węzeł w należy do grupy g **then**
          if znacznik czasu życia grupy T_g został wyzerowany **then**
            ułóż układ równań dla grupy g
            zastosuj algorytm grupowania dla grupy g
            ustaw znacznik czasu życia grupy $T_g = T_{gmax}$
          end if
          rozwiąż układ równań dla grupy g ▷ grupa węzłów
          $T_g = T_g - 1$
          kieruj węzły zależne do analizy
        else ▷ pojedynczy węzeł
          rozwiąż równanie dla węzła w
          if test stopu dla węzła w niespełniony **then**
            ustaw flagę braku zbieżności dla węzła $conv(w) = NIE$
            ustaw globalną flagę braku zbieżności $conv = NIE$
            kieruj węzły zależne do analizy
          end if
        end if
        zapamiętaj wartości z poprzedniej relaksacji
      end if
    end for ▷ koniec pętli węzłów
    if $k == k_{max} \& conv == NIE$ **then** ▷ brak zbieżności
      if $FlagaGrupowania ==$ NIE **then** ▷ cofanie i grupowanie
        grupuj węzły "niezbieżne"
        wyznaczenie punktu czasowego dla alg. cofania t dla grupy
        $FlagaGrupowania$ = TAK, $k_{max} = k_1$
        zeruj czas życia grupy $T_g = 0$
      else ▷ skrócenie kroku dla węzłów "niezbieżnych"
        for każdego węzła "niezbieżnego" w z listy EQ **do**
          $h = (t - t_{last})/2$ ▷ nowy krok
          $t_{next} = t_{last} + h$
          $t = MIN(t, t_{last})$ ▷ punkt, do którego należy się cofnąć
        end for
      end if ▷ brak zbieznosci, odrzuć, krok wstecz
      wstaw węzły "niezbieżne" na NEL(t)
      wykonaj kroku wstecz
    end if ▷ koniec pętli relaksacji
  end for
  przewidywanie nowych kroków analizy dla węzłów z EQ
  redukcja liczby zdarzeń
  synchronizacja czasów analizy grup SCC i T
  generacja zdarzeń
  wydruk wyników analizy
  zerowanie listy EQ
  pobranie czasu t najbliższego zdarzenia z NEL
until $t < t_{stop}$& NEL niepusta ▷ koniec pętli zdarzeń
KONIEC

zdarzeń NEL z czasem $t = t_{last}+h_{min}$. Wyniki analizy zostają odrzucone i analiza rozpoczyna się ponownie w punkcie $t = t_{last} + h_{min}$.

W przypadku słabej zbieżności grupowanie może być wielokrotnie wykonywane, co w skrajnym przypadku prowadzi do zgrupowania wszystkich węzłów układu i do zmiany analizy kierowanej zdarzeniami na zwykłą analizę czasową.

- Jeżeli po wyjściu z pętli relaksacji wszystkie węzły osiągnęły zbieżność, to należy wykonać predykcję nowych zdarzeń przez wyznaczenie następnych czasów analiz dla poszczególnych węzłów lub grup i wstawieniu ich w odpowiednie miejsce listy NEL.

  Dla każdego węzła z EQ predykowany jest indywidualny krok czasowy i wyznaczany jest czas następnej analizy. Dla grup wyznaczany jest wspólny krok czasowy będący najmniejszym spośród wszystkich przewidywanych kroków czasowych dla węzłów wchodzących w skład grupy (synchronizacja czasów analiz węzłów grup).

  Węzły/grupy umieszczane są na liście zdarzeń NEL z wyznaczonym wcześniej czasem analizy. Aby nie generować zdarzeń nadmiarowych i nie powodować nadmiernego skracania kroku analizy, wykorzystywany jest algorytm ograniczania liczby zdarzeń.

- Po wygenerowaniu zdarzeń aktualizowane są wektory przeszłości. Predykowany krok czasowy i czas ostatniej analizy zapamiętywane są indywidualnie dla każdego węzła/grupy węzłów. Jest to koniec analizy zdarzenia o czasie t. W tym miejscu następuje zapis wyników w zbiorze wyjściowym.

- W kolejnym kroku następuje czyszczenie listy EQ i usunięcie zdarzenia z listy NEL.

- Jeśli warunek kontynuacji analizy jest spełniony, tj. lista zdarzeń nie jest pusta i czas następnego zdarzenia mieści się w przedziale analizy, to następuje skok na początek pętli czasu. W przeciwnym przypadku proces analizy kończy się.

6.8 Małosygnałowa analiza częstotliwościowa

Rozważmy sieć opisaną kanonicznym równaniem algebraiczno-różniczkowym postaci (6.47) przy wymuszeniach sinusoidalnych o pulsacji ω.

$$f(x, \dot{x}, t) = s(t) \tag{6.47}$$

Równania tego typu, przy ustalonych warunkach początkowych maja rozwiązanie stanowiące sumę składowej swobodnej (stan przejściowy) i wymuszonej. Składowa swobodna jest odpowiedzią sieci na niezerowe warunki początkowe przy zerowych wymuszeniach. Dla sieci z założenia stabilnych składowa ta zanika do zera (stan przejściowy). Pozostaje składowa wymuszona, która dla wymuszeń sinusoidalnych $s(t)$ o pulsacji ω jest sygnałem sinusoidalnym o pulsacji ω, o amplitudzie zależnej od parametrów sieci opóźnionym w fazie w stosunku do wymuszeń. Po zaniknięciu odpowiedzi swobodnej (stanu nieustalonego) pozostaje niezależny od warunków początkowych sinusoidalny stan ustalony, zwany odpowiedzią zmiennoprądową *ang. alternate current response* (AC). Proces obliczania amplitud i faz odpowiedzi wymuszonej w poszczególnych punktach sieci w funkcji częstotliwości nazywany jest małosygnałową analizą częstotliwościową (*ang. AC analysis*). Najważniejszą metodą obliczania odpowiedzi sieci jest rozwiązywanie pomocniczej sieci immitancyjnej. Metoda ta oparta jest na transformacji algebraiczno-różniczkowych równań kanonicznych sieci oryginalnej w dziedzinę wartości symbolicznych. Otrzymane w ten sposób równania algebraiczne liniowe, o współczynnikach zespolonych zależnych od pulsacji ω, opisują składową wymuszoną rozwiązania równań sieci przy wymuszeniach sinusoidalnych (odpowiedź zmiennoprądowa). Moduł tego rozwiązania jest amplitudą, a argument - fazą przebiegu sinusoidalnego w sieci oryginalnej. Po transformacji symbolicznej równania opisują sieć strukturalnie identyczną z siecią oryginalną, o tej samej topologii, zwaną siecią immitancyjną.

Podstawą małosygnałowej analizy częstotliwościowej AC (*ang. Alternate Current*) jest założenie dostatecznie małych przyrostów składowych zmiennych na tle odpowiedzi statycznej sieci. Przy takim założeniu, po zastąpieniu nieliniowych funkcji gałęziowych modelami zlinearyzowanymi w punkcie pracy, gałęzie mogą być opisane zespolonymi modelami liniowymi. Sieć zlinearyzowana będąca siecią zastępczą poddawana jest transformacji z dziedziny czasu do dziedziny $j\omega$. W wyniku transformacji, sieć zastępcza staje się przyrostową, co powoduje w konsekwencji wyzerowanie wymuszeń stałoprądowych.

Analiza AC ma sens w przypadku, gdy w układzie zdefiniowano wymuszenia zespolone, których wydajność opisana jest amplitudą (MAGN) i fazą (PHASE). Amplituda (MAGN) takiego wymuszenia może być dowolna, co jest konsekwencją przyjętego modelu liniowego sieci.

6.8.1 Równania sieci

Rozważmy równania sieci nieliniowej postaci (6.47) zapisane w postaci zawierającej po prawej stronie wymuszenia $s(t)$. Odpowiedź stałoprądowa sieci jest rozwiązaniem równania

$$f(x_0, 0, 0) = s_0 \tag{6.48}$$

gdzie pochodne w równaniu zastąpiono zerami, a wymuszenia - stałymi s_0. Praca z małymi sygnałami przyrostowymi odpowiada przybliżeniu równania (6.47) w otoczeniu odpowiedzi stałoprądowej. Rozwijając obie strony równania (6.47), traktowane

jako funkcje zmiennych $(x, \dot{x}, s)$, w szereg Taylora z dokładnością do wyrazów rzędu pierwszego w otoczeniu odpowiedzi stałoprądowej (x_0, s_0) otrzymuje się

$$f(x_0, 0, 0) + \frac{\delta f}{\delta x}(x - x_0) + \frac{\delta f}{\delta \dot{x}}(\dot{x} - 0) = s_0 + [s(t) - s_0] \tag{6.49}$$

gdzie $\frac{\delta f}{\delta x}$, $\frac{\delta f}{\delta \dot{x}}$ są odpowiednimi pierwszymi pochodnymi funkcji f względem odpowiednich zmiennych x, $\dot{x}$ liczonymi w punkcie $(x_0, 0, 0)$.

W zlinearyzowanym równaniu (6.49) składniki rzędu zerowego są takie same jak w równaniu kanonicznym (6.48) i można je zredukować odejmując oba równania stronami.

$$\frac{\delta f}{\delta x}(x - x_0) + \frac{\delta f}{\delta \dot{x}}(\dot{x} - 0) = [s(t) - s_0]$$

Wprowadzając oznaczenie dla składowych przyrostowych $\hat{x} = x - x_0$ można zapisać równania w postaci (6.50).

$$\frac{\delta f}{\delta x}\hat{x} + \frac{\delta f}{\delta \dot{x}}\frac{d\hat{x}}{dt} = \hat{s}(t) \tag{6.50}$$

Otrzymane równanie liniowe stanowi opis kanoniczny sieci identyczny strukturalnie z siecią oryginalną (6.47). Gałęzie sieci powstały przez zlinearyzowanie gałęzi oryginalnych w otoczeniu rozwiązania stałoprądowego i zastąpienie wymuszeń zmiennych wymuszeniami przyrostowymi. Zmiennymi gałęziowymi sieci są zmienne przyrostowe odpowiednich oryginalnych zmiennych gałęziowych. Otrzymana sieć jest siecią małoprzyrostową.

Małosygnałowa analiza zmiennoprądowa sieci oryginalnej sprowadza się do analizy zmiennoprądowej sieci małoprzyrostowej (6.50) w stanie ustalonym przy sinusoidalnych wymuszeniach przyrostowych.

Po transformacji sieci immitancyjnej w dziedzinę $j\omega$ powstaje układ immitancyjny małoprzyrostowy postaci (6.51).

$$\left(\frac{\delta f}{\delta x} + j\omega\frac{\delta f}{\delta \dot{x}}\right)\bar{x} = \bar{s} \tag{6.51}$$

$\bar{x}$ oznacza zespolone przyrostowe zmienne gałęziowe sieci niosące informację o amplitudzie i fazie odpowiedzi sinusoidalnej, $\bar{s}$ zespolone sinusoidalne wymuszenia przyrostowe o zadanej amplitudzie i fazie.

6.9 Optymalizacja deterministyczna

Zdefiniujmy wektor parametrów projektowych $x = [x_1, \ldots, x_N]$, gdzie N oznacza liczbe parametrów projektowych, a $x_1, \ldots, x_i, \ldots, x_N$ są poszczeglnymi parametrami projektowymi[10].

Zdefiniujmy M funkcji układowych $y = f(x) = f_1(x), \ldots, f_j(x), \ldots, f_M(x)$ będące odpowiedziami układu w dziedzinie parametrów projektowych x zależnymi od tych parametrów projektowych. Zależność może być liniowa lub nieliniowa.

Zdefiniujmy M specyfikacji projektowych[11] $s = s_1, \ldots, s_M$ nałożonych na funkcje układowe $f(x) = f_1(x), \ldots, f_j(x), \ldots, f_M(x)$.

Zadanie optymalizacji polega na znalezieniu takich wartości parametrów projektowych x, dla których odpowiedź układu $y = f(x)$ przyjmie wartość zadaną przez specyfikacje projektowe s, tzn. $f(x) = s$.

W celu zrealizowania zadania optymalizacyjnego należyzbudować funkcję celu $FC(x)$, która jest konstruowana jest jako suma kwadratów różnic pomiędzy wartościami obliczonymi a zadanymi przez użytkownika.

$$FC(x) = ||s - f(x)||^2$$

Optymalizacja polega na znalezieniu minimum funkcji celu $FC(x)$

$$\min_{x \in R^N} |FC(x)| = \min_{x \in R^N} ||s - f(x)||^2$$

w dziedzinie parametrów projektowych x :

Ze względu na fakt, że funkcja celu zawiera składniki względne i bezwzględne, funkcję celu $FC(x)$ zapiszemy jako funkcję budowaną na podstawie zadanych specyfikacji względnych i bezwzględnych, jako ważona suma kwadratów różnic pomiędzy wartościami obliczonymi $f_i(x)$, a zadanymi s.

$$FC(x) = \sum_{i=1}^{k} \left(w_i \ \frac{f_i(x) - s_i}{s_i} \right)^2 + \sum_{j=k+1}^{k+l} [w_j \ (f_j(x) - s_j)]^2$$

gdzie: k - jest liczbą względnych specyfikacji projektowych, l - jest liczbą bezwzględnych specyfikacji projektowych, $f(x)$ - oznaczają odpowiedź układu (wartość obliczona), s - oznaczają wartości specyfikacji projektowych (wartości oczekiwane), w - oznacza wagę danej specyfikacji.

Zadanie optymalizacji sprowadza się do minimalizacji $FC(x)$ w dziedzinie parametrów projektowych x.

$$\min_{x \in R^N} |FC(x)|$$

gdzie: N jest liczbą zmiennych projektowych, x jest wektorem parametrów projektowych, $FC(x)$ jest funkcją budowaną na podstawie zadanych specyfikacji względnych i bezwzględnych.

W programie parametry projektowe poddawane są transformacji na wewnętrzne zmienne optymalizacji z. Transformacjom podlegają zmienne projektowe z ograniczonym jak i z nieograniczonym zakresem zmienności:

[10] Parametry projektowe ustawiane są dyrektywą '.VARP'

[11] Specyfikacje projektowe ustawiane są dyrektywą .SPEC

- dla i-tego parametru o nieograniczonym zakresie zmienności:

$$z_i = \frac{x_i - x_{i,0}}{scale_i}$$

Transformacja odwrotna wykonywana jest na podstawie zmiennej wewnętrznej x_{int}.

$$x_i = x_{i,0} + scale_i \; z_i$$

- dla i-tego parametru z ograniczonym zakresem zmienności:

$$z_i = \frac{x_{i,max} - x_{i,min}}{2 \; scale_i} \arcsin\left(\frac{2x_i - (x_{i,max} + x_{i,min})}{x_{i,max} - x_{i,min}}\right)$$

Transformacja odwrotna uzyskiwana jest po wyznaczeniu z powyższego równania x_i.

$$x_i = \frac{(x_{i,max} - x_{i,min})}{2} \sin\left(\frac{2z_i \; scale_i}{x_{i,max} - x_{i,min}}\right) + \frac{(x_{max,} + x_{i,min})}{2}$$

- transformacje zmiennych, których wartość leżypoza zakresem $< x_{min}, x_{max} >$:

dla $x_i < x_{i,min}$

$$\begin{aligned} z_i &= -\frac{\pi}{2}\frac{(x_{i,max} - x_{i,min})}{2scale_i} \\ x_i &= \frac{(x_{i,max} - x_{i,min})}{2} \sin\left(\frac{2z_i scale_i}{x_{i,max} - x_{i,min}}\right) + \frac{(x_{i,min} + x_{i,max})}{2} \end{aligned}$$

dla $x_i > x_{max}$

$$\begin{aligned} z_i &= \frac{\pi}{2}\frac{(x_{i,max} - x_{i,min})}{2scale_i} \\ x_i &= \frac{(x_{i,max} - x_{i,min})}{2} \sin\left(\frac{2z_i scale_i}{x_{i,max} - x_{i,min}}\right) + \frac{(x_{i,min} + x_{i,max})}{2} \end{aligned}$$

Dla przetransformowanych zmiennych zadanie optymalizacji można zapisać jako:

$$\min_{z \in R^N} |FC(z)|$$

Zadanie optymalizacji (optymalizacja w sensie najmniejszych kwadratów) polega na takiej modyfikacji wektora z, aby funkcja celu $FC(z)$ osiągnęła minimum (być może lokalne) w dziedzinie parametrów projektowych $x \; R^N$.

Szukanie minimum odbywa się przez iteracyjne wyznaczanie nowych wartości z^{k+1} dających minimalizację funkcji celu. Można to zapisać jako sekwencję generowanych wartości:

$$z^{(k+1)} = z^{(k)} + r^{(k)} d_U^{(k)}$$

gdzie: r jest współczynnikiem kroku, d_U jest tzw. wektorem poprawy - krokiem analizy, tzn. $d_U^{(k)}$ jest rozwiązaniem nieograniczonego zadania optymalizacji. Po wyznaczeniu nowych wartości z^{k+1} stosowana jest transformacja odwrotna celem wyznaczenia rzeczywistych parametrów układu x. Dla tych parametrów wykonywane są odpowiednie analizy celem wyznaczenia nowego wektora d_U.

W programie zastosowano algorytm Levenberga-Marquardta [K.M04] w wersji ze współczynnkimem tłumienia λ.

$$(J_k^T J_k + \lambda I) d_U = -J_k^T FC(z^k)$$

Z równania wyznaczany jest wektor d_U

$$d_U = -(J_k^T J_k + \lambda I)^{-1} J_k^T FC(z^k) \tag{6.52}$$

gdzie: J jest Jakobianem[12] - macierzą pierwszych pochodnych składowych funkcji celu F względem wewnętrznych zmiennych projektowych, λ jest współczynnikiem tłumienia algorytmu (*ang. damping parameter*) i nazywany jest parametrem Marquardta.

Wektor d_U jest tzw. wektorem poprawy i dla metody Levenberga-Marquardta oznaczymu go dla ustalenia uwagi $d_{LM} = d_U$. Dla $\lambda = \infty$ kierunek poprawy Levenberga-Marquardta d_{LM} przechodzi w kierunek najszybszego spadku d_{NS}. d_{NS} zapewnia jednak powolną zbieżność obliczeń - wbrew nazwie. Wektor d_U dany jest wtedy zależnością:

$$d_{NS} = -J^T FC(z^k) \tag{6.53}$$

W celu zapobieżenia omawianemu zjawisku można zastąpić macierz jednostkową I poprzez macierz składającą się z elementów diagonalnych $diag(J^T J)$. Wtedy algorytm Lavenberga-Marquardta przyjmuje postać

$$(J_k^T J_k + \lambda\ diag(J^T J)) d_U = -J_k^T FC(z)$$

Wektor d_U ma wtedy postać

$$d_U = -(J_k^T J_k + \lambda\ diag(J^T J))^{-1} J_k^T FC(z^k) \tag{6.54}$$

Proces optymalizacji można zapisać w postaci algorytmu 7 lub algorytmu 8.

W programie *Dero* zastosowano algorytm 9.

6.9.1 Wybrane elementy algorytmu

Obliczenie pochodnej funkcji celu względem parametrów projektowych

Pochodne składowych funkcji celu względem zmiennych przeskalowanych zmiennych projektowych z_i wyrażone są wzorem

$$\frac{\delta F_i}{\delta z_j} = \frac{[F_i(z_1, z_2, , \ldots, z_j + \Delta z, \ldots, z_N) - F_i(z_1, z_2, \ldots, z_j, \ldots, z_N)]}{\Delta z}$$

Pochodne funkcji celu względem wewnętrznych zmiennych projektowych dane są zależnością:

[12] Do estymacji Jakobianu stosuje się SR 1 rzędu

Algorytm 7 Pełny algorytm optymalizacji

1: Szacowanie Jakobianu J
2: Wykonanie pojedynczej analizy układu
3: Wykonanie N analiz układu zaburzonego
4: Wyznaczenie wektora kierunku poprawy d_U oraz współczynnika kroku r.
5: Obliczenie zmiennych optymalizacji na podstawie znajomości wektora d_U i współczynnika r
6: Obliczenie funkcji celu
7: Uaktualnienie Jakobianu
8: **if** test stopu optymalizacji niespełniony **then**
9: idź do 1
10: **end if**

Algorytm 8 Algorytm optymalizacji metodą Lavenberga-Marquardta

1: Wybierz $x^0 \in X$, $\lambda > 0$, $k = 0$
2: (START)
3: **if** $F(x^k)! = 0$ **then**
4: $(S.2)$ Wybierz $J_k \in R^{m \times n}$,
5: $\lambda_k = \lambda ||F(x^k)||^2$,
6: oblicz d_U^k z (6.54)
7: $(S.3)$ $x^{k+1} = P_X(x^k + x^{k+1})$, $k = k + 1$, idź do 2
8: **end if**
9: STOP

$$\frac{\delta F}{\delta z} = \frac{\delta F}{\delta z} \frac{\delta z}{\delta x}$$

Dla małych z składnik $\frac{\delta z}{\delta x} \approx scale$. Umożliwia to ręczną kontrolę poziomu pochodnych. Najkorzystniejsza z punktu widzenia algorytmu jest sytuacja, gdy pochodne co do modułu mają zbliżone wartości[13].

Estymacja Jakobianu

W celu wykonania estymacji Jakobianu J należywykonać jedną analizę układu oryginalnego i wyznaczyć $FC(z)$ oraz N analiz układu zaburzonego $FC(z_i + STIN)$, przy czym w każdej analizie zaburzana jest tylko jedna zmienna z_i. Estymuj Jakobian J przy pomocy schematu różnicowego pierwszego rzędu

$$J = \frac{FC(z_1, \ldots, z_i + STMIN, \ldots, z_N) - FC(z)}{STMIN}$$

Wyznaczanie wektora kierunku poprawy d_U

Do wyznaczenie wektora kierunku poprawy d_U (kroku) stosuje się przybliżenie Powell'a (Metoda Powell'a,Metoda Powell'a - Wikipedia) celem redukcji nakładów obliczeniowych. Uzyskiwany w ten sposób kierunek poprawy d_U jest pośredni pomiędzy kierunkiem najszybszego spadku, a kierunkiem Levenberga-Marquardta. Mnożnik kroku r jest

[13] Zmianę wartości uzyskuje się przez zadanie odpowiednich współczynników skalowania w linii dyrektywy '.VAR' - 4.4.2

Algorytm 9 Algorytm optymalizacji zastosowany w programie.

Require: M ▷ liczba specyfikacji projektowych
Require: N ▷ liczba zmiennych projektowych (Z)
Require: $STMIN, STMAX$ ▷ minimalny i maksymalny krok optymalizacji
Require: Z ▷ wewnętrzne zmienne projektowe $Z = trans(X)$
Require: $EOPT$ ▷ względna dokładność optymalizacji - (tab. 4.1)
1: $FC = FC(Z)$ ▷ funkcja celu FC
2: $FC_{EPS} = M * EOPT^2$ ▷ Dokładność obliczeń funkcji celu
3: $COUNTER_{FC} = N + 2$ ▷ maksymalna liczba obliczeń FCbez poprawy
4: $\tau_{inc} = 1$ ▷ mnożnik kroku optymalizacji
5: (START)
6: Oblicz $F_0 = F(Z)$ i funkcje celu $FC_i = \sum_{i \in 1..N} F(z_i)^2$
7: Wykonaj N obliczeń $F_d = F(Z + STMIN)$
8: Oblicz $J = \frac{F_d - F_0}{STMIN}$
9: **if** $\min_{i \in 1..N} FC_i < FC_{min}$ **then**
10: $FC_{min} = FC_i$
11: $Z_{BEST} = Z$ ▷ zapamiętaj najlepsze Z
12: $F_{BEST} = F$ ▷ zapamiętaj F
13: **end if**
14: **if then** $\min_i FC_i < FC_{EPS}$
15: Skocz do 33
16: **else**
17: $Z = Z_{BEST}$ ▷ brak poprawy więc odtwórz poprzednie wartości
18: **end if**
19: oblicz J^{-1}
20: oblicz wektory DZ_{MARQ} i DZ_{NS}
21: przewiduj wartość FC_{pred} możliwą do uzyskania w następnym kroku
22: **if** $FC_{pred} < FC_{EPS}$ **then**
23: Skocz do 33
24: **end if**
25: wybierz parametr Marquardta i mnoznik kroku τ_{inc}
26: **if** $COUNTER_{FC} == 0$ **then**
27: (BŁĄD) Przekroczono maksymalną liczbę obliczeń FCbez poprawy
28: Skocz do 33
29: **else**
30: $COUNTER_{FC}--$ ▷ zmniejsz licznik obl. FC
31: Skocz do 5
32: **end if**
33: (KONIEC)

ograniczany. Opcje STIN oraz STMAX umożliwiają ograniczenie wartości początkowej oraz maksymalnej r.

6.10 Wyznaczanie obszarów sprawności

6.10.1 Oznaczenia

W rozdziale zastosowano następujące oznaczenia:

p - parametr rzeczywisty,

$f_p = f(p)$ - funkcja zależna od p,

x - parametr zespolony zależny od p,

$x(p)$ - funkcja układowa zależna od p,

i, j, k - w indeksach dolnych to identyfikatory parametrów p dla odpowiednich kierunków wyznaczania obszarów sprawności.

6.10.2 Podstawy teoretyczne

W niniejszej rozdziale zostaną omówione teoretyczne podstawy techniki jednowymiarowych przeszukiwań ortogonalnych ODOS (*ang. One Dimensional Ortogonal Search*) wykorzystującą analizę biliniowej funkcji układowej. Wynikiem działania jest wyznaczony obszar sprawności przedstawiony na rys. 6.6. W symulatorze *Dero* zastosowano modyfikację techniki ODOS i wprowadzono trzeci parametr zmienny, co umożliwiło wyznaczanie przestrzeni sprawności w trzech wymiarach - rys. 6.7.

6.10.3 Biliniowa funkcja układowa

Rozpatrzmy klasę obwodów elektrycznych spełniających następujące założenia:

1. sieć typu SLS (Skupiona, Liniowa, Stacjonarna),
2. sieć w stanie ustalonym przy wymuszeniu stałoprądowych lub sinusoidalnych,
3. opis sieci elektrycznej realizowany jest za pomocą odpowiedzi stałej oraz odpowiedzi amplitudowej i fazowej w dziedzinie częstotliwości. Wielkości te są funkcjami funkcjami układowymi w dziedzinie parametrów wejściowych (R,L,C ...)
4. w skład sieci mogą wchodzić następujące elementy:
   - elementy bierne R,L,C,
   - osiem typów źródeł sterowanych, w tym źródła sterowane pochodną sygnału,
   - idealny wzmacniacz operacyjny
   - autonomiczne źródła napięcia i prądu.
5. Ponadto zakłada się, że sieć opisana jest układem równań liniowych o współczynnikach zespolonych, ułożonych za pomocą zmodyfikowanej metody potencjałów węzłowych (ZMPW).

Równanie sieci można zapisać jako:

$$Yx = B \tag{6.55}$$

gdzie: Y - macierz kwadratowa sieci, x - wektor kolumnowy niewiadomych, B - wektor wymuszeń.

Jeżeli założymy, że $detY \neq 0$, to wówczas istnieje jednoznaczne rozwiązanie postaci (6.56).

$$x = Y^{-1}B \tag{6.56}$$

Dla ustalonej pulsacji ω i ustalonej topologii sieci zdefiniujmy przestrzeń zespolonych parametrów wejściowych sieci $x = [x_i, \ldots, x_m] \in X_p$ opisujących elementy macierzy ZMPW tej sieci. Macierz ta jest liniowo zależna od tych elementów. W praktyce częściej posługujemy się przy projektowaniu rzeczywistymi parametrami sieci $p = [p_i, \ldots, p_k] \in P_p$. Parametrami rzeczywistymi sieci są wartości pojemności C, indukcyjności L, rezystancji R, dobroci Q itp. Każdemu parametrowi zespolonemu x_i może być przyporządkowany jeden lub więcej parametrów rzeczywistych p_i. Ogólnie można to zapisać w postaci (6.57).

$$x_i = x_i(p) \tag{6.57}$$

Np. jeżeli $y_c = j\omega C$ to $x_i = y_c = x_i(C)$ i $p = C$. Jeżeli we wzorze (6.56) B jest stałe (wymuszenia stałe) to wzór (6.58) opisuje zespoloną funkcją układową w przestrzeni rzeczywistych parametrów sieci P_p.

$$X(x(p)) = X(p) = Y^{-1}B \tag{6.58}$$

Skalarna zespolona funkcja układowa jest każdą ze składowych wektorowej funkcji układowej $X(p)$ lub ich kombinacją liniową mającą sens fizyczny $X(p) = x_k(p) - x_l(p)$. Analogicznie w przestrzeni P_x określa się $x(x)$ lub kombinację liniową $x(x) = x_k(x) - x_l(x)$.

Twierdzenie 6.10.1. *(Twierdzenie o biliniowości) Jeżeli dla dowolnej składowej wektora parametrów x_i macierz Y zawiera co najwyżej cztery pozycje liniowo zależna od x_i lub B zawiera co najwyżej dwie pozycje liniowo zależna od x_i, to funkcja układowa jest biliniową względem x_i, tzn. jest ilorazem funkcji liniowych*

$$X(p) = \frac{n(p)}{d(p)} = \frac{n_0 + n_l p}{d_0 + d_l p} \tag{6.59}$$

ze współczynnikami zależnymi od p. Opis przyrostowy opis biliniowej funkcji układowej (wzór Fidlera) w otoczeniu nominalnego punktu p_0 oparty na tzw. trzech współczynnikach (3C) wrażliwościowych. Twierdzenie to stosuje się do funkcji układowych przy założeniu, że pozostałe parametry p oprócz p_k są ustalone. W praktyce stosuje się przyrostowy opis biliniowej funkcji układowej w otoczeniu nominalnego punktu p_0 oparty na tzw. trzech współczynnikach wrażliwościowych (3C).

$$x(p_i) = x(p_{i0}) + \frac{x_i'(p_i - p_{i0})}{1 + q_i(p_i - p_{i0})} \tag{6.60}$$

gdzie:

$$\begin{aligned} x_0 &= x(p_{i0}) \\ x'_i &= \frac{\delta x(p_{i0})}{\delta p_{i0}} \\ q_i &= \frac{\delta x'_i}{\delta p_i} \end{aligned}$$

ze współczynnikami zależnymi od p, gdzie $x(p)$ to składowa wektora X lub ich kombinacja liniowa.

Obliczenie 3C dla przekroju sprawności

Obliczenie przekroju obszaru sprawności w dwóch kierunkach wymaga obliczenia w kolejnych punktach p_j zmodyfikowanych 3C w każdym z punktów p_i wg. wzorów modyfikacyjnych.

$$\begin{aligned} \hat{x}_0 &= x(p_{i0}) + \frac{x'_j(\Delta p_j)}{1 + q_j(\Delta p_j)} \\ \hat{x}_i &= x'_i + \frac{\Delta p_j[x''_{ij} + \Delta p_j(x''_{ij}q_j - x'_j q_{ji})]}{1 + q_j(\Delta p_j)} \\ \hat{q}_i &= q_i + \frac{q'_{ji}\Delta p_j}{1 + q_j(\Delta p_j)} \\ \hat{p}_{j0} &= p_{j0} + \Delta p_j \end{aligned}$$

Do obliczenia 3C potrzebnych jest tzw. siedem współczynników wrażliwościowych (7C)

$$\begin{aligned} x_0 &= x_{out} \\ x'_i &= x_i \frac{\Delta x_0}{\Delta f_i} \\ x'_j &= x_j \frac{\Delta x_0}{\Delta f_j} \\ q_i &= \frac{-\Delta x_i}{\Delta f_i} \\ q_j &= \frac{-\Delta x_j}{\Delta f_j} \\ x''_{ij} = x''_{ji} &= x_j \frac{\Delta x_i}{\Delta f_j}\frac{\Delta x_i}{\Delta f_i} + x_i \frac{\Delta x_j}{\Delta f_j}\frac{\Delta x_j}{\Delta f_i} \\ q''_{ij} = q''_{ji} &= -q_i q_j \end{aligned}$$

gdzie: x_i, x_j są odpowiednimi wartościami sygnałów na wyjściach elementów, $f_i = f_i(p_i)$ jest wartością wyjścia elementu zależnym od parametru p_i, $f_j = f_j(p_j)$ jest wartością wyjścia elementu zależnym od parametru p_j.

Współczynniki 3C oblicza się raz w punkcie nominalnym, a następnie przesuwając się w kierunku p_j oblicza się zmodyfikowane 3C wrażliwościowe funkcji układowej.

W analizie komputerowej do obliczenia 3C wymagane jest obliczenie tzw. siedmiu współczynników 7C. Wymaga to wykonania trzech analiz sieci: jednej analizy sieci oryginalnej i dwóch analiz sieci stowarzyszonych z parametrami p. Wzory do obliczenia 7C zostały podane dla obliczonych w przestrzeni zespolonych parametrów sieci, co dobrze pasuje do analizy sieci przy pomocy równań ułożonych metodą ZMPW (rozdział 7). W praktyce projektowej nie posługujemy się jednak parametrami zespolonymi, lecz parametrami rzeczywistymi sieci, takimi jak np. wartość rezystancji czy pojemności. Dlatego też 7C należy obliczyć w przestrzeni parametrów rzeczywistych (7CP). Ze względu na liniową zależność p od Y i B przejście do przestrzeni parametrów rzeczywistych można opisać zależnością biliniową w postaci:

$$p_i = p_{i0} + \frac{p'_i(p_i - p_{i0})}{1 + qq_i(p_i - p_{i0})} \tag{6.61}$$

Współczynniki we wzorze (6.61) obliczane są z modelu elementu na podstawie zależności (6.62).

$$\begin{array}{rcl} p_{i0} & = & f_i(p_{i0}) \\ p'_i & = & \frac{\delta f_i(p_{i0})}{\delta p_i} \\ qq_i & = & \frac{\delta p'_i}{\delta p_i} \approx \frac{\Delta p'_i}{\Delta p} \end{array} \tag{6.62}$$

Dzięki zależności (6.61) możliwe jest obliczenie 7CP w przestrzeni P gdy znamy 7C obliczonych w przestrzeni X. Wzory modyfikacyjne umożliwiają przejście z przestrzeni X do przestrzeni parametrów rzeczywistych P. Współczynniki 7CP po lewej stronie odnoszą się do przestrzeni parametrów rzeczywistych P.

$$\begin{array}{rcl} x'_i & = & p'_i x'_{ix} \\ q'_i & = & q'_i q' + qq_i \\ x''_{ij} & = & p''_j p''_j x''_{ijx} \\ q'_{ji} & = & p'_i q'_i q''_{ji} \end{array}$$

Algorytm obliczania 7C

Sposób obliczania 7C wrażliwościowych w przestrzeni X ($7CX$) przedstawia algorytm 10. Obliczenie 7C wymaga wykonania 3 analiz: jednej układu pierwotnego i dwóch układów zastępczych z wartościami współczynników p_i, p_j.

6.10.4 Analiza biliniowej funkcji układowej

W praktyce najczęściej bada się rzeczywiste funkcje układowe modułu i fazy zespolonej biliniowej funkcji układowej postaci (6.60), które można otrzymać z $X(p)$. Jeżeli $X(p_i)$ oznacza funkcję skalarną np. składową wektora $X(p)$ to kwadrat modułu zespolonej biliniowej funkcji układowej (ZBFU) można wyrazić wzorem (6.63).

$$\|x(p_i)\|^2 = \|x_0 + [x(p_i) - x_0]\|^2 \tag{6.63}$$

Wyznaczając z (6.60) różnicę $x(p_i) - x_0$ i podstawiając do (6.63) otrzymujemy wzór (6.64) na kwadrat modułu zespolonej biliniowej funkcji układowej (ZBFU) wyrażony przy pomocy trzech współczynników (3C).

Algorytm 10 Algorytm obliczania 7CX

1: Utwórz macierz Y
2: Utwórz wektor wymuszeń oryginalnych B
3: Rozwiąż $Yx = B$
4: Zapamiętaj wyniki
5: Utwórz wektor wymuszeń zastępczych skojarzonych z parametrem p_j
6: Rozwiąż $Yx = B_j$
7: Zapamiętaj wyniki
8: Utwórz wektor wymuszeń zastępczych skojarzonych z parametrem p_i
9: Rozwiąż $Yx = B_i$
10: Zapamiętaj wyniki
11: Na podstawie wyników oblicz wartości 7C
12: Dokonaj transformacji 7C do 7CP
13: Zapisz w pamięci wartości 7CP

$$\Delta p_i = p_i - p_{i0} \tag{6.64}$$

$$\|x(p_i)\|^2 = \|x_0\|^2 + \frac{a_1\Delta p_i + a_2\Delta p_i}{a_3\Delta p_i + a_4\Delta p_i + 1} \tag{6.65}$$

gdzie współczynniki a wyrażone są przy pomocy tzw. trzech współczynników (3C : x_0,x'_i, q_i).

$$\begin{aligned} a_1 &= \|x'_i\|^2 + 2Re(x_0^* x'_i q_i^*) \\ a_2 &= 2Re(x_0^* x'_i) \\ a_3 &= \|q_i\|^2 \\ a_4 &= 2Re(q_i) \end{aligned} \tag{6.66}$$

Analiza biliniowej funkcji układowej ma podstawowe znaczenie w projektowaniu układów liniowych, gdyż jest bardzo dobrą metodą analitycznego opisu obszaru sprawności R_s.

Definicja 6.10.1. *Obszarem sprawności R_s nazywany zbiór wartości parametrów p_i, dla których sieć elektryczna spełnia postawione jej wymagania techniczne.*

Wyznaczenie biliniowej funkcji układowej umożliwia wyznaczenie aproksymacji odcinkowo-liniowej obszaru sprawności, co stanowi istotę techniki ODOS.

Ograniczenia projektowe

Wyznaczenie obszaru sprawności wymaga zdefiniowania ograniczeń projektowych względem których obliczane są obszary sprawności. Technika ODOS akceptuje trzy typy ograniczeń projektowych:

1. ograniczenia stałoprądowe dla $\omega = 0$ (DC - analiza stałoprądowa) - rys. 6.4

$$S_L^2(\omega) \leq ||x(p, j\omega)||^2 \leq S_U^2(\omega) \tag{6.67}$$

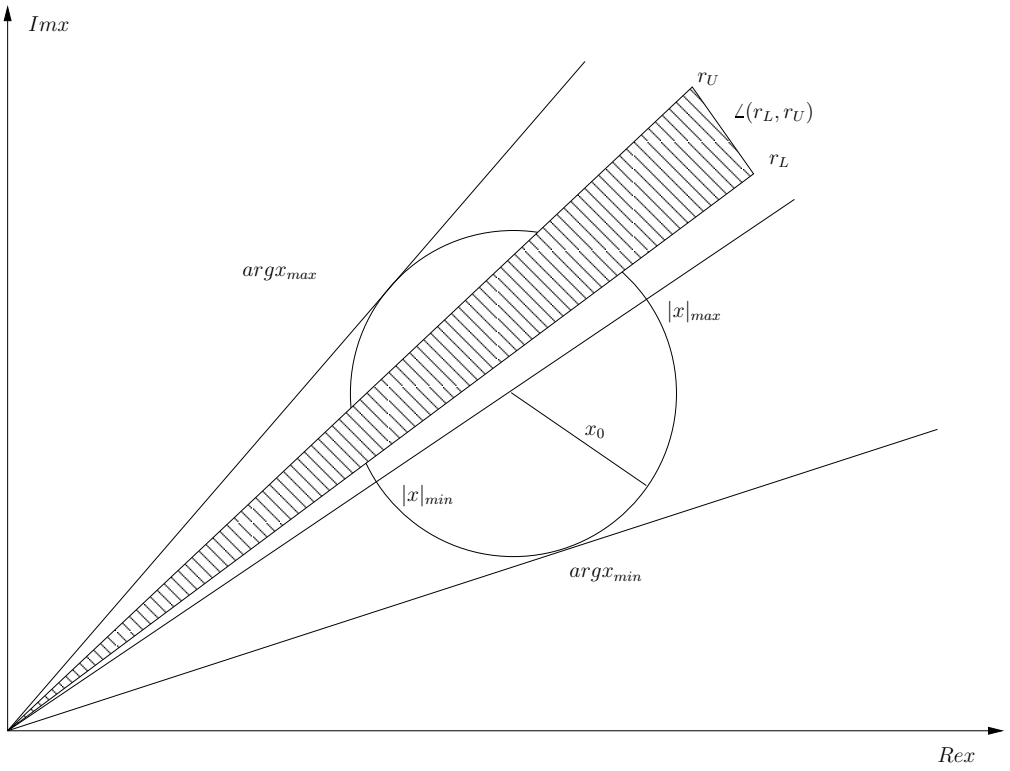
$$S_L^2 \leq x(p, 0) \leq S_U \tag{6.68}$$

2. ograniczenia amplitudowe i fazowe nałożone na charakterystyki amplitudowe i fazowe w dyskretnych punktach częstotliwości (AC - analiza częstotliwościowa) - rys. 6.3

$$x(p, j\omega) \in \angle(r_L, r_U) \tag{6.69}$$

gdzie : S_L, S_U są ograniczeniami dolnym i górnym nałożonym na charakterystykę amplitudową, $x(p, j\omega)$ jest zespoloną biliniową funkcją układową sieci, a $\angle(r_L, r_U)$ reprezentuje stożek na płaszczyźnie zespolonej wyznaczony przez dwie proste r_L,r_U określone przez dolne i górne ograniczenie nałożone na charakterystykę fazową.

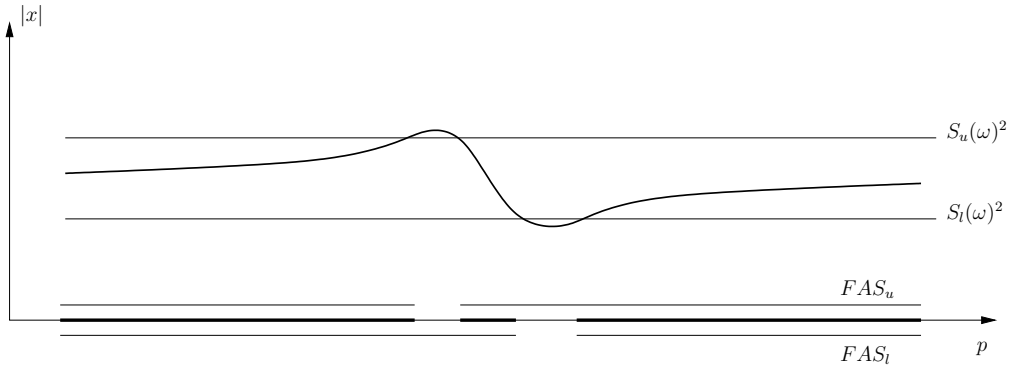
Funkcja układowa $x(p, j\omega)$ spełnia górne i dolne ograniczenia fazowe, gdy leży w stożku $\angle(r_L, r_U)$ (rys. 6.3).



Rys. 6.3: Interpretacja graficzna biliniowej funkcji układowej i ograniczeń fazowych na płaszczyźnie zespolonej dla $\xi > 0$.

W projektowaniu komputerowym wzory (6.67,6.68,6.69) wykorzystywane są do wyznaczania rodzin odcinków sprawności, przy pomocy których aproksymuje się obszary sprawności. Jeżeli założymy, że wszystkie składowe wektora rzeczywistych parametrów sieci przyjmują wartości nominalne p_0 oprócz jednego parametru p_i, to $x(p)$ opisuje funkcję układową jednej zmiennej, której dziedzina jest ograniczona do prostej przechodzącej przez punkt p_0 i równoległej do osi p_i.

$$\mathrm{LM_i(p_0) = \{p \in R^m p_k = p_{k0} \text{ dla } k = 1, ..m \text{ i } k \neq i\}} \tag{6.70}$$



Rys. 6.4: Przebieg kwadratu modułu biliniowej funkcji układowej z ograniczeniami amplitudowymi i elementarnymi rodzinami odcinków sprawności dla ograniczenia dolnego i górnego.

Wyznaczona część wspólną $LM_i(p_0)$ i obszaru sprawności R_s nazywa się rodziną odcinków sprawności (FAS). Rodzina odcinków sprawności jest w praktyce ograniczona do pewnego zakresu $[p_{min}, p_{max}]$ zmienności parametru p, może być pusta, lub zawierać skończoną liczbę odcinków. W zastosowaniach praktycznych mamy do czynienia z wieloma różnymi ograniczeniami projektowymi dla wielu pulsacji ω i dla różnych ograniczeń (fazowych i amplitudowych). Jeżeli obliczymy poszczególne rodziny odcinków sprawności odnoszące się do ograniczeń projektowych, to możliwe jest obliczenie rodziny wynikowej jako części wspólnej wszystkich rodzin odcinków sprawności.

Obliczania rodzin odcinków sprawności dla AC

Dla ustalonej ω_k i ograniczeń amplitudowych w analizie AC można obliczyć dwie rodziny odcinków sprawności FAS_{iL}, FAS_{iU} dla dolnego i górnego ograniczenia. Biorąc pod uwagę wzór (6.67) otrzymujemy dwie nierówności dla ograniczenia dolnego (6.71) i górnego (6.72) (rys. 6.4).

$$\mathrm{S_L^2}(\omega_k) - ||\mathrm{x(p, j}\omega_k)||^2 \leq 0 \tag{6.71}$$
$$||\mathrm{x(p, j}\omega_k)||^2 - \mathrm{S_U^2}(\omega_k) \leq 0 \tag{6.72}$$

Nierówności te można rozwiązać jako nierówność kwadratową (6.73).

$$c_2(p_i - p_{i0})^2 + c_1(p_i - p_{i0}) + c_0 \leq 0 \tag{6.73}$$

Współczynniki dla nierówności (6.71) dane są wzorami:

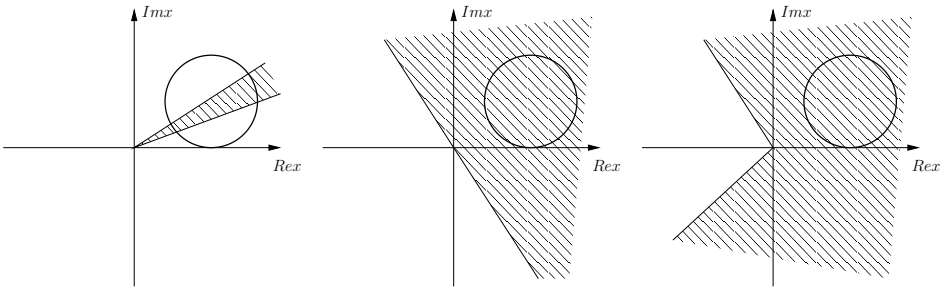
$$\begin{aligned} c_2 &= (S_L^2(\omega_k) - ||x_0||^2)a_3 - a_1 \\ c_1 &= (S_L^2(\omega_k) - ||x_0||^2)a_4 - a_2 \\ c_0 &= S_L^2(\omega_k) - ||x_0||^2 \end{aligned} \tag{6.74}$$

Współczynniki dla nierówności (6.73) dane są wzorem:

$$\begin{aligned} c_2 &= (||x_0||^2 - S_U^2(\omega_k))a_3 + a_1 \\ c_1 &= (||x_0||^2 - S_U^2(\omega_k))a_4 - a_2 \\ c_0 &= ||x_0||^2 - S_U^2(\omega_k) \end{aligned} \tag{6.75}$$

Współczynniki $a_1 \dots a_4$ są to współczynniki występujące we wzorze (6.66).

Dla ograniczeń fazowych rodziny elementarne znajdujemy w sposób podobny jak dla ograniczeń amplitudowych. Dla ustalonej ω_k należy rozwiązać dwie nierówności wynikające z zależności (6.69). Należy uwzględnić trzy przypadki kształtu obszaru ograniczeń na płaszczyźnie zespolonej. Kształt obszaru niech opisuje liczba $\xi = \angle(r_U, r_L^*)$. Jest to sinus kąta w stożku jak na rys. 6.3. Wygląd obszaru w zależności od wartości ξ przedstawiony został na rys. 6.5.



(a) $\xi > 0$ - stożek jest wypukły (b) $\xi = 0$ - półpłaszczyzna (c) $\xi < 0$ - stożek wklęsły

Rys. 6.5: Graficzna ilustracja ograniczeń fazowych na płaszczyźnie zespolonej.

.

Biorąc pod uwagę zależność (6.69) dla przypadku $\xi > 0$ otrzymujemy układ nierówności:

$$-Im(r_L)Re(x) + Re(r_L)Im(x) \geq 0 \tag{6.76}$$
$$Im(r_U)Re(x) - Re(r_U)Im(x) \geq 0 \tag{6.77}$$

Nierówności tę można rozwiązać tak jak nierówność kwadratową (6.73) dla nierówności (6.77) ze współczynnikami

$$\begin{aligned} c_2 &= Im[r_L q_i (x_0 q_i + x_i')^*] \\ c_1 &= Im[r_L (2(Re q_i) x_0 + x_i')^*] \\ c_0 &= Im[r_L x_0^*] \end{aligned} \tag{6.78}$$

Dla nierówności (6.77) w (6.78) należy zamienić r_L na r_U i zmienić znaki.

Obliczania rodzin odcinków sprawności dla DC

Dla ograniczeń nałożonych na odpowiedź stałoprądową (dla DC) należy rozwiązać dwie nierówności według wzoru (6.68).

$$S_L - x(p, 0) \leq 0 \tag{6.79}$$
$$x(p, 0) - S_U \leq 0 \tag{6.80}$$

Wyznaczając $x(p, o)$ ze wzoru 6.60 otrzymujemy dwie biliniowe nierówności typu

$$\frac{e_0 + e_1(p_i - p_{i0})}{1 + q_i(p_i - p_{i0})} \leq 0 \tag{6.81}$$

ze współczynnikami,

$$\begin{aligned} e_1 &= (S_L - x_0)q_i - x_i' \\ e_0 &= S_L - x_0 \end{aligned} \tag{6.82}$$

dla nierówności 6.80.

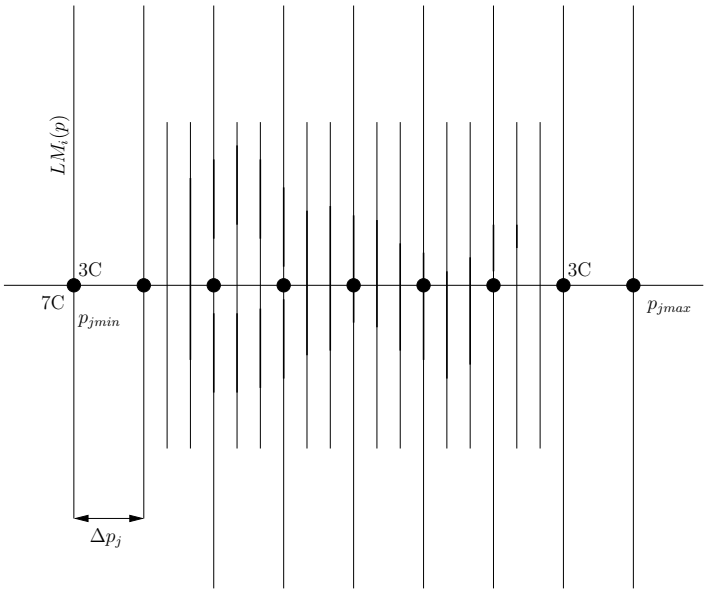
Dla nierówności 6.80 należy zamienić S_L z S_U i zmienić znaki. Mianownik nierówności 6.81 jest różny od zera, więc nierówność tą można rozwiązać tak jak nierówność kwadratową 6.73. W praktyce projektowej jednocześnie aktywnych jest wiele ograniczeń, co implikuje istnienie jednocześnie wielu rodzin odcinków sprawności. Rodzina wynikowa jest iloczynem mnogościowy rodzin elementarnych.

Wyznaczanie aproksymacji odcinkowo-liniowej obszarów sprawności

Definicja Jeśli przez $N = [p_0^1, p_0^p]$ *oznaczymy zbiór punktów w przestrzeni* R_m *i wybierzemy podzbiór* P *zbioru liczb naturalnych* $1, \ldots, m$*, to rodziny odcinków sprawności liczone w punktach* $k \in P$*, w kierunku* p_j*, będziemy nazywać aproksymacją odcinkowo liniową przekroju sprawności (SA).*

$$\mathrm{SA}_P(N) = \bigcup_{k=1}^{l} \bigcup_{i \in P} \mathrm{FAS}_i(p_p^{(k)}) \tag{6.83}$$

Z powyższej definicji wynika, że aproksymacja odcinkowo-liniowa wyznaczana jest dla przypadku dwóch parametrów zmiennych. Działanie algorytmu wyznaczania obszarów sprawności można łatwo wyjaśnić posługując się rys. 6.6.



Rys. 6.6: Aproksymacja odcinkowo-liniowa obszaru sprawności.

Ustalamy zbiór punktów z zakresu zmienności parametru p_j. Dla ustalonej pulsacji i ustalonej wartości parametru p_j wykonujemy analize ODOS z punktu nominalnego p_0 w kierunku p_i. Następnie przesuwamy się wzdłuż osi p_j do następnego punktu

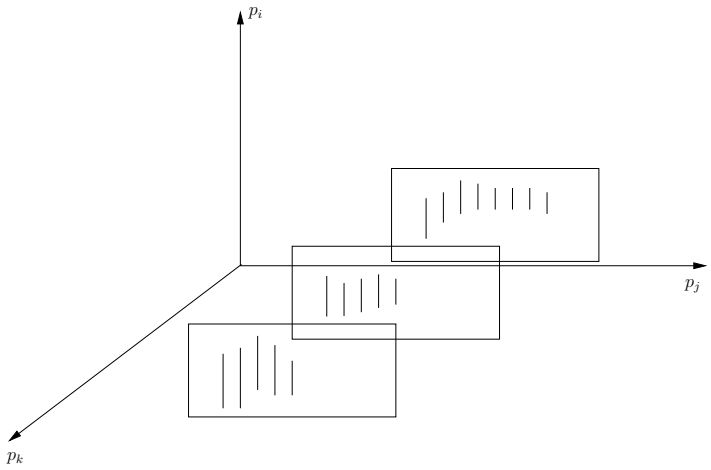
i ponownie wykonujemy analizę ODOS w kierunku p_i. W ten sposób wyznaczana jest aproksymacja odcinkowo-liniowa obszaru sprawności.

Dobór kroku z jakim przesuwamy się w kierunku p_j umożliwia redukcję nakładów obliczeń. Początkowo wykonujemy wstępną analizę z dużym krokiem, a następnie zmniejszamy krok i zawężamy zakres zmienności p_j wg wzoru

$$\Delta\hat{p_j} = \sqrt{[(p_{j_{max}} - p_{j_{min}})\Delta p_j} \tag{6.84}$$

Wyznaczanie aproksymacji odcinkowo-liniowej przestrzeni sprawności

Wprowadzając trzeci parametr p_k i wykonując algorytm 10 w kolejnych punktach p_k można wyznaczyć aproksymacje odcinkowo-liniową trójwymiarowego obszary sprawności 3D, tak jak zostało to przedstawione na rys. 6.7. Wyznaczanie obszarów sprawności w 3D zostało zaimplementowane z programie *Dero* .



Rys. 6.7: Trójwymiarowa aproksymacja odcinkowo-liniowa obszaru sprawności po wprowadzeniu parametru zmiennego p_k

Rozdział 7

Równania sieci w różnych dziedzinach

Do symulacji mikrosystemów wykorzystuje się dedykowane programy symulacji działające w wielu różnych dziedzinach[1] [E.S98a][LD98][E.S98b], ze względu na różne techniki symulacji. Na przykład techniki analityczne i techniki elementów skończonych są użyteczne w symulacji mikrostruktur [Duy94][S.C91]. Konieczność symulacji układów złożonych, w których poszczególne części pochodzą z różnych środowisk była powodem rozwoju specjalizowanych symulatorów (lata 80-te). Doprowadziło to do powstania szeregu systemów symulacji takich jak: MEMCAD, SENSM, CAPSIM, CAEMEMS-D, SENSOR [B+92], NM/SESES. Techniki symulacji sieci elektrycznych zastosowano do symulacji analogowych układów elektronicznych. Techniki analizy kierowanej zdarzeniami zastosowano do symulacji układów cyfrowych, natomiast techniki mieszane (*ang. mixed-mode*) do symulacji układów analogowo-cyfrowych. Szybki rozwój algorytmów symulacji sieci elektrycznych doprowadził do ich udoskonalenia i zastosowania w symulacji elementów z różnych dziedzin, w tym także sterowników. Było to możliwe dzięki opracowaniu dwóch podstawowych technik:

- analogii elektryczno-mechanicznej znanej jako metoda obwodów równoważnych ; Polega ona na znalezieniu takiego schematu elektrycznego, który opisany jest tymi samymi równaniami co modelowany mikrosystem (sterownik). Podstawową wadą tej metody jest konieczność linearyzacji elementów wokół punktów pracy.
- modelowanie behawioralne (języki HDL) pozwalające na symulację układu opisanego jawnie równaniami. Podstawowym ograniczeniem jest tu składnia języka.

Symulacja na poziomie mikrosystemów wymaga wykonania wielu symulacji na poziomie systemu, w którego skład wchodzą makromodele mikrourządzeń razem z elementami lub modelami części elektronicznej. Z powodu dużej różnorodności elementów stworzone zostało szereg technik opisujących zależności pomiędzy nimi. Należą do nich:

- metoda uogólnionych zmiennych,
- technika obwodów równoważnych,
- wykorzystanie języków opisu sprzętu.

[1] Np. elektrycznej, mechanicznej, termicznej, chemicznej, ...

Najniższym poziomem opisu zachowania urządzeń są makromodele elementów modelu tworzone na potrzeby symulacji. Posiadają one następujące cechy:

- opis analityczny umożliwia projektantowi łatwe modelowanie zaprojektowanych urządzeń,
- opisują prawidłowo zależności geometrii i właściwości fizycznych,
- opisują zachowanie dynamiczne i quasi-statyczne,
- wyrażone są w prostej formie równań (także różniczkowych) lub obwodów równoważnych,
- są łatwe do zastosowania na poziomie symulacji systemu.

W celu umożliwienia łatwej symulacji układów należących do różnych środowisk (*ang. nature*) wprowadzono zmienne uogólnione. W tabeli 7.1 zamieszczono zmienne uogólnione dla wybranych środowisk. Podejście to umożliwia tworzenie obwodów równoważ-

Tab. 7.1: Uogólnione zmienne dla kilku wybranych środowisk.

Zmienne uogólnione	Środowiska			
	Elektryczne	Przepływy cieczy	Mechaniczne	Obroty
e wymuszenie (effort)	v napięcie $[V]$	P ciśnienie	f siła	τ moment obrotowy
f przepływ (flow)	i prąd $[A]$	ϕ przepływ obj.	V przyśpieszenie	ω przyśpieszenie kątowe
p pęd (momentum)	ψ strumień $[Wb]$	p_p pęd ciśnienia	p pęd	P_t pęd
p położenie (state)	q ładunek $[C]$	V objętość	x przesunięcie	Θ kąt
W energia (energy)	$\int_q v dq$, $\int_\psi v d\psi$	$\int_V PdV$, $\int_{P_p} \phi dP_p$	$\int_x F dx$, $\int_p V dx$	$\int_\phi \tau d\phi$, $\int_{P_t} \omega dP_t$
P moc (power)	$v(t)i(t)$	$P(t)\phi(t)$	$F(t)V(t)$	$\tau(t)\omega(t)$

nych akceptowanych przez programy analizy układów elektronicznych i łatwe przeniesienie modeli do innych symulatorów. Modele te muszą spełniać zasadę zachowania energii. Jest to łatwe do spełnienia gdy model opisany jest językiem HDL-A. W języku tym zdefiniowano szereg środowisk dla których uogólnione wymuszenia i przepływy podano w tabeli 7.2. Wielkości te jednoznacznie opisują typy wyjść (portów) dostępnych w symulatorach, interpretację fizyczną wielkości płynących pomiędzy zaciskami (przepływy) i dostępnych na ich zaciskach (wymuszenia). Dla tak zdefiniowanych wyjść można określić dokładności z jakimi obliczane są przepływy i wymuszenia indywidualnie dla każdego środowiska.

Tab. 7.2: Środowiska zdefiniowane w języku HDL-A.

Środowisko	Wymuszenie	Przepływ
elektryczne	v, napięcie	i,prąd
mechaniczne 1	V, przyśpieszenie	f, siła
mechaniczne 2	d, przesunięcie	f, siła
obroty	ω, przyśp. kątowe	τ, moment obrotowy (*ang. torque*)
płyny	p, ciśnienie	fr, przepływ
termiczne	t, temperatura	hfr, (*ang. heat flow rate*)
NKN (nie Kirchoff)	a, wymuszenie	-
jakiekolwiek	a, wymuszenie	a, przepływ

Wspomniana wcześniej metoda obwodów równoważnych oparta jest na analogii. W układach elektryczno-mechanicznych można zdefiniować dwie analogie [Ped89] otrzymane z porównania równań różniczkowych pierwszego rzędu dla tych systemów (tabela 7.3): typu siła-prąd FI (*ang. force-current*) i siła-napięcie FV (*ang. force-voltage*).

Tab. 7.3: Podwójne analogie systemów elektryczno-mechanicznych

i	v	C	$1/R$	$1/L$	$Cv^2/2$	$Li^2/2$	FI
f	s	m	α	k	$Ms^2/2$	$f^2/2K$	
v	i	L	R	$1/C$	$Cv^2/2$	$Li^2/2$	FV

W metodzie obwodów równoważnych elementy składowe nie są opisane równaniami różniczkowymi lecz parametrami, które reprezentują ich właściwości jak np.: masa, pojemność, indukcyjność, ... Metoda ta jest szczególnie użyteczna do analizy złożonych systemów z elementami pracującymi w różnych środowiskach i co za tym idzie, z częstymi przejściami pomiędzy środowiskami.

Do analizy takich układów stosuje się uniwersalne symulatory układów elektronicznych jako programy rozwiązywania równań (*ang. solvers*):

- symulatory typu SPICE. Cechą charakterystyczną tych programów jest z góry ustalony zbiór dostępnych elementów i modeli.

- symulatory z wbudowanym językiem HDL umożliwiającym tworzenie modeli behawioralnych, w tym także modeli należących do różnych środowisk.

Do grupy pośredniej należy Dero z wbudowanym behawioralnym językiem opisu modeli użytkownika (języka MDL) i możliwością definiowania nowych zmiennych sieciowych oraz elementów należących do różnych środowisk.

7.1 Równania sieci w dziedzinie elektrycznej

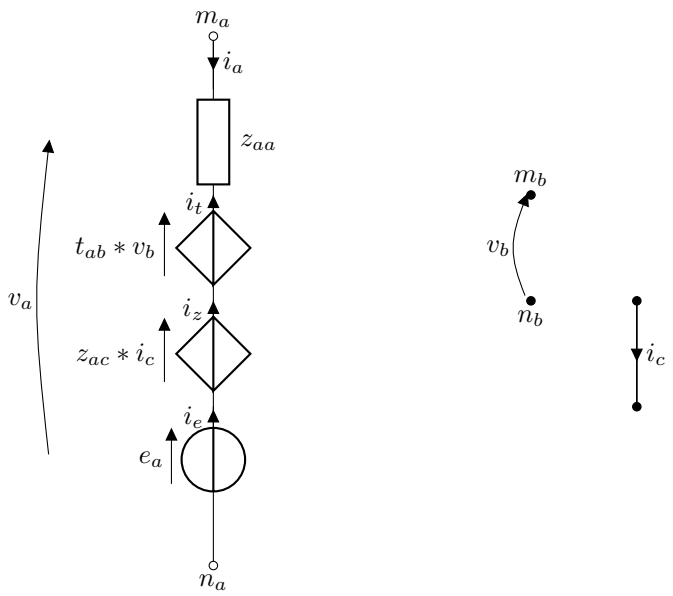
W dziedzinie elektrycznej x przyjmuje postać (7.1)

$$x = [v, i, q, \psi]^T \tag{7.1}$$

gdzie: v - zmienne napięciowe, i - zmienne prądowe, q - zmienne ładunkowe, ψ - zmienne strumieniowe. W skład sieci elektrycznej mogą wchodzić elementy opisane czterema podstawowymi typami równań opisującymi podstawowe typy gałęzi sieci:

1. gałąź typu prądowego o równaniu $i = f_i(x, \dot{x}, t)$,
2. gałąź typu napięciowego z prądem jako niewiadomą $v = f_v(x, \dot{x}, t)$,
3. gałąź opisana równaniem ładunkowym $q = f_q(x, \dot{x}, t)$,
4. gałąź opisana równaniem strumieniowym $\psi = f_\psi(x, \dot{x}, t)$,

W uniwersalnym symulatorze układów elektronicznych do układania równań używa się najczęściej zmodyfikowanej metody potencjałów węzłowych ZMPW [J.O94, J.O95]. Akceptuje ona następujące typy gałęzi: gałąź prądową, gałąź prądową z prądem jako niewiadomą, gałąź napięciową z prądem jako niewiadomą, gałąź opisaną równaniem strumieniowym i gałąź opisaną równaniem ładunkowym.

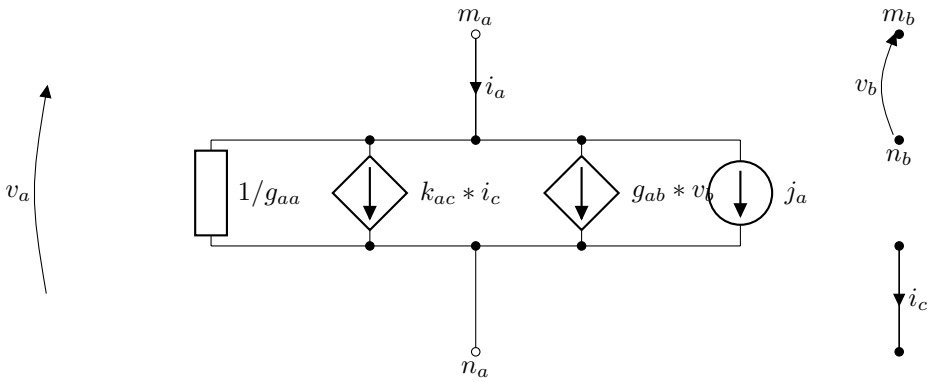


Rys. 7.1: Gałąź typu napięciowego

7.1.1 Gałąź typu prądowego

Gałąź typu prądowego przedstawiono na rys. 7.2. Gałąź opisana jest równaniem

$$i_a = G_{aa} v_a + G_{ab} v_b + K_{ac} i_c + J_a$$



Rys. 7.2: Gałąź typu prądowego

Rys. 7.3: Szablon ZMPW dla gałęzi typu prądowego

Y	m_a	n_a	m_b	n_b	i_c	B
m_a	$+G_{aa}$	$-G_{aa}$	$+G_{ab}$	$-G_{ab}$	$+K_{ac}$	$-J_a$
n_a	$-G_{aa}$	$+G_{aa}$	$-G_{ab}$	$+G_{ab}$	$-K_{ac}$	$+J_a$

Rys. 7.4: Szablon ZMPW dla gałęzi typu prądowego z prądem jako niewiadomą

Y	m_a	n_a	m_b	n_b	i_c	i_a	B
m_a						$+1$	
n_a						-1	
i_a	$+G_{aa}$	$-G_{aa}$	$+G_{ab}$	$-G_{ab}$	K_{ac}	-1	$-J_a$

7.1.2 Gałąź typu napięciowego

Gałąź typu napięciowego z prądem jako niewiadomą pokazana na rys. 7.1.

$$v_a = Z_{aa} i_a + T_{ab} v_b + Z_{ac} i_c + E_a$$

Rys. 7.5: Szablon ZMPW dla gałęzi typu napięciowego

Y	m_a	n_a	m_b	n_b	i_c	i_a	B
m_a						$+1$	
n_a						-1	
i_a	-1	$+1$	$+T_{ab}$	$-T_{ab}$	Z_{ac}	Z_{aa}	$-E_a$

7.1.3 Gałąź opisana równaniem ładunkowym

Gałąź opisana jest równaniem ładunkowym

$$\begin{aligned} i_a &= \frac{dq}{dt} \\ q_a &= C_{aa}v_a + C_{ab}v_b + q_{a0} \end{aligned} \tag{7.2}$$

gdzie: C_{xx} są to pojemności dynamiczne.

Rys. 7.6: Szablon ZMPW dla gałęzi opisanej równaniem ładunkowym

Y	m_a	n_a	i_a	v_a	v_b	q_a	B
m_a			$+1$				
n_a			-1				
g_a	$+1$	-1		-1			
g_a				C_{aa}	C_{ab}	-1	$-q_{e0}$
g_a			$+1$			$-\gamma$	$-\gamma p_{qa}$

7.1.4 Gałąź opisana równaniem strumieniowym

$$\begin{aligned} v_a &= \frac{d\psi}{dt} \\ \psi_a &= L_{aa}i_a + L_{ac}i_c + \psi_{a0} \end{aligned} \tag{7.3}$$

gdzie: L_{xx} są to indukcyjności.

Rys. 7.7: Szablon ZMPW dla gałęzi opisanej równaniem strumieniowym

Y	m_a	n_a	i_a	v_a	i_c	ψ_a	B
m_a			$+1$				
n_a			-1				
g_a	$+1$	-1		-1			
g_a			L_{aa}		L_{ac}	-1	$-\psi_{e0}$
g_a				$+1$		$-\gamma$	$-\gamma p_{\psi a}$

7.1.5 Szablony równań dla elementów liniowych

W rozdziale przedstawiono predefiniowane szablony elementów wykorzystywane do układania równań w klasycznych analizach AC, OP, DC, TR oraz optymalizacji (środowisko elektryczne). Szablony te są podstawą do wyprowadzenia pozostałych równań dla innych środowisk.

Rezystor liniowy

Szablon dla wszystkich rodzajów analiz. Równanie napięciowe z prądem jako niewiadomą.

$$R\, i_a = v_a$$

$$\begin{bmatrix} & & 1 \\ & & -1 \\ -1 & 1 & R \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.4}$$

Konduktancja liniowa

Szablon dla wszystkich rodzajów analiz. Wyjście konduktancyjne stosowane jest do obsługi wyjścia R, o ile wartość rezystancji $R > 0$.

$$G\ (v_{ma} - v_{na}) = i$$

$$\begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7.5}$$

Pojemność liniowa

Pojemność liniowa opisana jest równaniem prądowym postaci:

$$i = C\ \frac{dv}{dt} = C\ \dot{v} \tag{7.6}$$

Postać równania zależy od rodzaju analizy.

Analiza czasowa (TR) W analizie czasowej stanu nieustalonego pochodna $\dot{x}$ obliczona jest za pomocą schematu różnicowego postaci (6.12).

$$i_a = C(\gamma v_a + d_{v_a})$$

Równanie jest następnie linearyzowane (algorytm Newtona-Rapsona - rozdział 6.2). Po dyskretyzacji i linearyzacji równanie oraz po pogrupowaniu czynników otrzymujemy równanie gałęzi prądowej ZMPW, które może być układane przy pomocy szablonu opisanego w rozdziale 7.1.

$$\gamma C v_a = -C d_{v_a} \tag{7.7}$$

Zapisując równanie macierzowo otrzymamy szablon do automatycznego układania równania.

$$\begin{bmatrix} \gamma C & -\gamma C \\ -\gamma C & \gamma C \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} -C d_{v_{ma}} \\ C d_{v_{na}} \end{bmatrix} \tag{7.8}$$

gdzie $d_{v_a} = d_{v_{ma}} - d_{v_{na}}$ jest wektorem przeszłości zmiennej v_a.

Analiza punktu pracy (OP) i charakterystyk (DC) W analizach OP i DC przyjmuje się $\gamma = 0$, $d_{v_n} = 0$. Szablonu można nie stemplować. Pojemności są tutaj zastępowane *rozwarciami*, tzn. są pomijane. Powoduje to szereg problemów numerycznych związanych z pojawianiem się tzw. pływających węzłów[2]. Wyznaczenie wartości sygnału w takich węzłach można sprowadzić do analizy czasowej układu dla $t \to \infty$. Sieć zawierająca

[2]W literaturze spotyka się różne określenia, np.: węzły pływające, węzły odosobniene, węzły izolowane. Węzły takie spotyka się np. pomiędzy dwoma pojemnościami połączonymi w szereg. W klasycznej analizie punktu pracy wartość sygnału w takim węźle jest zawsze 0. W układach rzeczywistych na skutek stanu ustalonego, po włączeniu układu wartość sygnału może być różna od zera.

pojemności liniowe traktowana jest jak sieć nieliniowa. Stosując SR Eulera równania gałęzi (7.6) można zapisać:

$$i = C\frac{v(t_{n+1}) - v(t_n)}{\Delta t_n} = \frac{C}{\Delta t_n}v(t_{n+1}) + \underbrace{\left(-\frac{C}{\Delta t_n}v(t_n)\right)}_{d_i}$$

Wprowadzając $\gamma = \frac{1}{\Delta t}$ otrzymamy:

$$i = \gamma\ C\ v(t_{n+1}) + d_i$$

Przyjmując $d_i = 0$, $\gamma = 1$ oraz $v^{(p)} = v(t_{n+1})$ równanie można rozwiązać iteracyjnie (7.7). W każdej iteracji kontrolowana jest zmiana wartości sygnału w węźle $\Delta v = v^{(p)} - v^{(p-1)}$. Po uzyskaniu zbieżności, tzn. dla

$$||\Delta v|| < \epsilon_1 = 0.02||v|| + 0.002$$

przyjmowana jest wartość $\gamma_{i+1} = \gamma_i/10$. Po spełnieniu testu stopu

$$||\Delta v|| < \epsilon_2 = 0.002||v|| + 0.0002$$

wartość γ jest zmniejszana $\gamma_{i+1} = \gamma_i/10$. Kolejne spełnienie testu stopu (ϵ_2) powoduje 10-krotne zmniejszenie wartości γ.

Zmiana wartości γ powtarza się do momentu osiągnięcia wartości $\gamma = 1e{-}12$, Wtedy γ pozostaje stałe. Odpowiada to $\Delta t_{max} = 1e12$.

Analiza częstotliwościowa małosygnałowa (AC) W analizie częstotliwościowej małosygnałowej pojemność zastępowana jest konduktancją $g = j\omega\ C$ zależnymi od częstotliwości sygnału $f = \frac{\omega}{2\pi}$. Szablon ZMPW przyjmuje postać (7.9) podobną do (7.8), dla $\gamma = j\omega$ i $d_{v_a} = 0$.

$$\begin{bmatrix} j\ \omega C & -j\ \omega C \\ -j\ \omega C & j\ \omega C \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7.9}$$

Indukcyjność liniowa

$$v_a = L\ \frac{di_a}{dt}$$

Pochodną $\frac{di}{dt}$ obliczamy ze schematu różnicowego (dyskretyzacja).

$$v_a = L\ (\gamma i_a + d_i)$$

Równanie należy zlinearyzować (algorytm Newtona-Raphsona - rozdział 6.2), a następnie rozwiązać. Po pogrupowaniu otrzymujemy równanie gałęzi napięciowej ZMPW (roz. 7.1).

$$-v_{ma} + v_{na} + \gamma L i_a = -L d_{i_a}$$

Zapisując równanie macierzowo otrzymamy szablon do automatycznego układania równania.

$$\begin{bmatrix} & & 1 \\ & & -1 \\ -1 & 1 & \gamma L \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -L d_{i_a} \end{bmatrix} \tag{7.10}$$

Liniowe źródło prądowe

$$i = aux\ j_a$$

Analiza OP,DC,TR Wprowadzono mnożnik wydajności źródła $aux \in < 0, 1 >$ wykorzystywany w analizie kontynuacji. W klasycznej analizie OP,DC,TR mnożnik $aux = 1$. Szablon równania prądowego ZMPW przedstawiono poniżej.

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} -aux\ j_a \\ aux\ j_a \end{bmatrix} \tag{7.11}$$

W programie zastosowano także szablon równania prądowego ZMPW (7.12), z prądem gałęzi jako niewiadomą (Jx).

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} \\ \\ -aux\ j_a \end{bmatrix} \tag{7.12}$$

Analiza AC W analizie AC wymuszenia stałych źródeł prądowych i napięciowych są zerowane. Źródła prądowe stają się rozwarciami. Zerowanie wymuszeń realizuje wprowadzony mnożnik $aux = 0$. Można nie stemplować macierzy.

Liniowe prądowe źródło sterowane

Szablon dla wszystkich rodzajów analiz.

$$i = \sum_b G_{ab}\ (v_{mb} - v_{nb}) + \sum_c K_{ac} i_c$$

$$\begin{bmatrix} 0 & 0 & G_{ab} & -G_{ab} & -K_{ac} \\ 0 & 0 & -G_{ab} & G_{ab} & K_{ac} \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.13}$$

Liniowe napięciowe źródło sterowane

Szablon dla wszystkich rodzajów analiz.

$$v = \sum_b T_{ab}\ (v_{mb} - v_{nb}) + \sum_c Z_{ac} i_c$$

$$\begin{bmatrix} & & & & 1 \\ & & & & -1 \\ & & & & \\ & & & & \\ -1 & 1 & T_{ab} & -T_{ab} & Z_{ac} \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.14}$$

Idealny wzmacniacz operacyjny (liniowy)

Szablon dla wszystkich rodzajów analiz.

$$\begin{bmatrix} & & & & 1 \\ & & & & -1 \\ & & & & \\ & & & & \\ -1 & 1 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7.15}$$

7.1.6 Szablony równań dla elementów nieliniowych

Nieliniowe prądowe źródło sterowane

Wartości sterowań źródeł sterowanych x podlegają ograniczaniu, co gwarantuje lepszą zbieżność zastosowanego tłumionego algorytmu NR (rozdział 6.2.1).

$$i = f(x, \dot{x}, t)$$

$$\underbrace{\left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n} \right)}_{\delta f_x / \delta x} {x_n}^{(p)} = -\underbrace{f(x^{(p-1)})}_{f_x} + \left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n} \right) x_n^{(p-1)}$$

$$\begin{bmatrix} 0 & 0 & \frac{\delta f_x}{\delta v_b} & -\frac{\delta f_x}{\delta v_b} & -\frac{\delta f_i}{\delta i} \\ 0 & 0 & -\frac{\delta f_x}{\delta v_b} & \frac{\delta f_x}{\delta v_b} & \frac{\delta f_i}{\delta i} \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \end{bmatrix} = \begin{bmatrix} -f_x + \frac{\delta f_x}{\delta v_b}\ v_b + \frac{\delta f_x}{\delta i_c}\ i_c) \\ +f_x - \frac{\delta f_x}{\delta v_b}\ v_b - \frac{\delta f_x}{\delta i_c}\ i_c) \\ \\ \\ \end{bmatrix} \tag{7.16}$$

Nieliniowe napięciowe źródło sterowane

$$v = f(x, \dot{x}, t)$$

$$-v_{ma} + v_{na} + \underbrace{\left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n} \right)}_{\delta f_x / dx} {x_n}^{(p)} = -\underbrace{f(x^{(p-1)})}_{f_x} + \left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n} \right) x_n^{(p-1)}$$

$$\begin{bmatrix} & & & & & 1 \\ & & & & & -1 \\ & & & & & \\ & & & & & \\ & & & & & \\ -1 & 1 & \frac{\delta f_x}{\delta v_b} & -\frac{\delta f_x}{\delta v_b} & \frac{\delta f_x}{\delta i_c} & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ i_a \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \\ -f_x + \frac{\delta f_x}{\delta v_b}\ v_b + \frac{\delta f_x}{\delta i_c}\ i_c \end{bmatrix} \tag{7.17}$$

Wyjście ładunkowe (nieliniowe)

Gałąź opisana równaniem ładunkowym jest gałęzią nieliniową o równaniu postaci (7.18)

$$i = \frac{dq}{dt} \tag{7.18}$$

gdzie q jest funkcją nieliniową zależna od wartości sterowań $q(v)$. Pochodną $\frac{dq}{dt}$ obliczymy ze schematu różnicowego (6.12). Po podstawieniu otrzymamy pierwsze równanie.

$$i = \gamma q(v) + d_q \tag{7.19}$$

Po linearyzacji równania na $q(v)$ (rozwinięciu w szereg Taylora do wyrazów rzędu pierwszego), otrzymamy drugie równanie, które umożliwi obliczenie q.

$$q = q(v^{(p-1)}) + \frac{\delta q(v)}{\delta v}\ (v^{(p)} - v^{(p-1)})$$

Mamy do rozwiązania układ równań

$$i = \gamma q + d_q \tag{7.20}$$

$$q = q(v^{(p-1)}) + \frac{\delta q(v)}{\delta v}\ (v^{(p)} - v^{(p-1)}) \tag{7.21}$$

Po pogrupowaniu czynników stronami otrzymujemy układ równań

$$\gamma q = -d_q \tag{7.22}$$

$$\frac{\delta q(v_b^{(p-1)})}{\delta v}\ v_b^{(p)} - q = -q(v_b^{(p-1)}) + \frac{\delta q(v_b^{(p-1)})}{\delta v}\ v_b^{(p-1)} \tag{7.23}$$

gdzie $\frac{\delta q(v^{(p-1)})}{\delta v}$ jest pojemnością dynamiczną C.

$$\begin{bmatrix} & & & & \gamma \\ & & & & -\gamma \\ & & & & \cdot \\ & & & & \cdot \\ & & & & \cdot \\ . & . & \frac{\delta q(v_b^{(p-1)})}{\delta v} & -\frac{\delta q(v_b^{(p-1)})}{\delta v} & . & -1 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ q \end{bmatrix} = \begin{bmatrix} -d_q \\ d_q \\ \cdot \\ \cdot \\ \cdot \\ -q(v^{(p-1)}) + \frac{\delta q(v^{(p-1)})}{\delta v} \cdot v_b \end{bmatrix} \tag{7.24}$$

Gałąź opisana równaniem strumieniowym

Gałąź opisana równaniem strumieniowym jest gałęzią nieliniową o równaniu Postaci (7.25)

$$v = \frac{d\psi}{dt} \tag{7.25}$$

gdzie ψ jest funkcją nieliniową zależna od wartości sterowań $\psi(i)$. Pochodną $\frac{d\psi}{dt}$ obliczymy ze schematu różnicowego (6.12). Po podstawieniu otrzymamy pierwsze równanie.

$$v = \gamma\psi(i) + d_\psi \tag{7.26}$$

Po linearyzacji równania na $\psi(i)$ (rozwinięciu w szereg Taylora do wyrazów rzędu pierwszego), otrzymamy drugie równanie, które umożliwi obliczenie ψ.

$$\psi = \psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ (i^{(p)} - i^{(p-1)})$$

Mamy do rozwiązania układ równań

$$v = \gamma\psi + d_\psi \tag{7.27}$$

$$\psi = \psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ (i^{(p)} - i^{(p-1)}) \tag{7.28}$$

Grupując stronami otrzymamy

$$-v_{ma} + v_{na} + \gamma\psi = -d_\psi \tag{7.29}$$

$$\frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p)} - \psi = -\psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p-1)} \tag{7.30}$$

gdzie $\frac{\delta\psi(i^{(p-1)})}{\delta i}$ jest indukcyjnością dynamiczną L.

$$\begin{bmatrix} & 1 & & \cdot \\ & -1 & & \cdot \\ -1 & 1 & & \gamma \\ & & & \cdot \\ \cdot & \cdot & \cdot \quad \frac{\delta\psi(i^{(p-1)})}{\delta i} & -1 \end{bmatrix} \begin{bmatrix} v_{ma}^{(p)} \\ v_{na}^{(p)} \\ i_a^{(p)} \\ i^{(p)} \\ \psi \end{bmatrix} = \begin{bmatrix} \cdot \\ \cdot \\ -d_\psi \\ \cdot \\ -\psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p-1)} \end{bmatrix} \tag{7.31}$$

7.2 Równania sieci w dziedzinie mechaniki

Transformacja zmiennych i elementów do dziedziny mechaniki możliwa jest na dwa sposoby:

- analogia siła-prąd oznaczana jako środowisko mechaniczne 1,
- analogia siła-napięcie oznaczana jako środowisko mechaniczne 2.

Zmienne uogólnione transformacji zmiennych zamieszczono w tab. 7.1. W tab. 7.4 przedstawiono szczegółowe analogie pomiędzy zmiennymi występującymi w środowisku elektrycznym i ich transformacje na zmienne w środowisku mechanicznym 1 i mechanicznym 2.

W tab. 7.5 przedstawiono transformacje równań dla poszczególnych elementów mechanicznych.

W środowisku mechanicznym 1 (siła-prąd) analogia występuje pomiędzy prądowym prawem Kirchoff'a a prawem D'Alembert'a. Jedyna wadą analogii siła-prąd jest to, że łatwo ją stosować dla uziemionych kondensatorów. Widać to przez analogie energii w kondensatorze i energii masy oraz analogią pomiędzy uziemieniem (niezmienne napięcie = 0) i mechaniczną *masą* (nieruchoma pozycja). Ponieważ energia masy jest odpowiednio mierzona względem mechanicznego *podłoża* (tj., = prędkość $v = 0$), to energia pojemności musi być także mierzona w stosunku do masy elektrycznej (to znaczy, napięcie $v = 0$). Aby zastosować tę analogię, każdy węzeł w obwodzie elektrycznym staje

Tab. 7.4: Analogie pomiędzy środowiskiem elektrycznym, a mechanicznym 1 i 2.

Środowiska		
Elektryczne	Mechaniczne 1 (siła-prąd)	Mechaniczne 2 (siła-napięcie)
v Napięcie $[V]$	f Siła $[N]$	V Przyśpieszenie $[m/S^2]$
i Prąd $[A]$	V Przyśpieszenie $[m/S^2]$	f Siła $[N]$
Transformacja elementów		
Rezystancja [R]	L Smarność $[1/B]$	B Tarcie []
Pojemność [C]	m Masa $[kg]$	Odkształcalność [1/Sprężystość] $1/K$
Indukcyjność [L]	Odkształcalność [1/Sprężystość] $1/K$	m Masa $[kg]$
Transfomator $[N1 : N2]$	Dźwignia L1:L2	Dźwignia L1:L2
	K Sprężystość	K Sprężystość

się punktem w układzie mechanicznym. Masa staje się położeniem (stałą lokalizacją), rezystory stają się elementami ciernymi, kondensatory stają się masami, a indukcyjności stają się sprężynami. Źródło prądu staje się generatorem siły, a źródło napięcia staje się źródłem prędkości wejściowej.

Tab. 7.5: Analogie równań pomiędzy środowiskiem elektrycznym, a mechanicznym 1 i 2.

Analogie równań		
Śr. elektryczne	Śr. mechaniczne 1 (siła-prąd)	Śr. mechaniczne 2 (siła-napięcie)
$v = iR$	$v = \frac{f}{B}$	$f = vB$
$v = L\frac{di}{dt}$ $i = \frac{1}{L}\int vdt$	$v = \frac{1}{K}\frac{df}{dt}$ $f = K\int vdt = Kx$	$f = m\frac{dv}{dt} = ma$ $v = \frac{1}{m}\int fdt$
$v = \frac{1}{C}\int idt$ $i = C\frac{dv}{dt}$	$v = \frac{1}{m}\int fdt$ $v = m\frac{dv}{dt} = ma$	$f = K\int vdt = Kx$ $v = \frac{1}{K}\frac{df}{dt}$
moc $p = vi$	$p = vf$	$p = fv$
transformator $\frac{v_1}{v_2} = \frac{N_1}{N_2} = \frac{i_2}{i_1}$	$\frac{v_1}{v_2} = \frac{L_1}{L_2} = \frac{f_2}{f_1}$	$\frac{f_1}{f_2} = \frac{L_2}{L_1} = \frac{v_2}{v_1}$
energia pojemności $\frac{1}{2}Cv^2$	energia masy $\frac{1}{2}mv^2$	energia sprężystości $\frac{1}{2}Kx^2 = \frac{1}{2}K(\frac{f}{K})^2 = \frac{1}{2}\frac{f^2}{K}$
energia indukcyjności $\frac{1}{2}Li^2$	energia sprężystości $\frac{1}{2}Kx^2 = \frac{1}{2}K(\frac{f}{K})^2 = \frac{1}{2}\frac{f^2}{K}$	energia masy $\frac{1}{2}mv^2$
$\sum_{\text{węzły}} i = 0$	$\sum_{objekty} f = 0$	$\sum_{\text{pętle}} v = 0$
$\sum_{\text{pętle}} v = 0$	$\sum_{\text{pętle}} v = 0$	$\sum_{obiekty} f = 0$
$v_0 = 0$ każdy prąd można zewrzeć (węzeł) z ziemią v_0 i napięcie pozostanie 0	$v_0 = 0$ każdą siłę można przyłożyć do ziemi i przyśpieszenie v pozostanie 0	

Rozdział 8

Różniczkowanie symboliczne

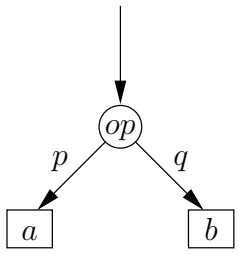
Algorytm różniczkowania symbolicznego wykorzystywany jest w modelach MDL do obliczania pochodnych funkcji względem zmiennych (wyjść względem sterowań oraz czasu). Są one wymagane przez algorytmy obliczeniowe symulatora.

Pochodne można zadać jawnie, co w przypadku bardziej skomplikowanych funkcji jest bardzo uciążliwe lub można je obliczyć numerycznie, co jest jednak bardzo nakładochłonne i spowalnia znacznie proces symulacji.

Najlepszym rozwiązaniem jest automatyczna generacja pochodnych w postaci symbolicznej w trakcie wczytywania opisu modelu. Automatyczna generacja pochodnych w postaci symbolicznej wymaga reprezentacji wzorów w postaci drzewa[Wir04] tworzonego na etapie parsowania wzoru. Przez odpowiednie przekształcenia drzewa parsowania można wyznaczyć pochodną w postaci symbolicznej.

8.1 Drzewo parsowania

Drzewo parsowania przedstawione na rys. 8.1 to struktura, która umożliwia reprezentację działań elementarnych (dodawania, odejmowania ...). Drzewo parsowania posiada



Rys. 8.1: Drzewo parsowania

jeden korzeń. W wierzchołkach drzewa 

, umieszczone są elementarne operacje arytmetyczne podzielone na dwie grupy:

- operacje podstawowe: dodawania, odejmowanie, mnożenie, dzielenie, potęgowanie,
- wywołania funkcji złożonych.

W liściach drzewa a umieszczone są zmienne (a,b) lub stałe. Gałęzie p, q wskazują na liście lub inne wierzchołki.

Tworzenie drzewa parsowania Drzewo parsowania tworzone jest w trakcie wczytywania (parsowania) wzoru przez modul parsera. Moduł ten można zrealizować za pomocą analizatora składni yacc[Gaw93] oraz współpracującego z nim parsera wejściowego lex[Gaw93]. Język programowania analizatora składniowego i parsera dobrze dostosowany jest do tego typu zadań. Po wczytaniu wzór powinien być reprezentowany w pamięci komputera w postaci drzewa parsowania. Liście drzewa powinny przechowywać zmienne/stałe występujące we wzorze. W wierzchołkach powinny być umieszczone elementarne operacje.

8.2 Różniczkowanie symboliczne

Algorytm różniczkowania symbolicznego wykorzystuje przekształcenia drzewa parsowania do wyznaczenia pochodnej w postaci symbolicznej względem wybranej zmiennej. Zmienna ta umieszczona jest w liściu. Drzewo podlega przekształceniu poczynając od korzenia, a kończąc na liściach. Przekształceniu podlegają wierzchołki drzewa. W przekształconym drzewie mogą pojawić się nowe gałęzie.

Po przekształceniu drzewa należy wygenerować zapis symboliczny danych przechowywanych w drzewie przez przekształcenie odwrotne. Etap ten musi realizować specjalny moduł generatora pochodnej symbolicznej.

8.2.1 Podstawowe przekształcenia drzewa

Na rys. 8.2 i 8.3 przedstawiono podstawowe przekształcenia drzewa, wynikające z podstawowych reguł obliczania pochodnych dla podstawowych działań matematycznych. ”Prim” w nazwie zmiennej oznacza pochodną. Na przykład pochodne wyrażeń:

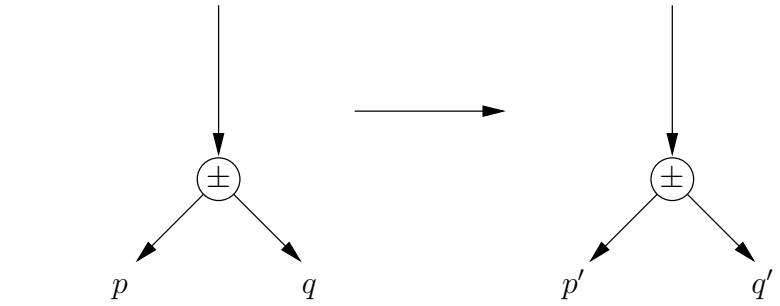
$$(pq)' = p'q + pq'$$

$$(p \pm q)' = p' \pm q'$$

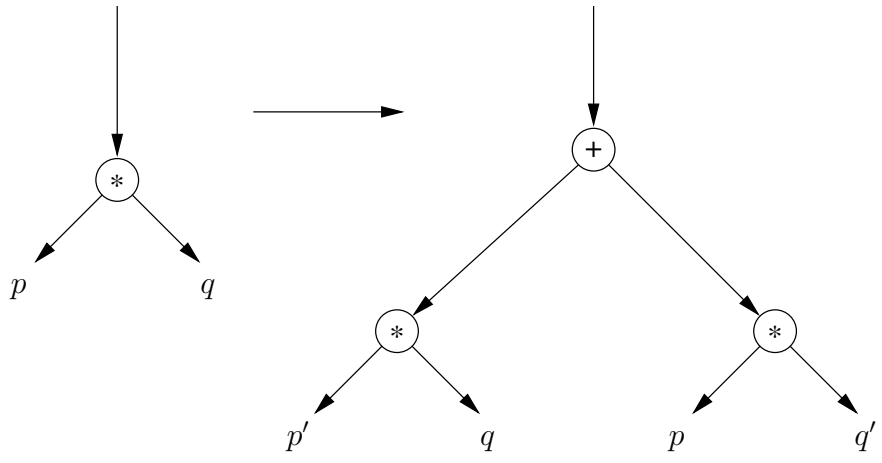
Opisane powyżej przekształcenia realizuje algorytm 11. Algorytm startuje od korzenia. Przekształca pierwszy węzeł drzewa w_1. Następnie przechodzi na poziom o jeden niższy i przekształca wszystkie wierzchołki z tej warstwy. Algorytm kończy działanie po dojściu do liści drzewa.

Algorytm 11 Przekształcanie drzewa parsowania

```
1: i - indeks warstwy, j - indeks w warstwie
2: n_i - liczba wierzchołków w warstwie
3: d - liczba warstw
4: for i = d...1 do
5:     for  j = 1...n_i  do
6:         przekształć węzeł w_j w warstwie i
7:     end for
8: end for
```



(a) dodawanie i odejmowanie $(p \pm q)' = p' \pm q'$



(b) mnożenie $(pq)' = p'q + pq'$

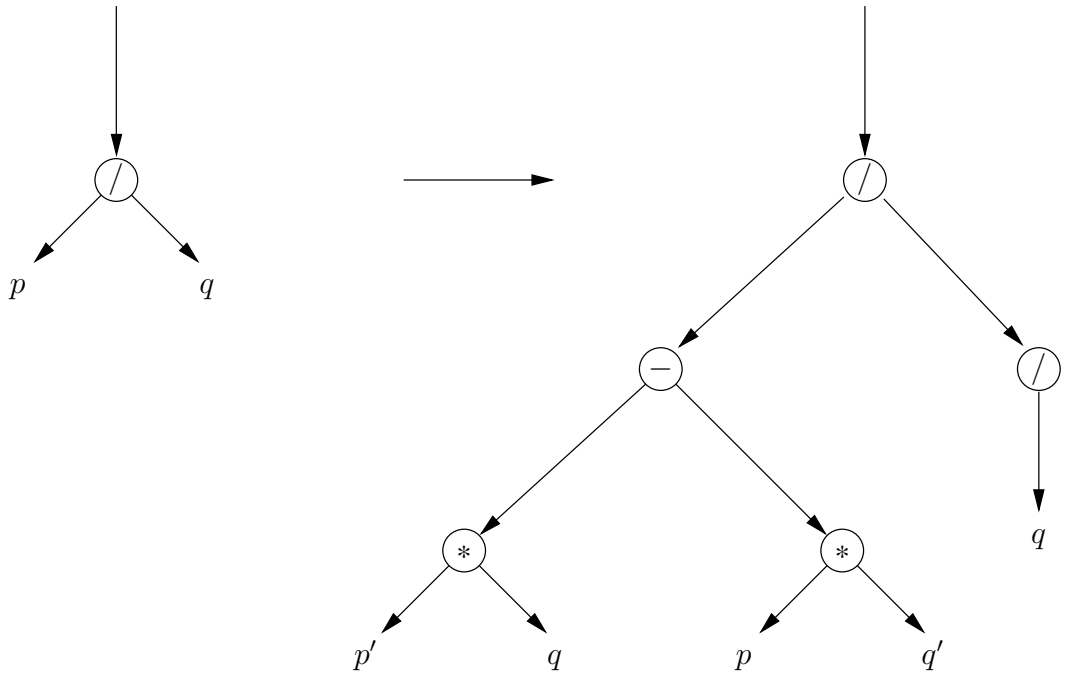
Rys. 8.2: Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych

8.2.2 Redukcja drzewa

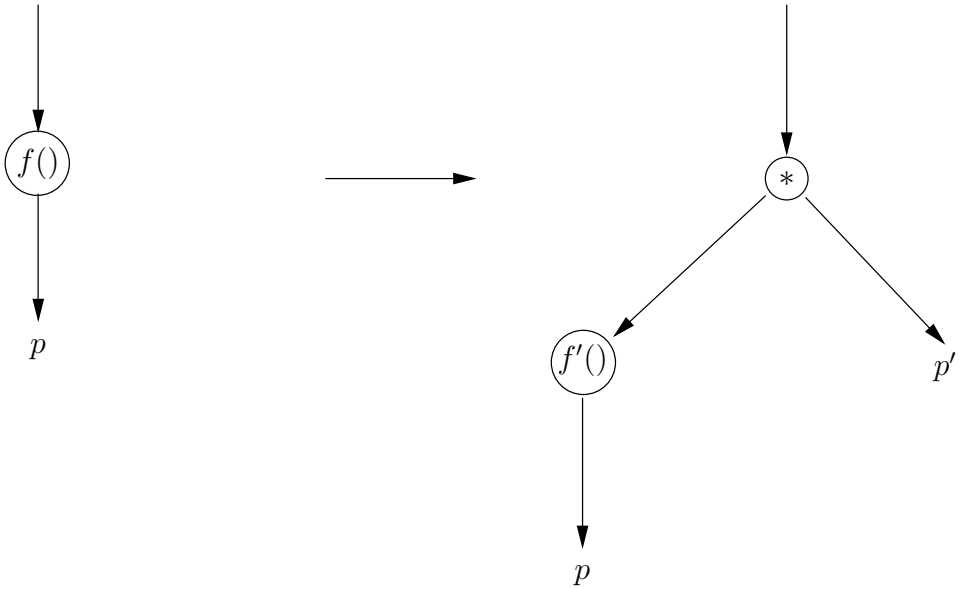
Przekształcone drzewo parsowania opisuje zróżniczkowane wyrażenie względem wszystkich zmiennych (liści drzewa). W celu wygenerowania pochodnej względem jednej zmiennej należy zredukować drzewo parsowania. Redukcja drzewa polega na usunięciu tych gałęzi drzewa, które na skutek wykonanych działań zerują się. Weźmy pod uwagę pochodną $(p + q)' = p'q + pq'$ przedstawioną w postaci drzewa na rys. 8.4. Obliczając pochodną względem zmiennej p, zmienna q traktowana jest jako stała i jej pochodna $q' = 0$. Czynnik pq' zeruje się. Działanie pq' zapisane jest odpowiednią gałęzią drzewa. Można ją w tym przypadku pominąć (zredukować), gdyż $pq' = 0$, bo $q' = 0$. Gałąź taka zostaje zastąpiona liściem, którego wartość równa jest 0.

Opisany tutaj sposób redukcji drzewa można zapisać w postaci algorytmu 12.

Algorytm przegląda wszystkie wierzchołki drzewa począwszy od wierzchołków najniższego poziomu, tzn. wierzchołków położonych możliwie najbliżej liści. Po przejrzeniu i redukcji wszystkich wierzchołków z danej warstwy algorytm przechodzi na wyższy poziom i powtórnie przeprowadza operację redukcji dla wierzchołków z danej warstwy. Algorytm kończy działanie po osiągnięciu korzenia drzewa parsowania. Po zakończeniu redukcji drzewo zawiera obliczoną pochodną względem wybranej zmiennej. Zmiennej i stałe występujące we wzorze zapisane są w liściach drzewa, natomiast działania



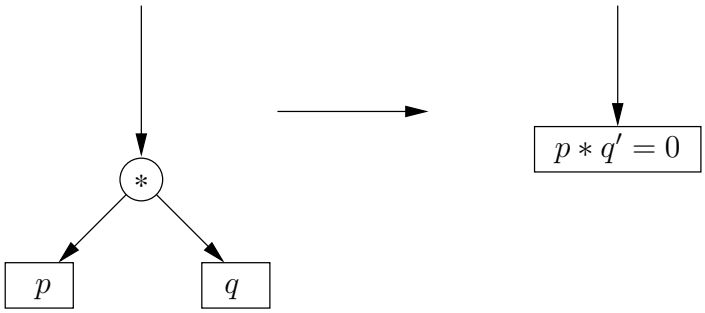
(a) dzielenie $(p/q)' = \frac{p'q - pq'}{q^2}$



(b) funkcja $f(p)' = f(p)' * p'$

Rys. 8.3: Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych

w wierzchołkach. Na podstawie danych umieszczonych w drzewie można wygenerować postać symboliczną pochodnej.

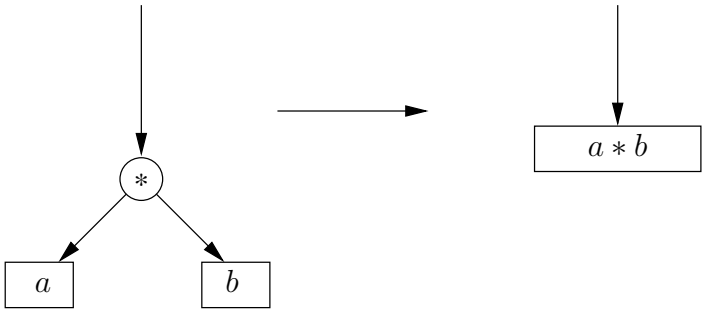


(a) operator mnożenia

Rys. 8.4: Redukcja drzewa

8.2.3 Generacja postaci symbolicznej

Dane zapisane w strukturze drzewa umożliwiają wygenerowanie postaci symbolicznej. Generacja postaci symbolicznej sprowadza się do zamiany gałęzi drzewa na liście. W liściach zapisanych jest wynik zamiany, czyli działanie w postaci symbolicznej. Zamianę gałęzi drzewa na liście przedstawiono na rys. 8.5.



(a) operator mnożenie

Rys. 8.5: Generacja postaci symbolicznej dla operacji mnożenia

Zamianę można zapisać w postaci algorytmu 13. Działanie algorytmu zaczyna się od zamiany wierzchołków najniższego poziomu. Wierzchołek zastępowany jest liściem, w którym zapisywana jest operacja wykonana na gałęziach p, q.

Algorytm kończy działanie z chwilą dojścia do korzenia drzewa. Drzewo zostaje zredukowane do pojedynczego liścia. W liściu zapisana jest symboliczna postać działań zapisanych w przekształcanym drzewie.

Przykład Weźmy pod uwagę wyrażenie arytmetyczne postaci (8.1). Drzewo parsowania dla wyrażenia przedstawiono na rys. 8.6.

$$a(b + dc) \tag{8.1}$$

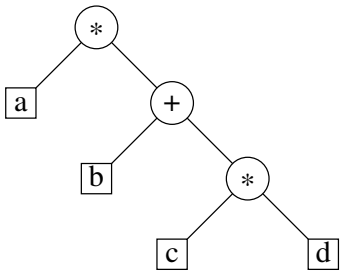
Przekształcone drzewo parsowania przedstawiono na rys. 8.7. Drzewo zostało zredukowane względem zmiennej a i zawiera pochodną wyrażenia po tej zmiennej.

Algorytm 12 Redukcja gałęzi drzewa

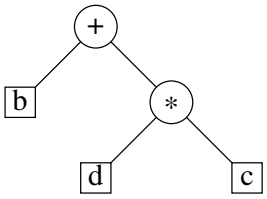
```
1: i - indeks warstwy, j - indeks w warstwie
2: n_i - liczba wierzchołków w warstwie
3: d - liczba warstw
4: for i = d ... 1 do
5:     for j = 1 ... n_i do
6:         if w_j ==' *' then
7:             if w_j → p == 0 w_j → q == 0 then
8:                 eliminuj węzeł w_j
9:             end if
10:        end if
11:        if w_j ==' /' then
12:            if w_j → q ==' 0' then
13:                błąd
14:            end if
15:        end if
16:        if w_j → p ==' 0' then
17:            eliminuj gałąź p
18:        end if
19:        if w_j → q ==' 0' then
20:            eliminuj gałąź q
21:        end if
22:    end for
23: end for
```

Algorytm 13 Generacja pochodnej w postaci symbolicznej

```
1: i - indeks warstwy, j - indeks w warstwie
2: n_i - liczba wierzchołków w warstwie
3: d - liczba warstw
4: for i = d ... 1 do
5:     for j = 1 ... n_i do
6:         if w_j ==' + - */' then
7:             zastąp w_j liściem l_j
8:             zapamiętaj działania p op q
9:             l_j ← q_{w_p} + w_j + q_{w_q}
10:        else
11:            zastąp w_j liściem l_j
12:            zapamiętaj działania p op q
13:            l_j ← f(p_{w_j})
14:        end if
15:    end for
16: end for
```



Rys. 8.6: Przykład drzewa parsowania dla wyrażenia (8.1).



Rys. 8.7: Drzewo pochodnej wyrażenia (8.1) obliczonej względem a.

Rozdział 9

Metody rozwiązywania układów równań liniowych

9.1 Zaawansowane metody klasyczne

Rozpatrzmy układ równań (9.1).

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n &= b_n \end{aligned} \tag{9.1}$$

Układ ten można zapisać macierzowo w postaci (9.2).

$$Ax = b \tag{9.2}$$

gdzie: $A \in R^{N\times N}$ jest macierzą współczynników a, $x \in R^N$ jest wektorem niewiadomych, $b \in R^N$ jest wektorem prawych stron. Układ równań posiada rozwiązania wtedy i tylko wtedy, gdy $rdA = rd(A\|b)$. Jeżeli $rdA = n$, to układ ma rozwiązanie jednoznaczne, jeżeli $rdA < n$, to rozwiązanie jest wieloznaczne. W praktyce mamy do czynienia z układami kwadratowymi $(N \times N)$. Jeżeli $detA \neq 0$, to rozwiązanie jest jednoznaczne dla każdego b, przy czym dla $b = 0$ rozwiązanie jest zerowe. Rozwiązanie układu równań (9.2) można zapisać w postaci (9.3).

$$x = A^{-1}b \tag{9.3}$$

Rozwiązanie układu równań wymaga odwrócenia macierzy A, co jest operacją bardzo nakładochłonną. Liczba działań (nakład obliczeniowy) jest proporcjonalna do n^3 , gdzie n jest wymiarem macierzy (liczbą niewiadomych).

Metoda rozkładu LU [J.O94, J.O95] jest jedną z najczęściej stosowanych technik rozwiązywania układów równań postaci (9.2). Metoda ta polega na znalezieniu takich macierzy L i U, aby $A = LU$. Zatem równania postaci (9.2) można zapisać

jako $LUx = b$. Prowadzi to do układu równań (9.4).

$$\begin{aligned} Ly &= b \\ Ux &= y \end{aligned} \tag{9.4}$$

Nakład obliczeń na rozwiązanie to $\frac{1}{2}n(n-1) + \frac{1}{2}n(n+1) = n^2$ mnożeń i dzieleń oraz $n^2 - n$ dodawań. Nakład jest taki sam jak przy obliczaniu $x = A^{-1}b$. Zaletą tej metody jest możliwość zapamiętania wyznaczonych macierzy L oraz U i rozwiązanie układu równań (9.2) dla różnych wektorów b.

W praktyce stosowane są pewne modyfikacje mające na celu zwiększenie dokładności, ograniczenie liczby działań i elementów niezerowych pojawiających się w czasie rozkładu. Najczęściej stosowany jest wybór elementu podstawowego o największym module spośród elementów macierzy. Po znalezieniu elementu podstawowego przestawia się wiersze i kolumny tak, aby element ten znalazł się na głównej przekątnej macierzy. Jeżeli macierz jest macierzą rzadką i zależy nam na minimalizacji liczby wprowadzanych w trakcie rozkładu wypełnień (elementów niezerowych), to stosuje się dodatkowe kryterium doboru elementu podstawowego (najczęściej kryterium Markowitza [J.O94, J.O95] - ze względu na prostotę i dobre oszacowanie). Na elementy podstawowe wybierane są elementy o jak najmniejszych miarach Markowitza i największych modułach.

Macierze pełne Jeżeli rozkład LU dla macierzy pełnych ma zostać zaimplementowany przy użyciu jednego z powszechnie używanych języków programowania (np.: C, Fortran, Pascal), to należy wziąć pod uwagę sposób przechowywania danych w pamięci komputera. Na przykład w języku C macierz jest przechowywana wierszami. Dla takiej organizacji najbardziej efektywne jest podejście Crouta [Lin92], w którym przez element podstawowy (z przekątnej macierzy) dzielony jest wiersz macierzy. Analogicznym algorytmem operujący na kolumnach jest algorytm Doolittle'a [Lin92]. Niestety podejścia te są niekorzystne z punktu widzenia wyboru elementu podstawowego [A.E86], dlatego też w praktyce najlepszym jest algorytm eliminacji Gaussa (alg. 14). Umożliwia on wybór elementu podstawowego z całej podmacierzy (*ang. complete pivoting*), a nie tylko z wiersza czy kolumny macierzy (*ang. partial pivoting*).

Techniki macierzy rzadkich wykorzystują rzadką strukturę macierzy. W strukturach reprezentowane są tylko elementy niezerowe. W zaawansowanych algorytmach rozwiązywania równań wykorzystuje się najczęściej metodę rozkładu LU macierzy w połączeniu z różnymi technikami gwarantującymi poprawność numeryczną rozwiązania. Techniki te wykorzystywane są na wszystkich etapach dekompozycji i rozwiązania.

- Na etapie przygotowania do dekompozycji (rozkład LU) należy tak przestawić wiersze i kolumny aby zminimalizować liczbę tzw. nowych wypełnień, czyli pojawiających się w trakcie dekompozycji elementów niezerowych. Wykorzystuje się zarówno informacje o postaci macierzy jak i informacje o właściwościach elementów niezerowych (typach elementów niezerowych).

- Na etapie dekompozycji bardzo ważnym zagadnieniem jest dbanie o poprawność numeryczną obliczeń i niedopuszczenie do kumulacji błędów. Realizuje się to przez:

  - wybór odpowiednich elementów podstawowych (*ang. pivot*) o największych modułach i najmniejszych miarach Markowitza [A.E86, J.O94],

  - wykorzystanie technik skalowania wierszy i kolumn, mających wpływ na dobór elementów podstawowych [A.E86].

Algorytm 14 Podstawowy algorytm rozkładu LU z normalizacją macierzy U.

```
1: DEKOMPOZYCJA LU
2: for k = 1...n − 1 do
3:     for j = 1...n do
4:         a_kj = a_kj/a_kj
5:         for i = k + 1...n do
6:             a_ij = a_ij − a_kj · a_ik
7:         end for
8:     end for
9: end for
10: PODSTAWIENIE W PRZÓD
11: for i = 1...n do
12:     b_i = b_i/a_ii
13:     s = b_i
14:     for k = i + 1...n do
15:         b(k) = b_j − a_ik · s
16:     end for
17: end for
18: PODSTAWIENIE WSTECZ
19: for i = n...1 do
20:     for j = 1...i − 1 do
21:         b_j = b_j − a_ij · b_i
22:     end for
23: end for
```

Techniki te zostaną szczegółowo omówione w dalszej części.

9.1.1 Klasyfikacja elementów niezerowych

W macierzy admitancyjnej układanej metodą ZMPW dla potrzeb analiz OP/TR/AC elementy macierzy można sklasyfikować według cech charakterystycznych. Można wyróżnić następujące typy reprezentowanych elementów:

0 - zera strukturalne,

1 - jedynki strukturalna,

C - elementy stałe nie zmieniające się w trakcie analizy (np. konduktancje wnoszone przez liniowe rezystancje),

+ - element zmienne, których wartość zmienia się w trakcie analiz:

- **OP, DC** w iteracjach Newtona-Raphsona (pochodne cząstkowe wyjść po sterowaniach),
- **TR, STDS** w iteracjach Newtona-Raphsona jak i w kolejnych punktach czasowych,
- **AC** w kolejnych punktach częstotliwości,

F - wypełnienie wprowadzone do macierzy w trakcie dekompozycji. Wartość lub zmiana wartości elementu nie są *a priori* znane.

9.1.2 Wstępne przygotowanie macierzy

Przed pierwszym rozwiązaniem układu równań wykonywane jest wstępne przestawienie elementów macierzy, celem zapewnienia poprawności numerycznej i minimalizacji liczby działań w trakcie rozkładu LU. W programie zastosowano trzy sposoby przenumerowania ustawiane opcją REXG (tab. 4.1): OFF - wyłączone przestawienia, ZERO - eliminacja zer z przekątnej, MIN - minimalizacja liczby elementów niezerowych.

Strategie przestawień niesymetrycznych wierszy realizują trzy usługi wywoływane w zależności od ustawionej opcji REXG (tab. 4.1).

nonsym_row_exchg_L_ONE() realizuje niesymetryczne przestawienia wierszy tak, aby jak najwięcej jedynek strukturalnych występujących w macierzy dolnej trójkątnej (L) przyciągnąć na główną przekątną. Przestawienie wykonywane jest tylko dla elementów podstawowych nie będących jedynkami strukturalnymi.

nonsym_row_exchg_ANY_COL_OR_ANY_SYM() realizuje niesymetryczne przestawienia wierszy, tak aby wyeliminować wszystkie zera z głównej przekątnej macierzy. Eliminacja następuje w dwóch etapach:

1. wprowadzenie na przekątną pierwszego niezerowego elementu w kolumnie (pierwszym elementem w kolumnie jest element zapisany w pierwszym wierszu). Jeśli istnieje taki element, to jest wykonywana zamiana wierszy. Jeśli wykonano przestawienia wierszy, to wykonujemy drugi etap, w przeciwnym przypadku następuje koniec działania.
2. ponowna próba zastąpienia zera na przekątnej przez zamianę wierszy. Zamiana wierszy nie może powodować wprowadzenia nowego zera na przekątną. Dlatego też, wyszukiwane są elementy symetryczne tak, aby przestawienie wierszy nie wprowadziło nowego zera w innym miejscu przekątnej.

   Weźmy pod uwagę macierz zawierającą na pozycji $(1,1)$ zero strukturalne.

	1	2	3
1	0		+
2		C	
3	+		1

$\rightarrow$

	1	2	3
3	+		1
2		C	
1	0		+

   Algorytm stara się wyeliminować za wszelką cenę zero z przekątnej przez przeszukanie wiersza 1 i znalezienie pierwszego elementu niezerowego dowolnego typu $(1,3)$ (tutaj element zmienny $(+)$). Jeśli istnieje niezerowy element symetryczny $(3,1)$, to wiersze są przestawiane. Na przekątną nie zostanie wprowadzone zero w trakcie zamiany wierszy 1 i 3. Analogiczny algorytm dla macierzy układanych może działać w oparciu o miary Markowitza, tzn. wszystkie elementy mające miarę 1 mają elementy symetryczne [A.E86].

nonsym_row_exchg_SYM_ONE() realizuje niesymetryczne przestawienia tak, aby na przekątną przyciągnąć jak najwięcej jedynek strukturalnych i wyeliminować jak najwięcej elementów zmiennych z przekątnej zastępując je jedynkami strukturalnymi (1) lub w ostateczności elementami stałymi (C). Przestawienia odbywają się w kilku etapach i *tylko dla elementów zerowych na przekątnej*. W każdym etapie przeszukiwane są wiersze zawierające elementy symetryczne:

1. w pierwszym etapie wyszukiwane są tylko te zera, które w kolumnie i wierszu posiadają pojedyncze jedynki strukturalne. W poniższej macierzy zastąpione zostaną wiersze 1 i 3 - przestawienia równań dla np. niezależnych źródeł napięciowych, do których nie dołączono żadnych elementów (3 kolumna, to zmienna prądowa wprowadzana przez źródło).

	1	2	3
1	0		1
2		C	
3	1		0

$\rightarrow$

	1	2	3
3	1		0
2		C	
1	0		1

2. w drugim etapie próbuje się zastąpić element zerowy lub zmienny na przekątnej innym elementem symetrycznym:

   - zastąpienie zera na przekątnej pozostałymi jedynkami symetrycznymi

   - zastąpienie elementów zmiennych na przekątnej jedynkami symetrycznymi

   - zastąpienie jakichkolwiek elementów na przekątnej przez jedynki symetryczne

3. w etapie trzecim (tylko dla REXG=MIN) próbuje się zastąpić elementy zmienne na przekątnej jedynkami symetrycznymi (jeśli jest to możliwe).

Algorytm 15 realizuje wstępne przestawienia niesymetryczne i symetryczne. Sposób działania zależy od opcji REXG.

Wynik działania algorytmu przedstawiony jest na rys. 9.1. W pierwszym etapie zostają wykonane wyeliminowane zera z pozycji (13,13) i (14,14) przez przestawienia symetrycznych jedynek. Następnie wykonywane zostają przestawienia elementów na przekątnej, tak aby na początku znalazły się elementy o najmniejszych miarach Markowitza. W trakcie przestawień następuje generacja brakujących wypełnień (F), np. na pozycji (14,11). Tak uporządkowana macierz nadaje się do dekompozycji, nie posiada zer na przekątnej i umożliwia efektywne zaimplementowanie algorytmu kontroli poprawności numerycznej dekompozycji.

9.1.3 Implementacja rozkładu LU

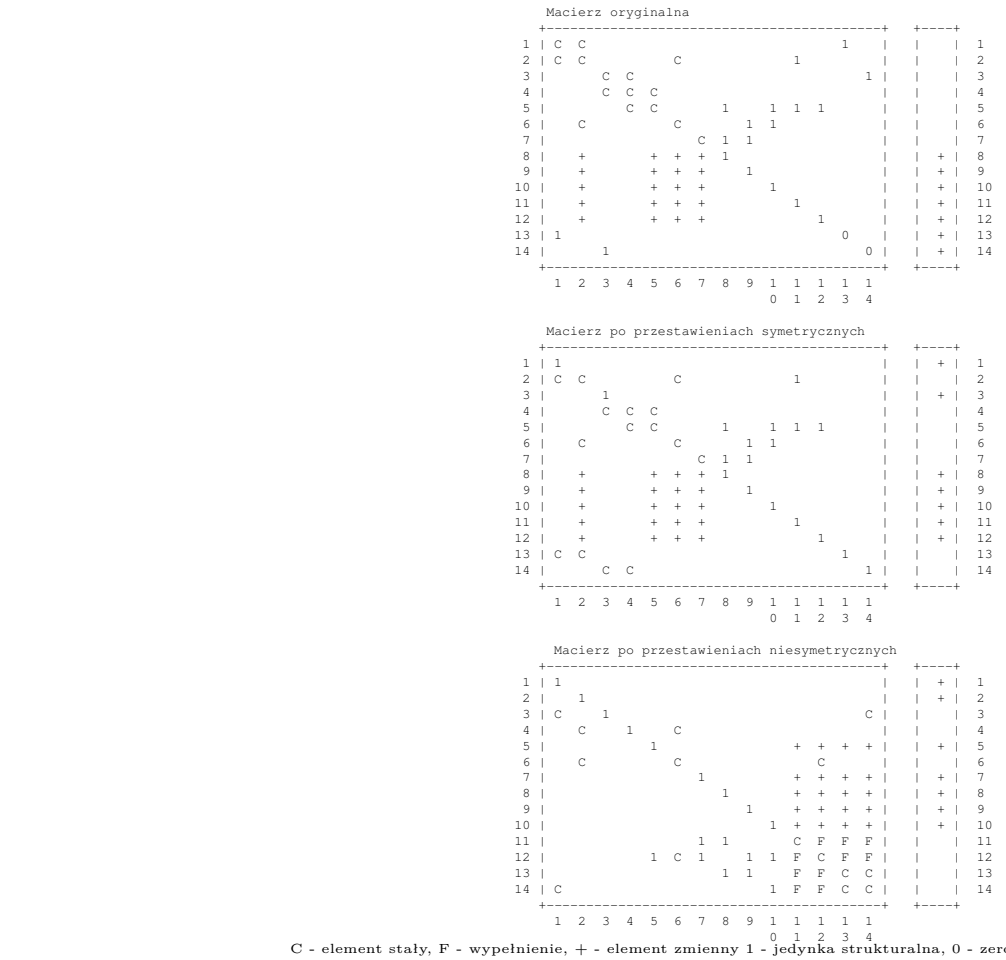
W programie zastosowano metodę rozkładu LU [J.O94] algorytmem Gaussa działającą na macierzach rzadkich. Przed dekompozycją wymagane są wstępne przestawienia niesymetryczne i symetryczne wierszy, tak aby wyeliminować wszystkie zera. Na początkowe miejsca powinny zostać wstawione elementy pojedyncze (*ang. singletons*, dla których nie ma potrzeby wykonywania dekompozycji. Zastosowano techniki kontroli dokładności dekompozycji przez kontrolę poziomu wartości elementów podstawowych. Stosowane są tu dwa podejścia: skalowanie kolumny lub przestawienia elementów podstawowych na przekątnej. Sterowanie algorytmem umożliwia opcja PIV, przy czym dla PIV=0 nie stosuje się kontroli dokładności rozwiązania (alg. 14).

9.1.4 Kontrola dokładności dekompozycji

Prawidłowy dobór elementów podstawowych macierzy jest bardzo ważny z punktu widzenia poprawności numerycznej algorytmu dekompozycji macierzy. Dla arytmetyki

Algorytm 15 Pełny algorytm wstępnego przenumerowania zastosowany w programie *Dero* .

1. utwórz listy przewlekane
2. wstępne przestawienia wierszy i kolumn (przestawienia symetryczne)
   Poniższe operacje wykonywane są iteracyjnie aż do momentu, gdy wszystkie zera z przekątnej zostaną wyeliminowane.
   (a) REXG=MIN - dla minimalizacji liczby wypełnień
   nonsym_row_exchg_L_ONE ();
   nonsym_row_exchg_SYM_ONE ();
   nonsym_row_exchg_ANY_COL_OR_ANY_SYM ();
   (b) REXG=ZERO: - normalne przestawienia i eliminacja zer,
   nonsym_row_exchg_ANY_COL_OR_ANY_SYM () Jeżeli nie udało się przestawić, to REXG=MIN i powrót do 2a
   (c) REXG=OFF - normalne przestawienia
   ERR (nonsym_row_exchg_SYM_ONE ( )); Jeżeli są zera na przekątnej (nie udało się przestawić) to REXG=MIN;
3. sprawdzenie poprawności elementów na głównej przekątnej. Na przekątnej, po przestawieniach symetrycznych nie może brakować elementów oraz nie mogą występować zera strukturalne. Jeśli taka sytuacja ma miejsce, to generowany jest komunikat o błędzie algorytmu (macierze układane metodą ZMPW są zawsze nieosobliwe).
4. ponowne połączenie elementów w kolumnach i wierszach: link_columns_r()
5. usunięcie elementów zerowych spoza głównej przekątnej
6. określenie kolejności elementów podstawowych na przekątnej na podstawie miar Markowitza (przestawienia niesymetryczne)
   (a) oblicz miary Markowitza dla elementów z przekątnej
   (b) oblicz liczbę elementów pojedynczych (*ang. singletons*): reorder_and_count_singletons_r()
   (c) usunięcie elementów zerowych spoza przekątnej: rem_off_diag_zero()
   (d) przestawienia elementów symetrycznych spoza przekątnej: search_and_reorder_symetric_elem ()
   (e) ustawienie elementów na przekątnej w kolejności rosnącej miary Markowitza (reorder())
7. usunięcie elementów zerowych spoza głównej przekątnej



Rys. 9.1: Struktura macierzy układu w trakcie wstępnego przestawienia wierszy i kolumn.

komputerowej o skończonej precyzji należy minimalizować błędy zaokrągleń przez wybór, największego co do modułu, elementu podstawowego macierzy[Wil63, Wil65]. Proces wyboru elementu podstawowego (*ang. pivot*) na przekątną nazywany jest *ang. pivoting*. Istnieją dwie strategie wyboru elementów podstawowych macierzy A:

- wybór największego (co do modułu) elementu podstawowego z całej macierzy (*ang. complete pivoting*. Podejście to sprowadza się do takiego wyboru r i s, aby

$$\|a_{rs}^{(k)}\| = \max_{i,j=k\ldots n} \|a_{ij}^{(k)}\| \tag{9.5}$$

  Znaleziony element (o największym module) a_{rs} powinien zostać wstawiony na przekątną przez zamianę k-tego i r-tego wiersza oraz k-tej i s-tej kolumny.

- wybór największego (co do modułu) elementu podstawowego z kolumny macierzy

(*ang. partial pivoting*). W podejściu tym wybieramy taki r-ty element w kolumnie k, że

$$\|a_{rk}^{(k)}\| = \max_{i=k\ldots n} \|a_{ik}^{(k)}\| \tag{9.6}$$

a następnie przestawiamy k-ty i i-ty wiersz. Podejście to jest także stosowane do wyboru elementu w wierszu, jednak jest rzadziej stosowane, gdyż wymaga zmiany kolejności zmiennych w wektorze rozwiązań.

Należy zauważyć, że jeśli macierz A posiada silnie dominującą przekątną, tzn.

$$\|a_{ii}\| \geq \sum_{i=1, i\neq j}^{n} \|a_{ij}\| \tag{9.7}$$

to wybór elementu podstawowego nie jest konieczny[1].

Skalowanie macierzy

Dokładność rozwiązania metodą rozkładu LU zależy od wartości elementów podstawowych. Jeśli wartości elementów podstawowych zbytnio różnią się od siebie, to może dojść do kumulacji błędów numerycznych podczas dekompozycji[2]. Może to spowodować znaczne zafałszowanie wyników[3] lub błędy nadmiaru (*ang. overflow*) lub niedomiaru (*ang. underflow*). Dlatego też w trakcie dekompozycji powinno się kontrolować wartości elementów podstawowych. Najczęściej stosuje się podejście polegające na skalowaniu kolumny macierzy celem wyrównania poziomu wartości elementów podstawowych. Oczywiście w trakcie rozwiązania należy przeskalować odpowiednio wartość rozwiązania[4].

Mnożnik powinien mieć wartość całkowitą i w przypadku systemów komputerowych używających arytmetyki binarnej, powinien mieć wartość będącą potęgą liczby 2*, co zapobiega błędom zaokrągleń.*

Skalowanie *jako takie* nie wpływa w większości przypadków [GD74, GEF67] na dokładność obliczeń wykonywanych w arytmetyce zmiennoprzecinkowej. Skalowanie ma wpływ na dokładność rozwiązania tylko w przypadku, gdy wpływa na wybór elementów podstawowych.

Weźmy pod uwagę $k+1$ krok eliminacji Gaussa[J.O94, J.O95]. Modyfikacji podlegają elementy podmacierzy a_{ij} zgodnie ze wzorem

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\, a_{kj}^{(k)} \tag{9.8}$$

[1] Strukturę symetryczną z silnie dominującą przekątną posiadają np. macierze admitancyjne układów pasywnych.

[2] Duży wpływ ma tzw. *numeryczne uwarunkowanie* równań. Układ równań jest *źle uwarunkowany*, jeśli wskutek błędów reprezentacji (zaokrągleń) liczb rzeczywistych oryginalne równania zostaną zaburzone w sposób uniemożliwiający dekompozycję.

[3] Jest to efekt maskowania elementów na skutek błędów zaokrągleń dla dużych różnic w wartościach elementów.

[4] Mnożenie kolumny przez τ powoduje konieczność podzielenia przez τ wartości rozwiązania $x_i = \hat{x}_i/\tau$. W przypadku skalowania wiersza rozwiązanie należy przemnożyć $x_i = \tau \cdot \hat{x}_i$.

Oznaczając przez β_i współczynnik skalowania i-tego wiersza, a przez α_j współczynnik skalowania j-tej kolumny i skalując zarówno kolumnę jak i wiersz otrzymamy

$$a_{ij}^{(k+1)} = \beta_i \ \alpha_j \ a_{ij}^{(k)} - \frac{\beta_i \ \alpha_k \ a_{ik}^{(k)}}{\beta_k \ \alpha_k \ a_{kk}^{(k)}} \ \beta_k \ \alpha_j \ a_{kj}^{(k)} \tag{9.9}$$

Po uproszczeniach otrzymamy

$$a_{ij}^{(k+1)} = \beta_i \ \alpha_j \ \left[a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \ a_{kj}^{(k)} \right] \tag{9.10}$$

Jak widać oba składniki są jednakowo skalowane i w przypadku użycia arytmetyki zmiennoprzecinkowej poprawa dokładności nie następuje.

Przestawienia elementów podstawowych

Kontrola wartości elementu podstawowego możliwa jest także przez zastosowanie przestawień elementów na przekątnej, co jednak wiąże się z potencjalną możliwością generowania nowych wypełnień. Jeżeli moduł wartości elementu podstawowego jest mniejszy niż wartość zadana, to poszukiwany jest nowy element podstawowy. Kryteriów wybierania nowego elementu jest wiele, lecz niestety żadne z nich nie jest do końca dobre. Poprawniejsze kryteria są obarczone dużym nakładem obliczeniowym. Z literatury [A.E86] wynika, że jednym z lepszych kryteriów jest wybór nowego elementu podstawowego tylko z przekątnej, jeśli wykonano wstępne przestawienia. Na przekątną wybierany jest element o najmniejszej miarze Markowitza spełniający kryterium dopuszczalnej wartości.

Sterowanie kontrolą dokładności rozwiązania możliwe jest przez zmianę wartości opcji PIV (tab. 4.1). Możliwe są następujące wartości opcji:

PIV=0 powoduje pominięcie kontroli poziomu elementów podstawowych (alg.14). Jest to najszybszy sposób rozwiązywania, lecz jednocześnie *niepewny* numerycznie. W trakcie dekompozycji na przekątnej może pojawić się zerowy element. W takim przypadku sygnalizowany jest błąd dekompozycji.

PIV=1 i 2 stosuje się empiryczne kryterium doboru elementu podstawowego. Dla i-tego elementu z przekątnej a_{ii} obliczana jest miara

$$\gamma = \epsilon_{PIV} \max_{j=1\ldots i} \|a_{jj}\| + \delta_{PIV} \tag{9.11}$$

gdzie: ϵ_{PIV} jest dopuszczalną względną różnicą wartości elementów podstawowych (opcja PIVR - tab. 4.1), δ_{PIV} - dopuszczalną bezwzględną różnicą (opcja PIVT - tab. 4.1), $\max_{j=1\ldots i} \|a_{jj}\|$ - maksymalnym dotychczasowym modułem wartości elementu podstawowego. Jeśli wartość modułu elementu podstawowego jest mała, tzn.

$$\|a_{ii}\| < \gamma \tag{9.12}$$

to dla:

to wykonywane operacje zależy od opcji PIV:

- dla PIV=1 skalowane są wszystkie elementy i kolumny, tzn. mnożone są one przez taką wartość τ, że

$$\tau \cdot a_{ii} > \gamma \tag{9.13}$$

  Realizuje to alg. 16.

- dla PIV=2 następuje wybór nowego elementu podstawowego z podmacierzy[5]. Realizuje to alg. 17.

[5]Wybierany jest nowy element tylko z przekątnej podmacierzy. Może to powodować generację nowych wypełnień.

Algorytm 16 Algorytm rozkładu LU - algorytm Gaussa ze skalowaniem kolumn macierzy.

1: $d_{max} = 0$
2: **for** $k = 1 \ldots n-1$ **do**
3: $d_{max} = max(d_{max}, \|d_{kk}\|)$ ▷ SKALOWANIE - MAX MODUŁ ELEMENTU PODSTAWOWEGO
4: $\gamma = \epsilon_{PIV}\ d_{max} + \delta_{PIV}$ ▷ POZIOM SKALOWANIA
5: **if** $\|d_{kk}\| < \gamma$ **then**
6: **if** $d_{kk} == 0$ **then** ▷ ZERO NA PRZEKĄTNEJ
7: $d_{kk} = pivot_{min}$
8: $d_{max} = max(d_{max}, \|d_{kk}\|)$
9: **end if**
10: $\gamma = \epsilon_{PIV}\ d_{max} + \delta_{PIV}$ ▷ POZIOM SKALOWANIA
11: **if** $\gamma > 0$ **then** ▷ SKALOWANIE KOLUMNY
12: $y = \frac{10 \cdot \gamma}{\|d_{kk}\|}$
13: $x = log_{10}(y)/log_{10}(2)$
14: **if** $x > 0$ **then**
15: $x = x + 0.5$
16: **else**
17: $x = x - 0.5$
18: **end if**
19: $r = 2^{rint(x)}$ ▷ ZAOKRĄGLENIE DO NAJBLIŻSZEJ LICZBY CAŁKOWITEJ
20: **for** $i = 1 \ldots n$ **do** ▷ SKALUJ KOLUMNĘ K
21: $a_{ik} = r \cdot a_{ik}$
22: **end for**
23: **end if**
24: **end if**
25: **end for**
26: **for** $j = 1 \ldots n$ **do** ▷ DEKOMPOZYCJA
27: $a_{kj} = a_{kj}/a_{kj}$
28: **for** $i = k+1 | \ldots | i <= n$ **do**
29: $a_{ij} = a_{ij} - a_{kj} \cdot a_{ik}$
30: **end for**
31: **end for**
32: **for** $i = 1 \ldots n$ **do** ▷ ALGORYTM PODSTAWIENIA W PRZÓD
33: $b_i = b_i/a_{ii}$
34: $s = b_i$
35: **for** $k = i+1 \ldots n$ **do**
36: $b(k) = b_j - a_{ik} \cdot s$
37: **end for**
38: **end for**
39: **for** $i = n \ldots 1$ **do** ▷ ALGORYTM PODSTAWIENIA WSTECZ
40: **for** $j = 1 \ldots i-1$ **do**
41: $b_j = b_j - a_{ij} \cdot b_i$
42: **end for**
43: **end for**
44: **for** $i = 1 \ldots n$ **do** ▷ SKALOWANIE ROZWIĄZANIA
45: $b_i = r_i \cdot b_i$
46: **end for**

Algorytm 17 Algorytm rozkładu LU z przestawieniem elementów na przekątnej.

DEKOMPOZYCJA LU
$d_{max} = 0$
for $k = singletons_{num} \dots n-1$ **do**
  $d_{max} = max(d_{max}, \|d_{kk}\|)$ ▷ MAKSYMALNY MODUŁ ELEMENTU PODSTAWOWEGO
  $\gamma = \epsilon_{PIV}\ d_{max} + \delta_{PIV}$ ▷ POZIOM PRZESTAWIENIA
  if $\|d_{kk}\| < \gamma$ **then** ▷ ZBYT MAŁA WARTOŚĆ PIVOTA - PRZESTAWIENIA
    OBLICZ MIARY MARKOWITZA DLA ELEMENTÓW $k \dots n$
    SORTUJ PIVOTY W KOLEJNOŚCI ROSNĄCEJ MIARY MARKOWITZA
    INDEKSY ZNAJDUJĄ SIĘ W WEKTORZE m_s O DŁUGOŚCI n
    for $i = k \dots n$ **do** ▷ ZNAJDŹ PIERWSZY PIVOT O NAJMNIEJSZEJ MIARZE MARKOWITZA
      if $d_{m_i} > \gamma$ **then** ZNALAZŁEM PIERWSZY DOBRY PIVOT
        ZNALEZIONY PIVOT MA MOŻLIWĄ NAJMNIEJSZĄ MIARĘ MARKOWITZA
        PRZESTAW SYMETRYCZNIE WIERSZE I KOLUMNY TAK
        ABY PRZESTAWIĆ k-TY Z m_i - TYM PIVOTEM
        PRZESTAW WIERSZ k-TY Z m_i -tym
        PRZESTAW KOLUMNĘ k -TĄ Z m_i -TĄ
        KONIEC PRZESTAWIENIA - WYSKOCZ Z PĘTLI for
      end if
    end for
  end if
  for $j = 1 \dots n$ **do** ▷ DEKOMPOZYCJA
    $a_{kj} = a_{kj}/a_{kj}$
    for $i = k + 1 toi <= n$ **do**
      $a_{ij} = a_{ij} - a_{kj} \cdot a_{ik}$
    end for
  end for
end for
for $i = 1 \dots n$ **do** ▷ PODSTAWIENIE W PRZÓD
  $b_i = b_i/a_{ii}$
  $s = b_i$
  for $k = i + 1 ton$ **do**
    $b(k) = b_j - a_{ik} \cdot s$
  end for
end for
for $i = n \dots 1$ **do** ▷ PODSTAWIENIE WSTECZ
  for $j = 1 toi - 1$ **do**
    $b_j = b_j - a_{ij} \cdot b_i$
  end for
end for
USTAW KOLEJNOŚĆ ZMIENNYCH W WEKTORZE ROZWIĄZAŃ

Część III

Modele elementów

Rozdział 10

Modele biblioteczne

Wraz z programem dostarczane są wybrane modele elementów w postaci plików bibliotecznych w języku MDL:

- definicje typów,
- definicje jednostek,
- definicje stałych,
- definicje zmiennych sieciowych,
- definicje wejść modeli MDL,
- definicje wyjść modeli MDL,
- modele elementów liniowych (R,L,C) zależnych od temperatury,
- modele źródeł napięciowych i prądowych,
- modele diody,
- modele tranzystorów bipolarnych BJT, odpowiadające poziomowi II z programu SPICE,
- modele tranzystorów MOS:

  z modelowaniem ładunku (model ładunkowy Ward-Dutton)

  z modelowaniem ładunku przy pomocy pojemności liniowych (model Meyer'a)
- model linii długiej

Modele można wprowadzać do opisu układu dyrektywą '.LIB' (2.6) lub '.INC' (2.6).

10.1 Predefiniowane typy

Predefiniowane typy danych znajdują się w pliku 'types.mdl' w katalogu bibliotek programu.

Wydruk 10.1: Predefiniowane typy (plik biblioteczny types.mdl)

```
1  # Revision : 1.1 (C) AIVA 2010
2  #
3  # BASIC TYPES
4  #
5
6  .TYPE NATURAL
7     LEFT = 0;
8     RIGHT = 1000000;
9  .END
10 #INFO = "liczby naturalne"
11
12 .TYPE INT
13    LEFT = -1000000;
14    RIGHT = 1000000;
15 .END
16 #INFO = "liczby calkowite"
17
18 # .TYPE LOGIC
19 # ENUM ZERO = 0;
20 # ENUM LOW = 0.2;
21 # ENUM HIGH = 3.8;
22 # ENUM ONE = 5.0;
23 # INFO = "typ wyliczeniowy"
24 # .END
25
26 .TYPE REAL
27    LEFT = -1E38;
28    RIGHT = 1E38;
29    TOLERANCE = 1.4E-12;
30 .END
31 # INFO = "liczby rzeczywiste"
```

10.2 Predefiniowane jednostki

Predefiniowane jednostki podstawowe znajdują się w pliku 'units.mdl' w katalogu bibliotek programu. Jednostki związane z danym środowiskiem zdefiniowano w plikach units-*.mdl, gdzie * oznacza nazwę danego środowiska.

Wydruk 10.2: Predefiniowane jednostki podstawowe.

```
# Revision : 1.1 (C) AIVA 2010
#
# base units
#

# Dimensionless multipliers
.UNIT
   yotta = 1e24; # yotta
   Y = 1yotta; # yotta
   zetta = 1e21; # zetta
   Z = 1zetta; # zetta
   exa = 1e15; # exa
   E = 1exa; # exa
   tera = 1e12; # tera
   T = 1tera; # tera
   giga = 1e9; # giga
   G = 1giga; # giga
   mega = 1e6; # mega
   M = 1mega; # mega
   kilo = 1e3; # kilo
   k = 1kilo; # kilo
   mili = 1e-3; # mili
   m = 1mili; # mili
   micro = 1e-6; # micro
   u = 1micro; # micro
   nano = 1e-9; # nano
   n = 1nano; # nano
   pico = 1e-12; # pico
   p = 1pico; # pico
   femto = 1e-15; # femto
   f = 1femto; # femto
   atto = 1e-18; # atto
   a = 1atto; # atto
   zepto = 1e-21; # zepto
   z = 1zepto; # zepto
   yocto = 1e-24; # yocto
   y = 1yocto; # yocto
.END

# Time
.UNIT
   S = 1; # 1 Second
   mS = 1e-3S;
   uS = 1e-6S;
   nS = 1e-9S;
   pS = 1e-12S;
   fS = 1e-15S;
   minut = 60S;
   minutes = 60S;
   hour = 60minut;
   hours = 60minut;
   day = 24h;
   days = 24h;
   week = 7day;
   weeks = 7day;
   month = 31day;
   months= 31day;
   year = 365days;
   years = 365day;
.END

# Frequency
.UNIT
   Hz = 1; # 1 Hertz
   kHz = 1e3Hz;
   megHz = 1e6Hz;
   gHz = 1e9Hz;
.END
```

Wydruk 10.3: Predefiniowane jednostki dla środowiska elektrycznego.

```
# Revision : 1.1 (C) AIVA 2010
#
# Electrical units
#

# EFFORT - Voltage [Volt]
.UNIT
   V = 1; # 1 Volt
   gigaV = 1e9V;
   GV = 1e9V;
```

```
11    megaV = 1e6V;
12    MV = 1e6V;
13    kiloV = 1e3V;
14    kV = 1e3V;
15    miliV = 1e-3V;
16    mV = 1e-3V;
17    microV = 1e-6V;
18    uV = 1e-6V;
19    nanoV = 1e-9V;
20    nV = 1e-9V;
21    picoV = 1e-12V;
22    pV = 1e-12V;
23    femtoV = 1e-15V;
24    fV = 1e-15V;
25  .END
26
27  # FLOW - Current [Amper]
28  .UNIT
29    A = 1; # 1 Amper
30    gigaA = 1e9A;
31    GA = 1e9A;
32    megaA = 1e6A;
33    MA = 1e6A;
34    kiloA = 1e3A;
35    kA = 1e3A;
36    miliA = 1e-3A;
37    mA = 1e-3A;
38    microA = 1e-6A;
39    uA = 1e-6A;
40    nanoA = 1e-9A;
41    nA = 1e-9A;
42    picoA = 1e-12A;
43    pA = 1e-12A;
44    femtoA = 1e-15A;
45    fA = 1e-15A;
46  .END
47
48  # MOMENTUM - magnetic flux [Wb - weber == T*m2 = J/A = V*s = kg*m2/(s2*A)]
49  .UNIT
50    Wb = 1; # 1 Wb
51    megaWb = 1e6Wb;
52    MWb = 1e6Wb;
53    kiloWb = 1e3Wb;
54    kWb = 1e3Wb;
55    miliWb = 1e-3Wb;
56    mWb = 1e-3Wb;
57    microWb = 1e-6Wb;
58    uWb = 1e-6Wb;
59    nanoWb = 1e-9Wb;
60    nWb = 1e-9Wb;
61    picoWb = 1e-12Wb;
62    pWb = 1e-12Wb;
63    femtoWb = 1e-15Wb;
64    fWb = 1e-15Wb;
65  .END
66
67  # STATE - Electric charge [C] = 6,24e18 e
68  # 1e = 1,602e-19C
69  .UNIT
70    C = 1; # 1 Coulomb = 1A * 1s = 1F*1V
71    gC = 1e9C;
72    megC = 1e6C;
73    kC = 1e3C;
74    mC = 1e-3C;
75    uC = 1e-6C;
76    nC = 1e-9C;
77    pC = 1e-12C;
78    fC = 1e-15C;
79  .END
80
81  # Resistance [Ohm]
82  .UNIT
83    Ohm = 1; # 1 Ohm
84    gigaOhm = 1e9Ohm;
85    GOhm = 1e9Ohm;
86    megaOhm = 1e6Ohm;
87    MOhm = 1e6Ohm;
88    kiloOhm = 1e3Ohm;
89    kOhm = 1e3Ohm;
90    miliOhm = 1e-3Ohm;
91    mOhm = 1e-3Ohm;
92  .END
93
94  # Inductance [Henr]
95  .UNIT
96    H = 1; # 1 Henr
97    miliH = 1e-3H;
98    mH = 1e-3H;
99    microH = 1e-6H;
100   uH = 1e-6H;
101   nanoH = 1e-9H;
102   nH = 1e-9H;
103   picoH = 1e-12H;
104   pH = 1e-12H;
105   femtoH = 1e-15H;
106   fH = 1e-15H;
107 .END
```

```
108
109 # Capacity [Farad]
110 .UNIT
111    F = 1; # 1 Farad
112    miliF = 1e-3F;
113    mF = 1e-3F;
114    microF = 1e-6F;
115    uF = 1e-6F;
116    nanoF = 1e-9F;
117    nF = 1e-9F;
118    picoF = 1e-12F;
119    pF = 1e-12F;
120    femtoF = 1e-15F;
121    fF = 1e-15F;
122 .END
```

Wydruk 10.4: Predefiniowane jednostki dla środowiska mechanicznego.

```
  1 # Revision : 1.1 (C) AIVA 2010
  2 #
  3 # Mechanical units
  4 #
  5
  6 # EFFORT - Force [Newton]
  7 .UNIT
  8    N = 1; # 1 Newton
  9    gN = 1e9N;
 10    megN = 1e6N;
 11    kN = 1e3N;
 12    mN = 1e-3N;
 13    uN = 1e-6N;
 14    nN = 1e-9N;
 15    pN = 1e-12N;
 16    fN = 1e-15N;
 17 .END
 18
 19 # FLOW - Velocity [m/S^2]
 20 .UNIT
 21    MS2 = 1; # 1 m/S2
 22    gmS2 = 1e9mS2;
 23    megmS2 = 1e6mS2;
 24    kmS2 = 1e3mS2;
 25    mmS2 = 1e-3mS2;
 26    umS2 = 1e-6mS2;
 27    nmS2 = 1e-9mS2;
 28    pmS2 = 1e-12mS2;
 29    fmS2 = 1e-15mS2;
 30 .END
 31
 32 # MOMENTUM - Momentum [kg*m^2/S]
 33 .UNIT
 34    KGm2S = 1; # 1 kg*m^2/S
 35    gKGm2S = 1e9KGm2S;
 36    megKGm2S = 1e6KGm2S;
 37    kKGm2S = 1e3KGm2S;
 38    mKGm2S = 1e-3KGm2S;
 39    uKGm2S = 1e-6KGm2S;
 40    nKGm2S = 1e-9KGm2S;
 41    pKGm2S = 1e-12KGm2S;
 42    fKGm2S = 1e-15KGm2S;
 43 .END
 44
 45 # STATE - State [Metr]
 46 .UNIT
 47    M = 1; # 1 metr
 48    gM = 1e9M;
 49    megM = 1e6M;
 50    kM = 1e3M;
 51    mM = 1e-3M;
 52    uM = 1e-6M;
 53    nM = 1e-9M;
 54    pM = 1e-12M;
 55    fM = 1e-15M;
 56 .END
 57
 58 # Electrical -> Mechanical1
 59 # Lubricity - smarnosc - bezwymiarowa
 60 # friction - tarcie - bezwymiarowa
 61
 62 # Inductance [Henr]
 63 .UNIT
 64    H = 1; # 1 Henr
 65    miliH = 1e-3H;
 66    mH = 1e-3H;
 67    microH = 1e-6H;
 68    uH = 1e-6H;
 69    nanoH = 1e-9H;
 70    nH = 1e-9H;
 71    picoH = 1e-12H;
 72    pH = 1e-12H;
 73    femtoH = 1e-15H;
 74    fH = 1e-15H;
 75 .END
 76
 77 # Capacity -> Mass [kg]
```

```
78 .UNIT
79    kg = 1; # 1 kg
80    milig = 1e-3kg;
81    mg = 1e-3kg;
82    microg = 1e-6kg;
83    ug = 1e-6kg;
84 .END
```

10.3 Predefiniowane stałe

Najczęściej używane stałe (nie tylko fizyczne) zostały zdefiniowane pliku bibliotecznym 'const.mdl' (wydruk 10.5)w katalogu bibliotek programu.

Wydruk 10.5: Predefiniowane stałe

```
1  # Revision : 1.1 (C) AIVA 2010
2
3  .CONST PI
4    VAL=3.149;
5    INFO="Liczba PI";
6  .END
7
8  .CONST C2K
9    VAL= 273.17;
10   INFO="addytywny przelicznik stopni C na K [K]";
11 .END
12
13 .CONST EPS0
14   VAL= 8.854215;
15   INFO="bezwzględna przenikalność elektryczna próżni [F/m]";
16 .END
17
18 .CONST EU
19   VAL= 2.718281;
20   INFO="podstawa logarytmu naturalnego";
21 .END
22
23 .CONST EUCONST
24   VAL= 2.718281;
25   INFO="stała Eulera";
26 .END
27
28 .CONST KBOLTZ
29   VAL= 1.380623E-23;
30   INFO="stała Boltzmana [J/K]";
31 .END
32
33 .CONST MI0
34   VAL= 1.25663E-6;
35   INFO="stała magnetyczna [H/m]";
36 .END
37
38 .CONST NISI0
39   VAL= 1.450000E10;
40   INFO="koncentracja nośników samoistnych w krzemie w temperaturze 273.15K [1/m^3]";
41 .END
42
43 .CONST PI
44   VAL= 3.141592653;
45   INFO="liczba PI";
46 .END
47
48 .CONST QELE
49   VAL= 1.602189E-19;
50   INFO="ładunek elementarny [C]";
51 .END
52
53 .CONST RAD2DEG
54   VAL= 57.295779;
55   INFO="multiplikatywny przelicznik radianów na stopnie [o/rad]";
56 .END
```

W programie wykorzystywane są także stałe zależne od systemu (wydruk 10.6), na którym wykonywane są obliczenia. Stałe te wykorzystywane są np. do ograniczania wartości funkcji wykładniczych.

Wydruk 10.6: Predefiniowane stałe systemowe

```
1  # Revision : 1.2 (C) AIVA 2010
2  #
3  # STALE ZALEZNE OD SYSTMEU KOMPUTEROWEGO
4  #
5
6  .CONST MAXEXPX
7     VAL=40;
8     INFO="MAKSYMALNY WYKLADNIK EXP(40)";
9  .END
10
11 .CONST MINEXPX
12    VAL=-200;
13    INFO="MINIMALNY WYKLADNIK EXP(-200)";
14 .END
15
16 .CONST MAXEXPOX
17    VAL=85;
18    INFO="MAKSYMALNY WYKLADNIK EXPO(85)";
19 .END
```

10.4 Zmienne sieciowe

Predefiniowane typy zmiennych sieciowych znajdują się w plikach 'vars-*.mdl' w katalogu bibliotek programu. Dla każdego ze środowisk istnieje osobny plik z definicjami wyjść.

Wydruk 10.7: Predefiniowane zmienne dla środowiska elektrycznego.

```
# Revision (C) AIVA 2010
#
# ELECTRICAL VARIABLES
#

.VARIABLE V
   TYPE = REAL;
   ABSTOL = 1uV;
.END

.VARIABLE I
   TYPE = REAL;
   ABSTOL = 100nA;
.END

.VARIABLE F
   TYPE = REAL;
   ABSTOL = 1u;
.END

.VARIABLE Q
   TYPE = REAL;
   ABSTOL = 1p;
.END
```

Wydruk 10.8: Predefiniowane zmienne dla środowiska mechanicznego.

```
# Revision (C) AIVA 2010
#
# MECHANICAL VARIABLES
#

# E-EFFORT
.VARIABLE FORCE
   TYPE = REAL;
   ABSTOL = 1uN; # 1e-6 N
.END

# F-FLOW
.VARIABLE VELOCITY
   TYPE = REAL;
   ABSTOL = 100mMS2; # 1e-3 m/S^2
.END

# PRZESUNIECIE
.VARIABLE X
   TYPE = REAL;
   ABSTOL = 1mm;
.END

# PED
.VARIABLE MOMENTUM
   TYPE = REAL;
   ABSTOL = 1u;
.END
```

10.5 Definicje wejść modeli

Predefiniowane wejścia znajdują się w plikach 'inputs-*.mdl' w katalogu bibliotek programu. Dla każdego ze środowisk istnieje osobny plik z definicjami wejść.

Wydruk 10.9: Predefiniowane stałe dla środowiska elektrycznego.

```
1  # Revision : 1.1 (C) AIVA 2010
2  #
3  # Prdefinded electrical inputs
4  #
5
6  # Electrical voltage input
7  .INPUT VV
8     PLUS => V;
9     MINUS => V;
10 .END
11
12 # Electrical current input
13 .INPUT II
14    PLUS => I;
15 .END
16
17 # Electrical flux input
18 .INPUT FF
19    PLUS => F;
20 .END
21
22 # Electrical charge input
23 .INPUT QQ
24    PLUS => Q;
25 .END
```

Wydruk 10.10: Predefiniowane stałe dla środowiska mechanicznego.

```
1  # Revision : 1.1 (C) AIVA 2010
2  #
3  # Prdefinded mechanical inputs
4  #
5
6  # Mechanical 1 velocity
7  .INPUT M1V
8     PLUS => V;
9     MINUS => V;
10 .END
11
12 # Mechanical 1 force input
13 .INPUT M1F
14    PLUS => MF;
15 .END
16
17 # Mechanical 1 - ?? Electrical flux input
18 .INPUT M1F
19    PLUS => M1F;
20 .END
21
22 # Mechanical 1 - ?? Electrical charge input
23 .INPUT M1Q
24    PLUS => M1Q;
25 .END
26
27 # Mechanical 2 velocity input
28 .INPUT M1V
29    PLUS => V;
30    MINUS => V;
31 .END
32
33 # Mechanical 2 force input
34 .INPUT M1F
35    PLUS => MF;
36 .END
37
38 # Mechanical 2 flux
39 .INPUT M2F
40    PLUS => M2F;
41 .END
42
43 # Mechanical 2 charge
44 .INPUT M2Q
45    PLUS => M2Q;
46 .END
```

10.6 Definicje wyjść modeli

Predefiniowane wyjść znajdują się w plikach 'outputs-*.mdl' w katalogu bibliotek programu. Dla każdego ze środowisk istnieje osobny plik z definicjami wyjść.

Wydruk 10.11: Predefiniowane stałe dla środowiska elektrycznego.

```
# Revision : 1.1 (C) AIVA 2010
#
# Prdefinded electrical outputs
#
# Resistance

.OUTPUT R
        TYPE => R;
        PLUS => V;
        MINUS => V;
.END

# Capacitance
.OUTPUT C
        TYPE => C;
        PLUS => V;
        MINUS => V;
.END

# Inductance
.OUTPUT L
        TYPE => L;
        PLUS => V;
        MINUS => V;
        THROUGH => I;
.END


# Electrical voltage source
.OUTPUT V
        TYPE => V;
        PLUS => V;
        MINUS => V;
        THROUGH => I;
.END

# Electrical current source
.OUTPUT J
        TYPE => J;
        PLUS => V;
        MINUS => V;
.END

# Electrical flux output
.OUTPUT F
        TYPE => F;
        PLUS => V;
        MINUS => V;
        THROUGH => I;
        VARIABLE => F;
.END

# Electrical charge output
.OUTPUT Q
        TYPE => Q;
        PLUS => V;
        MINUS => V;
        THROUGH => I;
        VARIABLE => Q;
.END
```

Wydruk 10.12: Predefiniowane stałe dla środowiska mechanicznego.

```
# Revision : 1.1 (C) AIVA 2010
#
# Prdefinded mechanical inputs
#

# ------------------------------------------------
# Mechanical 1
# ------------------------------------------------

# Mechanical 1 smarnosc (Resistance)
.OUTPUT M1L
        TYPE => R;
        PLUS => FORCE;
        MINUS => FORCE;
.END

# Mechanical 1 mass (Capacitance)
.OUTPUT M1M
        TYPE => C;
        PLUS => FORCE;
```

```
21          MINUS => FORCE;
22  .END
23
24  # Mechanical 1 odksztalcalnosc (Inductance)
25  .OUTPUT M1O
26          TYPE => L;
27          PLUS => FORCE;
28          MINUS => FORCE;
29          THROUGH => VELOCITY;
30  .END
31
32  # Mechanical 1 sila - (voltage source)
33  .OUTPUT M1F
34          TYPE => V;
35          PLUS => FORCE;
36          MINUS => FORCE;
37          THROUGH => VELOCITY;
38  .END
39
40  # Mechanical 1 przyspieszenie (current source)
41  .OUTPUT M1V
42          TYPE => J;
43          PLUS => FORCE;
44          MINUS => FORCE;
45  .END
46
47  # ------------------------------------------------------
48  # Mechanical 2
49  # ------------------------------------------------------
50
51  # Mechanical 2 tarcie (Resistance)
52  .OUTPUT M2B
53          TYPE => R;
54          PLUS => VELOCITY;
55          MINUS => VELOCITY;
56  .END
57
58  # Mechanical 2 odksztalcalnosc (Capacitance)
59  .OUTPUT M2O
60          TYPE => C;
61          PLUS => VELOCITY;
62          MINUS => VELOCITY;
63  .END
64
65  # Mechanical 2 masa (Inductance)
66  .OUTPUT M2M
67          TYPE => L;
68          PLUS => VELOCITY;
69          MINUS => VELOCITY;
70          THROUGH => FORCE;
71  .END
72
73  # Mechanical 2 przyspieszenie (voltage source)
74  .OUTPUT M2V
75          TYPE => V;
76          PLUS => VELOCITY;
77          MINUS => VELOCITY;
78          THROUGH => FORCE;
79  .END
80
81  # Mechanical 2 sila (current source)
82  .OUTPUT M2F
83          TYPE => J;
84          PLUS => VELOCITY;
85          MINUS => VELOCITY;
86  .END
```

10.7 Model rezystancji

Biblioteki modeli zawierają dwa podstawowe modele rezystancji:

- model podstawowy R,
- model z uzależnieniami termicznymi RT.

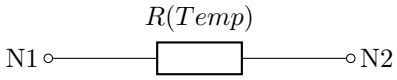
Modelowanie zależności termicznych w rezystorze umożliwia powiązanie wartości parametru modelu - rezystancji z temperaturą zgodnie z zależnością:

$$R(T) = AREA * R * TC \tag{10.1}$$

gdzie: RT jest wartością rezystancji zależną od temperatury, a TC dane jest wzorem (10.2).

$$\begin{aligned} TC &= 1 + TC1 * \Delta T + TC2 * \Delta T^2 + TC3^{TCE*\Delta T} \\ \Delta T &= TEMP - TNOM \end{aligned} \tag{10.2}$$

Schemat modelu rezystancji przedstawiono na rys. 10.1. Model opisany jest następują-



Rys. 10.1: Model rezystancji (plik biblioteczny rlc/rt.mdl).

cymi parametrami zamieszczonymi w tab. 10.1.

Tab. 10.1: Lista parametrów modelu rezystancji RT.

Nazwa	Opis	Wartość
parametry elementu		
R	wartość rezystancji	1
T	temperatura elementu	300.15
AREA	współczynnik powierzchni	1
parametry modelu		
TC1	współczynnik 1 stopnia temperatury	0
TC2	współczynnik 2 stopnia temperatury	0
TC3	podstawa zależności wykładniczej	1.01
TCE	współczynnik w wykładniku dla TC3	0
TNOM	temperatura nominalna (K)	300.15

Kod modelu RT przedstawiono na wydruku 10.13

Wydruk 10.13: Model rezystancji RT (rlc/rt.mdl).

```
1 # Revision (C) AIVA 2010
2
3 # TERMAL RESISTOR MODEL RT
4 #
5 # Element parameters
6 # R resistance (>0) (1)
7 # TEMP temperature (300.15K)
8 # AREA area factor (1)
```

```
#
# Model parameters
#
# TC1 1st order temperature coefficient (0)
# TC2 2nd order temperature coefficient (0)
# TC3 value (1.01)
# TCE TC3 exponent coefficient (0)
# TNOM nominal temperature in K (300.15K)
#
# RT=AREA*R*(1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)^2+TC3^(TCE*(TEMP-TNOM)));

# ------------------------------------------------------------
# Library:
# .LIB "math/temp3.mdl"
# ------------------------------------------------------------

.MODEL RT
.EXTERNAL N1,N2;
# ELEMENT PARAMETER
.PARAM REAL R "Resistance " = 1;
.PARAM REAL TEMP "Temperature " = 300.15;
.PARAM REAL AREA "Area factor " = 1;
# MODEL PARAMETERS
.COMMON REAL TC1 "1st order temperature coefficient for ROUT" = 0;
.COMMON REAL TC2 "2nd order temperature coefficient for ROUT" = 0;
.COMMON REAL TC3 "3rd value " = 1.01;
.COMMON REAL TCE "TC3 exponent coefficient " = 0;
.COMMON REAL TNOM "Nominal temperature (300.15K) " =300.15 ;
.OUTPUT R(N1,N2):REAL RT;
.BEGIN

RT=AREA*R*TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE ); # lib/math/temp3.mdl

.END # RT
```

10.8 Model pojemności

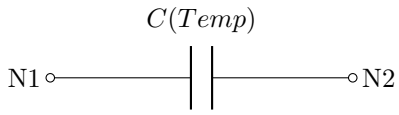
Biblioteki modeli zawierają dwa podstawowe modele pojemności:

- model podstawowy C,
- model z uzależnieniami termicznymi CT.

Modelowanie zależności termicznych w pojemności umożliwia powiązanie wartości parametru modelu - pojemności z temperaturą zgodnie z zależnością:

$$C(T) = AREA * C * TC \tag{10.3}$$

gdzie: CT jest wartością pojemności zależną od temperatury, a TC dane jest wzorem (10.2). Schemat modelu pojemności przedstawiono na rys. 10.2. Model opisany jest



Rys. 10.2: Model pojemności (plik biblioteczny rlc/ct.mdl).

następującymi parametrami zamieszczonymi w tab. 10.1.

Tab. 10.2: Lista parametrów modelu pojemności CT.

Nazwa	Opis	Wartość
parametry elementu		
C	wartość pojemności	1
T	temperatura elementu	300.15
AREA	współczynnik powierzchni	1
parametry modelu		
TC1	współczynnik 1 stopnia temperatury	0
TC2	współczynnik 2 stopnia temperatury	0
TC3	podstawa zależności wykładniczej	1.01
TCE	współczynnik w wykładniku dla TC3	0
TNOM	temperatura nominalna (K)	300.15

Kod modelu CT przedstawiono na wydruku 10.14

Wydruk 10.14: Model pojemności CT (rlc/ct.mdl).

```
1  # Revision (C) AIVA 2010
2  #
3  # TERMAL CAPACITOR MODEL CT
4  #
5  # Element parameters
6  # C capacity (0)
7  # TEMP temperature (300.15K)
8  # AREA area factor (1)
9  #
10 # Model parameters
11 # TC1 1st order temperature coefficient (0)
12 # TC2 2nd order temperature coefficient (0)
13 # TC3 value (1.01)
14 # TCE TC3 exponent coefficient (0)
15 # TNOM nominal temperature in K (300.15K)
```

```
16 #
17 # CT=AREA*C*(1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)^2+TC3^(TCE*(TEMP-TNOM)));
18 #
19 # ------------------------------------------------------------
20 # Library:
21 # .LIB "math/temp3.mdl"
22 # .LIB "rlc/ct.mdl"
23 # Calling:
24 # .MODEL yyyy CT [common_par_name=value,...]
25 # yyyy.name n1 n2 [individual_par_name=value,...]
26 # ------------------------------------------------------------
27
28 .MODEL CT
29 .EXTERNAL N1,N2;
30 # ELEMENT PARAMETER
31 .PARAM REAL C "Capacity " = 1;
32 .PARAM REAL TEMP "Temperature " = 0;
33 .PARAM REAL AREA "Area factor " = 1;
34 # MODEL PARAMETERS
35 .COMMON REAL TC1 "1st order temperature coefficient for ROUT" = 0;
36 .COMMON REAL TC2 "2nd order temperature coefficient for ROUT" = 0;
37 .COMMON REAL TC3 "3rd value " = 1.01;
38 .COMMON REAL TCE "TC3 exponent coefficient " = 0;
39 .COMMON REAL TNOM "Nominal temperature (300.15K) " =300.15 ;
40 .OUTPUT C(N1,N2):REAL CT;
41 .BEGIN
42
43 CT=AREA*C*TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE ); # lib/math/temp3.mdl
44
45 .END # CT
```

10.9 Model indukcyjności

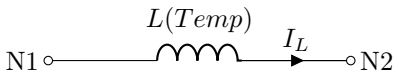
Biblioteki modeli zawierają dwa podstawowe modele indukcyjności:

- model podstawowy L,
- model z uzależnieniami termicznymi LT.

Modelowanie zależności termicznych w indukcyjności umożliwia powiązanie wartości parametru modelu - indukcyjności z temperaturą zgodnie z zależnością:

$$L(T) = AREA * L * TC \tag{10.4}$$

gdzie: $L(T)$ jest wartością rezystancji zależną od temperatury, a TC dane jest wzorem (10.2). Schemat modelu indukcyjności przedstawiono na rys. 10.3. Model opisany jest



Rys. 10.3: Model indukcyjności (plik biblioteczny rlc/lt.mdl).

następującymi parametrami zamieszczonymi w tab. 10.3.

Tab. 10.3: Lista parametrów modelu indukcyjności LT.

Nazwa	Opis	Wartość
parametry elementu		
L	wartość indukcyjności	1
T	temperatura	300.15
AREA	współczynnik powierzchni	1
parametry modelu		
TC1	współczynnik 1 stopnia temperatury	0
TC2	współczynnik 2 stopnia temperatury	0
TC3	podstawa zależności wykładniczej	1.01
TCE	współczynnik w wykładniku dla TC3	0
TNOM	temperatura nominalna (K)	300.15

Kod modelu LT przedstawiono na wydruku 10.15

Wydruk 10.15: Model indukcyjności LT (rlc/lt.mdl).

```
1  # Revision (C) AIVA 2010
2  #
3  # TERMAL INDUCTANCE MODEL
4  #
5  # Element parameters
6  # L inductance (1)
7  # TEMP temperature (300.15K)
8  # AREA area factor (1)
9  #
10 # Model parameters
11 # TC1 1st order temperature coefficient (0)
12 # TC2 2nd order temperature coefficient (0)
13 # TC3 value (1.01)
14 # TCE TC3 exponent coefficient (0)
15 # TNOM nominal temperature in K (300.15K)
16
17 # ------------------------------------------------
```

```
18 # Library:
19 # .LIB "math/temp3.mdl"
20 # .LIB "rlc/lt.mdl"
21 # Calling:
22 # .MODEL yyyy LT [common_par_name=value,...]
23 # yyyy.name n1 n2 [individual_par_name=value,...]
24 # ------------------------------------------------------------
25
26 .MODEL LT
27 .EXTERNAL N1,N2;
28 .FLOW IL;
29 # ELEMENT PARAMETER
30 .PARAM REAL L "Inductance " = 1.0;
31 .PARAM REAL TEMP "Temperature " = 0.0;
32 .PARAM REAL AREA "Area factor " = 1.0;
33 # MODEL PARAMETERS
34 .COMMON REAL TC1 "1st order temperature coefficient for ROUT" = 0.0;
35 .COMMON REAL TC2 "2nd order temperature coefficient for ROUT" = 0.0;
36 .COMMON REAL TC3 "3rd value " = 1.01;
37 .COMMON REAL TCE "TC3 exponent coefficient " = 0.0;
38 .COMMON REAL TNOM "Nominal temperature (300.15K) " = 300.15;
39 .OUTPUT L(N1,N2,IL):REAL LT;
40 .BEGIN
41
42 LT=AREA*L*TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
43
44 .END # LT
```

10.10 Modele źródeł

Biblioteki modeli zawierają dwa podstawowe modele:

- źródło napięciowe zależne od temperatury VT,
- źródło prądowe zależne od temperatury IT.

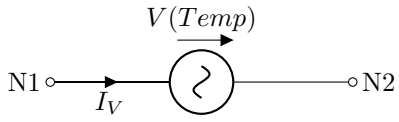
Oba źródła mogą generować przebiegi okresowe zależne od czasu: przebieg prostokątny i sinusoidalny. Mogą także pełnić rolę źródeł zasilających. W analizie małosygnałowej (AC) umożliwiają modelowanie wymuszeń małosygnałowych o określonej częstotliwości i fazie.

10.10.1 Model źródła napięciowego

Modelowanie zależności termicznych źródła umożliwia powiązanie podstawowych wartości parametrów z temperaturą zgodnie z zależnościami:

$$DCT = DC1 * TC \tag{10.5}$$
$$V1T = V1 * TC \tag{10.6}$$
$$V2T = V2 * TC \tag{10.7}$$
$$VSIT = VSI * TC \tag{10.8}$$

gdzie: DCT, $V1T$, $V2T$, $VSIT$ są odpowiednimi wartościami parametrów zależnych od temperatury, a TC dane jest wzorem (10.2). Schemat modelu źródła napięciowego przedstawiono na rys 10.4. Model opisany jest parametrami zamieszczonymi w tab. 10.4.



Rys. 10.4: Model źródła napięciowego (plik biblioteczny src/vt.mdl)

Tab. 10.4: Lista parametrów źródła napięciowego VT

Nazwa	Opis	Wartość
parametry elementu		
MAGN	Amplituda w analizie małosygnałowej AC	0
PHAS	Faza w analizie małosygnałowej AC	0
TEMP	Temperatura elementu	300.15
parametry modelu - prostokąt		
DC	Poziom odniesienia DC	0
V1	Napięcie poziomu V1	0
V2	Napięcie poziomu V2	0

kontynuacja na następnej stronie

kontynuacja z poprzedniej strony		
Nazwa	**Opis**	**Wartość**
FREQ	Częstotliwość	50
TD	Czas opóźnienia impulsu	$1e-30$
TR	Czas narastania	$1e-30$
TF	Czas opadania	$1e-30$
PW	Szerokość impulsu	$1e-30$
PER	Okres	$1e30$
parametry modelu - sinusoida		
VSI	Amplituda	0
TDSI	Czas opóźnienia	0
TAU	Opóźnienie	$1e30$
parametry termiczne modelu		
TC1	współczynnik 1 stopnia temperatury	0
TC2	współczynnik 2 stopnia temperatury	0
TC3	podstawa zależności wykładniczej	1.01
TCE	współczynnik w wykładniku dla TC3	0
TNOM	temperatura nominalna (K)	300.15

Kod modelu źródła napięciowego przedstawiono na wydruku 10.16

Wydruk 10.16: Model źródła napięciowego VT (src/vt.mdl)

```
# Revision : 1.2 (C) AIVA 2003-2010
#
# Voltage source model (time and temperature dependent)
#
# Element parameters
#
# MAGN Small signal magnitude (AC) (0)
# PHAS Small signal phase (AC) (0)
#
# Model parameters
#
# DC Bias voltage (0)
# V1 Voltage value V1 (0)
# V2 Voltage value V2 (0)
# FREQ Frequency (50)
# TD Time delay (1e-30)
# TR Time raise (1e-30)
# TF Time fall (1e-30)
# PW Pulse width (1e-30)
# PER Period (1e30)
#
# Sinusoidal response
#
# VSI Sinusoidal amplitude (0)
# TDSI Time delay (sinusoid) (0)
# TAU Delay (1e30)
#
# Thermal parameters
#
# TEMP temperature (300.15K)
#
# Model parameters
#
# TC1 1st order temperature coefficient (0)
# TC2 2nd order temperature coefficient (0)
# TC3 value (1.01)
# TCE TC3 exponent coefficient (0)
# TNOM nominal temperature in K (300.15K)
#
# TERMPCOEFF = 1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)^2+TC3^(TCE*(TEMP-TNOM));

# ------------------------------------------------------
# Library:
# .LIB "math/temp3.mdl"
# ------------------------------------------------------

.MODEL VT
.EXTERNAL N1,N2;
.FLOW II;
# ELEMENT PARAMETERS
.PARAM REAL MAGN "Small signal magnitude " = 0;
.PARAM REAL PHAS "Small signal phase " = 0;
.PARAM REAL TEMP "Temperature " = 300.15;
```

```
.PARAM REAL DC "Bias voltage " = 0;
.PARAM INT DEBUG "Wydruki kontrolne " = 0;
# MODEL PARAMETER
.COMMON REAL V1 "Voltage V1 " = 0;
.COMMON REAL V2 "Voltage V2 " = 0;
.COMMON REAL FREQ "Frequency " = 50;
.COMMON REAL TD "Time delay " = 1e-30;
.COMMON REAL TR "Time raise " = 1e-30;
.COMMON REAL TF "Time fall " = 1e-30;
.COMMON REAL PW "Pulse width " = 1e-30;
.COMMON REAL PER "Period " = 1e30;
# SINUSOIDAL
.COMMON REAL VSI "Sinusoidal amplitude " = 0;
.COMMON REAL TDSI "Time delay (sinusoid) " = 0;
.COMMON REAL TAU "Delay " = 1e30;
# TERMAL MODEL PARAMETERS
.COMMON REAL TC1 "1st order temperature coefficient " = 0;
.COMMON REAL TC2 "2nd order temperature coefficient " = 0;
.COMMON REAL TC3 "3rd value " = 1.01;
.COMMON REAL TCE "TC3 exponent coefficient " = 0;
.COMMON REAL TNOM "Nominal temperature (300.15K) " =300.15 ;
# OUTPUTS
.OUTPUT V(N1,N2,II):COMPLEX V;
# MODEL VARIABLES
.VAR REAL PULSE "pulse value ";
.VAR REAL TBKP "next timebreakpoint";
# INITIAL OR TEMPERATURE DEPENDENT PARAMETRS
.MEM INT DCT ;
.MEM INT V1T ;
.MEM INT V2T ;
.MEM INT VSIT;
.MEM INT TEMP_OLD;
.BEGIN

# CHECK PERIOD PARAMETERS
#ASSERT ((TR+PW+TF) > PER,": PER < (TR+PW+TF) ");

IF ( TEMP != TEMP_OLD ){
  DCT = DC * TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  V1T = V1 * TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  V2T = V2 * TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  VSIT = VSI* TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  TEMP_OLD = TEMP; # REMEMBER TEMPERATURE
}

IF(DC_ANALYSIS){
    V=DCT+V1T;
}

IF(TR_ANALYSIS){

    (PULSE, TBKP) = PULSE(TIME,V1T,V2T,TD,TR,PW,TF,PER);
    IF( TRAN:NR == 1 && V1T!=V2T && TBKP > TIME){
    IF(TRAN:EDITA){
        TRAN:SHEDULE ( N1, TBKP );
        TRAN:SHEDULE ( N2, TBKP );
    }
    TRAN:TIMEBREAKPOINT ( TBKP );
    }
    V= DCT+VSIT*ONE(TIME-TDSI)*SIN(2*PI*FREQ*(TIME-TDSI))*EXP(-(TIME-TDSI)/TAU) + PULSE;
}

IF(AC_ANALYSIS){ # AC
  (V.RE,V.IM)=MAGN*CEXP(COMPLEX(0,PHAS)); # COMPLEX
  IF ( DEBUG ){
    PRINT("AC ", ENAME, " V = (", V.RE, ", " V.IM, ") MAGN = ", MAGN , " PHAS = ", PHAS );
  }
}


IF ( DEBUG ){
    PRINT(NAME, "." , ENAME );
    PRINT("TIME =", TIME );
    PRINT("TEMP =", TIME );
    PRINT("# PULSE");
    PRINT("DC =", DC );
    PRINT("V1 =", V1 );
    PRINT("V2 =", V2 );
    PRINT("V1(T) =", V1T );
    PRINT("V2(T) =", V2T );
    PRINT("DC(T) =", DCT );
    PRINT("TD =", TD );
    PRINT("TR =", TR );
    PRINT("TF =", TF );
    PRINT("PW =", PW );
    PRINT("PER =", PER );
    PRINT("# SINUS");
    PRINT("VSI =", VSI );
    PRINT("VSI(T) =", VSIT );
    PRINT("TDSI =", TDSI );
    PRINT("TAU =", TAU );
    PRINT("FREQ =", FREQ );
    PRINT("MAGN =", MAGN );
    PRINT("PHAS =", PHAS );
    PRINT("# OUTPUT VALUE");
    PRINT("V = " , V );
}
```

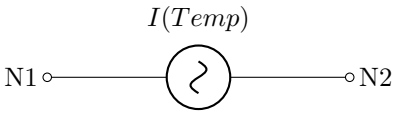
```
151
152 .END
```

10.10.2 Model źródła prądowego

Modelowanie zależności termicznych źródła umożliwia powiązanie podstawowych wartości parametrów z temperaturą zgodnie z zależnościami:

$$DCT = DC1 * TC \tag{10.9}$$
$$V1T = V1 * TC \tag{10.10}$$
$$V2T = V2 * TC \tag{10.11}$$
$$VSIT = VSI * TC \tag{10.12}$$

gdzie: DCT, $V1T$, $V2T$, $VSIT$ są odpowiednimi wartościami parametrów zależnych od temperatury, a TC dane jest wzorem (10.2). Schemat modelu źródła prądowego przedstawiono na rys 10.5. Model opisany jest parametrami zamieszczonymi w tab. 10.5.



Rys. 10.5: Model źródła prądowego (plik biblioteczny src/it.mdl)

Tab. 10.5: Lista parametrów modelu źródła prądowego IT

Nazwa	**Opis**	**Wartość**
parametry elementu		
MAGN	Amplituda w analizie małosygnałowej AC	0
PHAS	Faza w analizie małosygnałowej AC	0
TEMP	Temperatura elementu	300.15
parametry modelu - prostokąt		
DC	Poziom odniesienia DC	0
V1	Prąd dla poziomu V1	0
V2	Prąd dla poziomu V2	0
FREQ	Częstotliwość	50
TD	Czas opóźnienia impulsu	$1e-30$
TR	Czas narastania	$1e-30$
TF	Czas opadania	$1e-30$
PW	Szerokość impulsu	$1e-30$
PER	Okres	$1e30$
parametry modelu - sinusoida		
		kontynuacja na następnej stronie

kontynuacja z poprzedniej strony		
Nazwa	**Opis**	**Wartość**
VSI	Amplituda	0
TDSI	Czas opóźnienia	0
TAU	Opóźnienie	$1e30$
parametry termiczne modelu		
TC1	współczynnik 1 stopnia temperatury	0
TC2	współczynnik 2 stopnia temperatury	0
TC3	podstawa zależności wykładniczej	1.01
TCE	współczynnik w wykładniku dla TC3	0
TNOM	temperatura nominalna (K)	300.15

Kod modelu źródła prądowego przedstawiono na wydruku 10.17

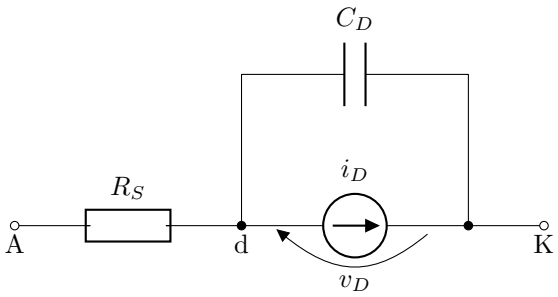
Wydruk 10.17: Model źródła prądowego IT (src/it.mdl)

```
# Revision : 1.2 (C) AIVA 2003-2010
#
# Current source model (time and temperature dependent)
#
#
# Element parameters
#
# MAGN Small signal magnitude (AC) (0)
# PHAS Small signal phase (AC) (0)
#
# Model parameters
#
# DC Bias voltage (0)
# V1 Current value V1 (0)
# V2 Current value V2 (0)
# FREQ Frequency (50)
# TD Time delay (1e-30)
# TR Time raise (1e-30)
# TF Time fall (1e-30)
# PW Pulse width (1e-30)
# PER Period (1e30)
#
# Sinusoidal response
#
# VSI Sinusoidal amplitude (0)
# TDSI Time delay (sinusoid) (0)
# TAU Delay (1e30)
#
# Thermal parameters
#
# TEMP temperature (300.15K)
#
# Model parameters
#
# TC1 1st order temperature coefficient (0)
# TC2 2nd order temperature coefficient (0)
# TC3 value (1.01)
# TCE TC3 exponent coefficient (0)
# TNOM nominal temperature in K (300.15K)
#
# TERMPCOEFF = 1+TC1*(TEMP-TNOM)+TC2*(TEMP-TNOM)^2+TC3^(TCE*(TEMP-TNOM));
#
# ------------------------------------------------------
# Library:
# .LIB "math/temp3.mdl"
# ------------------------------------------------------

.MODEL IT
.EXTERNAL N1,N2;
# ELEMENT PARAMETERS
.PARAM REAL MAGN "Small signal magnitude " = 0;
.PARAM REAL PHAS "Small signal phase " = 0;
.PARAM REAL TEMP "Temperature " = 300.15;
.PARAM REAL DC "Bias voltage " = 0;
# MODEL PARAMETER
.COMMON REAL V1 "Current value V1 " = 0;
.COMMON REAL V2 "Current value V2 " = 0;
.COMMON REAL FREQ "Frequency " = 50;
.COMMON REAL TD "Time delay " = 1e-30;
.COMMON REAL TR "Time raise " = 1e-30;
.COMMON REAL TF "Time fall " = 1e-30;
.COMMON REAL PW "Pulse width " = 1e-30;
.COMMON REAL PER "Period " = 1e30;
# SINUSOIDAL
.COMMON REAL VSI "Sinusoidal amplitude " = 0;
.COMMON REAL TDSI "Time delay (sinusoid) " = 0;
.COMMON REAL TAU "Delay " = 1e30;
# TERMAL MODEL PARAMETERS
```

```
.COMMON REAL TC1 "1st order temperature coefficient " = 0;
.COMMON REAL TC2 "2nd order temperature coefficient " = 0;
.COMMON REAL TC3 "3rd value " = 1.01;
.COMMON REAL TCE "TC3 exponent coefficient " = 0;
.COMMON REAL TNOM "Nominal temperature (300.15K) " =300.15 ;
# OUTPUTS
.OUTPUT V(N1,N2,II):COMPLEX OV;
.OUTPUT I(N1,N2):REAL I;
# MODEL VARIABLES
.VAR REAL PULSE "pulse value ";
.VAR REAL TBKP "next timebreakpoint";
# INITIAL OR TEMPERATURE DEPENDENT PARAMETRS
.MEM INT DCT ;
.MEM INT V1T ;
.MEM INT V2T ;
.MEM INT VSIT;
.MEM INT TEMP_OLD;
.BEGIN

# CHECK PERIOD PARAMETERS
ASSERT ((TR+PW+TF) > PER,": PER < (TR+PW+TF) ");

IF ( TEMP != TEMP_OLD ){
  DCT = DC1* TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  V1T = V1 * TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  V2T = V2 * TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  VSIT = VSI* TEMP3( TEMP, TNOM, TC1, TC2, TC3, TCE );
  TEMP_OLD = TEMP; # REMEMBER TEMPERATURE
}

IF(DC_ANALYSIS){
    I=DCT+V1T;
}

IF(TR_ANALYSIS){

    (PULSE, TBKP) = PULSE(TIME,V1T,V2T,TD,TR,PW,TF,PER);
    IF( TRAN:NR == 1 && V1T!=V2T && TBKP > TIME){
    IF(TRAN:EDITA){
        TRAN:SHEDULE ( N1, TBKP );
        TRAN:SHEDULE ( N2, TBKP );
    }
    TRAN:TIMEBREAKPOINT ( TBKP );
    }
    I= DCT+VSIT*ONE(TIME-TDSI)*SIN(2*PI*FREQ*(TIME-TDSI))*EXP(-(TIME-TDSI)/TAU) + PULSE;
}

IF(AC_ANALYSIS){ # AC
  (I.RE,I.IM)=MAGN*CEXP(COMPLEX(0,PHAS)); # COMPLEX
  IF ( DEBUG ){
    PRINT("AC ", ENAME, " V = (", I.RE, ", " I.IM, ") MAGN = ", MAGN , " PHAS = ", PHAS );
  }
}


IF ( DEBUG ){
    PRINT(NAME, "." , ENAME );
    PRINT("TIME =", TIME );
    PRINT("TEMP =", TIME );
    PRINT("# PULSE");
    PRINT("DC =", DC );
    PRINT("V1 =", V1 );
    PRINT("V2 =", V2 );
    PRINT("V1(T) =", V1T );
    PRINT("V2(T) =", V2T );
    PRINT("DC(T) =", DCT );
    PRINT("TD =", TD );
    PRINT("TR =", TR );
    PRINT("TF =", TF );
    PRINT("PW =", PW );
    PRINT("PER =", PER );
    PRINT("# SINUS");
    PRINT("VSI =", VSI );
    PRINT("VSI(T) =", VSIT );
    PRINT("TDSI =", TDSI );
    PRINT("TAU =", TAU );
    PRINT("FREQ =", FREQ );
    PRINT("MAGN =", MAGN );
    PRINT("PHAS =", PHAS );
    PRINT("# OUTPUT VALUE");
    PRINT("I = " , I );
}

.END
```

10.11 Model diody

Model DS jest w pełni zgodny z modelem diody z programu SPICE 2G6[L.W80]. Schemat modelu diody przedstawiono na rys 10.6.



Rys. 10.6: Model wielkosygnałowy diody (plik biblioteczny d/ds.mdl)

Model opisany jest następującymi parametrami zamieszczonymi w tab. 10.6.

Tab. 10.6: Lista parametrów diody

Nazwa	Opis	Wartość
parametry elementu		
AREA	współczynnik zwielokrotnienia	1
parametry modelu		
IS	prąd nasycenia	$1e-14A$
RS	rezystancja szeregowa	$1e-6\Omega$
N	współczynnik emisji	1
BV	napięcie przebicia wstecznego	$0.0=\infty$
IBV	prąd przebicia wstecznego dla $v_D=-BV$	$1e-3$
NBV	współczynnik	1
TT	czas przelotu (sec)	0
CJO	pojemność złączowa diody (pF)	0
VJ	potencjał złącza p-n (V)	1
M	współczynnik dla złącza p-n	0.5
FC	współczynnik dla pojemności dyfuzyjnej	0.5
EG	bariera potencjałów (barrier height) (eV)	1.11
XTI	współczynnik temperatury wykładnika IS	3
TNOM	temperatura nominalna (K)	300.15

Równania modelu diody w odpowiednich zakresach pracy przedstawiono poniżej:

dla $v_D \le -BV_{TEMP}$

$$i_D = -IS_{TEMP}\left[\exp\left(\frac{-BV_{TEMP}-v_D}{V_T}\right)-1-\frac{BV_{TEMP}}{V_T}\right]$$

dla $v_D = BV_{TEMP}$

$$i_D = IBV$$

dla $-BV_{TEMP} < v_D < -5 * N * V_T$

$$i_D = -IS_{TEMP} + GMIN * v_D$$

dla $v_D \geq -5 * N * V_T$

$$i_D = -IS_{TEMP} * \left[\exp\left(\frac{v_D}{N * V_T}\right) - 1\right]$$

Pojemność C_D modeluje jest sumą pojemności złączowej i dyfuzyjnej

$$C_D = C_j + C_d \tag{10.13}$$

Pojemność złączowa C_j opisana jest równaniami:

dla $v_D \leq VJ * FC$

$$C_j = CJO_{TEMP} * \left(1 - \frac{v_D}{VJ_{TEMP}}\right)^{-M}$$

dla $v_D > VJ * FC$

$$C_j = \frac{CJO_{TEMP}}{(1 - FC)^{(M+1)}} * \left(1 - FC * (M + 1) + M * \frac{v_D}{VJ_{TEMP}}\right)$$

C_d opisano wzorem:

$$C_d = IS_{TEMP} * \frac{TT}{N * V_T} * \exp\left(\frac{v_D}{N * V_T}\right)$$

Parametry zależne od temperatury opisane są równaniami:

$$\begin{aligned}
IS_{TEMP} &= AREA * I_S * \left(\frac{T}{t_R EF}\right)^{\left(\frac{XTI}{N}\right)} * \exp\left(\frac{EG * (T - T_{NOM})}{V_T * T_{NOM} * N}\right) \\
VJ_{TEMP} &= VJ * \left(\frac{T}{T_{NOM}}\right) \\
&\quad -3 * V_T * ln\left(\frac{T}{T_{NOM}}\right) - EG_{TEMP} + EG * \left(\frac{T}{T_{NOM}}\right) \\
EG_{TEMP} &= 1.16 - 70.2e-4 * \frac{T^2}{T + 1108} \\
CJO_{TEMP} &= AREA * CJO * \\
&\quad \left[1 + M * \left(4e-4 * (T - T_{NOM}) + 1 - \frac{VJ_{TEMP}}{VJ}\right)\right]
\end{aligned}$$

Wydruk 10.18: Model diody półprzewodnikowej (d/ds.mdl)

```
1 # Revision : 1.1 (C) AIVA 2003
2 #
3 # SPICE like diode model
4 #
5 # Element parameters :
6 # AREA - area factor (1)
```

```
7  # Model parameters :
8  # IS - saturation curent (1e-14 A)
9  # N - emission coefficient (1)
10 # BV - reverse breakdown "knee" voltage (infinite)
11 # IBV - reverse breakdown "knee" current (1e-3)
12 # NBV - reverse breakdown ideality factor (1)
13 # TT - transient time (0 sec)
14 # CJO - zero-bias junction capacitance (0 pF)
15 # VJ - p-n potential (1 V)
16 # M - p-n grading coefficient (0.5)
17 # FC - forward-bias depletion capacitance coefficient(0.5)
18 # EG - bangap voltage (barrier height) (1.11 eV)
19 # XTI - IS temeprature exponent (3)
20 # TNOM - nominal temperature (300.15°K)
21
22 .MODEL DS
23 .OPTI LIMU=OFF,CLUB=8,CLLB=2, DEBU=0;
24 .EXTERNAL A,K; # EXTERNAL NODES
25 .INTERNAL AI; # INTERNAL NODES
26 # INPUT
27 .INPUT V(AI,K):REAL V_D;
28 # OUTPUTS
29 .OUTPUT J(AI,K):REAL I_D;
30 .OUTPUT C(AI,K):REAL C_D; # J(AI,K):
31 .OUTPUT R(A,AI):REAL R_S;
32 # ELEMENT CONTROL PARAMETERS
33 .PARAM INT LIM_ON "turn on/off input limit " = 0;
34 .PARAM INT DEBUG "debug printout " = 0;
35 .PARAM INT CAP_ON "on-off capacitors " = 1;
36 # ELEMENT PARAMETERS
37 .PARAM REAL TEMP "temperature " = 300.15;
38 .PARAM REAL AREA "area factor (1) " = 1;
39 # MODEL PARAMETERS
40 .COMMON REAL IS "saturation curent (1e-14 A) " = 1E-14;
41 .COMMON REAL RS "resistance (Ohm) " = 1e-9;
42 .COMMON REAL N "emission coefficient (1) " = 1;
43 .COMMON REAL BV "reverse breakdown knee voltage (infinite) " = 100;
44 .COMMON REAL IBV "reverse breakdown knee current (1e-10) " = 1E-3;
45 .COMMON REAL NBV "reverse breakdown ideality factor (1) " = 1.0;
46 .COMMON REAL TT "transient time (0 sec) " = 0;
47 .COMMON REAL CJO "zero-bias junction capacitance (0 pF) " = 1e-6;
48 .COMMON REAL VJ "p-n potential (1 V) " = 1;
49 .COMMON REAL M "p-n grading coefficient (0.5) " = 0.5;
50 .COMMON REAL FC "forward-bias depletion capacitance coefficient (0.5) " =0.5;
51 .COMMON REAL EG "bangap voltage (barrier height) (1.11 eV)" = 1.11;
52 .COMMON REAL XTI "IS temeprature exponent (3) " = 3;
53 .COMMON REAL TNOM "nominal temperature (300.15K) " = 300.15;
54 # PREPROCESSED VARS
55 .MEM REAL TEMP_OLD=-1;
56 # ELEMENT VARIABLES
57 .MEM REAL EG0,REAL RAT,REAL EGT,REAL TOL,REAL IT,REAL IBVV;
58 .MEM REAL IST,REAL NVT,REAL VT,REAL BVV,REAL BVTN;
59 .MEM REAL CJT,REAL VJT,REAL FCVJ,REAL FC1,REAL FC2,REAL FC3,REAL VCRIT;
60 # OUTPUT VARIABLES
61 .OUTPUT REAL CJUN;
62 .OUTPUT REAL CDIFF;
63 .OUTPUT REAL GD;
64 .OUTPUT REAL VDTEMP;
65 .BEGIN
66
67 IF( TEMP != TEMP_OLD ){
68 # TEMPERATURE CHANGED - FIRST CALL
69   TEMP_OLD = TEMP;
70
71 # CHECKING VALUES
72   IF((AREA<=0)||(IS<=0)||(N<1)) { ASSERT(1); }
73   IF((EG<0)||(TNOM<=0)||(XTI<0)){ ASSERT(2); }
74   IF((BV<0)||(IBV<=0)||(NBV<1)) { ASSERT(3); }
75
76 # PREPROCESSING
77   IST=IS*AREA;
78   VJT=VJ;
79   CJT=CJO*AREA;
80   EG0=1.16-7.02E-4*(TNOM*TNOM)/(TNOM+1108);
81   VT=(KBOLTZ/QELE)*TEMP;
82   NVT=N*VT;
83   VCRIT=NVT*LOG(NVT/(SQRT(2)*IST));
84   BVTN=NBV*VT;
85   RAT=TEMP/TNOM ;
86 # CHECKING VALUES
87   ASSERT ( NVT ==0 );
88   ASSERT ( VJ ==0 );
89   ASSERT ( VJT ==0 );
90   ASSERT ( BVTN ==0);
91   ASSERT ( IST ==0 );
92   IF(RAT!=1){
93     EGT=1.16-7.02E-4*(TEMP*TEMP)/(TEMP+1108);
94     IST=IST*(RAT^(XTI/N))*EXP((RAT-1)*EG/NVT);
95     VJT=VJT*RAT-3*VT*LOG(RAT)-EG0*RAT+EGT;
96     CJT=CJT*(1+M*(4E-4*(TEMP-TNOM)+(1-VJT/VJ)));
97   }
98   FCVJ=FC*VJT;
99   FC1=(1-FC)^(1+M);
100  FC2=1-FC*(1+M);
101  FC3=M/VJT;
102  BVV=BV;
103  TOL=1E-4*IBV;
```

```
104     IF(BV){
105       BVV=BV-BVTN*LOG(1+IBV/IST);
106       IT=0;
107       IBVV=0;
108       WHILE((ABS(IBVV-IBV)>TOL)&&(IT<=25)){
109         BVV=BV-BVTN*LOG(IBV/IST+1-BVV/BVTN);
110         IBVV=IST*(EXP((BV-BVV)/BVTN)-1+BVV/BVTN);
111         IT=IT+1;
112       }
113     }
114   }
115
116   IF( OPT_NRME == 0 && NR <= 1 && DC_ANALYSIS ){
117   # AUTOMATIC START POINT
118       V_D=VCRIT;
119   }ELSE{
120     IF( LIM_ON ){
121   # LIMIT INPUT VALUE INSIDE MODEL (LIM_ON!=0)
122       IF((BVV!=0)&&V_D<MIN(0,-BVV+10.0*NVT)){
123         VDTEMP= -V_D-BVV;
124         (VDTEMP)=PNJLIM(VDTEMP,(-(V_D_+BVV)),NVT,VCRIT);
125         V_D=-VDTEMP-BVV;
126       }ELSE{
127         (V_D)=PNJLIM(V_D,V_D_,NVT,VCRIT);
128       }
129     }
130   }
131
132   # OUTPUTS CALCULATION BLOCK
133
134   R_S = 0; # RESISTANCE RSef
135   IF ( RS != 0 ){
136     R_S = RS/AREA;
137   }
138
139   IF(V_D>(-5*NVT)){
140   # NORMAL AND SUBTHRESHOLD REGION (I)
141       I_D=IST*(EXPO(V_D/NVT)-1);
142   }ELSE{
143     IF((BVV!=0)&&(V_D<(-BVV))){
144   # BREAKDOWN REGION (III)
145       I_D=-IST*((EXPO(-(BVV+V_D)/(VT*NBV))-1)+BVV/(VT*NBV));
146     }ELSE{
147   # CUTOFF REGION (II)
148       I_D=-IST;
149     }
150   }
151
152   # CAPACITANCE C_D = D_dif + D_jun
153   C_D = 0;
154   IF( CAP_ON ){
155   # JUNCTION CAPACITANCE C_j
156     IF(V_D>FCVJ){
157       CJUN=(FC2+V_D*FC3)*CJT/FC1;
158     }ELSE{
159       CJUN=CJT/((1-V_D/VJT)^M);
160     }
161   # DIFFUSION CAPACITANCE C_d
162     GD=(TT*IST/NVT)*EXPO(V_D/NVT);
163     CDIFF=TT*GD;
164     C_D=CDIFF+CJUN;
165   }
166
167
168   IF(DEBUG==1){
169     PRINT("# ", NAME,".",ENAME, " TIME:", TIME, " NR:", NR);
170     PRINT(" VD = ", V_D, " VD_ = ", V_D_ );
171
172     IF(V_D>(-5*NVT)){
173       IF(DEBUG>1){ PRINT(" NORMAL AND SUBTHRESHOLD REGION (I)" ); }
174     }ELSE{
175       IF((BVV!=0)&&(V_D<(-BVV))){
176         IF(DEBUG>1){ PRINT(" BREAKDOWN REGION (III)"); }
177       }ELSE{
178         IF(DEBUG>1){ PRINT(" CUTOFF REGION (II)"); }
179       }
180     }
181     PRINT(" ID = ", I_D );
182     PRINT(" CD = ", C_D );
183     PRINT("# END OF ",NAME,".",ENAME);
184   }
185   .END
```

10.12 Model diody tunelowej

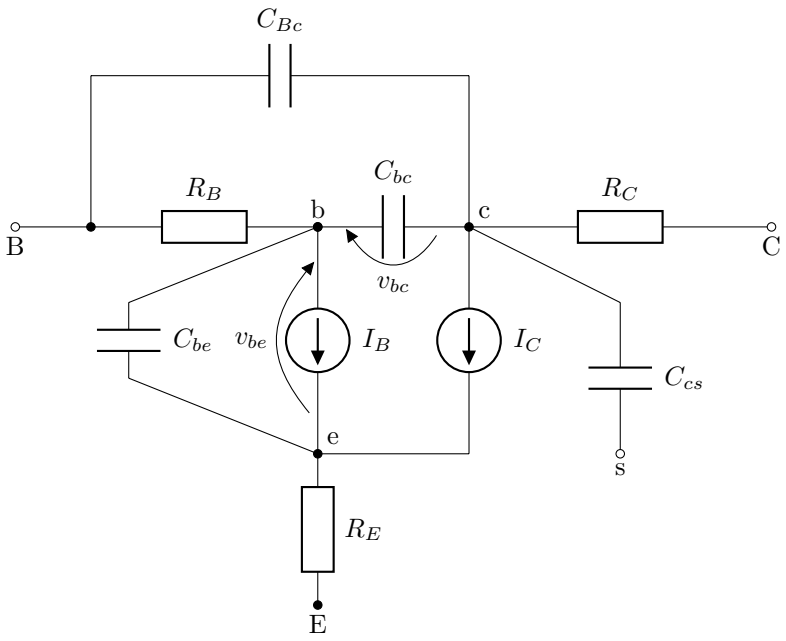
Model diody tunelowej przedstawiono na wydruku 10.19.

Wydruk 10.19: Model diody tunelowej (plik biblioteczny d/dtun.mdl).

```
# Revision
#
# Tunel diode model
#
# Individual parameters :
# AREA area factor (1).
# Common parameters :
# N material factor for GaAs: N <3.6-4.2> for
# Ge: N <3.2-3.6> (4),
# IP peak current in A (10 m),
# UP peak voltage in V (0.12),
# UV valley voltage in V (0.52),
# UPP parameter of linear approximation of static characteristic
# in high current region in V (1),
# FI junction potential in V (1.42),
# CV zero-bias junction capacitance in F (25p).
#
#-----------------------------------------------
# Calling :
# .MODEL model_line_name DS [model_par_name=value,...]
# model_line_name.element_name a n [element_par=value, ...]
# where:
# a - anode
# n - cathode
#-----------------------------------------------
.MODEL DTUN
.EXTERNAL A,K
.INPUT V(A,K):REAL VD;
.OUTPUT G(A,K):REAL ID;
# PREPROCESSED VARS
.MEM REAL IS;
.MEM REAL UT;
.MEM REAL A1;
.MEM REAL B1;
.MEM REAL C1;
.MEM REAL TEMP_OLD=-1;
# ELEMENT PARAMETERS
.PARAM REAL AREA "area factor (1) " = 1;
.PARAM REAL TEMP "temperature " = 300.15;
# MODEL PARAMETERS
.COMMON REAL N "emission coefficient (4) " = 4;
.COMMON REAL IP "peak current in A (10 m) " = 10miliA;
.COMMON REAL UP "peak voltage in V (0.12) " = 0.12;
.COMMON REAL UV "valley voltage in V (0.52) " = 0.52;
.COMMON REAL UPP "parameter of linear approximation of static characteristic in high current region in V (1)" = 1;
.COMMON REAL FI "junction potential in V (1.42) " = 1.42;
.COMMON REAL CV "zero-bias junction capacitance in F (25p) " = 25picoF;
IF( TEMP != TEMP_OLD ){
    A=(UV/UP)^N;
    GTV=N*(IP/UP)*((1-N)*A+N-1)/((N-1+A)^2);
    B=UV-UPP;
    C=-IP/GTV;
    PP1=UP;
    WHILE(ABS(PP1-PP)>1E-5){
      DFPP1=(-B*C/(PP1^2))*EXP(B/PP1);
      PP=(C*EXP(B/PP1)-DFPP1*PP1)/(1-DFPP1);
      DFPP=(-B*C/(PP^2))*EXP(B/PP);
      PP1=(C*EXP(B/PP)-DFPP*PP)/(1-DFPP);
    }
    UT=PP1;
    IS=IP/EXP(UPP/UT);
    A1=N*(IP/UP)*(N-1)/((N-1)^2)+IS/UT;
    A=(1.1*UPP/UP)^N;
    B1=N*(IP/UP)*((1-N)*A+N-1)/((N-1+A)^2)+IS*EXP(1.1*UPP/UT)/UT;
    C1=IS*(EXP(1.1*UPP/UT)-1)+IP*N*1.1*UPP/(UP*(N-1+A));
} # END OF PREPROCESSING

    IF(VD<0){
      ID=A1*VD;
    } ELSE {
      IF(VD>1.1*UPP){
        ID=B1*(VD-1.1*UPP)+C1;
      } ELSE {
        A=(VD/UP)^N;
        ID=IS*(EXP(VD/UT)-1)+IP*N*VD/(UP*(N-1+A));
        POM=N*(IP/UP)*(N-1+A)-N*IP*VD*N*(VD/UP)^(N-1)/(UP^2);
      }
    }
    CAP=AREA*CV*SQRT(MAX(1E-20,FI-UV)/MAX(1E-20,FI-VD));
    ID=AREA*ID+CAP*VD';

.END
```

10.13 Model tranzystora BJT

Model tranzystora bipolarnego jest zgodny z modelem Gummel-Poon zastosowanym w programie SPICE 2G6[L.W80]. Schemat modelu tranzystora przedstawiono na rys. 10.7. Parametry modelu zostały podzielone na dwie grupy jak pokazano w tab. 10.7



Rys. 10.7: Model tranzystora BJT (plik biblioteczny bjt/bjt2.mdl).

Tab. 10.7: Lista parametrów modelu tranzystora bipolarnego BJT.

Nazwa	**Opis**	**Wartość**
	individual parameters	
AREA	współczynnik zwielokrotnienia	1
TYPE	typ tranzystora : 1 npn, -1 pnp	1
	common parameters	
BF	idealna maksymalna β_F dla pracy normalnej	100
BR	idealna maksymalna β_R dla pracy inwersyjnej	1
IS	prąd nasycenia	$1e-16A$
NF	współczynnik emisji dla pracy normalnej	1
NR	współczynnik emisji dla pracy wstecznej	1
ISC	prąd upływu złącza B-C	$0A$
ISE	prąd upływu złącza B-E	$0A$
		kontynuacja na następnej stronie

kontynuacja z poprzedniej strony

Nazwa	**Opis**	**Wartość**
NC	współczynnik emisji prądu upływu złącza B-C	2
NE	współczynnik emisji prądu upływu złącza B-E	1.5
VAF	napięcie Early'ego dla pracy normalnej	$0V = \infty$
VAR	napięcie Early'ego dla pracy inwersyjnej	$0V = \infty$
IKF	prąd kolana BF dla pracy normalnej dla dużych prądów	$0A = \infty$
IKR	prąd kolana BR dla pracy normalnej dla dużych prądów	$0A = \infty$
IRB	prąd bazy, dla którego rezystancja $r_B b$ osiąga połowę wartości maksymalnej RBM	$0A$
RB	rezystancja bazy	0Ω
RBM	minimalna rezystancja bazy	0Ω
CJC	pojemność złączowa B-C przy zerowej polaryzacji - C_{jc}	$0F$
CJE	pojemność złączowa B-E przy zerowej polaryzacji - C_{je}	$0F$
FC	współczynnik przedłużenia liniowego charakterystyk pojemności złączowych dla pracy normalnej	0.5
MJC	współczynniki potęgowy pojemności złączowej B-C - C_{jc}	0.33
MJE	współczynniki potęgowy pojemności złączowej B-E - C_{je}	0.33
VJC	potencjał wbudowany złącza B-C	$0.75V$
VJE	potencjał wbudowany złącza B-E	$0.75V$
XCJC	wewnętrzny współczynnik podziału pojemności dyfuzyjnej złącza of B-C	1
ITF	parametr zależności czasu przelotu TF dla dużych prądów	$0A$
PTF	nadmiar fazy dla częstotliwości $1/(TF*2*PI)$ w deg	$0deg$
TF	czas przelotu dla pracy normalnej	$0sec$
TR	czas przelotu dla pracy inwersyjnej	$0sec$
VTF	napięcie opisujące zależność czasu przelotu TV od napięcia U_{BC}	$0V = \infty$
XTF	zależność TF od napięcia polaryzacji	0
EG	przerwa energetyczna	$1.11eV$
TKF1	1 współczynnik temperaturowy dla IKF w $1/deg$	0
TKF2	2 współczynnik temperaturowy dla IKF w $1/deg^2$	0
TKR1	1 współczynnik temperaturowy dla IKR w $1/deg$	0
TKR2	2 współczynnik temperaturowy dla IKR w $1/deg^2$	0
TIB1	1 współczynnik temperaturowy dla IRB w $1/deg$	0
TIB2	2 współczynnik temperaturowy dla IRB w $1/deg^2$	0
TRM1	1 współczynnik temperaturowy dla RBM w $1/deg$	0
TRM2	2 współczynnik temperaturowy dla RBM w $1/deg^2$	0

kontynuacja na następnej stronie

kontynuacja z poprzedniej strony		
Nazwa	**Opis**	**Wartość**
TRB1	1 współczynnik temperaturowy dla RB w $1/deg$	0
TRB2	2 współczynnik temperaturowy dla RB w $1/deg^2$	0
XTB	wykładnik temperaturowej zależności BF (β_F) i BR (β_R)	0
XTI	wykładnik temperaturowej zależności dla prądu nasycenia	3
TNOM	temperatura nominalna	$300.15K$

Model stałoprądowy Prąd kolektora opisany jest zależnością

$$i_C = \frac{ibe - ibc}{q_B} - \frac{ibc}{brt} - ibcn$$

Prąd bazy opisany jest zależnością

$$i_B = \frac{ibe}{BF_{TEMP}} + iben + \frac{ibc}{BR_{TEMP}} + ibcn$$

gdzie:

$$\begin{aligned}
ibe &= \begin{cases} IS_{TEMP} * (\exp(\frac{v_{be}}{NF\ V_T}) - 1) + GMIN * v_{be} & v_{be} > -5\ NF\ V_T \\ (-IS_{TEMP}/v_{be} + GMIN) * v_{be} & v_{be} \le -5\ NF\ V_T \end{cases} \\
iben &= \begin{cases} ISE_{TEMP} * (\exp(\frac{v_{be}}{NE\ V_T}) - 1) & v_{be} > -5\ NF\ V_T \\ ISE_{TEMP} & v_{be} \le -5\ NF\ V_T \end{cases} \\
ibc &= \begin{cases} IS_{TEMP} * (\exp(\frac{v_{bc}}{NR\ V_T}) - 1) \\ +GMIN * v_{bc} & v_{bc} > -5\ NR\ V_T \\ (-IS_{TEMP}/v_{bc} + GMIN) * v_{bc} & v_{bc} \le -5\ NR\ V_T \end{cases} \\
ibcn &= \begin{cases} ISC_{TEMP} * (\exp(\frac{v_{bc}}{NC\ V_T}) - 1) & v_{bc} > -5\ NF\ V_T \\ ISC_{TEMP} & v_{bc} \le -5\ NF\ V_T \end{cases} \\
q_B &= \frac{q_1}{2}[1 + (1 + 4\ q_2)^{NK}] \\
q_1 &= \frac{1}{1 - \frac{v_{be}}{VAR} - \frac{v_{bc}}{VAF}} \\
q_2 &= \frac{IS_{TEMP}}{IKF_{TEMP}}(\exp(\frac{v_{be}}{NF\ V_T} - 1) + \frac{IS_{TEMP}}{IKR_{TEMP}}(\exp(\frac{v_{bc}}{NR\ V_T} - 1)
\end{aligned}$$

$$\begin{aligned}
IS_{TEMP} &= IS_{TNOM}\left(\frac{T}{TNOM}\right)^{XTI} \exp\left[\frac{q * EG * (T - TNOM)}{k * T * TNOM}\right] \\
BF_{TEMP} &= BF_{TNOM}\left(\frac{T}{TNOM}\right)^{XTB} \\
BR_{TEMP} &= BR_{TNOM}\left(\frac{T}{TNOM}\right)^{XTB} \\
ISE_{TEMP} &= ISE_{TNOM}\left(\frac{T}{TNOM}\right)^{(XTI-XTB)} \exp\left[\frac{q * EG * (T - TNOM)}{k * T * TNOM * NE}\right] \\
ISC_{TEMP} &= ISC_{TNOM} * \left(\frac{T}{TNOM}\right)^{(XTI-XTB)} \exp\left[\frac{q * EG * (T - TNOM)}{k * T * TNOM * NC}\right]
\end{aligned}$$

Uwzględnione są rezystancje obszarów bazy, kolektora i emitera RC,RB,RE. Rezystancja bazy R_B zależy od parametrów RBM, IRB. W zależności o podanych parametrów występuje różny sposób obliczania R_B.

- jeżeli podano parametr RBM i nie podano parametru IRB

$$RB_I = RBM + \frac{RB - RBM}{QB}$$

- jeżeli oba parametry RBM i IRB zostały ustawione, to

$$RB_I = 3 * (RB - RBM) * \frac{\tan(z) - z}{z * [\tan(z)]^2} + RBM$$

gdzie:

$$z = \frac{-1 + \sqrt{\frac{144 * i_b}{\pi^2 * IRB + 1}}}{\frac{244}{\pi^2} * \sqrt{\frac{i_b}{IRB}}}$$

Pojemności Pojemności dyfuzyjne i złączowe są modelowane za pomocą funkcji nieliniowych zależnych od wartości zmiennych sterujących. Pojemności C_{be},C_{bc},C_{Bc} i C_{cs} opisują gromadzenie ładunku elektrycznego w różnych obszarach tranzystora. Pojemność złączowa C_{be} modeluje gromadzenie ładunku w obszarze bazy. Dlatego też pojemność C_{be} została podzielona na dwie pojemności: pojemność dyfuzyjną i złączową. Pojemność dyfuzyjna opisana jest zależnością.

$$\begin{aligned} C_{dbe} &= TF * (GBE * QB - DQBDVBE * IBE)/QB2 \\ C_{be} &= C_{dbe} + CJS(CJE_{TEMP}, UBE, VJE_{TEMP}, MJE, FC) \\ C_{bc} &= TR * GBC + CJS(CJC_{TEMP}, UBC, VJC_{TEMP}, MJC, FC) \end{aligned}$$

gdzie:

$$GBE = \frac{\delta\ ibe}{\delta\ v_{be}}$$
$$GBC = \frac{\delta\ ibc}{\delta\ v_{bc}}$$

$$C_{Bc} = CJS(CJXT, UBX, VJC_{TEMP}, MJC, FC)$$
$$C_{cS} = CJS(CJS_{TEMP}, UCS, VJS_{TEMP}, MJS, FC)$$

Zależności temperaturowe podano poniżej.

$$\begin{aligned} CJE_{TEMP} &= CJE * \{1 + MJE * \left[0.0004 + (T - TNOM) + \left(1 - \frac{VJE_{TEMP}}{VJE}\right)\right]\} \\ CJC_{TEMP} &= CJC * \{1 + MJC * \left[0.0004 + (T - TNOM) + \left(1 - \frac{VJC_{TEMP}}{VJC}\right)\right]\} \\ CJS_{TEMP} &= CJS * \{1 + MJS * \left[0.0004 + (T - TNOM) + \left(1 - \frac{VJS_{TEMP}}{VJS}\right)\right]\} \end{aligned}$$

$$VJE_{TEMP} = VJE * \left(\frac{T}{TNOM}\right) - 3 * V_T * \ln\left(\frac{T}{TNOM}\right) - EG_{TEMP} + EG * \left(\frac{T}{TNOM}\right)$$

$$VJC_{TEMP} = VJC * \left(\frac{T}{TNOM}\right) - 3 * V_T * \ln\left(\frac{T}{TNOM}\right) - EG_{TEMP} + EG * \left(\frac{T}{TNOM}\right)$$

$$VJS_{TEMP} = VJS * \left(\frac{T}{TNOM}\right) - 3 * V_T * \ln\left(\frac{T}{TNOM}\right) - EG_{TEMP} + EG * \left(\frac{T}{TNOM}\right)$$

gdzie EG_{TEMP} jest dany równaniem 10.18.

Plik biblioteczny modelu tranzystora BJT zamieszczono na wydruku 10.20.

Wydruk 10.20: Model tranzystora bipolarnego BJT (bjt/bjt2.mdl).

```
# Revision (C) AIVA 2010
#
# Modified (HSPICE level 2 like) Gummel-Poon bipolar transistor's model.
# Includes quasi-saturation model based on the paper by G.M.Kull et. al.
# and a variable base resistance model
#
# Effects NOT included in comparison with SPICE :
# - collector-substrate capacitance,
# - noises.
#
# Element parameters :
# AREA area factor (1),
# TYPE transistor's type selector : 1 npn, -1 pnp (1).
# Model parameters :
# BF ideal maximum forward beta (100),
# BR ideal maximum reverse beta (1),
# IS transport saturation current in A (0.1f),
# NF forward current emission coefficient (1),
# NR reverse current emission coefficient (1),
# ISC B-C leakage saturation current in A (0),
# ISE B-E leakage saturation current in A (0),
# NC B-C leakage emission coefficient (2),
# NE B-E leakage emission coefficient (1.5),
# VAF forward Early voltage in V (0=infinity),
# VAR reverse Early voltage in V (0=infinity),
# IKF corner for BF high current roll-off in A (0=infinity),
# IKR corner for BR high current roll-off in A (0=infinity),
# IRB current where rbb' falls halfway to RBM in A (0),
# RB base resistance in Ohms (0),
# RBM minimum high current resistance in Ohms (0),
# CJC B-C zero bias depletion capacitance in F (0),
# CJE B-E zero bias depletion capacitance in F (0),
# FC coefficient for forward bias depletion capacitance (0.5),
# MJE B-E junction grading factor (0.33),
# MJC B-C junction grading factor (0.33),
# VJC B-C built-in potential in V (0.75),
# VJE B-E built-in potential in V (0.75),
# XCJC internal base fraction of B-C depletion capacitance (1),
# ITF high-current parameter for effect on TF in A (0),
# PTF excess phase at frequency 1/(TF*2*PI) in deg (0),
# TF ideal forward transit time in sec. (0),
# TR reverse transit time in sec. (0),
# VTF voltage describing UBC dependence of TF in V (0=infinity),
# XTF TF bias dependence coefficient (0),
# EG energy gap in eV (1.11),
# TKF1 1st order temperature coefficient for IKF in 1/deg (0),
# TKF2 2nd order temperature coefficient for IKF in 1/deg^2 (0),
# TKR1 1st order temperature coefficient for IKR in 1/deg (0),
# TKR2 2nd order temperature coefficient for IKR in 1/deg^2 (0),
# TIB1 1st order temperature coefficient for IRB in 1/deg (0),
# TIB2 2nd order temperature coefficient for IRB in 1/deg^2 (0),
# TRM1 1st order temperature coefficient for RBM in 1/deg (0),
# TRM2 2nd order temperature coefficient for RBM in 1/deg^2 (0),
# TRB1 1st order temperature coefficient for RB in 1/deg (0),
# TRB2 2nd order temperature coefficient for RB in 1/deg^2 (0),
# XTB forward and reverse beta temperature exponent (0),
# XTI saturation current temperature exponent (3),
# RC (0)
# TNOM nominal temperature in K (300.15).
#


.FUNCTION (REAL RR) GETEX(REAL COEFF1,REAL COEFF2,REAL X)
.BEGIN
    RR=1+COEFF1*X+COEFF2*X*X;
.END
#
# Modelling depletion capacitance (as in spice)
#
.FUNCTION (REAL RR1) CJS(REAL C0,REAL UJ,REAL VJ,REAL MJ,REAL FC)
.BEGIN
    IF(UJ<FC*VJ){
      RR1=C0*EXP(-MJ*LOG(1-UJ/VJ));
    }ELSE{
      RR1 = C0*(1-FC*(1+MJ)+MJ*UJ/VJ)/EXP((1+MJ)*LOG(1-FC));
    }
```

```
77  .END
78
79  .FUNCTION (REAL RR2) DCJS(REAL C0,REAL UJ,REAL VJ,REAL MJ,REAL FC)
80  .BEGIN
81      IF(UJ<FC*VJ){
82        RR2=MJ*C0*((1-UJ/VJ)^(-MJ-1))/VJ;
83      }
84      RR2=C0*MJ/(VJ*EXP((1+MJ)*LOG(1-FC)));
85  .END
86
87  .MODEL BJT2
88  .OPTI LIMU=BOTH,CLUB=40,CLLB=5;
89  # NODES
90  .EXTERNAL CE,BE,EE;
91  .INTERNAL C,B,E;
92  # ELEMENT PARAMETERS
93  .PARAM INT DEBUG "debug printout " = 0;
94  .PARAM INT DEBUG_PRE "debug printout " = 0;
95  .PARAM INT CAP_ON "on-off capacitors " = 1;
96  .PARAM REAL TEMP "temperature " = 300.15;
97  .PARAM REAL AREA "area factor (1) " = 1;
98  # MODEL PARAMETER
99  .COMMON REAL LEVEL "Model level (BJTL2) - for compatibility" = 2 ;
100 .COMMON REAL TNOM "Nominal temperature (300.15K)" =300.15 ;
101 .COMMON REAL TYPE "(1) - NPN, -1 - PNP" =1 ;
102 .COMMON REAL IS "transport saturation current in A (0.1f)" =1E-16 ;
103 .COMMON REAL BF "Forward beta (100)" =100 ;
104 .COMMON REAL BR "ideal maximum reverse beta (1)" =1 ;
105 .COMMON REAL NF "reverse current emission coefficient (1)" =1 ;
106 .COMMON REAL NR1 "reverse current emission coefficient (1)" =1 ;
107 .COMMON REAL ISE "B-E leakage saturation current in A (0)" =0 ;
108 .COMMON REAL ISC "B-C leakage saturation current in A (0)" =0 ;
109 .COMMON REAL NE "B-E leakage emission coefficient (1.5)" =1.5 ;
110 .COMMON REAL NC "B-C leakage emission coefficient (2)" =2 ;
111 .COMMON REAL VAF "forward Early voltage in V (0=infinity)" =0 ;
112 .COMMON REAL VAR "reverse Early voltage in V (0=infinity)" =0 ;
113 .COMMON REAL IKF "corner for BF high current roll-off in A (0=infinity)" =0 ;
114 .COMMON REAL IKR "corner for BR high current roll-off in A (0=infinity)" =0 ;
115 .COMMON REAL IRB "current where rbb' falls halfway to RBM in A (0)" =0.1 ;
116 .COMMON REAL RB "base resistance in Ohms (0)" =100 ;
117 .COMMON REAL RBM "minimum high current resistance in Ohms (0)" =10 ;
118 .COMMON REAL VJC "B-C built-in potential in V (0.75) " =0.75 ;
119 .COMMON REAL VJE "B-E built-in potential in V (0.75) " =0.75 ;
120 .COMMON REAL VJS "Substarte-junction built-in potential in V (0.75) " =0.75 ;
121 .COMMON REAL MJE "B-E junction grading factor (0.33) " =0.333 ;
122 .COMMON REAL MJC "B-C junction grading factor (0.33) " =0.333 ;
123 .COMMON REAL MJS "Substarte-junction grading factor (0.33)" =0.333 ;
124 .COMMON REAL FC "coefficient for forward bias depletion capacitance (0.5)"=0.5 ;
125 .COMMON REAL TF "ideal forward transit time in sec. (0), " =0 ;
126 .COMMON REAL TR "reverse transit time in sec. (0), " =0 ;
127 .COMMON REAL PTF "excess phase at frequency 1/(TF*2*PI) in deg (0)"=0 ;
128 .COMMON REAL XTF "TF bias dependence coefficient (0)" =0 ;
129 .COMMON REAL VTF "voltage describing UBC dependence of TF in V (0=infinity)"=0 ;
130 .COMMON REAL ITF "high-current parameter for effect on TF in A (0)"=0 ;
131 .COMMON REAL RC "Collector resistance(0)" =10 ;
132 .COMMON REAL RE "Emitter resistance(0)" =1 ;
133 .COMMON REAL EG "energy gap in eV (1.11)" =1.11 ;
134 .COMMON REAL TKF1 "1st order temperature coefficient for IKF in 1/deg (0)" =0;
135 .COMMON REAL TKF2 "2nd order temperature coefficient for IKF in 1/deg^2 (0)" =0;
136 .COMMON REAL TKR1 "1st order temperature coefficient for IKR in 1/deg (0)" =0;
137 .COMMON REAL TKR2 "2nd order temperature coefficient for IKR in 1/deg^2 (0)" =0;
138 .COMMON REAL TRB1 "1st order temperature coefficient for RB in 1/deg (0)" =0;
139 .COMMON REAL TRB2 "2nd order temperature coefficient for RB in 1/deg^2 (0)" =0;
140 .COMMON REAL TIB1 "1st order temperature coefficient for IRB in 1/deg (0)" =0;
141 .COMMON REAL TIB2 "2nd order temperature coefficient for IRB in 1/deg^2 (0)" =0;
142 .COMMON REAL TRM1 "1st order temperature coefficient for RBM in 1/deg (0)" =0;
143 .COMMON REAL TRM2 "2nd order temperature coefficient for RBM in 1/deg^2 (0)" =0;
144 .COMMON REAL XTB "forward and reverse beta temperature exponent (0)" =0;
145 .COMMON REAL XTI "saturation current temperature exponent (3)" =3;
146 .COMMON REAL P_MAX "Max power (1W)" =1;
147 # MODEL CURRENTS
148 .OUTPUT J(C, E):REAL I_COL;
149 .OUTPUT J(B , E):REAL I_BAS;
150 # R - RESISTORS
151 .OUTPUT R(BE, B):REAL R_BAS;
152 .OUTPUT R(EE, E):REAL R_EMI;
153 .OUTPUT R(CE, C):REAL R_COL;
154 # C - CAPACITORS CURRENTS
155 .OUTPUT J(B, E):REAL I_CBE;
156 .OUTPUT J(B, C):REAL I_CBC;
157 .OUTPUT J(BE, C):REAL I_CBC1;
158 # CONTROLS
159 .INPUT V(B ,E ):REAL cVBE;
160 .INPUT V(B ,C ):REAL cVBC;
161 .INPUT V(BE,C ):REAL cVBX;
162 # ELEMENT VARIABLES
163 .MEM REAL pOIKF,REAL pOIKR ;
164 .MEM REAL pVT,REAL pIST,REAL pVCRIT,REAL NFxVT,REAL NRxVT,REAL NExVT,REAL
165 .MEM REAL NCxVT,REAL BRT,REAL BFT,REAL pISCT,REAL pISET;
166 .MEM REAL pCJXT,REAL pCJCT,REAL pCJET,REAL CJST,REAL pIRBP,REAL pRBP,REAL pRBMP;
167 .MEM REAL pRCP,REAL pREP,REAL pOVAF,REAL pOVAR;
168 .MEM REAL pTD;
169 .MEM REAL TEMP_OLD=0.1;
170 # MODEL VARIABLES
171 .VAR REAL XKT,REAL RAT;
172 .VAR REAL RATL,REAL BFAC,REAL FACL,REAL FAC,REAL PBFAC,REAL EGEF;
173 .VAR REAL IKFP;
```

```
174 .VAR REAL IKRP;
175 .VAR REAL pVJCT;
176 .VAR REAL pVJET;
177 .VAR REAL GBE;
178 .VAR REAL CBC;
179 .VAR REAL CBC1;
180 .VAR REAL DELT,REAL VT;
181 # OUTPUT VARIABLES
182 .OUTPUT REAL UBE;
183 .OUTPUT REAL UBC;
184 .OUTPUT REAL UBX;
185 #.OUTPUT REAL UCS;
186 .OUTPUT REAL IC;
187 .OUTPUT REAL IB;
188 .OUTPUT REAL IBE;
189 .OUTPUT REAL IBC;
190
191 .OUTPUT REAL IBEN;
192 .OUTPUT REAL IBCN;
193
194 .OUTPUT REAL Q1;
195 .OUTPUT REAL Q2;
196 .OUTPUT REAL SQ2;
197 .OUTPUT REAL QB2;
198 .OUTPUT REAL QB;
199
200 .OUTPUT REAL RBB;
201 .OUTPUT REAL TMP;
202 .OUTPUT REAL TMP1;
203 .OUTPUT REAL TMP2;
204 .OUTPUT REAL TMP3;
205 .OUTPUT REAL TMP4;
206 .OUTPUT REAL DK;
207
208 # CAPACITORS
209 .OUTPUT REAL CBE;
210 .OUTPUT REAL CDBE;
211 .VAR REAL pVTFP, REAL pITFP;
212 .VAR REAL VJST;
213 .VAR REAL ISCT;
214 # POWER
215 .OUTPUT REAL P_COL;
216 .OUTPUT REAL P_EMI;
217
218 .VAR REAL ARG;
219 .VAR REAL IST;
220 .VAR REAL CJET;
221 .VAR REAL CJCT;
222 .BEGIN # MODEL BODY
223
224 IF( TEMP != TEMP_OLD ){
225   TEMP_OLD = TEMP;
226 # CHECKING PARAMETERS VALUES
227   IF(LEVEL!=2){
228     PRINT(NAME, "WRONG MODEL LEVEL (BJTL2) = ",LEVEL);
229     ASSERT(-100);
230   }
231   ASSERT( TYPE!=1 && TYPE!=-1 );
232
233 # CALCULATING OF TEMP DEPEND PARAMETERS
234   XKT=KBOLTZ/QELE;
235   pVT=XKT*TEMP;
236   pIST=AREA*IS;
237   pVCRIT=pVT*LOG(pVT/(SQRT(2)*pIST));
238 # EMISSION COEFFICIENTS
239   NExVT=NE*pVT;
240   NCxVT=NC*pVT;
241   NFxVT=NF*pVT;
242   NRxVT=NR1*pVT;
243   BFT=BF;
244   BRT=BR;
245   pISET=AREA*ISE;
246   pISCT=AREA*ISC;
247
248   pVJET=VJE;
249   pVJCT=VJC;
250
251   pIRBP=IRB*AREA;
252   pRBP=RB/AREA;
253   pRBMP=MAX(RBM,RB)/AREA;
254   pRCP=RC/AREA;
255   pREP=RE/AREA;
256   RAT=TEMP/TNOM;
257
258
259   VT=pVT;
260
261 IF(RAT!=1){
262     DELT=TEMP-TNOM;
263     RATL=LOG(RAT);
264     BFAC=EXP(XTB*RATL);
265     FACL=(RAT-1)*EG/VT+XTI*RATL;
266     FAC=EXP(FACL);
267     PBFAC=-2*VT*(1.5*RATL+QELE*ARG);
268     EGEF=1.16-(7.02E-4*TEMP*TEMP)/(TEMP+1108);
269     ARG= -EGEF/(2*XKT)+1.1150877/(2*KBOLTZ*TNOM);
270 # transport saturation current*/
```

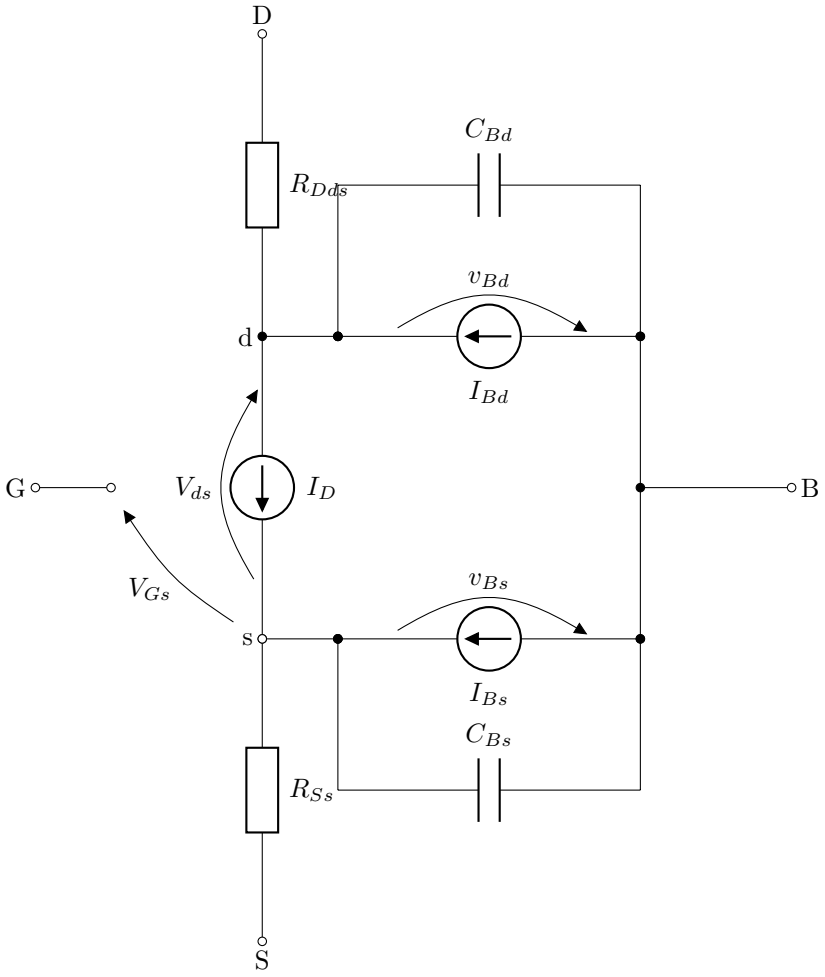
```
271     IST=IST*FAC;
272 # forward and reverse beta*/
273     BFT=BFT*BFAC;
274     BRT=BRT*BFAC;
275 # depletion capacitance parameters*/
276     pVJET=RAT*VJE+PBFAC;
277     IF(VJE!=0) {
278       CJET=CJET*(1+MJE*(4E-4*DELT-(pVJET-VJE)/VJE));
279     }
280     pVJCT=RAT*VJC+PBFAC;
281     IF(VJC!=0) {
282       CJCT=CJCT*(1+MJC*(4E-4*DELT-(pVJCT-VJC)/VJC));
283     }
284     VJST=RAT*VJS+PBFAC;
285     IF(VJS!=0){
286       CJST=CJST*(1+MJS*(4E-4*DELT-(VJST-VJS)/VJS));
287     }
288 #
289 # junctions leakage saturation currents
290 # ISET=ISET*EXP(FACL/NE)/BFAC;
291     ISCT=ISCT*EXP(FACL/NC)/BFAC;
292
293 # # high current beta degradation parameters
294 # IKFP=POW(IKFP,GETEX(TKF1,TKF2,DELT));
295 # IKRP=POW(IKRP,GETEX(TKR1,TKR2,DELT));
296 # # base resistance parameters*/
297 # IRBP=POW(IRBP,GETEX(TIB1,TIB2,DELT));
298 # RBP=RBP*GETEX(TRB1,TRB2,DELT);
299 # RBMP=RBMP*GETEX(TRM1,TRM2,DELT);
300 }
301
302 # OTHERS
303
304   pOVAF=0;
305   pOVAR=0;
306   pOIKF=0;
307   pOIKR=0;
308   IF(VAF!=0) { pOVAF=1/VAF; }
309   IF(VAR!=0) { pOVAR=1/VAR; }
310   IF(IKF!=0) { pOIKF=1/IKFP/AREA;}
311   IF(IKR!=0) { OIKR=1/IKRP/AREA; }
312   IF(VTF!=0) { {pVTFP=1/VTF/1.44;}
313
314   pTD=PTF*TF/RAD2DEG;
315   pITFP=ITF*AREA;
316   pVTFP=0;
317
318
319   IF( DEBUG_PRE ){
320     PRINT("PREPOCESSING: ", NAME, " TYPE ", TYPE, " TEMP ", TEMP, " TEMP_OLD ", TEMP_OLD, " pVT ",
            pVT ," pIST ", pIST
321       , " pVCRIT ", pVCRIT, " BRT = ", BRT , " BFT = ", BFT );
322   }
323
324   IF( DEBUG_PRE > 2 ){
325     PRINT ( " pOIKF = ", pOIKF ,
326     " pOIKR = ", pOIKR ,
327     " NFxVT = ", NFxVT ,
328     " NRxVT = ", NRxVT ,
329     " NExVT = ", NExVT ,
330     " NCxVT = ", NCxVT ,
331     " pISCT = ", pISCT ,
332     " pISET = ", pISET ,
333     " pCJXT = ", pCJXT ,
334     " pCJCT = ", pCJCT ,
335     " pCJET = ", pCJET ,
336     " CJST = ", CJST ,
337     " pIRBP = ", pIRBP ,
338     " pRBP = ", pRBP ,
339     " pRBMP = ", pRBMP ,
340     " pRCP = ", pRCP ,
341     " pREP = ", pREP ,
342     " pOVAF = ", pOVAF ,
343     " pOVAR = ", pOVAR ,
344     " pTD = ", pTD ,
345     " GMIN = ", GMIN );
346   }
347 }
348
349 # MODEL BODY
350
351 UBE=TYPE*cVBE;
352 UBC=TYPE*cVBC;
353
354 IF( OPT_NRME == 0 && NR <= 1 && DC_ANALYSIS ){
355     cVBE= TYPE*pVCRIT;
356     cVBC=-TYPE*pVCRIT;
357     UBE=TYPE*cVBE;
358     UBC=TYPE*cVBC;
359     IF(DEBUG_PRE){
360       PRINT(ENAME, " NRME: ", OPT_NRME, " NR AUTOMATYCZNY PUNKT STARTOWY UBE: ", UBE, " UBC:
              ", UBC);
361   }
362 }
363
364 # Main components of the collector and base currents
365 # IBE - current of the B-E junction (ideal),
```

```
# IBC − same as IBE but for B−C junction,
# IBEN − nonideal component of the B−E current (low level injection effects),
# IBCN − same as IBEN but for B−C current.

IF(UBE>(−5*NFxVT)) {
    IBE =pIST*(EXPO(UBE/NFxVT)−1)+GMIN*UBE;
    IBEN=pISET*(DXPO(UBE/NExVT)−1);
}ELSE{
    IBE =−pIST+GMIN*UBE;
    IBEN=−pISET;
}

IF(UBC>(−5*NRxVT)){
    IBC =pIST*(EXPO(UBC/NRxVT)−1)+GMIN*UBC;
    IBCN=pISCT*(DXPO(UBC/NCxVT)−1);
}ELSE{
    IBC =−pIST+GMIN*UBC;
    IBCN=−pISCT;
}


# GUMMEL'S CONSTANT AND ITS PARTIAL DERIVATIVES
Q1=1/(1−pOVAR*UBE−pOVAF*UBC);
Q2=pOIKF*IBE+pOIKR*IBC;
SQ2=SQRT(1+4*Q2);
QB=Q1*(1+SQ2)/2;
QB2=QB*QB;

ASSERT( QB <= 0 );

# DC PART OF THE COLLECTOR TO EMITTER CURRENT
IC = (IBE−IBC)/QB−IBC/BRT−IBCN;

# COLLECTOR CURRENT SOURCE
I_COL = TYPE*IC;

# DC PART OF THE BASE TO EMITTER CURRENT AND DERIV
IB = IBE/BFT+IBEN+IBC/BRT+IBCN;

I_BAS= TYPE*IB;

# BASE RESISTANCE
TMP=pRBP−pRBMP;
RBB=pRBMP+TMP/QB;
IF(pIRBP!=0){
    TMP2=MAX(IB/pIRBP,1E−9);
    TMP3=(−1+SQRT(1+14.59025*TMP2))/2.4317/SQRT(TMP2);
    TMP4=TAN(TMP3);
    RBB=pRBMP+(3*TMP*(TMP4−TMP3)/(TMP3*TMP4*TMP4));
}

ASSERT( RBB <= 0 );

# CHARGE STORAGE
IF(CAP_ON){
# Capacitance Cbc (CI,BI)
# internal base to collector depletion capacitance Cbc
  (CBC) = CJS(pCJCT,UBC,pVJCT,MJC,FC);
  CBC=CBC+TR*IBC:cVBC;
  ASSERT( CBC < 0 );

  I_CBC=CBC*cVBC';


# Capacitance CBc (B,CI) external base to collector depletion capacitance CBc
  (CBC1)=CJS(pCJXT,UBX,pVJCT,MJC,FC);
  ASSERT( CBC1 < 0 );

  I_CBC1=CBC1*cVBX';

# B−E diffusion capacitance (bias−dependency of the TF parameter)
  GBE=IBE:cVBE;
  CDBE=TF*(GBE*QB−QB:cVBE*IBE)/QB2;
  ASSERT( CDBE < 0 );

  IF(IBE>0){
    TMP=EXP(UBC*pVTFP);
    TMP2=IBE/(IBE+pITFP);
    TMP3=XTF*TMP*TMP2^2;
    TMP4=pITFP/IBE/IBE;
    TMP1=2*TMP2*TMP3*GBE*TMP4;
    DK=2*TMP2*GBE*TMP4;
    CDBE=CDBE*(1+TMP3)+TMP1*TF*IBE/QB;
  }

  ASSERT( CDBE < 0 );

# B−E depletion capacitance Cje
  (CBE)= CJS(pCJET,UBE,pVJET,MJE,FC);
  CBE=CDBE+CBE;

  ASSERT( CBE < 0 );
  I_CBE = CBE * cVBE';

}ELSE{
# Capacitors are OFF
  I_CBE = 0;
```

```
463     I_CBC = 0;
464     I_CBC1 = 0;
465   }
466
467   # Collector resistance
468   R_COL=pRCP;
469   ASSERT( R_COL <= 0 );
470
471   # Emitter resistance
472   R_EMI=pREP;
473   ASSERT( R_EMI <= 0 );
474
475   # Base resistance
476   R_BAS=RBB;
477   ASSERT( R_BAS <= 0 );
478
479   P_COL = UBC * IC;
480   P_EMI = UBE * IB;
481
482   # Control printouts
483   IF( DEBUG>=1 ) {
484     PRINT(NAME,".",ENAME, " NR", NR);
485     PRINT(" IB = IBE/BFT+IBEN+IBC/BRT+IBCN");
486     PRINT(" UBE =",UBE );
487     PRINT(" UBC =",UBC );
488   #PRINT(" I_BAS =",I_BAS);
489     PRINT(" IB =",IB );
490     PRINT(" IBE(junc ideal) =",IBE );
491   #PRINT(" I_COL =",I_COL);
492     PRINT(" IC =",IC );
493     PRINT(" IBC(junc ideal) =",IBC );
494     PRINT(" BETA=IC/IB =",IC/IB );
495     PRINT(" BFT =",BFT );
496     PRINT(" BRT =",BRT );
497     PRINT(" IBEN =",IBEN );
498     PRINT(" IBCN =",IBCN );
499     PRINT(" RE =",R_EMI);
500     PRINT(" RC =",R_COL);
501     PRINT(" RB =",R_BAS);
502     PRINT(" Cbe =",CBE );
503     PRINT(" Cbc =",CBC );
504     PRINT(" Cbc1 =",CBC1 );
505     IF( P_COL + P_EMI > P_MAX ) {
506       PRINT(" MAX POWER REACHED ", " P_COL=" , P_COL, " P_EMI=", P_EMI, " > P_MAX=" , P_MAX);
507     }
508   }
509   .END # End of BJT2
```

10.14 Model tranzystora MOS

Model tranzystora MOS jest kompatybilny z modelem MOS LEVEL 2 z programu SPICE 3F4 [A+81] w zakresie stałoprądowym (DC). Ładunek bramki może być opisany modelem Ward-Dutton'a lub modelem Meyer'a. Model wielkosygnałowy jest pokazany na rys. 10.8.



Rys. 10.8: Model wielkosygnałowy tranzystora MOS.

Model opisany jest szeregiem parametrów zamieszczonych w tab. 10.8.

Tab. 10.8: Lista parametrów modelu tranzystora MOS.

Nazwa	Opis	Wartość
parametry elementu		
OFF	Tranzystor początkowo wyłączony	0
L	Długość kanału	$1e-4$
W	Szerokość kanału	$1e-4$
AD	Powierzchnia drenu	$1e-6$
AS	Powierzchnia źródła	$1e-6$
PD	Parametr drenu	0.0
PS	Parametr źródła	0.0
NRD	Liczba kwadratów drenu	1.0
NRS	Liczba kwadratów źródła	1.0
parametry modelu		
TYPE	Typ kanału (1)-N kanałowy lub (-1)-P kanałozy	1
VTO	Napięcie progowe	$-1e5$
KP	Transkonduktancja	$2e-5$
GAMMA	Współczynnik polaryzacji podłoża	0
PHI	Potencjał Fermiego dla podłoża	0.6
LAMBDA	Modulacja długości kanału	0.0
RD	Rezystancja drenu	0.0
RS	Rezystancja źródła	0.0
CBD	Pojemność złączowa B-D	0.0
CBS	Pojemność złączowa B-S	0.0
IS	Prąd nasycenia złącz (diod) podłoża	$1e-14$
PB	Potencjał wbudowany złącz (diod) podłoża	0.5
CGSO	Pojemność jednostkowa zakładki B-S	0.0
CGDO	Pojemność jednostkowa zakładki G-D	0.0
CGBO	Pojemność jednostkowa zakładki G-B	0.0
RSH	Rezystancja powierzchniowa	1.0
CJ	Pojemność jednostkowa złącz (diod) podłoża	0.0
MJ	Bottom grading coefficient	0.5
CJSW	Side junction cap per area	0.0
MJSW	Side grading coefficient	0.3
JS	Prąd nasycenia złącz (diod) podłoża	0.0
TOX	Grubość warstwy tlenku	$1e-7$
LD	Lateral diffusion	0.0
WD	Lateral diffusion of the channel	0.0
UO	Ruchliwość nośników dla zerowej polaryzacji	600
FC	Współczynnik przedłużenia liniowego charakterystyk C złączowych dla dodatnich napięć	0.5

kontynuacja na następnej stronie

kontynuacja z poprzedniej strony		
Nazwa	**Opis**	**Wartość**
NSUB	Koncentracja nośników w podłożu	$1.45e10$
TPG	Typ bramki: 0-aluminiowa, ! = 0 - polikrzemowa. Typ przewodnictwa: = −1 - zgodny z podłożem, = 1 - przeciwny do podłoża	-1
NSS	Gęstość stanów powierzchniowych	0.0
DELTA	Efekt wąskiego kanału na napięcie przełączania	0.0
UEXP	*ang. Crit. field exp for mob. deg.*	0.0
UCRIT	*ang. Crit. field for mob. degradation*	$1e4$
VMAX	*ang. Maximum carrier drift velocity*	0.0
XJ	Głębokość złącza	0.0
NEFF	*ang. Total channel charge coeff.*	1.0
NFS	*ang. Fast surface state density*	0.0
TNOM	Temperatura nominalna	300.15
KF	*ang. Flicker noise coefficient*	0.0
AF	*ang. Flicker noise exponent*	1.0
NISI	Koncentracja nosników (dodatkowy parametr)	$1.45e16$
REFTEMP	Temperatura pomiaru parametrów	300.15
XQC	Współczynnik ładunku w kanale	0.5

Prąd drenu

Prąd drenu w ogólności opisany jest równaniem (10.14).

$$i_{ds} = \beta_{eff}\left(vgsx - vbin - \frac{1}{2}\eta\ vdsx\right) - \frac{2}{3}\gamma_{SD}\left[(PHI - (v_{bs} - vdsx))^{1.5} - (PHI - v_{bs})^{1.5}\right] \tag{10.14}$$

gdzie:

$$\begin{aligned} \beta_{eff} &= \beta\ \frac{\mu_{fact}}{l_f act} \\ vdsx &= \begin{cases} v_{ds} & v_{ds} \leq v_{dsat} \\ v_{dsat} & v_{ds} > v_{dsat} \end{cases} \end{aligned}$$

Modyfikacje równania (10.14) występują dla zakresu odcięcia dla $v_{gs} < vth$,

$$i_{ds} = 0$$

i dla zakresu podprogowego dla $v_{on} > vth$, $v_{gs} \leq v_{on}$ i $NFS > 0.0$.

$$i_{ds} = i_{ds} * \exp\left(\frac{v_{gs} - v_{on}}{xn * V_T}\right)$$

Napięcie odcięcia

Równanie na napięcie odcięcia wykorzystuje model ładunkowy do obliczenia efektu skracania kanału przy napięciu odcięcia. Równania modelują także efekt wąskiego kanału. Napięcie v_{on} jest wykorzystywane do modelowania obszaru podprogowego.

$$
\begin{aligned}
vth &= vbi + F_N * (PHI - v_{bs}) + \gamma * sarg \\
v_{on} &= vth + V_T * xn \\
vbin &= vbi + F_N * (PHI - v_{bs}) \\
V_T &= \frac{k * T}{q} \\
xn &= 1 + \frac{cfs}{COX} + \frac{cd}{COX} + F_N \\
cfs &= q * NFS \\
cd &= \frac{\delta Q_{dep}}{\delta v_{bs}} \\
F_N &= \frac{1}{8} * \frac{2 * \pi * \epsilon_{Si} * DELTA}{COX * W_e} \\
Q_{dep} &= COX * \gamma_{SD} * sarg \\
\gamma_{SD} &= GAMMA * (1 - argss - argsd) \\
argss &= \frac{XJ}{2 * L_e} * \left(\sqrt{1 + \frac{2 * xd * sarg}{XJ}} - 1 \right) \\
argsd &= \frac{XJ}{2 * L_e} * \left(\sqrt{1 + \frac{2 * xd * barg}{XJ}} - 1 \right)
\end{aligned}
$$

Redukcja ruchliwości nośników

Redukcja ruchliwości nośników μ_{eff} spowodowana jest poprzeczną składową pola elektrycznego wywołaną napięciem v_{gs} (10.15).

$$
\begin{aligned}
\mu_{fact} &= \left[\frac{UCRIT\ \epsilon_{Si}}{COX\ (v_{gs} - v_{on} - UTRA\ v_{ds}} \right]^{UEXP} \\
\mu_{eff} &= \mu_{fact}\ UO_{TEMP}
\end{aligned}
\tag{10.15}
$$

Napięcie nasycenia

Jeżeli parametr $VMAX$ nie został podany, to napięcie nasycenia v_{dsat} obliczane jest z zależności poniżej.

$$
\begin{aligned}
\eta &= 1 + F_N \\
\gamma_D &= \frac{\gamma_S D}{\eta} \\
vgsx &= \begin{cases} v_{gs} & if\ NFS = 0.0 \\ v_{gs} & if\ NFS > 0.0\ and\ v_{gs} \geq v_{on} \\ v_{on} & if\ NFS > 0.0\ and\ v_{gs} < v_{on} \end{cases} \\
v_{dsat} &= \frac{vgsx - vbin}{\eta} + \\
&\quad + \frac{1}{2}\gamma_D^2 \left[1 - \sqrt{1 + \frac{4}{\gamma_D^2} \left(\frac{vgsx - vbin}{\eta} + PHI - v_{bs} \right)} \right]
\end{aligned}
$$

Modulacja długości kanału

Jeżeli $LAMBDA > 0.0$,

$$l_{fact} = 1 - LAMBDA\ v_{ds}$$

Jeżeli $LAMBDA \leq 0.0$ i $NSUB > 0.0$ i $VMAX == 0.0$,

$$l_{fact} = 1 - \frac{xd}{L_e}\left[\frac{v_{ds} - v_{dsat}}{4} + \sqrt{1 + \left(\frac{v_{ds} - v_{dsat}}{4}^2\right)^2}\right]^{0.5}$$

Jeżeli parametr $VMAX$ został podany, i jeżeli $LAMBDA \leq 0.0$ i $NSUB > 0.0$ i $VMAX > 0.0$ wtedy,

$$l_{fact} = 1 - \frac{xd}{\sqrt{NEFF}\ L_e}\left[\sqrt{\left(\frac{VMAX\ xd}{2\sqrt{NEFF}\ \mu_{eff}}\right)^2 + v_{ds} - v_{dsat}} + \frac{VMAX\ xd}{2\ \sqrt{NEFF}\ \mu_{eff}}\right]$$

W modelu zagwarantowano, że długość kanału nie stanie się wartością ujemną:

$$\begin{aligned} xwb &= xd\ \sqrt{PB} \\ l_{eff} &= \begin{cases} L_e\ l_{fact} & if\ l_{eff} \geq xwb \\ \frac{xwb}{1+\frac{xwb - l_{eff}}{xwb}} & if\ l_{eff} < xwb \end{cases} \end{aligned}$$

Diody podłożowe

Model wykorzystuje modele diod podłożowych, które są ciągłe w każdym zakresie pracy (w odróżnieniu od modeli z programu SPICE [Fot97]. W modelu wykorzystano równania modelu diod tranzystora przedstawione poniżej:

- dla $v_{db} > -5V_T$

$$\begin{aligned} i_D &= TYPE\left[IS_{TEMP} * (\exp 1(\frac{v_D}{V_T}) - 1) + GMIN * v_D\right] \\ g_D &= TYPE\left[\frac{IS_{TEMP}}{V_T} * dxp1(\frac{v_D}{V_T}) + GMIN\right] \end{aligned}$$

- dla $v_{db} \leq -5 * V_T$

$$\begin{aligned} i_D &= TYPE\left[IS_{TEMP} * \exp 1(-5)\ (1 - \exp 1(5) + 5 + \frac{v_D}{V_T}) + GMIN * v_D\right] \\ g_D &= TYPE\left[\frac{IS_{TEMP}}{V_T} * dxp1(-5) + GMIN\right] \end{aligned}$$

gdzie:

i_D prąd diody

v_D napięcie na diodzie

g_D konduktancja diody

$V_T = \frac{k*T}{q}$

Model ładunkowy Ward-Dutton

W modelu WD obliczane są ładunki w bramki i podłoża z jednowymiarowego równania Poisson'a. Ładunek w kanale obliczany jest z zależności:

$$Q_c = -(Q_g + Q_b) \tag{10.16}$$

Równanie to odpowiada można zapisać w postaci wiążącej ładunki źródła i drenu:

$$Q_c = Q_s + Q_d \tag{10.17}$$

Niestety, ładunki dla drenu i źródła nie są tak dobrze zdefiniowane jak ładunki gromadzone w bramce i podłożu i muszą zostać obliczone przez podział ładunku w kanale. W modelu zastosowano podany poniżej sposób podziału ładunku kanału:

$$\begin{aligned} Q_d &= \begin{cases} \frac{1}{2}Q_c & linear\ region \\ XQC * Q_c & saturation\ region \end{cases} \\ Q_s &= \begin{cases} \frac{1}{2}Q_c & linear\ region \\ (1 - XQC) * Q_c & saturation\ region \end{cases} \end{aligned}$$

Ten sposób podziału powoduje nieciągłości na granicach pomiędzy obszarem liniowym i nasycenia. Równania ładunków są ciągłe dla $XQC = \frac{1}{2}$. Poniżej zamieszczono szereg użytecznych parametrów używanych w równaniach:

$$\begin{aligned} \gamma &= \frac{vth - vbi}{sarg} \\ C_o &= COX * W_e * L_e \\ v_g &= v_{gs} - v_{bs} - v_{bi} + PHI \\ v_s &= PHI - v_{bs} \\ v_z &= \begin{cases} PHI - v_{bs} + v_{ds} & v_{ds} \leq v_{dsat} \\ PHI - v_{bs} + v_{dsat} & v_{ds} > v_{dsat} \end{cases} \end{aligned}$$

Zakres akumulacji (Accumulation region) ($v_g \leq 0$):

$$\begin{aligned} Q_g &= C_o v_g \\ Q_b &= -Q_g \\ Q_c &= 0 \\ Q_d &= 0 \\ Q_s &= 0 \end{aligned}$$

Zakres odcięcia (Cut-off region) ($v_{gs} \leq vth$):

$$\begin{aligned} Q_g &= C_o * \gamma * \left[\sqrt{\frac{\gamma^2}{4} + v_g} - \frac{\gamma}{2} \right] \\ Q_b &= -Q_g \\ Q_c &= 0 \\ Q_d &= 0 \\ Q_s &= 0 \end{aligned}$$

Zakres włączenia (On region) ($v_{gs} > vth$):

$$\begin{aligned}
i &= v_g * \left(v_z^{0.5} + v_s^{0.5}\right) - \frac{2}{3}\gamma\left(v_z + v_z^{0.5}v_s^{0.5} + v_s\right) \\
&\quad -\frac{1}{2}(v_z + v_s)\left(v_z^{0.5} + v_s^{0.5}\right) \\
Q_g &= C_o * v_g - \frac{C_o}{i}\left\{\frac{1}{2}v_g(v_z + v_s)({v_z}^{0.5} + {v_s}^{0.5})\right. \\
&\quad -\frac{2}{5}\gamma\left[v_z^2 + v_z^{0.5}v_s^{0.5}(v_z + v_s) + v_s^2\right] \\
&\quad \left.-\frac{1}{3}(v_z^{0.5} + v_s^{0.5})(v_z^2 + v_z v_s + v_s^2)\right\} \\
Q_b &= -\frac{C_o\gamma}{i}\left\{\frac{2}{3}v_g(v_z + v_z^{0.5}v_s^{0.5} + v_s)\right. \\
&\quad -\frac{1}{2}\gamma(v_z + v_s)(v_z^{0.5} + v_s^{0.5}) \\
&\quad \left.-\frac{2}{5}\left[v_z^2 + v_z^{0.5}v_s^{0.5}(v_z + v_s) + v_z v_s + v_s^2\right]\right\} \\
Q_c &= -(Q_g + Q_b) \\
Q_d &= \begin{cases} \frac{1}{2}Q_c & v_{ds} \le v_{dsat} \\ XQC * Q_c & v_{ds} > v_{dsat} \end{cases} \\
Q_s &= \begin{cases} \frac{1}{2}Q_c & v_{ds} \le v_{dsat} \\ (1 - XQC) * Q_c & v_{ds} > v_{dsat} \end{cases}
\end{aligned}$$

Model Meyer'a wykorzystujący pojemności

$$v_{gb} = v_{gs} - v_{bs}$$

- Zakres odcięcia (Cut-off region) ($v_{gs} \le vth$):

$$\begin{aligned}
C_{gs} &= 0 \\
C_{gd} &= 0 \\
C_{gb} &= \begin{cases} C_O & v_{gb} \le vfb \\ \frac{C_O}{[1 + \frac{4}{\gamma^2}(v_{gb} - vfb)]^{0.5}} & v_{gb} > vfb \end{cases}
\end{aligned}$$

- Zakres nasycenia (Saturation region) ($v_{ds} > v_{dsat}$):

$$\begin{aligned}
C_{gs} &= \frac{2}{3}C_O \\
C_{gd} &= 0 \\
C_{gb} &= 0
\end{aligned}$$

- Zakres liniowy (Linear region) ($v_{ds} < v_{dsat}$):

$$\begin{aligned}
C_{gs} &= \frac{2}{3}C_O\left[1 - \frac{(v_{dsat} - v_{ds})^{0.5}}{(2v_{dsat} - v_{ds})^{0.5}}\right] \\
C_{gd} &= \frac{2}{3}C_O\left[1 - \frac{v_{dsat}^{0.5}}{(2v_{dsat} - v_{ds})^{0.5}}\right] \\
C_{gb} &= 0
\end{aligned}$$

Zależności temperaturowe

Oznaczenia:

$TNOM$ temperatura nominalna ($300.15K$)

T temperatura analizowanego układu

Ruchlowość

$$UO_{TEMP} = UO_{TNOM}\left(\frac{T}{TNOM}\right)^{BEX}$$
$$KP_{TEMP} = KP_{TNOM}\left(\frac{T}{TNOM}\right)^{BEX}$$

gdzie BEX jest parametrem modelu (-1.5). Napięcie odcięcia może zostać obliczone jako:

$$\begin{aligned} EG_{TEMP} &= 1.16 - \frac{7.02e^{-4}\ T^2}{T+1108} \\ n_i \quad \alpha \quad & T^{1.5}\exp\left(-\frac{EG_{TEMP}}{2kT}\right) \end{aligned}$$

EG_{TNOM} przerwa energetyczna w temperaturze $TNOM$

EG_{TEMP} przerwa energetyczna w temperaturze T

$$\begin{aligned} PHI &= 2V_T \ln\left(\frac{NSUB}{n_i}\right) \\ PHI &= 2\frac{T}{TNOM}PHI_{TNOM} - 2V_T\left[\frac{3}{2}\ln\left(\frac{T}{TNOM}\right) + \frac{1}{2k}\left(-\frac{EG_{TEMP}}{T} + \frac{EG_{TNOM}}{TNOM}\right)\right] \\ v_{bi} &= vbi_{TNOM} - 0.5[EG_{TEMP} - EG_{TNOM}]\ TPG \\ &\quad +0.5[PHI - PHI_{TNOM}] \\ vfb &= vbi - PHI \\ VTO &= vbi + GAMMA\sqrt{PHI} \end{aligned}$$

Kod modelu tranzystora MOS znajduje się w plikach bibliotecznych:

- lib/mos/mos2wd.mdl - model Ward-Dutton
- lib/mos/mos2mey.mdl - model Meyer'a

Wydruk 10.21: Model tranzystora MOS (Ward-Dutton).

```
1  # Revision : 1.1 (C) AIVA 2003-2010
2  #
3  # MODEL TRANZYSTORA MOS LEVEL 2 - VERSION 4
4  #
5  # OPIS MODELU:
6  # A. MODEL DLA DC PODOBNY DO MODELU SPICE 3F4/2G6 LEVEL 2 (**)
7  # 1. MODELOWANIE REZYSTANCJI SZEREGOWYCH DRENU I ZRODLA
8  # 2. Z MINIMALNA WARTOSCIA KONDUKTANCJI DS
9  # 3. OBLICZANIE VDSAT ZGODNIE Z TEORIA BAUM'A
10 # B. DIODY PODLOZOWE - DWA RODZAJE:
11 # 1. MODELOWANE CIAGLYMI FUNKCJAMI I POCHODNYMI
12 # 2. MODELOWANE NIECIAGLYMI FUNKCJAMI I POCHODNYMI ZGODNIE Z SUGESTIAMI
```

```
13 # ZAWARTYMI W KSIAZCE (*)
14 # C. MODELOWANIE LADUNKOW - WARD-DUTTON NA PODSTAWIE KSIAZKI (*) I
15 # MODYFIKACJA USUWAJACA BLAD MODELOWANIA LADUNKU QG
16 # D. ZABUDOWANE OGRANICZANIE NAPIEC NA ZLACZACH TRANZYSTORA MOS I DIOD
17 # 1. DOMYSLNIE WYLACZONE
18 # 2. STOSOWANE OGRANICZANIE LOGARYTMICZNE NAPIEC NA ZLACZACH
19 #
20 # (*) "FET MODELING FOR CIRCUIT SIMULATION" DILEEP A. DIVEKAR
21 # (**) KOD ZRODLOWEGO SPICE3F4/2G6
22
23 # FUNCTIONS USED IN MODEL BODY
24 #.INC "lib/fun.mdl"
25 .INC "lib/limvds.mdl"
26 .INC "lib/ctrlim.mdl"
27 .INC "lib/fetlim.mdl"
28 .INC "lib/pnjlim1.mdl"
29
30 .MODEL MOS2WD
31 .OPTI DERI=1,LIMU=BOTH,CLUB=10,CLLB=3;
32 # PINS
33 .EXTERNAL DE,GE,SE,BE,ZERO;
34 .INTERNAL DI,SI;
35 # FLOWS
36 .FLOW QG, QB, QD, QS;
37 # MODEL CONTROL 1-On, 0-Off
38 .PARAM REAL LIMIT_VDS "LIMIT VOLTAGE VDS FLAG" =0;
39 .PARAM REAL LIMIT_VBD "LIMIT VOLTAGE VBD FLAG" =0;
40 .PARAM REAL PN_ALFA "wspolczynnik skalowania dla pnjlimi()" =1;
41 .PARAM REAL DIODE_ON "0-OFF, 1-CONT. EQ., 2-NONCONT. EQ." =2;
42 .PARAM REAL DIODE_CAP_ON"DIODE CAPACITANCE 0-off, 1-on" =1;
43 .PARAM REAL CHARGE_ON "WARD-DUTTON CAP MODEL" =1;
44 # PRINTOUT CONTROL 1-On, 0-Off
45 .PARAM REAL PRN_ID "Control printouts" =0;
46 .PARAM REAL PRN_DIO "Control printouts" =0;
47 .PARAM REAL PRN_CTRL "Control printouts" =0;
48 .PARAM REAL PRN_CTRL_OLD"Control printouts" =0;
49 .PARAM REAL PRN_DCREG "Control printouts" =0;
50 .PARAM REAL PRN_QREG "Control printouts" =0;
51 .PARAM REAL PRN_PREP "Control printouts" =0;
52 .PARAM REAL PRN_AC "Control printouts" =0;
53 # MODEL PARAMETERS
54 .COMMON REAL LEVEL "Level parameter for compatibility with SPICE" = 2;
55 .COMMON REAL TYPE "(1)-Nchannel or (-1)-Pchannel MOS" =1;
56 .COMMON REAL VTO "Threshold voltage (0.1)" ;
57 .COMMON REAL KP "Transconductance parameter (2e-5)" = 2E-5;
58 .COMMON REAL GAMMA "Bulk threshold parameter (0.01)" = 0.01;
59 .COMMON REAL PHI "Surface potential" = 0.6;
60 .COMMON REAL LAMBDA "Channel length modulation" = 0.0;
61 .COMMON REAL RD "Drain ohmic resistance" = 0.0;
62 .COMMON REAL RS "Source ohmic resistance" = 0.0;
63 .COMMON REAL CBD "B-D junction capacitance" = 0.0;
64 .COMMON REAL CBS "B-S junction capacitance" = 0.0;
65 .COMMON REAL IS "Bulk junction sat. current" = 1E-14;
66 .COMMON REAL PB "Bulk junction potential" = 0.5;
67 .COMMON REAL CGSO "Gate-source overlap cap." = 0.0;
68 .COMMON REAL CGDO "Gate-drain overlap cap." = 0.0;
69 .COMMON REAL CGBO "Gate-bulk overlap cap." = 0.0;
70 .COMMON REAL RSH "Sheet resistance" = 1.0;
71 .COMMON REAL CJ "Bottom junction cap per area (0.0)" ;
72 .COMMON REAL MJ "Bottom grading coefficient" = 0.5;
73 .COMMON REAL CJSW "Side junction cap per area" = 0.0;
74 .COMMON REAL MJSW "Side grading coefficient" = 0.3;
75 .COMMON REAL JS "Bulk jct. sat. current density" = 0.0;
76 .COMMON REAL TOX "Oxide thickness" =1E-7;
77 .COMMON REAL LD "Lateral diffusion" = 0.0;
78 .COMMON REAL WD "Lateral diffusion of the channel" = 0.0;
79 .COMMON REAL UO "Surface mobility" =600;
80 .COMMON REAL FC "Forward bias jct. fit parm." = 0.5;
81 .COMMON REAL NSUB "Substrate doping" = 1.45E10;
82 .COMMON REAL TPG "Gate type" =1;
83 .COMMON REAL NSS "Surface state density" = 0.0;
84 .COMMON REAL DELTA "Width effect on threshold voltage" = 0.0;
85 .COMMON REAL UEXP "Crit. field exp for mob. deg." = 0.0;
86 .COMMON REAL UCRIT "Crit. field for mob. degradation" =1E4;
87 .COMMON REAL VMAX "Maximum carrier drift velocity (5m/s)" ;
88 .COMMON REAL XJ "Junction depth" = 0.0;
89 .COMMON REAL NEFF "Total channel charge coeff." = 1.0;
90 .COMMON REAL XQC "Coef. of channel charge" = 0.5;
91 .COMMON REAL NFS "Fast surface state density" = 0.0;
92 .COMMON REAL TNOM "Parameter measurement temperature" = 300.15;
93 .COMMON REAL KF "Flicker noise coefficient" = 0.0;
94 .COMMON REAL AF "Flicker noise exponent" = 1.0;
95 .COMMON REAL NISI "MY PARAMETER NISI " =1.45E16 ;
96 .COMMON REAL REFTEMP "Parameter measurement temperature" = 300.15;
97 .COMMON REAL AD "Drain area" = 1E-6 ;
98 .COMMON REAL AS "Source area" = 1E-6;
99 .COMMON REAL PD "Drain perimeter" = 0.0;
100 .COMMON REAL PS "Source perimeter" = 0.0;
101 .COMMON REAL NRD "Drain squares" = 1.0;
102 .COMMON REAL NRS "Source squares" = 1.0;
103 # END OF COMMON PARAMETERS
104 .COMMON REAL ESI=(11.7 * EPS0),EOX=(3.9 * EPS0),WG1=1.16,WG2=7.02E-4;
105 .COMMON REAL WG3=1108E0,WG=1.1150877E0;
106 # ELEMENT PARAMETERS
107 .PARAM REAL OFF "Device initially off" = 0.0;
108 .PARAM REAL L "Length" = 1E-4 ;
109 .PARAM REAL W "Width" = 1E-4 ;
```

```
.PARAM REAL TEMP "Instance operating temperature" = 300.15;
.PARAM REAL VON "THRESHOLD VOLTAGE - .VAR REAL VARIABLE ASSOCIATED WITH ELEMENT" = 0.0;
# ELEMENT VARIABLES
.MEM REAL pBETA ;
.MEM REAL pCBD ;
.MEM REAL pCBS ;
.MEM REAL pCFS ;
.MEM REAL pCJ ;
.MEM REAL pCO "OxideCap";
.MEM REAL pCOX "The oxide capacitance per unit gate area" = 3.453e-4;
.MEM REAL pCJSW ;
.MEM REAL pCBDSW ;
.MEM REAL pCZBSSW ;
.MEM REAL CZBS ;
.MEM REAL ETA "Static feedback" ;
.MEM REAL pFN ;
.MEM REAL pFCPB ;
.MEM REAL pF2S ;
.MEM REAL pF3S ;
.MEM REAL pF4S ;
.MEM REAL pF2D ;
.MEM REAL pF3D ;
.MEM REAL pF4D ;
.MEM REAL pIDSAT ;
.MEM REAL pISSAT ;
.MEM REAL pKP "Effective Transconductance";
.MEM REAL pPHI2 ;
.MEM REAL pPB "Bulk potential";
.MEM REAL pSBIARG ;
.MEM REAL pVSCRIT "Critical source voltage";
.MEM REAL pVDCRIT "Critical drain voltage";
.MEM REAL pVCRIT ;
.MEM REAL pVBI ;
.MEM REAL pVT_0 "Init vgs";
.MEM REAL pVT ;
.MEM REAL pVTO "Recalculated threshold voltage VTO" ;
.MEM REAL pVFB ;
.MEM REAL pXD ;
.MEM REAL pWE "Electrical Width";
.MEM REAL pLE "Electrical lenght L";
.MEM REAL SIG1[4],SIG2[4];
.MEM REAL TEMP_OLD=TEMP-0.1;
.MEM REAL START_CND=0;
# ZMIENNE ELEMENTU WIDZIANE Z ZEWNATRZ
.PARAM REAL REGIONDC"DC REGION FLAG" =0;
.PARAM REAL REGIONQ"Q REGION FLAG" =0;
.PARAM REAL VBD_, VDS_, VGD, VGD_, VGS_, VBS_ ;
# MODEL INPUTS
.INPUT REAL V(DI,SI):cVDS "Drain-source control"; #=TYPE*3*pVCRIT;
.INPUT REAL V(GE,SI):cVGS "Gate-source"; #=TYPE*2*pVCRIT;
.INPUT REAL V(GE,DI):cVGD "Gate-drain "; #=0;
.INPUT REAL V(BE,SI):cVBS "Bulk-Source"; #=-1;
.INPUT REAL V(BE,DI):cVBD "Bulk-Drain "; #=-1;
# MODEL OUTPUT
.OUTPUT REAL J(DI,SI):ID "Drain current";
.OUTPUT REAL J(BE,DI):IBD "B-D junction current - diode";
.OUTPUT REAL J(BE,SI):IBS "B-S junction current - diode";
# Uwaga na inne wezly podlaczenia (jezeli nie ma Rd i Rs)
.OUTPUT REAL C(BE,DI):CDB_OUT "B-D junction capacitor CBd";
.OUTPUT REAL C(BE,SI):CBS_OUT "B-S junction capacitor CBs";
# WARD-DUTTON MODEL Q
.OUTPUT REAL Q(GE,ZERO,QG):QG;
.OUTPUT REAL Q(BE,ZERO,QB):QB;
.OUTPUT REAL Q(DI,ZERO,QD):QD;
.OUTPUT REAL Q(SI,ZERO,QS):QS;
# RD and RS
.OUTPUT REAL R(DE,DI):RD_OUT"Drain resistance" ;
.OUTPUT REAL R(SE,SI):RS_OUT"Source resistance";
# MODEL BODY
.VAR REAL VSWD;
.VAR REAL VGWD;
.VAR REAL VZWD;
# Q DERIVATIVES
.VAR REAL Q2G;
.VAR REAL Q2B;
.VAR REAL IWD "CURRENT";
# OTHER VARS
.VAR REAL SQVZ,SQVS;
.VAR REAL AAA;
.VAR REAL BBB;
.VAR REAL CCC;
.VAR REAL DDD;
.VAR REAL EEE;
.VAR REAL V1,V2,XV,A1,B1,C1,D1,A,B,C,R,S,R3,S2,P,P0,P2,RO,FI,Y3,P3,P4,IKNT;
.VAR REAL A3,B3,I,DELTA4,JKNT,J,XVALID;
# VECTORS
.VAR REAL A4[4],B4[4];
.VAR REAL POLY4[8],X4[8];
#
.VAR REAL VGS "Gate-Source voltage" ;
.VAR REAL VDS "Drain-Source voltage" ;
.VAR REAL VBS "Bulk-Source voltage" ;
.VAR REAL VBD "Bulk-Drain voltage" ;
.VAR REAL VDSAT"Saturation voltage";
#
.VAR REAL MODE"INVERSION(-1) / NORMAL(1) MODE"=0;
.VAR REAL WKFNG,WKFNGS,FERMIG,FERMIS;
```

```
.VAR REAL BETAEFF,SPHI;
.VAR REAL UFACT,CLFACT;
.VAR REAL PHIMINVBS;
.VAR REAL RATIO4,UO_IND,PHIO,IS_IND,ISDENS,PBO,GMANEW,GMAOLD,CAPFACT,CZBD,CZBDSW;
.VAR REAL DSRGDB,D2SDB2,TMP,TMP1,VBIN,XWD,XWS,ARGSS,ARGSD;
.VAR REAL VTNOM,RATIO,FACT1,FACT2,EGFET,EGFET1,ARG1,PBFACT,PBFACT1;
.VAR REAL ARG,SARG,SARGSW,XLAMDA,SPHI3,BARG,DBRGDB;
.VAR REAL DBARGS,DBARGD,ARGXS,ARGXD,ARGS,ARGD;
.VAR REAL GAMMAD,GAMASD,DBXWD,DBXWS,DGDDVB;
.VAR REAL DDXWD,CDONCO,XN,ARGG,VGST,SARG3,BODY;
.VAR REAL UEFF,VGSX,GAMMD2,ARGV;
.VAR REAL BSARG,DBSRDB,BODYS,GDBDVS,SARGV,XLFACT,DLDSAT,XDV,XLV,VQCHAN;
.VAR REAL VL,XLS,XWB,XLD;
.VAR REAL XLEFF,DELTAL,DFACT,CDRAIN,VDSON,CDSON,EXPG,GMW;
.VAR REAL CAP_BS"Cbs − diode capacitance";
.VAR REAL CAP_BD"Cbd − diode capacitance";
.VAR REAL GOTO1"GOTO FLAG";
.VAR REAL GOTO, BAD;
.VAR REAL tVON,tVTH;
.VAR REAL tKP "KP";

.BEGIN # MODEL BODY

# PREPROCESSING
IF( TEMP != TEMP_OLD ){
  BAD = 0; # INICJALIZACJA

# INICJALIZACJA WSPOLCZYNNIKOW WEKTOROW SIGx

  SIG1[0] = 1 ;
  SIG1[1] = −1 ;
  SIG1[2] = 1 ;
  SIG1[3] = −1 ;

  SIG2[0] = 1 ;
  SIG2[1] = 1 ;
  SIG2[2] = −1 ;
  SIG2[3] = −1 ;

# SPRAWDZANIE POPRAWNOSCI DANYCH

# sprawdz poziom modelu
  IF(LEVEL!=2) {
    PRINT("WRONG MODEL LEVEL (MOSL2WD)");
    EXIT(−100);
  }

# kontrola poprawnosci parametow elementu

  IF(W<=0) {
    PRINT("W less than 1E−4 : ",W);
    BAD=−1;
  }
  IF(L<=0) {
    PRINT("L less than 3E−4 : ",L);
    BAD=−2;
  }

  pLE= L − 2 ∗ LD;
  pWE= W − 2 ∗ WD;

  IF(pWE<=0) {
    PRINT("pWE less than 0 : ",pWE);
    BAD=−22;
  }
  IF(pLE<=0) {
    PRINT("pLE less than 0 : ",pLE);
    BAD=−23;
  }

# uwzglednienie parametrw elementu w wyrazeniach obliczanych
  IF(TOX<=0) {
    PRINT("TOX less than 0 : ",TOX);
    BAD=−3;
  }
  IF(PB<=0) {
    PRINT("PB less than 0 : ",PB);
    BAD=−4;
  }
  IF(AS<=0) {
    PRINT("AS less than 0 : ",AS);
    BAD=−5;
  }
  IF(AD<=0) {
    PRINT("AS less than 0 : ",AS);
    BAD=−6;
  }

  IF(BAD!=0) {
    EXIT(BAD!=0);
  }

# OBLICZANIE PARAMETROW POSREDNICH I ZALEZNOSCI OD TEMPERATURY

  FACT1=0;
  FACT1 = TNOM/REFTEMP;
  VTNOM=KBOLTZ∗TNOM/QELE;
```

```
304
305   EGFET1 = 1.16-(7.02E-4*TNOM*TNOM)/(TNOM+1108);
306   ARG1 = -EGFET1/(2*KBOLTZ*TNOM)+1.1150877/(KBOLTZ*(REFTEMP+REFTEMP));
307   PBFACT1 = -2*VTNOM *(1.5*LOG(FACT1)+QELE*ARG1);
308
309   pCOX=EOX/TOX;
310
311   tKP=KP;
312   IF(tKP.ISSET){
313     tKP = UO * 1e-4 * pCOX; # 1E-4 -> (m**2/cm**2)
314 #PRINT("KP =",tKP);
315   }
316
317   IF( NSUB.ISSET ) {
318     IF( (NSUB*1E6)<NISI){
319 #PRINT(" NSUB < NI ");
320       NSUB=NISI;
321       PRINT(" NSUB le NI -> NSUB=",NSUB);
322     }
323     IF(CJ.ISSET) {
324 # cm**3/m**3
325       CJ=SQRT(ESI*QELE*NSUB*1E6/(2*PB));
326       PRINT("CJ=",CJ);
327     }
328     IF(PHI.ISSET) {
329       IF(PHI<0.1) {
330 # (cm**3/m**3)
331         PHI= 2*VTNOM* LOG(NSUB*1E6/NISI);
332         PHI= MAX(0.1,PHI);
333 #PRINT("PHI=",PHI);
334       }
335     }
336
337     IF(GAMMA.ISSET) {
338 # (cm^3/m^3)
339       GAMMA=SQRT(2 * ESI * QELE * NSUB * 1E6 )/pCOX;
340       PRINT("GAMMA=",GAMMA);
341     }
342
343     FERMIS = TYPE * 0.5 * PHI;
344     WKFNG = 3.2;
345     IF(TPG != 0) {
346       FERMIG = TYPE * TPG*0.5*EGFET1;
347       WKFNG = 3.25 + 0.5 * EGFET1 - FERMIG;
348     }
349     WKFNGS = WKFNG - (3.25 + 0.5 * EGFET1 +FERMIS);
350
351     IF(VTO) { # NIE ZADANE VTO
352       IF(NSS.ISSET){ # NIE ZADANE NSUB
353         PRINT("NSS less or equal 0 : ",NSS);
354         NSS=5e-8;
355         PRINT("NSS=",NSS);
356         BAD=-10;
357         EXIT(BAD);
358       }
359 # (cm^2/m^2)
360       pVFB = WKFNGS - NSS * 1E4 * QELE /pCOX;
361       pVTO = pVFB + TYPE * (GAMMA * SQRT(PHI)+ PHI);
362         PRINT("pVTO =",pVTO);
363       IF(pVTO<0) {
364         PRINT("pVTO less or equal 0");
365         pVTO =0;
366         PRINT("pVTO =",pVTO);
367       }
368     }ELSE{ # ZADANE VTO
369       pVFB = pVTO - TYPE * (GAMMA* SQRT(PHI)+PHI);
370     }
371 # (cm**3/m**3)
372     pXD = SQRT((ESI+ESI)/(QELE * NSUB * 1E6 ));
373
374   }ELSE{ # cm^3/m^3
375     IF(VTO.ISSET) { # NIE ZADANE VTO
376       pVTO = TYPE * 0.1;
377 #PRINT("pVTO =",pVTO);
378     }
379     IF(CJ.ISSET) { # NIE ZADANE CJ
380       CJ=0;
381 # PRINT("CJ=",CJ);
382     }
383     IF(PHI.ISSET) { # NIE ZADANE PHI
384       PHI=0.6;
385 #PRINT("PHI=",PHI);
386     }
387   }
388
389
390   pVT=KBOLTZ*TEMP/QELE;
391   RATIO=TEMP/TNOM;
392   FACT2 = TEMP/REFTEMP;
393   EGFET = 1.16-(7.02E-4*TEMP*TEMP)/(TEMP+1108);
394   ARG = -EGFET/(2*KBOLTZ*TEMP)+1.1150877/(KBOLTZ*(REFTEMP+REFTEMP));
395   PBFACT = -2*pVT *(1.5*LOG(FACT2)+QELE*ARG);
396
397   IF(pLE<=0) {
398     PRINT(" : effective channel length less than zero");
399   }
400
```

```
401   RATIO4 = RATIO * SQRT(RATIO) ;
402   pKP = tKP / RATIO4;
403     pBETA = pKP*W/pLE;
404     pCO=pCOX*W*pLE;
405
406   IF(pCO<=0) {
407     PRINT("ERROR: pCO <= 0 : ",pCO);
408     PRINT(" pCOX <= 0 : ",pCO);
409     PRINT(" W = ",W);
410     PRINT(" pLE = ",pLE);
411     BAD=-21;
412   }
413
414
415   UO_IND = UO/RATIO4;
416   PHIO= (PHI-PBFACT1)/FACT1;
417   pPHI2 = FACT2 * PHIO + PBFACT;
418   pVBI = pVTO - TYPE * (GAMMA*SQRT(PHI)) + 0.5*(EGFET1-EGFET) + TYPE*0.5*(pPHI2-PHI);
419   pVT_0 = pVBI + TYPE * GAMMA * SQRT(pPHI2);
420
421     IF(IS<=0){
422       IS=1E-14;
423       PRINT("IS=",IS);
424     }
425   IS_IND = IS * EXP(-EGFET/pVT+EGFET1/VTNOM);
426   ISDENS = JS * EXP(-EGFET/pVT+EGFET1/VTNOM);
427
428   IF( (ISDENS == 0) || (AD == 0) || (AS == 0) ) {
429     pIDSAT = IS_IND;
430       pISSAT = IS_IND;
431       pVSCRIT = pVT*LOG(pVT/(SQRT(2)*IS_IND));
432     pVDCRIT = pVSCRIT;
433   }ELSE{
434     pIDSAT = ISDENS * AD;
435       pISSAT = ISDENS * AS;
436       pVDCRIT = pVT * LOG( pVT / (SQRT(2) * ISDENS * AD));
437     pVSCRIT = pVT * LOG( pVT / (SQRT(2) * ISDENS * AS));
438   }
439
440   PBO = (PB - PBFACT1)/FACT1;
441   pPB = FACT2 * PBO+PBFACT;
442
443   pSBIARG = SQRT(pPB);
444
445   GMANEW = (pPB-PBO)/PBO;
446   GMAOLD = (PB-PBO)/PBO;
447
448 # ####
449
450   CAPFACT = 1/(1+MJ* (4E-4*(TNOM-REFTEMP)-GMAOLD));
451   pCBD = CBD * CAPFACT;
452   pCBS = CBS * CAPFACT;
453   pCJ = CJ * CAPFACT;
454
455   CAPFACT = 1/(1+MJSW* (4E-4*(TNOM-REFTEMP)-GMAOLD));
456   pCJSW = CJSW * CAPFACT;
457
458   CAPFACT = (1+MJ* (4E-4*(TEMP-REFTEMP)-GMANEW));
459   pCBD = pCBD * CAPFACT;
460   pCBS = pCBS * CAPFACT;
461   pCJ = pCJ * CAPFACT;
462   CAPFACT = (1+MJSW* (4E-4*(TEMP-REFTEMP)-GMANEW));
463   pCJSW = pCJSW * CAPFACT;
464   pFCPB = FC * pPB;
465
466   IF(pCBD>0) {
467     CZBD = pCBD;
468   }ELSE{
469     CZBD=pCJ*AD;
470   }
471   IF(CJSW>0) {
472     CZBDSW= pCJSW * PD;
473   }ELSE{
474     CZBDSW=0;
475   }
476
477 # ####
478
479   ARG = 1-FC;
480   SARG = EXP( (-MJ) * LOG(ARG) );
481   SARGSW = EXP( (-MJSW) * LOG(ARG) );
482
483   pCBD = CZBD;
484   pCBDSW = CZBDSW;
485   pF2D = CZBD*(1-FC* (1+MJ))* SARG/ARG + CZBDSW*(1-FC* (1+MJSW))* SARGSW/ARG;
486   pF3D = CZBD * MJ * SARG/ARG/ pPB + CZBDSW * MJSW * SARGSW/ARG / pPB;
487   pF4D = CZBD*pPB*(1-ARG*SARG)/ (1-MJ) + CZBDSW*pPB*(1-ARG*SARGSW)/ (1-MJSW) -pF3D/2* (
          pFCPB*pFCPB) -pFCPB * pF2D;
488   IF(pCBS>0) {
489     CZBS=pCBS;
490   }ELSE{
491     CZBS=pCJ*AS;
492   }
493   IF(pCJSW>0) {
494     pCZBSSW = pCJSW * PS;
495   }ELSE{
496     pCZBSSW=0;
```

```
497     }
498
499     pF2S = CZBS*(1-FC* (1+MJ))* SARG/ARG + pCZBSSW*(1-FC* (1+MJSW))* SARGSW/ARG;
500     pF3S = CZBS * MJ * SARG/ARG/ pPB + pCZBSSW * MJSW * SARGSW/ARG / pPB;
501     pF4S = CZBS*pPB*(1-ARG*SARG)/ (1-MJ) + pCZBSSW*pPB*(1-ARG*SARGSW)/ (1-MJSW) -pF3S/2* (
            pFCPB*pFCPB) -pFCPB * pF2S;
502
503   # constant per device [Dev.90]
504   # pFN = 1/8 * (DELTA*2*PI * ESI * TOX)/(E_SIO2 * W);
505   #
506     pFN = 0.125*DELTA*2.0*PI*ESI/pCO*pLE;
507     ETA = 1.0+pFN;
508
509   # constant per model [Dev.90]
510   #
511     pCFS = QELE*NFS*1E4; #(CM**2/M**2)
512     pVCRIT=ABS(pVBI+TYPE*(pFN*pPHI2+GAMMA*SQRT(pPHI2))) ;
513
514       IF(PRN_PREP){
515         PRINTNAME();
516         PRINT("# PREPROCESSED PARAMETERS");
517         PRINT(" pBETA = ", pBETA );
518         PRINT(" pCBD = ", pCBD );
519         PRINT(" pCBD = ", pCBD );
520         PRINT(" pCBS = ", pCBS );
521         PRINT(" pCFS = ", pCFS );
522         PRINT(" pCJ = ", pCJ );
523         PRINT(" pCO = ", pCO );
524         PRINT(" pCOX = ", pCOX );
525         PRINT(" pCJSW = ", pCJSW );
526         PRINT(" pCBDSW = ", pCBDSW );
527         PRINT(" pCZBSSW = ", pCZBSSW );
528         PRINT(" CZBS = ", CZBS );
529         PRINT(" ETA = ", ETA );
530         PRINT(" pFN = ", pFN );
531         PRINT(" pFCPB = ", pFCPB );
532         PRINT(" pF2S = ", pF2S );
533         PRINT(" pF3S = ", pF3S );
534         PRINT(" pF4S = ", pF4S );
535         PRINT(" pF2D = ", pF2D );
536         PRINT(" pF3D = ", pF3D );
537         PRINT(" pF4D = ", pF4D );
538         PRINT(" pIDSAT = ", pIDSAT );
539         PRINT(" pISSAT = ", pISSAT );
540         PRINT(" pKP = ", pKP );
541         PRINT(" pPHI2 = ", pPHI2 );
542         PRINT(" pPB = ", pPB );
543         PRINT(" pSBIARG = ", pSBIARG );
544         PRINT(" pVSCRIT = ", pVSCRIT );
545         PRINT(" pVDCRIT = ", pVDCRIT );
546         PRINT(" pVCRIT = ", pVCRIT );
547         PRINT(" pVBI = ", pVBI );
548         PRINT(" pVT_0 = ", pVT_0 );
549         PRINT(" pVT = ", pVT );
550         PRINT(" pVFB = ", pVFB );
551         PRINT(" pXD = ", pXD );
552         PRINT(" pWE = ", pWE );
553         PRINT(" pLE = ", pLE );
554       }
555
556   }
557   # END OF PREPROCESSING
558
559   # MODEL BODY
560
561   REGIONDC=0;
562
563   IF(PRN_DCREG||PRN_QREG||PRN_CTRL||PRN_CTRL_OLD){
564     PRINTNAME();
565     PRINT(" ITER: ",NR_ITER);
566   }
567
568   # RS AND RD
569
570   RD_OUT=RSH*NRD;
571   RS_OUT=RSH*NRS;
572
573   # DO NOT DELETE
574
575   XLAMDA = LAMBDA;
576
577   # GET CONTROLS VALUES
578
579   VGS = TYPE*cVGS;
580   VGS_= TYPE*cVGS_;
581
582   VGD = TYPE*cVGD;
583   VGD_= TYPE*cVGD_;
584
585   VDS = TYPE*cVDS;
586   VDS_= TYPE*cVDS_;
587
588   VBS = TYPE*cVBS;
589   VBS_= TYPE*cVBS_;
590
591   VBD = TYPE*cVBD;
592   VBD_= TYPE*cVBD_;
```

```
593
594 IF(PRN_CTRL!=0){
595   PRINT(" INPUTS");
596   PRINT(" VGS = ",VGS);
597   PRINT(" VGD = ",VGD);
598   PRINT(" VDS = ",VDS);
599   PRINT(" VBS = ",VBS);
600   PRINT(" VBD = ",VBD);
601   IF(PRN_CTRL_OLD!=0){
602     PRINT(" VGS_ = ",VGS_);
603     PRINT(" VGD_ = ",VGD_);
604     PRINT(" VDS_ = ",VDS_);
605     PRINT(" VBS_ = ",VBS_);
606     PRINT(" VBD_ = ",VBD_);
607   }
608 }
609
610 # VDS OSCILATIONS
611
612 IF((VDS<0)&&(VDS_>0)&&(VDS<1E-10)){
613   VDS=0;
614 }
615
616 VBD=VBS-VDS;
617 VGD=VGS-VDS;
618
619 IF(VDS<0){
620     MODE=-1; # SET INVERSION MODE
621 }ELSE{
622     MODE= 1; # SET NORMAL MODE
623 }
624
625 # START CONDITIONS
626 IF(START_CND==1) {
627     IF((VDS==0)&&(VGS==0)&&(VBS==0)&&(OFF==0)){
628         cVBS = -1;
629         cVBD = 0;
630         cVGS = TYPE * pVT_0;
631         cVDS = 0;
632     }
633     IF(OFF==1) {
634         cVDS =0;
635         cVGS =0;
636         cVBS =0;
637         cVBD =0;
638         cVDS_ =0;
639         cVGS_ =0;
640         cVBS_ =0;
641         cVBD_ =0;
642         ID=0;
643     }
644 }ELSE{
645
646 # NORMAL MODE
647
648     IF(MODE<0){
649 # INPUTS
650         VDS = -VDS;
651         VDS_ = -VDS_;
652
653         TMP = VGS;
654         VGS = VGD;
655         VGD = TMP; # TMP == VGS
656
657 # LAST NR ITERATION
658         TMP = VGS_;
659         VGS_ = VGD_;
660         VGD_ = TMP; # TMP == VGS
661
662 # DIODES
663         TMP = VBS;
664         VBS = VBD;
665         VBD = TMP; # TMP == VBS
666
667 # LAST NR ITERATION
668         TMP = VBS_;
669         VBS_ = VBD_;
670         VBD_ = TMP; # TMP == VBS
671     }
672
673 # LIMIT NONLINEAR BRANCH VOLTAGES
674     tVON=TYPE*VON;
675
676     IF(LIMIT_VDS!=0){
677         IF(MODE>0){
678             VGS=FETLIM(VGS,VGS_,tVON);
679             VDS=VGS-VGD;
680             VDS=LIMVDS(VDS,VDS_) ;
681             VGD=VGS-VDS;
682         }ELSE{
683             VGD=FETLIM(VGD,VGD_,tVON);
684             VDS=VGS-VGD;
685             VDS=-LIMVDS(-VDS,-VDS_);
686             VGS=VGD+VDS;
687         }
688     } # LIMIT_VDS
689
```

```
690 # LIMIT DIODE VOLTAGES
691     IF(LIMIT_VBD!=0){
692         IF(VDS>=0){
693             VBS=PNJLIM1(VBS,VBS_,pVT,pVSCRIT,PN_ALFA);
694             VBD=VBS-VDS;
695         }ELSE{
696             VBD=PNJLIM1(VBD,VBD_,pVT,pVDCRIT,PN_ALFA);
697             VBS=VBD+VDS;
698         }
699     } # LIMIT_VBD
700 } # END OF START CONDITIONS
701
702
703 # DIODES - CONTINUES EQUATION MODEL
704 IF(DIODE_ON==1) {
705   IF(VBD>-5*pVT) {
706     IBD=TYPE*(pIDSAT*(EXPO(VBD/pVT)-1)+GMIN*VBD);
707   }ELSE{
708     IBD=TYPE*(pIDSAT*EXPO(-5)*(1-EXP1(5)+5+VBD/pVT)+GMIN*VBD);
709   }
710
711   IF(VBS>-5*pVT) {
712     IBS=TYPE*(pISSAT*(EXPO(VBS/pVT)-1)+GMIN*VBS);
713   }ELSE{
714     IBS=TYPE*(pISSAT*EXPO(-5)*(1-EXP1(5)+5+VBS/pVT)+GMIN*VBS);
715   }
716 }
717
718
719 # DIODES - NONCONTINUES EQUATION MODEL (SPICE LIKE)
720 IF(DIODE_ON==2) {
721   IF(VBD<=0) {
722     IBD=TYPE*pIDSAT*VBD/pVT;
723   }ELSE{
724     IBD=TYPE*pIDSAT*(EXP1(VBD/pVT)-1);
725   }
726   IF(VBS<=0) {
727     IBS=TYPE*pISSAT*VBS/pVT;
728   }ELSE{
729     IBS=TYPE*pISSAT*(EXP1(VBS/pVT)-1);
730   }
731 }
732
733 IF(MODE<0){
734 # DIODES EXCHANGE
735     TMP=IBS;
736     IBS=IBD;
737     IBD=TMP;
738 }
739
740 # COMPUTE SOME USEFUL QUANTITIES
741 PHIMINVBS=pPHI2-VBS;
742
743 # SARG
744 # VBS<=0 SARG=SQRT(pPHI2-VBS)
745 # VBS> 0 SARG=SQRT(pPHI2)/(1+0.5*VBS/pPHI2+0.375*(VBS/pPHI2)^2)
746
747 IF(VBS<=0) {
748     SARG=SQRT(PHIMINVBS) ;
749     DSRGDB = -0.5/SARG;
750     D2SDB2 = 0.5*DSRGDB/(PHIMINVBS);
751 }ELSE{
752     SPHI = SQRT(pPHI2);
753     SPHI3 = pPHI2*SPHI;
754     SARG = SPHI/(1.0+0.5*VBS/pPHI2);
755     TMP = SARG/SPHI3;
756     DSRGDB = -0.5*SARG*TMP;
757     D2SDB2 = -DSRGDB*TMP;
758 }
759
760 SARG3 = SARG*SARG*SARG;
761
762 # BARG
763 # VBD<=0 BARG=SQRT(pPHI2-VBD)
764 # VBD> 0 BARG=SQRT(pPHI2)/(1+0.5*VBD/pPHI2+0.375*(VBD/pPHI2)^2)
765
766 IF( VBD <= 0) {
767     BARG=SQRT(PHIMINVBS+VDS);
768     DBRGDB = -0.5/BARG;
769 }ELSE{
770     BARG = SPHI/(1.0+0.5*(VBS-VDS)/pPHI2);
771     TMP = BARG/SPHI3;
772     DBRGDB = -0.5*BARG*TMP;
773 }
774
775 # CALCULATE THRESHOLD VOLTAGE (VON)
776 # NARROW-CHANNEL EFFECT
777
778 # ETA and pFN in preprocessing -> constants per device
779
780 VBIN = pVBI*TYPE + pFN*PHIMINVBS;
781
782 IF((GAMMA>0)||(NSUB>0)){
783     XWD=pXD*BARG;
784     XWS=pXD*SARG;
785 # short-channel effect with vds != 0
786     ARGSS=0.0;
```

```
787     ARGSD=0.0;
788     DBARGS=0.0;
789     DBARGD=0.0;
790 # ARGSS ARGSD [Dev.90]
791     IF(XJ>0.0){
792         TMP=2/XJ;
793         ARGXS = 1.0+XWS*TMP;
794         ARGXD = 1.0+XWD*TMP ;
795         ARGS = SQRT(ARGXS);
796         ARGD = SQRT(ARGXD);
797         TMP = 0.5*XJ/pLE;
798         ARGSS = TMP*(ARGS-1.0);
799         ARGSD = TMP*(ARGD-1.0) ;
800     }
801     GAMASD=GAMMA*(1-ARGSS-ARGSD);
802     DBXWD = pXD*DBRGDB;
803     DBXWS = pXD*DSRGDB;
804     IF(XJ>0){
805         TMP = 0.5/pLE;
806         DBARGS = TMP*DBXWS/ARGS;
807         DBARGD = TMP*DBXWD/ARGD;
808     }
809     DGDDVB = -GAMMA*(DBARGS+DBARGD);
810     IF(XJ>0){
811         DDXWD = -DBXWD;
812     }
813 }ELSE{
814     GAMASD=GAMMA;
815     GAMMAD=GAMMA;
816     DGDDVB = 0.0;
817 }
818
819 tVON = VBIN + GAMASD*SARG;
820 tVTH = tVON;
821
822 GOTO=0;
823
824 IF(NFS != 0.0 && pCO != 0.0) {
825
826 # pCFS - constant per model
827
828     CDONCO = -(GAMASD*DSRGDB+DGDDVB*SARG)+pFN;
829     XN = 1.0+pCFS/pCO*W*pLE+CDONCO; # tu pCFS/pCOX <=> pCFS/pCO*W*pLE;
830     TMP = pVT*XN;
831     tVON = tVON+TMP;
832     ARGG = 1.0/TMP;
833     VGST = VGS-tVON;
834 }ELSE{
835
836     VGST = VGS-tVON;
837
838 # GOTO CUTOFF REGION
839
840     IF(VGS <= tVTH){
841         REGIONDC=11;
842     }
843 }
844
845 IF(REGIONDC==0) {
846
847 # compute some more useful quantities
848
849 # pSBIARG constant per model
850     GAMMAD = GAMASD;
851     BODY = BARG*BARG*BARG-SARG3;
852     GOTO1=0;
853         IF(NFS != 0.0 ) {
854             IF(pCO == 0.0) {
855                 GOTO1=1;
856             }
857         }
858
859     IF((pCO > 0.0)&&(GOTO1==0)){
860         TMP = UCRIT * 100 * ESI/pCOX ; # CM/M
861             IF(VGST <= TMP){
862                 GOTO1=1;
863             }ELSE{
864                 UFACT = EXP(UEXP*LOG(TMP/VGST));
865                 UEFF = UO *1E-4*UFACT; # (M**2/CM**2)
866             }
867     }
868
869     IF(GOTO1>0) {
870         UFACT = 1.0;
871         UEFF = UO * 1E-4; # (M**2/CM**2)
872     }
873
874 # EVALUATE SATURATION VOLTAGE AND ITS DERIVATIVES ACCORDING TO
875 # GROVE-FROHMAN EQUATION
876
877     VGSX = VGS;
878     GAMMAD = GAMASD/ETA;
879     IF(NFS != 0 && pCO != 0) {
880         VGSX = MAX(VGS,tVON);
881     }
882
883     TMP1=(VGSX-VBIN)/ETA;
```

```
884|
885|     IF(GAMMAD > 0) {
886|         GAMMD2 = GAMMAD*GAMMAD;
887|         ARGV = TMP1+PHIMINVBS;
888|         IF(ARGV <= 0.0) {
889|             VDSAT = 0.0;
890|         }ELSE{
891|             ARG = SQRT(1.0+4.0*ARGV/GAMMD2);
892|             VDSAT = TMP1+GAMMD2*(1.0-ARG)/2.0;
893|             VDSAT = MAX(VDSAT,0.0);
894|         }
895|     }ELSE{
896|         VDSAT = TMP1;
897|         VDSAT = MAX(VDSAT,0.0);
898|     }
899|
900|     IF(VMAX.ISSET) {
901| # EVALUATE SATURATION VOLTAGE AND ITS DERIVATIVES
902| # ACCORDING TO BAUM'S THEORY OF SCATTERING VELOCITY
903| # SATURATION
904|         GAMMD2 = GAMMAD*GAMMAD;
905|         V1 = (VGSX-VBIN)/ETA+PHIMINVBS;
906|         V2 = PHIMINVBS;
907|         XV = VMAX*pLE/UEFF;
908|             A1 = GAMMAD/0.75;
909|         B1 = -2.0*(V1+XV);
910|         C1 = -2.0*GAMMAD*XV;
911|         D1 = 2.0*V1*(V2+XV)-V2*V2-4.0/3.0*GAMMAD*SARG3;
912|         A = -B1;
913|         B = A1*C1-4.0*D1;
914|         C = -D1*(A1*A1-4.0*B1)-C1*C1;
915|         R = -A*A/3.0+B;
916|         S = 2.0*A*A*A/27.0-A*B/3.0+C;
917|         R3 = R*R*R;
918|         S2 = S*S;
919|         P = S2/4.0+R3/27.0;
920|         P0 = ABS(P);
921|         P2 = SQRT(P0);
922|         IF(P < 0) {
923|             RO = SQRT(S2/4.0+P0);
924|             RO = LOG(RO)/3.0;
925|             RO = EXP(RO);
926|             FI = ATAN(-2.0*P2/S);
927|             Y3 = 2.0*RO*COS(FI/3.0)-A/3.0;
928|         }ELSE{
929|             P3 = (-S/2.0+P2);
930|             P3 = EXP(LOG(ABS(P3))/3.0);
931|             P4 = (-S/2.0-P2);
932|             P4 = EXP(LOG(ABS(P4))/3.0);
933|             Y3 = P3+P4-A/3.0;
934|         }
935|         IKNT = 0;
936|         A3 = SQRT(A1*A1/4.0-B1+Y3);
937|         B3 = SQRT(Y3*Y3/4.0-D1);
938|         I=1;
939|         WHILE(I<=4){
940|             A4[I-1] = A1/2.0+SIG1[I-1]*A3;
941|             B4[I-1] = Y3/2.0+SIG2[I-1]*B3;
942|             DELTA4 = A4[I-1]*A4[I-1]/4.0-B4[I-1];
943|             IF(DELTA4 >= 0){
944|                 IKNT = IKNT+1;
945|                 TMP = SQRT(DELTA4);
946|                 X4[IKNT-1] = -A4[I-1]/2.0+TMP;
947|                 IKNT = IKNT+1;
948|                 X4[IKNT-1] = -A4[I-1]/2.0-TMP;
949|             }
950|             I=I+1;
951|         }
952|         JKNT = 0;
953|         J=1;
954|         WHILE(J<=IKNT){
955|             IF(X4[J-1] > 0){
956|
957| # IMPLEMENT THIS SANELY
958|
959|                 POLY4[J-1] = X4[J-1]*X4[J-1]*X4[J-1]*X4[J-1]+A1*X4[J-1]*X4[J-1]*X4[J-1];
960|                 POLY4[J-1] = POLY4[J-1]+B1*X4[J-1]*X4[J-1]+C1*X4[J-1]+D1;
961|                 IF(ABS(POLY4[J-1]) <= 1.0E-6){
962|                     JKNT = JKNT+1;
963|                     IF(JKNT <= 1) {
964|                         XVALID = X4[J-1];
965|                     }
966|                     IF(X4[J-1] <= XVALID){
967|                         XVALID = X4[J-1];
968|                     }
969|                 }
970|             }
971|             J=J+1;
972|         }
973|         IF(JKNT > 0) {
974|             VDSAT = XVALID*XVALID-PHIMINVBS;
975|         }
976|     }
977|
978| # EVALUATE EFFECTIVE MOBILITY AND ITS DERIVATIVES
979|
980|     IF(VDS != 0.0) {
```

```
981         GAMMAD = GAMASD;
982         IF((VBS-VDSAT) <= 0){
983             BSARG = SQRT(VDSAT+PHIMINVBS) ;
984             DBSRDB = -0.5/BSARG;
985         }ELSE{
986             BSARG = SPHI/(1.0+0.5*(VBS-VDSAT)/pPHI2);
987             DBSRDB = -0.5*BSARG*BSARG/SPHI3;
988         }
989         BODYS = BSARG*BSARG*BSARG-SARG3;
990         GDBDVS = 2.0*GAMMAD*(BSARG*BSARG*DBSRDB-SARG*SARG*DSRGDB);
991         IF(VMAX.ISSET) {
992             IF(NSUB == 0.0 || XLAMDA > 0.0) {
993             }ELSE{
994                 ARGV = (VDS-VDSAT)/4.0;
995                 SARGV = SQRT(1.0+ARGV*ARGV);
996                 ARG = SQRT(ARGV+SARGV);
997                 XLFACT = pXD/(pLE*VDS);
998                 XLAMDA = XLFACT*ARG;
999                 DLDSAT = VDS*XLAMDA/(8.0*SARGV);
1000            }
1001        }ELSE{
1002            ARGV = TMP1-VDSAT;
1003            XDV = pXD/SQRT(NEFF);
1004            XLV = VMAX*XDV/(2.0*UEFF);
1005                VQCHAN = ARGV-GAMMAD*BSARG;
1006            VL = VMAX*pLE;
1007
1008            IF(NSUB == 0.0 || XLAMDA > 0.0) {
1009            }ELSE{
1010                ARGV = VDS-VDSAT;
1011                ARGV = MAX(ARGV,0.0);
1012                XLS = SQRT(XLV*XLV+ARGV);
1013                DLDSAT = XDV/(2.0*XLS);
1014                XLFACT = XDV/(pLE*VDS);
1015                XLAMDA = XLFACT*(XLS-XLV);
1016                DLDSAT = DLDSAT/pLE;
1017            }
1018        }
1019    }
1020
1021 # LIMIT CHANNEL SHORTENING AT PUNCH-THROUGH
1022
1023    XWB = pXD*pSBIARG;
1024    XLD = pLE-XWB;
1025    CLFACT = 1.0-XLAMDA*VDS;
1026    XLEFF = pLE*CLFACT;
1027    DELTAL = XLAMDA*VDS*pLE;
1028    IF(NSUB==0.0){
1029      XWB = 0.25E-6;
1030    }
1031    IF(XLEFF<XWB) {
1032        XLEFF = XWB/(1.0+(DELTAL-XLD)/XWB);
1033        CLFACT = XLEFF/pLE;
1034        DFACT = XLEFF*XLEFF/(XWB*XWB);
1035    }
1036
1037 # EVALUATE EFFECTIVE BETA (EFFECTIVE KP)
1038
1039    BETAEFF = pBETA*UFACT/CLFACT;
1040
1041    GAMMAD = GAMASD;
1042 }
1043
1044 CDRAIN = 0.0;
1045
1046 IF(((REGIONDC==11)||(REGIONDC==0)) && (VDS <= 1.0E-10)) {
1047 # CUTOFF REGION
1048    CDRAIN = 0.0;
1049    IF(REGIONDC!=11){
1050      REGIONDC=1;
1051    }
1052 }
1053
1054
1055 IF((NFS != 0.0)&&(REGIONDC==0)&&(VGS <= tVON)) {
1056    REGIONDC=2;
1057 # SUBTHRESHOLD REGION
1058    IF(VDSAT<=0){
1059        IF(VGS<tVON) {
1060            CDRAIN = 0.0;
1061        }
1062        REGIONDC=111;
1063    }ELSE{
1064        VDSON = MIN(VDSAT,VDS);
1065        IF(VDS > VDSAT) {
1066            BARG = BSARG;
1067            DBRGDB = DBSRDB;
1068            BODY = BODYS;
1069        }
1070        CDSON = BETAEFF*((tVON-VBIN-ETA*VDSON*0.5)*VDSON-GAMMAD*BODY/1.5);
1071        EXPG = EXP(ARGG*(VGS-tVON));
1072        CDRAIN = CDSON*EXPG;
1073        GMW = CDRAIN*ARGG;
1074    }
1075 }
1076
1077 IF(REGIONDC==0) {
```

```
1078      IF(VDS <=VDSAT) {
1079  # LINEAR REGION VDS<VDSAT && VGS>tVON
1080          REGIONDC=3;
1081          CDRAIN = BETAEFF*((VGS-VBIN-ETA*VDS/2.0)*VDS-GAMMAD*BODY/1.5);
1082      }ELSE{
1083  # SATURATION REGION
1084          REGIONDC=4;
1085          CDRAIN = BETAEFF*((VGS-VBIN-ETA*VDSAT/2.0)*VDSAT-GAMMAD*BODYS/1.5);
1086      }
1087  }
1088
1089
1090  # CAPACITANCES
1091
1092  IF(CHARGE_ON!=0){ # CHARGE IS ON
1093
1094  # WARD-DUTTON CHARGE MODEL
1095  # WARD-DUTTON VOLTAGES
1096  # ACCUMULATION AND CUTOFF - ONLY VG -> VGWD
1097      VGWD=0;
1098      VSWD=0;
1099      VZWD=0;
1100
1101      VGWD=VGS-VBS-TYPE*pVBI+pPHI2 ;
1102
1103      REGIONQ=0;
1104      IF((REGIONDC==2)||(REGIONDC==1)||(REGIONDC==11)||(REGIONDC==111)){
1105          IF(VGWD<=0){
1106  # 1 - CHARGE ACCUMULATION
1107              REGIONQ=1;
1108              IF(PRN_QREG!=0){
1109                  PRINT("CHARGE ACCUMULATION");
1110              }
1111              QG=pCO*VGWD;
1112              QB=-QG;
1113          }ELSE{
1114  # 2 - CHARGE CUT-OFF
1115              REGIONQ=2;
1116              IF(PRN_QREG!=0){
1117                  PRINT("CHARGE CUT-OFF");
1118              }
1119              TMP=VGWD+GAMMA*GAMMA/4;
1120  #IF(TMP<0)TMP=0;
1121              TMP=SQRT(TMP);
1122              QG=pCO*GAMMA*(TMP-GAMMA/2);
1123              QB=-QG;
1124          }
1125      }
1126      IF((REGIONDC==3)||(REGIONDC==4)){
1127  # ON REGION - 3-LINEAR AND 4-SATURATION
1128          IF(PRN_QREG!=0){
1129              PRINT("CHARGE ON");
1130          }
1131  #
1132  # ON REGION - VG->VGWD,VS->VSWD,VZ->VZWD
1133  #
1134  # VG - ALREADY CALCULATED
1135  # VS
1136          VSWD=pPHI2-VBS;
1137  # VZ
1138          IF(VDS<=VDSAT){
1139              VZWD=pPHI2-VBS+VDS;
1140          }ELSE{
1141              VZWD=pPHI2-VBS+VDSAT;
1142          }
1143
1144  # SQVS = SQRT(VS)
1145
1146          SQVS=VSWD;
1147          IF(SQVS<=0){
1148              SQVS=1E-15;
1149          }ELSE{
1150              SQVS=SQRT(SQVS);
1151          }
1152
1153  # SQVZ = SQRT(VZ)
1154
1155          SQVZ=VZWD;
1156          IF(SQVZ<=0){
1157              SQVZ=1E-15;
1158          }ELSE{
1159              SQVZ=SQRT(SQVZ);
1160          }
1161
1162          AAA = SQVZ+SQVS;
1163          BBB = VZWD+SQVZ*SQVS+VSWD;
1164          CCC = VZWD*VZWD+SQVZ*SQVS*(VZWD+VSWD)+VSWD*VSWD;
1165          DDD = VZWD*VZWD+VZWD*VSWD+VSWD*VSWD;
1166          EEE = 0.5*AAA*(VSWD+VZWD);
1167
1168          IWD = VGWD*AAA-2*GAMMA*BBB/3-EEE;
1169
1170          IF( IWD==0 ){
1171            PRINT(" IWD = ", IWD);
1172            EXIT(-1485);
1173          }
1174
```

```
1175 # Q2G - PART OF THE QG
1176         Q2G = VGWD*EEE - 0.4*GAMMA*CCC - AAA*DDD/3;
1177
1178 # QG
1179         QG = pCO*(VGWD-Q2G/IWD);
1180
1181 # Q2B - PART OF THE QB
1182         Q2B = 2*VGWD*BBB/3 - GAMMA*EEE - 2*(CCC+VSWD*VZWD)/5;
1183
1184 # QB
1185         QB = (-pCO*GAMMA*Q2B)/IWD;
1186
1187         IF(REGIONDC==3){
1188 # 3 - Q LINEAR RANGE
1189             REGIONQ=3;
1190             IF(PRN_QREG!=0){
1191                 PRINT("CHARGE IN LINEAR REGION");
1192             }
1193             QD=-0.5*(QG+QB);
1194             QS=QD;
1195         }ELSE{
1196 # 4 - Q SATURATION RANGE
1197             REGIONQ=4;
1198             IF(PRN_QREG!=0){
1199                 PRINT("CHARGE IN SATURATION REGION");
1200             }
1201             QD=-XQC*(QG+QB);
1202             QS=-(1-XQC)*(QG+QB);
1203         }
1204     }
1205 } # END OF CHARGE_ON
1206
1207 # DIODE CAPACITANCES
1208
1209 IF(DIODE_CAP_ON!=0){
1210
1211     IF((CZBS != 0) || (pCZBSSW != 0)) {
1212         IF(VBS < pFCPB){
1213             ARG=1-VBS/pPB;
1214 # THE FOLLOWING BLOCK LOOKS SOMEWHAT LONG AND MESSY,
1215 # BUT SINCE MOST USERS USE THE DEFAULT GRADING
1216 # COEFFICIENTS OF 0.5, AND SQRT IS MUCH FASTER THAN AN
1217 # EXP(LOG()) WE USE THIS SPECIAL CASE CODE TO BUY TIME.
1218 # (AS MUCH AS 10% OF TOTAL JOB TIME!)
1219 #! IT IS OF COURSE TRUTH FOR BUILD IN MODEL NOT FOR EMDL ONE !
1220             IF(MJ == MJSW) {
1221                 IF(MJ ==0.5) {
1222                     SARG = 1/SQRT(ARG);
1223                     SARGSW = SARG;
1224                 }ELSE{
1225                     SARG = EXP(-MJ* LOG(ARG));
1226                     SARGSW = SARG;
1227                 }
1228             }ELSE{
1229                 IF(MJ ==0.5) {
1230                     SARG = 1/SQRT(ARG);
1231                 }ELSE{
1232                     SARG = EXP(-MJ* LOG(ARG));
1233                 }
1234                 IF(MJSW ==0.5) {
1235                     SARGSW = 1/SQRT(ARG);
1236                 }ELSE{
1237                     SARGSW = EXP(-MJSW* LOG(ARG));
1238                 }
1239             }
1240             CAP_BS=CZBS*SARG+ pCZBSSW*SARGSW;
1241         }ELSE{
1242             CAP_BS=pF2S+ pF3S*VBS;
1243         }
1244     }ELSE{
1245         CAP_BS=0;
1246     }
1247     CBS_OUT=CAP_BS;
1248
1249     IF((pCBD != 0) || (pCBDSW != 0) ) {
1250         IF(VBD < pFCPB) {
1251             ARG=1-VBD/pPB;
1252 # THE FOLLOWING BLOCK LOOKS SOMEWHAT LONG AND MESSY,
1253 # BUT SINCE MOST USERS USE THE DEFAULT GRADING
1254 # COEFFICIENTS OF 0.5, AND SQRT IS MUCH FASTER THAN AN
1255 # EXP(LOG()) WE USE THIS SPECIAL CASE CODE TO BUY TIME.
1256 # (AS MUCH AS 10% OF TOTAL JOB TIME!)
1257             IF((MJ ==0.5) && (MJSW ==0.5)) {
1258                 SARG = 1/SQRT(ARG);
1259                 SARGSW = SARG;
1260             }ELSE{
1261                 IF(MJ ==0.5) {
1262                     SARG = 1/SQRT(ARG);
1263                 }ELSE{
1264                     SARG = EXP(-MJ* LOG(ARG));
1265                 }
1266                 IF(MJSW ==0.5) {
1267                     SARGSW = 1/SQRT(ARG);
1268                 }ELSE{
1269                     SARGSW =EXP(-MJSW* LOG(ARG));
1270                 }
1271             }
```

```
1272            CAP_BD=pCBD*SARG+ pCBDSW*SARGSW;
1273        }ELSE{
1274            CAP_BD=pF2D + VBD * pF3D;
1275        }
1276    }ELSE{
1277        CAP_BD = 0;
1278    }
1279
1280    CDB_OUT=CAP_BD ;
1281
1282 } # END OF DIODE_CAP_ON
1283
1284 # ASSIGN OUTPUT VALUES
1285
1286 QG = TYPE*QG;
1287 QD = TYPE*QD;
1288 QS = TYPE*QS;
1289 QB = TYPE*QB;
1290
1291 # DRAIN CURRENT
1292
1293 IF(MODE>=0) {
1294 # NORMAL
1295
1296    ID = TYPE*CDRAIN + GMIN*VDS;
1297
1298    CDB_OUT=CAP_BD;
1299    CBS_OUT=CAP_BS;
1300
1301 }ELSE{
1302 # INVERSION
1303
1304    ID = -TYPE*CDRAIN + GMIN*VDS ;
1305
1306    CDB_OUT=CAP_BS;
1307    CBS_OUT=CAP_BD;
1308
1309    QG=-QG;
1310    TMP=QD;
1311    QD=-QS;
1312    QS=-TMP;
1313    QB=-QB;
1314 }
1315
1316 # REMEMBER VON VALUE
1317
1318 VON = TYPE * tVON;
1319
1320 # CONTROL PRINTOUTS
1321
1322 IF(PRN_DCREG!=0){
1323
1324    IF(REGIONDC==11){
1325        PRINT(" DC => 1,11,11-CUTOFF, 2-SUB,3-LIN,4-SAT: ",REGIONDC);
1326    }
1327    IF(MODE<0){
1328        PRINT(" INVERSION MODE");
1329    }ELSE{
1330        PRINT(" NORMAL MODE");
1331    }
1332 }
1333 IF(PRN_CTRL!=0){
1334    PRINT(" INPUTS");
1335    PRINT(" VGS = ",VGS);
1336    PRINT(" VGD = ",VGD);
1337    PRINT(" VDS = ",VDS);
1338    PRINT(" VBS = ",VBS);
1339    PRINT(" VBD = ",VBD);
1340    IF(PRN_CTRL_OLD!=0){
1341        PRINT(" VGS_ = ",VGS_);
1342        PRINT(" VGD_ = ",VGD_);
1343        PRINT(" VDS_ = ",VDS_);
1344        PRINT(" VBS_ = ",VBS_);
1345        PRINT(" VBD_ = ",VBD_);
1346    }
1347 }
1348 IF(PRN_ID){
1349    PRINT(" DRAIN CURRENT");
1350    PRINT(" ID = ",ID);
1351 }
1352 IF(PRN_DIO){
1353    PRINT(" PARASITIC DIODES");
1354    PRINT(" IBD = ",IBD);
1355    PRINT(" IBS = ",IBS);
1356 }
1357 IF(PRN_AC){
1358    PRINT(" SMALL SIGNAL PARAMETERS");
1359    PRINT(" CDRAIN = ",CDRAIN);
1360    PRINT(" INNE ");
1361    PRINT(" pVBI = ",pVBI);
1362    PRINT(" tVTH = ",tVON);
1363    PRINT(" pVTO = ",pVTO);
1364    PRINT(" VDSAT = ",VDSAT);
1365    PRINT(" PHIMINVBS= ",PHIMINVBS);
1366    PRINT(" pKP = ",pKP);
1367    PRINT(" TOX = ",TOX);
1368    PRINT(" pCO = ",pCO);
```

```
1369      PRINT(" pCOX = ",pCOX);
1370      PRINT(" QG = ",QG);
1371      PRINT(" VGWD = ",VGWD);
1372      PRINT(" GAMMA = ",GAMMA);
1373  }
1374
1375  IF(REGIONDC==0) {
1376      PRINT("REGION ERROR");
1377      ASSERT (-2);
1378  }
1379
1380  # DEBUG PRINTOUTS
1381
1382  IF( DEBUG>=1 ){
1383
1384    PRINT(" ");
1385    PRINT("# OPERATING POINT INFORMATION *");
1386    PRINT(" ");
1387    PRINT("# INPUTS ");
1388    PRINT(" VGS = ", cVGS);
1389    PRINT(" VDS = ", cVDS);
1390    PRINT(" VBS = ", cVBS);
1391    PRINT("# CURRENTS ");
1392    PRINT(" ID = ",ID);
1393
1394    IF(1){
1395      PRINT("# OTHERS ");
1396      PRINT(" pVBI = ",pVBI);
1397      PRINT(" tVTH = ",TYPE*VON) ;
1398      PRINT(" pVTO = ",pVTO);
1399      PRINT(" pCO = ",pCO);
1400      PRINT(" pCOX = ",pCOX);
1401      PRINT(" QG = ",QG);
1402      PRINT(" QB = ",QB);
1403      PRINT(" QD = ",QD);
1404      PRINT(" QS = ",QS);
1405    }
1406  }
1407
1408  .END
```

Część IV

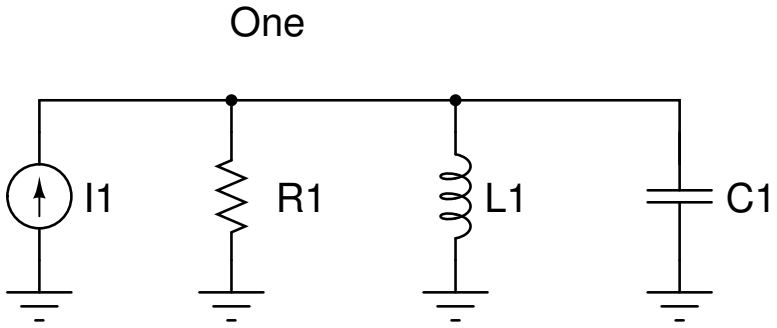
Przykłady użycia programu

Rozdział 11

Układy liniowe

11.1 Dwójnik RLC

Na rys.11.1 przedstawiono układ rezonansowy RLC, który pobudzany jest przez niezależne źródło prądowe I_1, które generuje sygnał o przebiegu prostokątnym ze składową stałą. Zbiór wejściowy symulatora przedstawiono na wydruku 11.1.



Rys. 11.1: Układ RLC (plik: linear-filters/rlc.cir).

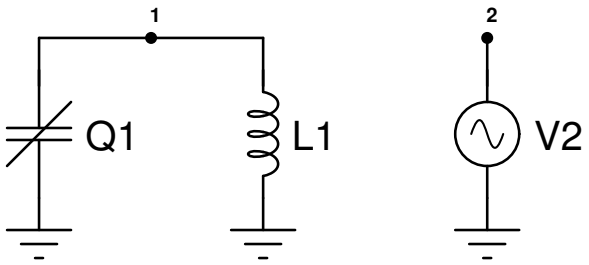
Wydruk 11.1: Układ RLC.

```
1  # Revision (C) AIVA 2003-2010
2  .TASK "Linear RLC (schematic: rlc-sch.eps)"
3
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "vars.mdl"
7  .LIB "math/temp3.mdl"
8  .LIB "src/pulse.mdl"
9  .LIB "src/i.mdl"
10 .LIB "rlc/r.mdl"
11 .LIB "rlc/l.mdl"
12 .LIB "rlc/c.mdl"
13
14 .MODEL L L
15 .MODEL R R
16 .MODEL C C
17 .MODEL I IT (TDSI=0.1,VSI=0.99,FREQ=1)
18 .MODEL I I (V1=0,V2=10,TD=2uS,TR=1uS,PW=60uS,TF=2uS,PER=1e3S)
19
20 I.I1 ONE 0 DC=0.1A
21 L.R1 ONE 0 R=1kOhm
22 L.L1 ONE 0 L=1mH
23 C.C1 ONE 0 C=1uF
24
```

```
.CMD
.OPTI REXG=2, PIV=3, ITL5=1000000
.PRINT ADD TR(ONE)
.PROBE ADD TR(ONE) TR(L1.IL) MD(L1.IL) PH(*)
.OP TITLE="TRANSIENT ANALYSIS (TRAN)"
#.IC L1.IL=1
#.IC ONE=1
.TRAN STEP=0.01 TMIN=0.1 TMAX=500 HMAX=0.1 UIC TITLE="TRANSIENT ANALYSIS (TRAN)"
.OPTI WAC=3
.AC SCALE=LIN FPTS=2000 FMIN=4.5k FMAX=5.5k TITLE="SMALL SIGNAL ANALYSIS (AC)"
.END
```

11.2 Sterowany ładunek

Na rys.11.2 przedstawiono dwójnik zawierający sterowany ładunek Q_1 oraz liniową indukcyjność. Element Q_1 posiada zgromadzony ładunek i jest sterowany ze źródła napięciowego V_2. Element Q_1 opisany jest za pomocą modelu QQ. Zbiór wejściowy symulatora przedstawiono na wydruku 11.2.



Rys. 11.2: Układ ze sterowanym ładunkiem (plik: linear-filters/charge.cir)

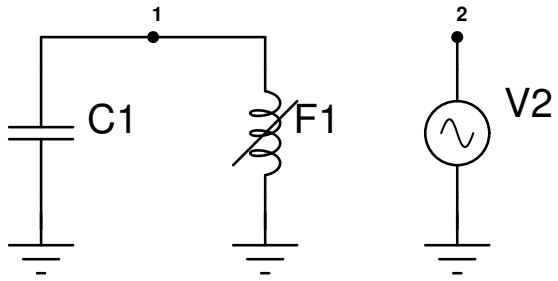
Wydruk 11.2: Sterowany ładunek

```
# Revision (C) AIVA 2003-2010
.TASK "Controlled charge (schematic: charge-sch.eps)"

.LIB "types.mdl"
.LIB "units.mdl"
.LIB "math/pulse.mdl"
.LIB "src/v.mdl"
.LIB "rlc/r.mdl"
.LIB "rlc/l.mdl"
.LIB "rlc/c.mdl"

.OPTI ECHO=ON;

.MODEL QQ;
.OPTI LIMU=OFF,CLUB=5,CLLB=5;
.COMMON REAL UG=1;
.PARAM REAL C=1;
.EXTERNAL N1,N2,N3,N4;
.FLOW QQ;
.OUTPUT Q(N1,N2,QQ):REAL Q;
.INPUT V(N1,N2):REAL U;
.INPUT V(N3,N4):REAL US;
.BEGIN

Q=C*(1+US/UG)*U;

.END

.MODEL L L;
.MODEL R R;
.MODEL Q QQ;
.MODEL V SRCV_E TDSI=10 VSI=0.99 FREQ=0.03 V1=0 V2=0;

L.L2 1 0 L=1;
Q.CP1 1 0 IN 0 C=1;
V.V2 IN 0 DC=0;

.CMD
.OPTI REXG=2;
```

```
.PRINT ADD TR(1) TR(IN) TR(CP1.QQ) PAR(HN) PAR(HN) PAR(RLTE) # PAR(ORDER);
.IC CP1.QQ=1;
.OPTI PIV=1, WDCT=1, HMIN=1e-9, HINI=1E-5, POST=0, ERTR=1e-5,RELT=1e-7, ALLP=1;
.TRAN STEP=0.1S TMIN=0.1S TMAX=41S HMAX=0.2S UIC TITLE="Transient analysis (TRAN)";
.END
```

11.3 Sterowany strumień magnetyczny

Na rys.11.3 przedstawiono dwójnik zawierający sterowany strumień magnetyczny F_1 oraz liniową pojemność C_1. Dla elementu F_1 zadano wartość początkową strumienia magnetycznego. Wartość strumienia sterowana jest ze źródła napięciowego V_2. Zbiór wejściowy symulatora przedstawiono na wydruku 11.3. Element F_1 opisany jest za pomocą modelu *FLUX*.



Rys. 11.3: Układ ze sterowanym strumieniem magnetycznym (plik: linear-filters/flux.cir).

Wydruk 11.3: Sterowany strumień magnetyczny.

```
# Revision (C) AIVA 2003-2010
.TASK "Controlled magnetic flux (schematic: flux-sch.eps)"

.LIB "types.mdl"
.LIB "units.mdl"
.LIB "vars.mdl"
.LIB "math/pulse.mdl"
.LIB "src/v.mdl"
.LIB "rlc/l.mdl"
.LIB "rlc/c.mdl"

.OPTI ECHO=1 ,DEBU_LEX=0 ,DEBU_CIR=0 ,DEBU_ELE=0 ,DEBU_MOD=0 ,DEBU_OUT=0 ,DEBU_MTX=0

.MODULE NONLINEAR FLUX
#.OPTI LIMU=BOTH,CLUB=2,CLLB=0;
.EXTERNAL N1,N2,N3,N4;
.FLOW II,FLU;
.COMMON REAL L=1,REAL UG=1;
.INPUT V(N3,N4):REAL US;
.OUTPUT F(N1,N2,II,FLU):REAL FLOW;
.INPUT I(II):REAL IL;
.BEGIN
FLOW=IL*(L+(L*US)/UG);
.END

.MODEL C C
.MODEL F FLUX
.MODEL V V TDSI=0.9 VSI=0.99 FREQ=0.03

C.C1 ONE 0 C=1
V.V2 TWO 0 DC=1
F.F ONE 0 TWO 0

.CMD
.OPTI REXG=MIN, ECHO=1, PIV=0, ORDE=1
.PROBE ADD TR(F.II) TR(F.FLU) TR(ONE) TR(TWO) PAR(HN) PAR(NR)
.PRINT ADD TR(ONE) TR(TWO) TR(F.II) TR(F.FLU) PAR(HN) PAR(ORDE)
.IC F.FLU=1
.OPTI WDCT=1,WNRI=0, ITL4=20,ITL1=50,ITL5=900000
.OPTI HINI=1mS, HMIN=1nS, FIXS=0, ALLP=0, POST=0, ERTR=1e-2, RELT=1e-5, LTEA=0.9
```

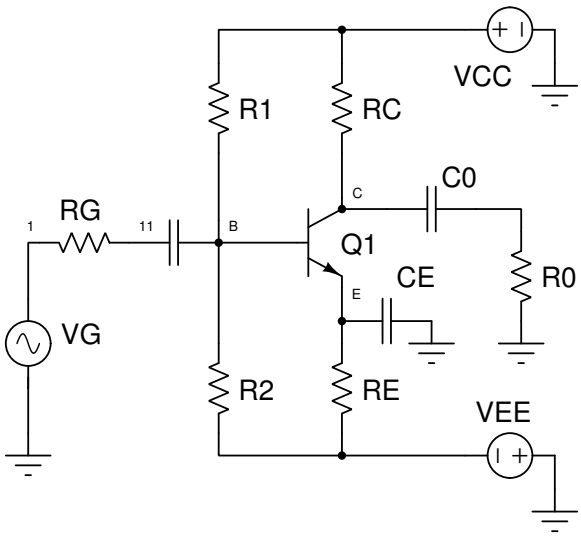
```
41 .TRAN STEP=0.1 TMIN=0 TMAX=30 HMAX=0.5 UIC TITLE="TRANSIENT ANALYSIS"
42 .STDS STEP=0.5 TMIN=0.1 TMAX=30 PER=0.1884 TITLE="STEADY STATE ANALYSIS"
43 .END
```

Rozdział 12

Symulacja układów nieliniowych

12.1 Wzmacniacz z tranzystorem pracującym w układzie wspólnego emitera

Na rys.12.1 przedstawiono układ wzmacniacza z tranzystorem pracującym w układzie wspólnego emitera zasilanego symetrycznymi źródłami zasilania $+V_{CC}$, $-V_{EE}$. Pętlę ujemnego sprzężenia zwrotnego realizuje rezystor R_E. Wzmocnienie układu zależy od parametrów tranzystora g_m (jego punktu pracy) oraz wartości rezystancji R_C i wartości obciążenia R_0. Zbiór wejściowy symulatora przedstawiono na wydruku 11.1. Wzmacniacz sterowany jest ze źródła napięciowgo V_G.



Rys. 12.1: Wzmacniacz z tranzystorem pracującym w układzie wspólnego emitera.

Wydruk 12.1: Układ wzmacniacza z tranzystorem pracującym w układzie OE.

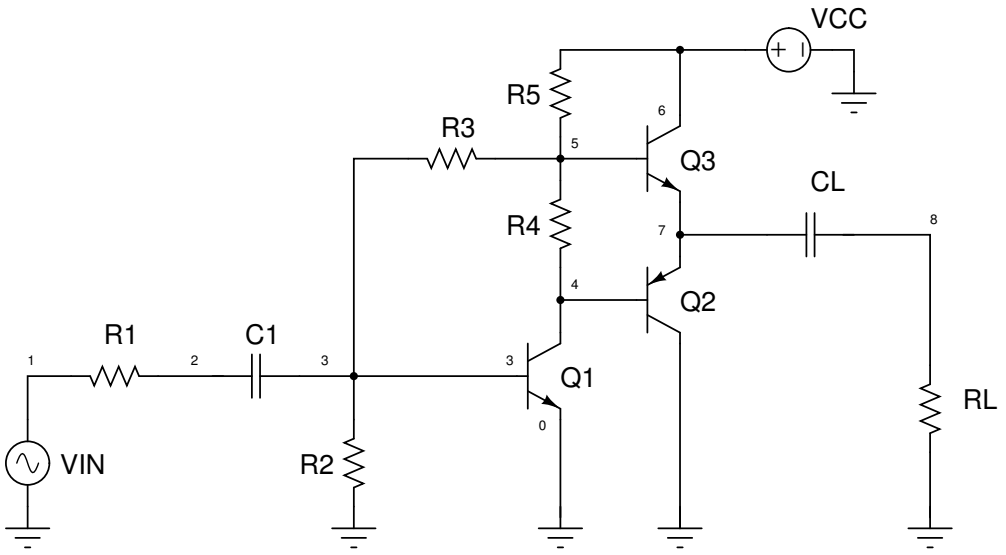
```
1 # Revision (C) AIVA 2002-2010
2 #
```

```
.TASK "Open emitter amplifier (schematic: open_emitter-sch.eps)"
# Library
.LIB "types.mdl"
.LIB "units.mdl"
.LIB "vars.mdl"
.LIB "bjt/pnjlim.mdl"
.LIB "d/ds.mdl"
.LIB "bjt/bjt2.mdl"
.LIB "math/pulse.mdl"
.LIB "src/v.mdl"
.LIB "rlc/r.mdl"
.LIB "rlc/c.mdl"
# Model lines
.MODEL R R
.MODEL C C
.MODEL VC SRCV_E V1=0 V2=0 TD=10mS TR=0.1mS PW=2mS TF=0.1mS FREQ=0
.MODEL VG SRCV_E V1=0 V2=1 TD=1mS TR=2mS PW=3mS TF=1mS PER=9mS VSI=10mV TDSI=5mS FREQ=100Hz
.MODEL BC547B BJT2_E TYPE=1 BF=100

VG.VG 1 0 (DC=10,MAGN=1V,PHAS=0,DEBUG=0)
R.RG 1 11 R=22k
C.C1 11 B C=1.5uF

R.R1 B VCC R=330k
R.R2 B VEE R=100k

BC547B.QT1 C B E

R.RC C VCC R=18k
R.RE E VEE R=3.3k
C.CE E 0 C=10uF

C.C0 C WY C=1uF
R.RO WY 0 R=47k

VC.VCC VCC 0 DC=15V
VC.VEE VEE 0 DC=-15V

.CMD

.OPTI ECHO=0
.OPTI PIV=2,ITL4=50,ITL1=50, ITL5=200000, WDCT=2, WNRI=0, ALLP=1
.PRINT ADD OP("[BCE]*")
.OPTI WDCT=0, WNRI=0
.OP TITLE="Operation point analysis"
# Specifications
#.SP ADD OP 0 REL VCC.II S1=-2.5e-3 W1=1 S2=-1.0e-3 W2=1
.SP ADD OP 0 REL 1 S1=5 W1=1 S2=6 W2=1
.SP ADD OP 0 REL IN S1=2 W1=1 S2=3 W2=1
.SP ADD OP 0 REL C S1=-4 W1=1 S2=-3 W2=1
#.SP ADD OP 0 REL VCC S1=14.0 W1=1 S2=16.0 W2=0.5
#.SP ADD OP 0 REL VEE S1=-16.0 W1=1 S2=-14.0 W2=0.5
# Variables
#.VAR ADD RG.R SCALE=1 MIN=1k MAX=10k
.VAR ADD RG1.R SCALE=1 MIN=10k MAX=25k
.VAR ADD RC.R SCALE=1 MIN=3k MAX=20k
.VAR ADD RE.R SCALE=1 MIN=1k MAX=5k
.VAR ADD VG.DC SCALE=1 MIN=1 MAX=10
# ------------------------------------------------------------
.OPTI MXAN=700, DEBU_CENT=0
#.ERRC
#.CENT
#.VAR PRI FIX
#.VAR PRI VARP
#.VAR PRI VAR
.SP PRI ACT
.VAR PRI VAR
.PRINT ADD OP(VCC) OP(VEE) OP("*.II") PAR(NR)
.PRINT ADD TR(WY) TR(1) PAR(HN) PAR(RLTE) PAR(NR) PAR(ORDER) # TR(IN) TR(B) TR(C) TR(B) TR(E)
.OPTI HMIN=1e-7
.OPTI PIV=2,ITL4=50,ITL1=150, ITL5=200000, WDCT=2, WNRI=0, ALLP=1, ORDE=5
.OP TITLE="Operating point analysis (OP)"
.TRAN STEP=1mS TMIN=0.1mS TMAX=40mS HMAX=1mS TITLE="Transient analysis (TRAN)"
.OPTI WDCT=1
#.ITA STEP=1mS TMIN=0.1mS TMAX=40mS HMAX=1mS TITLE="Iterated timing analysis (ED-ITA)"
.AC SCALE=DEC FPTS=100 FMIN=1Hz FMAX=1megHz TITLE="Small signal analysis (AC)"
.OPTI PIV=1,ITL4=50,ITL1=150, ITL5=200000, WDCT=0, WNRI=0, ALLP=1, ORDE=5
.CENT # Optimization
.SVER
.SP PRI ACT
.END
```

12.2 Wzmacniacz mocy

Na rys.12.2 przedstawiono układ wzmacniacza mocy zasilanego z pojedynczego źródła napięciowego V_{CC}. Zbiór wejściowy symulatora przedstawiono na wydruku 12.2. Wzmacniaczem steruje źródło napięcia sinusoidalnego V_{IN} opisanego linią modelu VV. W sekcji komend wykonywane są trzy analizy: punktu pracy, czasowa oraz stanu usta-

lonego.



Rys. 12.2: Wzmacniacz mocy (plik: nonlin-amp/powamp.cir).

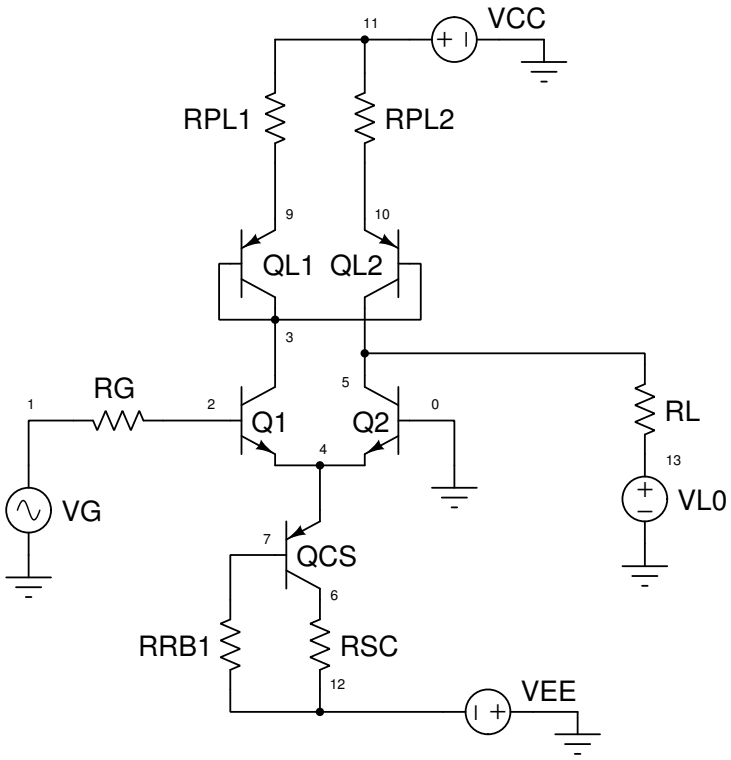
Wydruk 12.2: Wzmacniacz mocy.

```
1  # Revision : 1.1 (C) AIVA 2003-2010
2  #
3  .TASK "POWER AMPLIFIER"
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "math/pulse.mdl"
7  .LIB "src/v.mdl"
8  .LIB "rlc/r.mdl"
9  .LIB "rlc/l.mdl"
10 .LIB "rlc/c.mdl"
11 .LIB "math/one.mdl"
12 .LIB "bjt/pnjlim.mdl"
13 .LIB "d/ds.mdl"
14 .LIB "bjt/bjt2.mdl"
15
16 .MODEL NPN BJT2 (TYPE= 1,RC=1,RE=1,BF=80,RB=1,TF=1e-9,TR=20e-9,CJE=3e-12,CJC=2e-12,VAF=50,VAR=200,
       CJS=2e-12)
17 .MODEL PNP BJT2 (TYPE=-1,RC=1,RE=1,BF=80,RB=1,TF=1e-9,TR=20e-9,CJE=3e-12,CJC=2e-12,VAF=50,VAR=210,
       CJS=2e-12)
18 .MODEL V V (TDSI=0,VSI=0,FREQ=0,V1=0,V2=0)
19 .MODEL VV V (V1=0,TDSI=100uS,VSI=2V,FREQ=1kHz)
20 .MODEL R R
21 .MODEL C C
22
23 VV.VIN 1 0 DC=0
24 R.R1 1 2 R=1e3
25
26 CJ.C1 2 3 C=1e-6
27 R.RC1 2 3 R=1e6
28
29 R.R2 3 0 R=11e3
30 R.R3 3 5 R=39e3
31 R.R4 4 5 R=120
32 R.R5 5 VCC R=1e3
33 NPN.Q1 4 3 0
34 PNP.Q2 0 4 7
35 NPN.Q3 VCC 5 7
36
37 R.RCL 7 8 R=1e3
38 C.CL 7 8 C=10e-6
39 R.RL 8 0 R=100
40 V.VCC VCC 0 DC=20
41
42 .CMD
43 .OPTI PIV=2
44 .OPTI ITL1=100,ITL4=50, ITL5=400000
45 .PROBE ADD TR(1) TR(3) TR(4) TR(7) TR(8) PAR(ORDER) PAR(HN) PAR(NR)
46 .PRINT ADD TR(1) TR(3) TR(4) TR(7) PAR(HN) PAR(NR)
```

```
47 .OPTI WDCT=1, HINI=1e-6, RELT=3e-5, ERTR=1.1e-3
48 .OP TITLE="Operating Point analysis (OP)"
49 .TRAN STEP=0.5uS TMIN=0mS TMAX=3mS HMAX=0.1mS UIC TITLE="Transient analysis (TRAN)"
50 .STDS STEP=0.1mS TMIN=0mS TMAX=1mS PER=1uS TITLE="Steady state analysis (STDS)"
51 .END
```

12.3 Wzmacniacz różnicowy

Na rys.12.3 przedstawiono układ wzmacniacza różnicowego sterowanego ze źródła V_G z wyjściem symetrycznym obciążonym rezystancją R_L. Na wyjściu dodatkowo włączono źródło napięciowe $VL0$ umożliwiające pomiar prądu płynącego przez obciążenie R_L. Zbiór wejściowy symulatora przedstawiono na wydruku 12.3. W sekcji komend wykonywane analizę punktu pracy i analizę czasową. Przedstawiony wzmacniacz można zrealizować z użyciem układu UL1111 (CEMI) - http://www.datasheetcatalog.com/datasheets_pdf/U/L/1/1/UL1111.shtml.



Rys. 12.3: Schemat ze wzmacniaczem różnicowym (plik: nonlin-amp/diff-pair.cir).

Wydruk 12.3: Układ ze wzmacniaczem różnicowym (plik: nonlin-amp/diff-pair.cir).

```
1 # Revision : 1.2 (C) AIVA 2003-2010
2 #
3 .TASK "Differential pair (schematic: diff_pair-sch.eps)"
4 # Library
5 .LIB "types.mdl"
6 .LIB "units.mdl"
7 .LIB "vars.mdl"
8 .LIB "pnjlim.mdl"
9 .LIB "math/pulse.mdl"
```

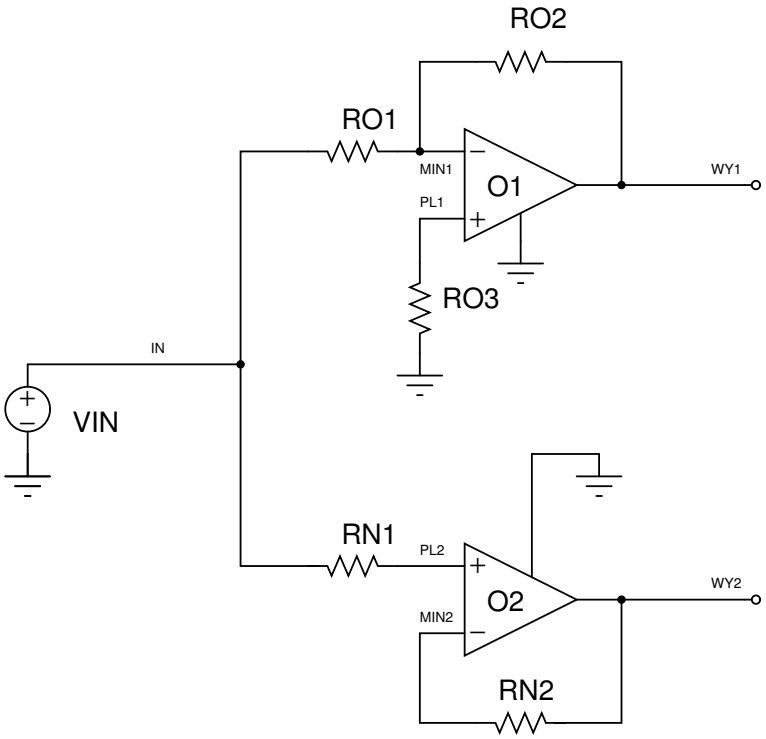
```
10 .LIB "src/vt.mdl"
11 .LIB "src/it.mdl"
12 .LIB "rlc/l.mdl"
13 .LIB "rlc/c.mdl"
14 .LIB "bjt/pnjlim.mdl"
15 .LIB "bjt/bjt2.mdl"
16 .LIB "d/ds.mdl"
17 # Model lines
18 .MODEL R R
19 .MODEL C C
20 .MODEL V VT (V1=0, V2=0,TD=10mS,TR=0.1mS,PW=2mS,TF=0.1mS,FREQ=0)
21 .MODEL I IT (V1=0, V2=0mV,TD=10mS,VSI=0mA,FREQ=0kHz)
22
23 .MODEL VIN V (VSI=5mV,FREQ=2.5kHz)
24 .MODEL MOD V (V1=0,V2=0,TR=1mS,TF=1mS,PW=2mS,PER=6mS)
25 .MODEL VZ V (V1=0,V2=0)
26 .MODEL UL1111N BJT2 (TYPE=1,IS=1E-16,BF=50,CJE=1pF,CJC=1pS,VAF=80,RB=50,RC=10)
27 .MODEL UL1111P BJT2 (TYPE=1,IS=1E-16,BF=50,CJE=1pF,CJC=1pS,VAF=80,RB=50,RC=10)
28 .MODEL BC157 BJT2 (TYPE=-1,IS=0.2nS,BF=180,CJE=10pF,CJC=4.5pF,RB=50,VAF=30,TF=10nS,TR=10nS)
29
30 VIN.VG 1 0 (MAGN=1,DC=0.0)
31 R.RG 1 2 R=10
32 VZ.VCC 11 0 DC=15
33 VZ.VEE 12 0 DC=-15
34 VZ.VLO 13 0 DC=5
35 UL1111N.Q1 3 2 4
36 UL1111N.Q2 5 0 4
37 BC157.QL1 3 3 9
38 BC157.QL2 5 3 10
39 R.RPL1 11 9 R=3kOhm
40 R.RPL2 11 10 R=3kOhm
41 R.RL 13 5 R=2kOhm
42 C.CS2 5 28 C=2.2uF
43 R.ROBC 28 0 R=1k
44 UL1111N.QCS 4 7 6
45 R.RSC 12 6 R=3.6kOhm
46 R.RB1 12 7 R=62kOhm
47 C.CS1 27 7 C=10uF
48 MOD.VMOD 27 0 (DC=0)
49 R.RB2 7 0 R=82kOhm
50 .CMD
51 .OPTI PIV=1,ITL1=160, WDCT=2
52 .PRINT ADD TR(5) TR(7) TR(1) TR(28) TR(13)
53 .PRINT ADD AC(5) AC(28) AC(1)
54 .OP
55 #.AC DEC 2 1 1G
56 #.DC VG DC -1 1 0.1
57 #.ALT .MODEL MOD V1=-4
58 #.ALT .MODEL MOD V2=4
59 .OPTI HMIN=1E-5,ITL4=30, WDCT=1
60 .TRAN STEP=0.05mS TMIN=0uS TMAX=10mS HMAX=0.1mS
61 .END
```

12.4 Idealny wzmacniacz operacyjny

Na rys.12.4 przedstawiono układ testowy idealnych wzmacniaczy operacyjnych odwracającego i nieodwracająaego. Zbiór wejściowy symulatora przedstawiono na wydruku 12.4.

Wydruk 12.4: Układ ze wzmacniaczem operacyjnym (plik: nonlin-oper/oper-amp-ideal.cir).

```
1  # Revision : 1.1 (C) AIVA 2003-2010
2  #
3  .TASK "Ideal Operational Amplifier"
4
5  .LIB "types.mdl"
6  .LIB "units.mdl"
7  .LIB "math/pulse.mdl"
8  .LIB "src/v.mdl"
9  .LIB "src/r.mdl"
10 .LIB "src/c.mdl"
11 .LIB "src/o.mdl"
12
13 .MODEL R R
14 .MODEL C C
15 .MODEL V V (VSI=25 FREQ=1K)
16
17 V.VIN IN 0 (MAGN=1,DC=0)
18
19 # Uklad odwracajacy o k=-1V/V
20 O.O1 PL1 MIN1 WY1 0
21 R.RO1 IN MIN1 100
22 R.RO2 MIN1 WY1 100
23 R.RO3 PL1 0 50
24 # Uklad nieodwracajacy o k=1V/V
25 O.O2 PL2 MIN2 WY2 0
```



Rys. 12.4: Schemat układu idealnego wzmacniacza operacyjnego (plik: oper-amp-ideal.cir).

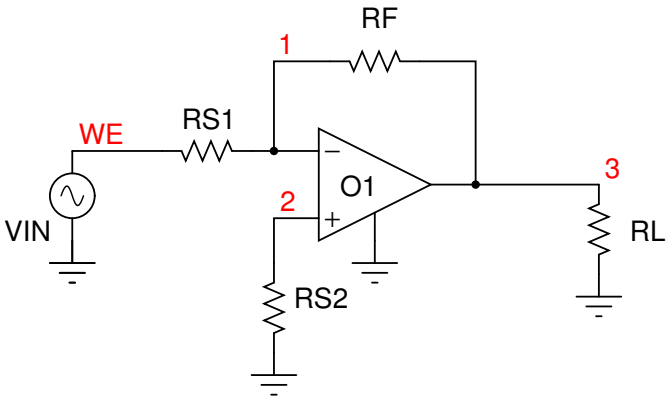
```
R.RN1 IN PL2 100
R.RN2 MIN2 WY2 100
.CMD
#.OPTI WDCT=4,LIMU=0,PIV=1,MATV=1
.OPTI PIV=1
#.PRINT AC V(5) V(4)
#.AC DEC 250 200 30K
.OP
.PRINT ADD TR(PL1) TR(MIN1) TR(WY1)
.PRINT ADD TR(PL2) TR(MIN2) TR(WY2) TR(IN)
.TRAN TMIN=0.2mS TMAX=2mS TITLE="ANALIZA TR UA741"
.END
```

12.5 Wzmacniacz operacyjny ua741

Na rys.12.5 przedstawiono układ testowy popularnego wzmacniacza operacyjnego *ua741*. Wzmacniacz został opisany w postaci podukładu zamieszczonego na wydruku 12.5. Zbiór wejściowy symulatora układu testowego z układem *ua741* przedstawiono na wydruku 12.6.

Wydruk 12.5: Wzmacniacz operacyjny ua741.

```
# Revision : 1.1 (C) AIVA 2003-2010
# Model wzmacniacza operacyjnego ua741
#
.SUBCKT O741 1 2 WY zero
.MODEL C C;
.MODEL R R;
.MODEL QNL BJT2_E TYPE= 1;
.MODEL QPL BJT2_E TYPE=-1;
```



Rys. 12.5: Schemat układu ua741 (plik: nonlin-oper/ua741x1.cir).

```
.MODEL V SRCV_E TD=20E-9 TR=9E-9 PW=30E-9 TF=12E-9 FREQ=0;

QNL.Q1 3 2 4;
QNL.Q2 3 1 5;
QPL.Q3 7 6 4;
QPL.Q4 8 6 5;
QNL.Q5 7 9 10;
QNL.Q6 8 9 11;
QNL.Q7 VCC 7 9;
QNL.Q8 6 15 12;
QNL.Q9 15 15 VEE;
QPL.Q10 3 3 VCC;
QPL.Q11 6 3 VCC;
QPL.Q12 17 17 VCC;
QPL.Q14 22 17 VCC;
QNL.Q15 22 22 21;
QNL.Q16 22 21 20;
QNL.Q17 13 13 VEE;
QNL.Q18 VCC 8 14;
QNL.Q19 20 14 18;
QNL.Q20 22 23 WY;
QPL.Q21 13 25 WY;
QNL.Q22 VCC 22 23;
QPL.Q23 VEE 20 25;

R.R1 10 VEE R=1e3;
R.R2 9 VEE R=50e3;
R.R3 11 VEE R=1e3;
R.R4 12 VEE R=3e3;
R.R5 15 17 R=39e3;
R.R6 21 20 R=40e3;
R.R7 14 VEE R=50e3;
R.R8 18 VEE R=50;
R.R9 WY 25 R=25;
R.R10 23 WY R=50;
R.R11 13 VEE R=50e3;

*#C.COMP 22 8 C=30pF

V.VCC VCC zero DC=15;
V.VEE VEE zero DC=-15;

.END
```

Wydruk 12.6: Układ ze wzmacniaczem operacyjnym ua741.

```
# Revision : 1.1 (C) AIVA 2003-2010
.TASK "Operational Amplifiers ua741"
.LIB "types.mdl"
.LIB "units.mdl"
.LIB "vars.mdl"
.LIB "bjt/pnjlim.mdl"
.LIB "math/one.mdl"
.LIB "math/pulse.mdl"

.LIB "src/v.mdl"
.LIB "rlc/r.mdl"
.LIB "rlc/c.mdl"
.LIB "bjt/bjt2.mdl"

.INC "ua741.sub"

.MODEL R R;
.MODEL C C;
.MODEL V VT V1=0 V2=5,TD=5mS,TR=1mS,PW=7mS,TF=1mS,FREQ=0;

V.VIN IN 0 DC=-1;
R.RS1 IN 1 R =1kOhm;
R.RS2 2 0 R =0.5kOhm;

# - + out 0
O741:XO1 1 2 OUT 0;
C.CL OUT 1 C=0.1uF;
R.RF 1 OUT R=1kOhm;
R.RL OUT 0 R=100kOhm;
#
.CMD
.PRINT ADD TR("IN") TR("OUT") PAR(HN) PAR(RLTE) PAR(ORDER);
.PRINT ADD OP("IN") OP("OUT") PAR(NR);
.OP TITLE="OP - ua741";
.TRAN STEP=1mS TMIN=0mS TMAX=20mS HMAX=10mS TITLE="TRAN - ua741";
.END
```

Rozdział 13

Symulacja układów cyfrowych

Dero umożliwia symulację układów o mieszanym typie sygnałów, w tym układów cyfrowych. Bramki cyfrowe można opisać na kilka sposobów:

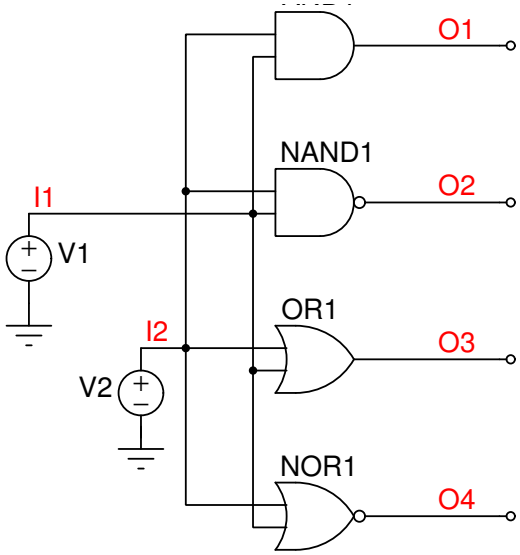
- jako układy analogowe w postaci podukładu,
- można zamodelować za pomocą wbudowanego języka MDL - np. układ o charakterystyce odcinkowo-liniowej.

13.1 Bramki AND, NAND, OR, NOR

Na rys.13.1 przedstawiono układ testowy do symulacji podstawowych bramek cyfrowych. Bramki zamodelowano wykorzystując język MDL. Zbiór wejściowy symulatora przedstawiono na wydruku 13.1. Zbiór biblioteczny zawierający modele podstawowych bramek przedstawiono na wydruku 13.2.

Wydruk 13.1: Bramki AND, NAND, OR, NOR.

```
1  # Revision : 1.3 (C) AIVA 2003-2010
2  #
3  .TASK "DIGITAL GATES: AND NAND OR NOR "
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "vars.mdl"
7  .LIB "math/pulse.mdl"
8  .LIB "src/v.mdl"
9  .LIB "rlc/r.mdl"
10 .LIB "rlc/c.mdl"
11 .LIB "dig/gates.mdl"
12
13 # MODEL LINES
14 .MODEL R R
15 .MODEL C C
16 .MODEL VG1 V (V1=-2.5,V2=2.5,TD=1mS,TR=2mS,PW=3mS,TF=1mS,PER=9mS)
17 .MODEL VG2 V (V1=-2.5,V2=2.5,TD=3mS,TR=2mS,PW=3mS,TF=1mS,PER=9mS)
18
19 .MODEL AND AND
20 .MODEL NAND NAND
21 .MODEL OR OR
22 .MODEL NOR NOR
23
24 # INPUTS
25 VG1.V1 I1 0 DC=2.5 DEBUG=0
26 VG2.V2 I2 0 DC=2.5 DEBUG=0
27
28 # GATES
29 AND.AND1 I1 I2 O1 0
30 NAND.NAND1 I1 I2 O2 0
31 OR.OR2 I1 I2 O3 0
32 NOR.NOR2 I1 I2 O4 0
33
34 .CMD
35 .PRINT ADD TR(I*) TR(O*) PAR(HN)
36 .PROBE ADD TR(I*) TR(O*) PAR(HN)
37 .OP TITLE="Operating point analysis"
```



Rys. 13.1: Schemat układu do symulacji bramek (plik: nonlin-digital/gates.cir).

```
38 .TRAN STEP=1mS TMIN=1mS TMAX=10mS HMAX=1mS TITLE="Transient analysis (TRAN)"
39 .ITA STEP=0.1mS TMIN=1mS TMAX=10mS HMAX=1mS TITLE="Iterateed timing analysis (ED-ITA)"
40 .END
```

Wydruk 13.2: Modele MDL podstawowych bramek.

```
1  # Revision : 1.2 (C) AIVA 2007-2010
2  #
3
4  .MODEL AND
5  .OPTI LIMU=OFF, CLUB=2, CLLB=2, DEBU=0;
6  .PIN I1,I2,O,ZERO;
7  .FLOW IX;
8  .PARAM REAL ONE=5;
9  .PARAM REAL ZERO=0;
10 .PARAM REAL SW1=0.5;
11 .PARAM REAL SW2=3.0;
12 .INPUT V(I1,ZERO):REAL IN1;
13 .INPUT V(I2,ZERO):REAL IN2;
14 .OUTPUT D(O,O,IX):DIGITAL OUT;
15 .BEGIN
16 OUT=(IN2+IN1)/2;
17 OUT=(SW2+SW1)/2;
18
19 IF ( IN1>SW2 && IN2>SW2 ){
20   OUT=ONE;
21 }
22
23 IF ( IN1<SW1 && IN2<SW1 ){
24   OUT=ZERO;
25 }
26 .END
27
28
29
30 .MODEL NAND
31 .OPTI LIMU=OFF, CLUB=2, CLLB=2, DEBU=1;
32 .PIN I1,I2,O,ZERO;
33 .FLOW IX;
34 .PARAM REAL ONE=5;
35 .PARAM REAL ZERO=0;
36 .PARAM REAL SW1=0.5;
37 .PARAM REAL SW2=3.0;
38 .INPUT V(I1,ZERO):REAL IN1;
39 .INPUT V(I2,ZERO):REAL IN2;
40 .OUTPUT D(O,O,IX):DIGITAL OUT;
41 .BEGIN
42 OUT=(IN2+IN1)/2;
43 OUT=(SW2+SW1)/2;
44
45 IF ( IN1>SW2 && IN2>SW2 ){
46   OUT=ZERO;
```

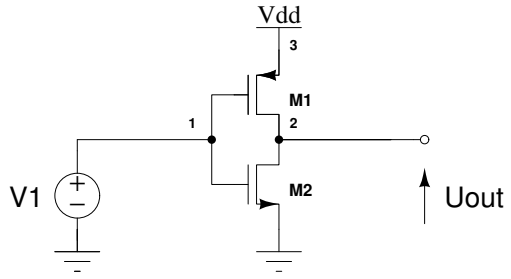
```
}ELSE{
  IF ( IN1<SW1 || IN2<SW1 ){
    OUT=ONE;
  }
}
.END



.MODEL OR
.OPTI LIMU=OFF, CLUB=2, CLLB=2, DEBU=1;
.PIN I1,I2,O,ZERO;
.FLOW IX;
.PARAM REAL ONE=5;
.PARAM REAL ZERO=0;
.PARAM REAL SW1=0.5;
.PARAM REAL SW2=3.0;
.INPUT V(I1,ZERO):REAL IN1;
.INPUT V(I2,ZERO):REAL IN2;
.OUTPUT D(O,O,IX):DIGITAL OUT;
.BEGIN
OUT=(IN2+IN1)/2;
OUT=(SW2+SW1)/2;

IF ( IN1>SW2 || IN2>SW2 ){
  OUT=ONE;
}

IF ( IN1<SW1 && IN2<SW1 ){
  OUT=ZERO;
}
.END



.MODEL NOR
.OPTI LIMU=OFF, CLUB=2, CLLB=2, DEBU=1;
.PIN I1,I2,O,ZERO;
.FLOW IX;
.PARAM REAL ONE=5;
.PARAM REAL ZERO=0;
.PARAM REAL SW1=0.5;
.PARAM REAL SW2=3.0;
.INPUT V(I1,ZERO):REAL IN1;
.INPUT V(I2,ZERO):REAL IN2;
.OUTPUT D(O,O,IX):DIGITAL OUT;
.BEGIN
OUT=(IN2+IN1)/2;
OUT=(SW2+SW1)/2;

IF ( IN1>SW2 || IN2>SW2 ){
  OUT=ZERO;
}

IF ( IN1<SW1 && IN2<SW1 ){
  OUT=ONE;
}
.END
```

13.2 Inwerter MOS

Na rys.13.2 przedstawiono układ testowy do symulacji inwertera cyfrowego (NOT) zbudowanego z tranzystorów MOS. Tranzystory MOS zostały opisane modelami MDL, natomiast cała bramka została zapisana w postaci podukładu przedstawionego na wydruku 13.3. W pliku zmieszczono linie modeli dla tranzystorów MOS w dwóch wersjach: model Meyer'a i Warda-Duttona (rozdział 10.14). Zbiór wejściowy symulatora przedstawiono na wydruku 13.4.

Wydruk 13.3: Podukład inwertera MOS.

```
# Revision : 1.1 (C) AIVA 2003-2010
.SUBCKT INV IN OUT VDD ZERO
## INWERTER CMOS

#.MODEL NMOS MOS2MEY (TYPE= 1,VTO=0.5,PHI=0.7,KP=1.0E-3,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e-12,CGDO
    =1e-12,CBD=5e-12,CBS=5e-12)
#.MODEL PMOS MOS2MEY (TYPE=-1,VTO=0.5,PHI=0.7,KP=1.0E-3,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e-12,
    CGDO=1e-12,CBD=5e-12,CBS=5e-12)
.MODEL NMOS MOS2WD (TYPE= 1,VTO=0.5,PHI=0.7,KP=1.0E-3,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e
    -12,CGDO=1e-12,CBD=5e-12,CBS=5e-12)
.MODEL PMOS MOS2WD (TYPE=-1,VTO=0.5,PHI=0.7,KP=1.0E-3,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e
    -12,CGDO=1e-12,CBD=5e-12,CBS=5e-12)

PMOS.MP VDD IN OUT VDD (W=25e-6,L=5e-6)
NMOS.MN OUT IN ZERO ZERO (W=25e-6,L=5e-6)
```



Rys. 13.2: Inwerter zbudowany z tranzystorów MOS.

```
12
13 .END
```

Wydruk 13.4: Układ inwertera MOS.

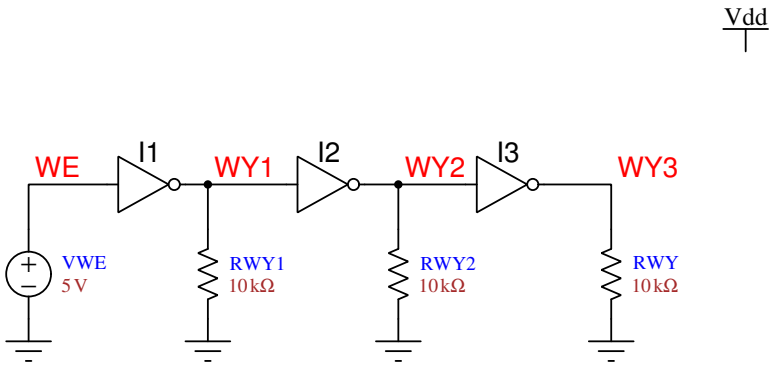
```
1  # Revision : 1.2 (C) AIVA 2003-2010
2  # MOS INVERTER
3  .INC lib/mos2mey.mdl
4  .LIB "math/pulse.mdl"
5  .LIB "src/v.mdl"
6  .LIB "rlc/r.mdl"
7  .LIB "rlc/c.mdl"
8  #
9  .MODEL VCC V
10 .MODEL V V
11 .MODEL R R
12 .MODEL C C
13 #
14 VCC.VDD 2 0 5
15 V.VWE 1 0 (DC=0V)
16 PMOS.M1P 2 1 3 2 (W=250U,L=5U)
17 NMOS.M2N 0 1 3 0 (W=250U,L=5U)
18 R.RWY 3 0 1E3
19 .MODEL NMOS MOS2MEY_E (TYPE= 1,VTO=0.5,PHI=0.7,KP=5E-4,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1U,
       CGDO=1U,CBD=50P,CBS=50P)
20 .MODEL PMOS MOS2MEY_E (TYPE=-1,VTO=-0.5,PHI=0.7 ,KP=5E-4,GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=
       1U,CGDO=1U,CBD=50P,CBS=50P)
21 .MODEL VSRC SRCV_E (V1=0,V2=5,TD=2E-9,TR=1E-9,PW=30E-9,TF=2E-9)
22 .CMD
23 .PRINT DC(1) DC(3)
24 .PRINT TR(1) TR(3)
25 .TRAN TMIN=1E-9 TMAX=70E-9
26 .DC VAR1=VWE.DC START1=0 STOP1=5 STEP1=0.05
27 .END
```

13.3 Trzy inwertery MOS

Na rys.13.3 przedstawiono układ testowy do symulacji trzech inwerterów zbudowanego z tranzystorów MOS. Tranzystory MOS zostały opisane modelami MDL, natomiast pojedyncza bramka została zapisana w postaci podukładu przedstawionego na wydruku 13.3. Zbiór wejściowy symulatora przedstawiono na wydruku 13.5.

Wydruk 13.5: Trzy inwertery MOS

```
1  # Revision : 1.1 (C) AIVA 2003-2010
2  .TASK "Trzy inwertery MOS"
3  .LIB "fun.mdl"
4  .LIB "types.mdl"
5  .LIB "pnjlim.mdl"
6  .LIB "mos/mos2mey.mdl"
7  .LIB "mos/mos2wd.mdl"
8  .LIB "rlc/r.mdl"
9  .LIB "rlc/c.mdl"
10 .LIB "src/v.mdl"
11
12 .INC "mosinv.sub"
13
```



Rys. 13.3: Trzy inwertery MOS.

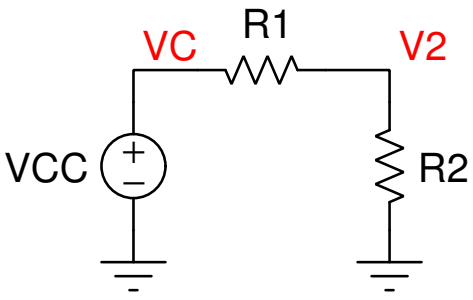
```
14 .MODEL R R
15 .MODEL V SRCV_E V1=0 V2=0 TD=1 TR=1E-5 PW=20E-5 TF=2E-5
16 .MODEL VWE SRCV_E V1=0 V2=10 TD=2e-5 TR=5E-5 PW=50E-5 TF=2E-5 PER=1E3
17 .MODEL NMOS MOS2MEY_E (TYPE= 1, VTO=0.5,PHI=0.7, KP=1.0E-4) #GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e
       -12,CGDO=1e-12,CBD=5e-12,CBS=5e-12)
18 .MODEL PMOS MOS2MEY_E (TYPE=-1, VTO=0.5,PHI=0.7, KP=1.0E-4)
19 #GAMMA=1.83,LAMBDA=0.115,LEVEL=2,CGSO=1e-12,CGDO=1e-12,CBD=5e-12,CBS=5e-12)_
20 VWE.WE WE 0 DC=-5
21 R.WE WE 0 R=10e3
22 ## STEROWANIE NAPIECIECIOWE JAKO ZRODLO PRADOWE
23
24 # INWERTER 1
25 INV:I1 WE WY1 VDD 0
26 R.WY1 WY1 0 R=10e3
27
28 # INWERTER 2
29 INV:I2 WE2 WY2 VDD 0
30
31 # INWERTER 3
32 INV:I3 WE3 WY3 VDD 0
33
34 R.RWY WY3 0 R=10e3
35
36 ## ZASILANIE
37 #V.VDD VDD 0 DC=5 DEBUG=0
38
39 .CMD
40 .OPTI PIV=2, WDCT=1, POST=0, ITL1=100,ITL4=30, HMIN=1e-8, RELT=1e-6
41 #, I_ABS=1e-6
42 #.OPTI NRST=VAR , OPTS
43 .PRINT ADD OP(WE) TR("W*")
44 .PROBE ADD TR("*")
45 #.OPTI DEBU_MTX=0
46 .OP
47 .TRAN STEP=1e-6 TMIN=0 TMAX=40e-5 HMAX=1e-5 TITLE="ANALIZA TRAN"
48 .END
```

Rozdział 14

Optymalizacja

14.1 Optymalizacja układu RR

Na rys.14.1 przedstawiono schemat optymalizowanego układ dwójnika R-R zasilanego ze źródła napięciowego. Zbiór wejściowy symulatora przedstawiono na wydruku 14.1. W zbiorze wejściowym zadano ograniczenia na wartości napięć w węzłach układu dla składowej stałej (w węźle VC) oraz ograniczenia na amplitudę sygnału w węźle $V2$ dla dwóch częstotliwości: $1kHz$ i $2kHz$.



Rys. 14.1: Schemat optymalizowanego układu RR.

Wydruk 14.1: Optymalizacja układu RR

```
1  # Revision : 1.1 (C) AIVA 2002-2010
2  .TASK "OPTYMALIZACJA UKLADU RR"
3
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "units-e.mdl"
7  .LIB "vars.mdl"
8
9  .LIB "rlc/r.mdl"
10 .LIB "rlc/c.mdl"
11 .LIB "src/v.mdl"
12
13 .MODEL R R
14 .MODEL C C
15 .MODEL V V (V1=0, V2=0)
16
17 V.VCC VC 0 (DC=2V)
18
19 R.R1 VC V2 R=1kOhm
20 R.R2 V2 0 R=1kOhm
21
22 .CMD
```

```
23
24 .OPTI REXG=1,PIV=1,ITL4=100,ITL1=40,POST=1,HMIN=1E-8
25 .OP TITLE="Analiza punktu pracy OP"
26 .AC SCALE=DEC FPTS=5 FMIN=1 FMAX=16 TITLE="Analiza AC"
27 # ------------------------------------------------------------
28 .SP ADD OP 0 REL VC S1=2 W1=1 S2=2.2 W2=1
29 #.SP ADD OP 0 REL V2 S1=1 W1=1 S2=1.25 W2=1
30 .SP ADD MAGN 1kHz REL V2 S1=1 W1=1 S2=1.25 W2=1
31 .SP ADD MAGN 2kHz REL V2 S1=1 W1=1 S2=1.25 W2=1
32 #.SP ADD OP 0 REL VCC.I S1=0.0001 W1=1 S2=0.05 W2=1
33 # ------------------------------------------------------------
34 #.VAR ADD R0.R SCALE=1000 MIN=10 MAX=1000
35 .VAR ADD R1.R SCALE=1 MIN=100 MAX=1000
36 .VAR ADD VCC.DC SCALE=1 MIN=1 MAX=3
37 .VAR ADD R2.R SCALE=1 MIN=100 MAX=2000
38 # ------------------------------------------------------------
39 .OPTI WOPT=1,WDCT=2,WAC=0,DEBU_CENT=0,ECHO=ON
40 .OPTI EOPT=1e-7,STIN=0.1,STMX=0.5,MXAN=1000
41 .OPTI DEBU_MDL=0
42 #.ALT R2.R=5k
43 .CENT
44 .ERRC
45 .SVER
46 .SP PRI ACT
47 .VAR PRI VAR
48 .OPTI DEBU_DC=0, WDCT=1, DEBU_OP=0
49 .PRINT ADD DC(V2)
50 .DC VAR1=VCC.DC START1=1 STOP1=2 STEP1=0.1 VAR2=R1.R START2=100 STOP2=1000 STEP2=100 TITLE="Analiza
       charakterystyk DC"
51 .END
```

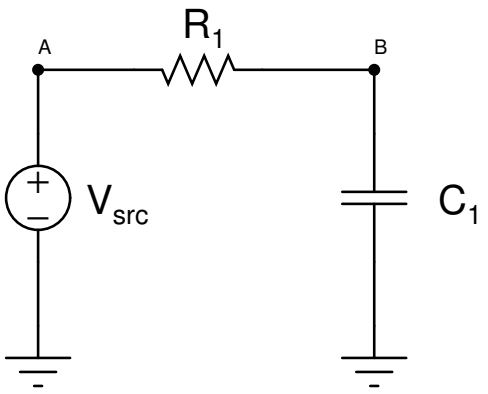
Wykonano optymalizację układu (CENT), wydruk otrzymanych zoptymalizowanych parametrów układu. Wykonano także analizy (OP,AC,DC).

Rozdział 15

Analiza czasowa kierowana zdarzeniami

15.1 Układ RC1

Na rys.15.1 przedstawiono układ *RC* wykorzystywany do testowania analizy czasowej kierowanej zdarzeniami ED-ITA. wzmacniacza Zbiór wejściowy symulatora przedstawiono na wydruku 15.1.



Rys. 15.1: Schemat układu RC.

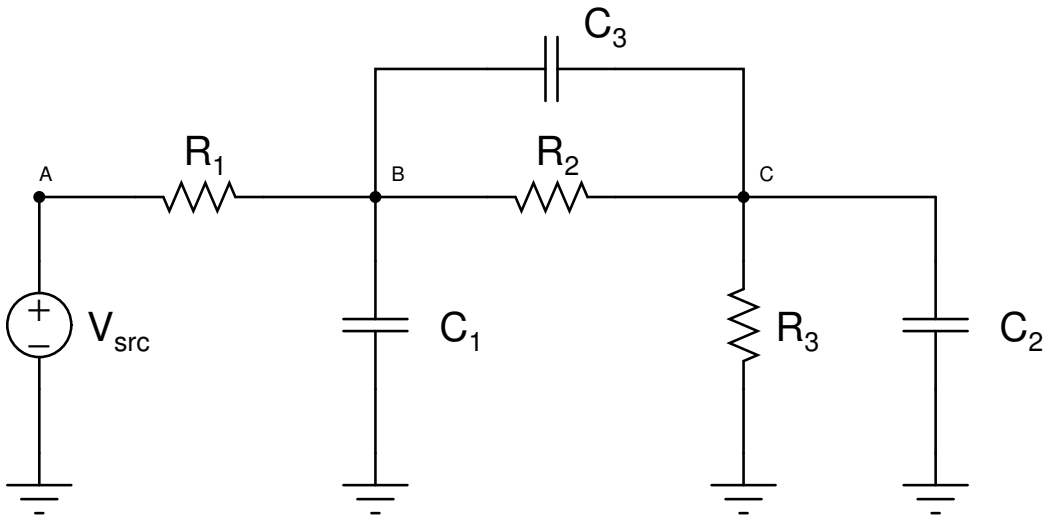
Wydruk 15.1: Zbiór wejściowy symulatora dla układu RC.

```
1  # Revision : 1.1 (C) AIVA 2002-2010
2  .TASK "Event-Driven transient analysis"
3
4  .LIB "fun.mdl"
5  .LIB "types.mdl"
6  .LIB "units.mdl"
7  .LIB "rlc/r.mdl"
8  .LIB "rlc/c.mdl"
9  .LIB "src/v.mdl"
10
11 .MODEL V V
12 .MODEL R R
13 .MODEL C C
14
15 V.VIN A 0 (TD=20E-2,PW=3E-2,V1=1,V2=4)
```

```
16 R.R1 A B R=5K
17 R.R2 B 0 R=5K
18 C.C1 B 0 C=2E-6
19
20 .CMD
21 .PRINT TRAN V(A) V(B)
22 .OP
23 .TRAN STEP=1mS TMAX=500mS TITLE="Analiza czasowa TRAN"
24 #.OPTI HMIN=1E-4
25 .OP
26 .EDITA STEP=1mS TMAX=500mS TITLE="Analiza czasowa kierowana zdrzeniami EDITA"
27 .END
```

15.2 Układ RC2

Na rys.15.2 przedstawiono bardziej złożony układ *RC* wykorzystywany do testowania analizy czasowej kierowanej zdarzeniami ED-ITA. Układ ten jest trudny do analizy przez analizator kierowany zdarzeniami, ze względu na występowanie sprzężenia pomiędzy węzłami sieci (głównie element C_3). Zbiór wejściowy symulatora przedstawiono na wydruku 15.2.



Rys. 15.2: Schemat układu RC2.

Wydruk 15.2: Zbiór wejściowy symulatora dla układu RC2.

```
1  # Revision : 1.1 (C) AIVA 2002-2010
2  .TASK "Event-Driven transient analysis"
3
4  .LIB "fun.mdl"
5  .LIB "types.mdl"
6  .LIB "units.mdl"
7  .LIB "rlc/r.mdl"
8  .LIB "rlc/c.mdl"
9  .LIB "src/v.mdl"
10
11 .MODEL V V
12 .MODEL R R
13 .MODEL C C
14
15 VIN A 0 VSRC (TD=1E-3,PW=3E-3,V1=1,V2=4)
16 R1 A B R=5K
17 R2 B C R=5K
18 C1 B 0 C=2E-6
19 R3 C 0 R=10K
20 C2 C 0 C=5E-6
```

```
21 C3 B C C=1E-6
22
23 .CMD
24 .PRINT TRAN V(A) V(B) V(C)
25 .OP
26 .TRAN STEP=1mS TMAX=30mS TITLE="Analiza czasowa TRAN"
27 #.OPTI HMIN=1E-4
28 .OP
29 .EDITA STEP=1mS TMAX=30mS TITLE="Analiza czasowa kierowana zdrzeniami EDITA"
30 .END
```

Rozdział 16

Testy modeli bibliotecznych

16.1 Niezależne źródło napięciowe

Zbiór wejściowy symulatora do testowania niezależnych źródeł napięciowych i prądowych zależnych od temperatury przedstawiono na wydruku 16.1.

Wydruk 16.1: Niezależne źródła.

```
1  # Revision : 1.2 (C) AIVA 2010
2  #
3  .TASK "Linear voltage and current source"
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "src/vt.mdl"
7  .LIB "src/it.mdl"
8  .MODEL VEXT VT (V1=0,V2=5,TD=10uS,TR=3uS,PW=20uS,TF=4uS,PER=40uS)
9  .MODEL VSIN VT (VSI=1,FREQ=100K)
10 .MODEL IEXT IT (V1=0,V2=5,TD=10uS,TR=3uS,PW=20uS,TF=4uS,PER=40uS)
11 .MODEL ISIN IT (VSI=1,FREQ=100K)
12
13 V1.VEXT EXT 0 (DC=0V)
14 V2.VSIN SIN 0 (DC=0V)
15
16 I1.IEXT EXTI 0 (DC=0V)
17 I2.ISIN SINI 0 (DC=0V)
18
19 R1.IEXT EXTI 0 (R=1Ohm)
20 R2.ISIN SINI 0 (R=1Ohm)
21
22 .CMD
23 .PRINT ADD TR(EXT) TR(SIN) TR(EXTI) TR(SINI)
24 .TRAN STEP=0.1uS TMIN=10nS TMAX=250uS HMAX=5uS TITLE="Analiza czasowa"
25 .END
```

16.2 Dioda półprzewodnikowa - DS

Na rys.16.1 przedstawiono schemat układu testowego modelu diody półprzewodnikowej opisanej modelem DS - rozdział 10.11. Zbiór wejściowy symulatora przedstawiono na wydruku 16.2.

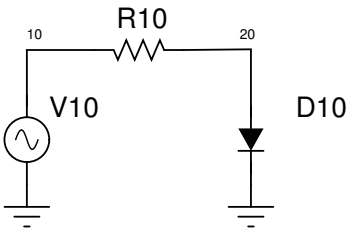
16.3 Tranzystor bipolarny - BJTL2

Na rys.16.2 przedstawiono schemat układu testowego modelu tranzystora bipolarnego (bjt) opisanego modelem BJT2 - rozdział 10.13. Testowane są dwa tranzystory typu *pnp* i *npn*. Zbiór wejściowy symulatora przedstawiono na wydruku 16.3.

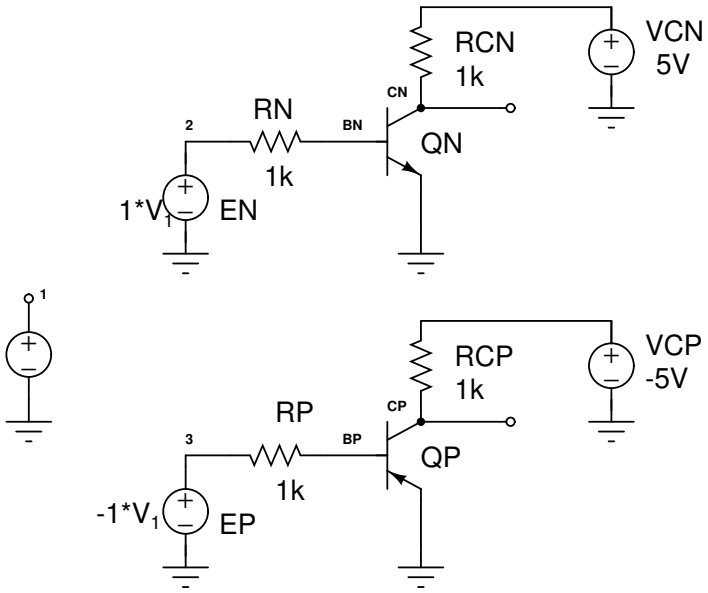
Wydruk 16.3: Test modelu tranzystora bipolarnego BJT (plik: test-models/bjtnp.cir).

Wydruk 16.2: Dioda półprzewodnikowa.

```
# Revision : 1.2 (C) AIVA 2002-2010
.TASK "Diode test"

.LIB "units.mdl"
.LIB "types.mdl"
.LIB "bjt/pnjlim.mdl"
.LIB "math/pulse.mdl"
#.LIB "fun.mdl"
.LIB "src/v.mdl"
.LIB "src/i.mdl"
.LIB "rlc/r.mdl"
.LIB "rlc/l.mdl"
.LIB "rlc/c.mdl"
.LIB "d/ds.mdl"

.MODEL R R
.MODEL C C
.MODEL L L
.MODEL CJ CJ
.MODEL CJX CJX
.MODEL V SRCV_E V1=0 V2=0 TD=2e-6 TR=1E-6 PW=60E-6 TF=2E-6 VSI=10V FREQ=10kHz PER=70e-6
.MODEL J SRCI_E V1=0 V2=10 TD=2e-6 TR=1E-6 PW=60E-6 TF=2E-6 PER=1E3
.MODEL D DS

# ROZNE WYNIKI JESLI ZRODLA NA POCZATKU LUB NA KONCU OPISU
V.VB A 0 DC=1

R.R2 A D R=1kOhm
CJ.C2 D 0 C=10nF
D.D1 D 0 DEBUG=1 CAP_ON=1 LIM_ON=0 TEMP=330
D.D2 0 D DEBUG=0 CAP_ON=1 LIM_ON=0 TEMP=320

.CMD
.OPTI ITL1=500, PIV=1, REXG=ZERO, RELT=1e-6, POST=1, WNRI=0, ALLP=1
.OP
.PROBE ADD TR(*) MD("[C-E]") # PH(*)
.OPTI WDCT=0, DEBU_MOD=0, DEBU_AC=0 , DEBU_OUT=0, DEBU_ELE=0, DEBU_TRAN=0
.OPTI SR=TRAP
.OPTI SR=GEAR
.TRAN STEP=1E-6 TMIN=1E-8 TMAX=300E-6 HMAX=5E-6 TITLE="ANALIZA TRAN (1)"
.TRAN STEP=2E-6 TMIN=2E-8 TMAX=500E-6 HMAX=5E-6 TITLE="ANALIZA TRAN (2)"
.STDS STEP=1E-6 TMAX=300E-6 PER=70E-6 AUT TITLE="ANALIZA STDS"
.OPTI WAC=0, DEBU_AC=0, DEBU_OUT=0, DEBU_ELE=0
.AC SCALE=LIN FPTS=2000 FMIN=4.5k FMAX=5.5k TITLE="ANALIZA AC (1)"
.AC SCALE=LIN FPTS=2000 FMIN=4.5k FMAX=5.5k TITLE="ANALIZA AC (2)"
.END
```



Rys. 16.1: Układ testowy diody (plik: test-models/d.cir).



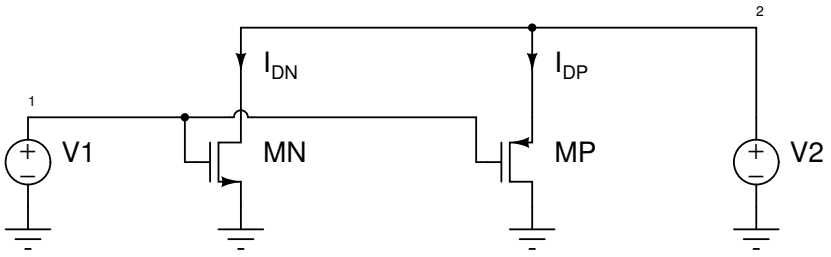
Rys. 16.2: Układ testowy tranzystorów bipolarnych (plik: /test-models/bjtnp.cir).

```
1  # Revision : 1.2 (C) AIVA 2002-2010
2  .TASK "bjt.cir - test of NPN i PNP"
3  # Library
4  .LIB "types.mdl"
5  .LIB "units.mdl"
6  .LIB "math/pulse.mdl"
7  .LIB "src/v.mdl"
8  .LIB "rlc/r.mdl"
9  .LIB "rlc/l.mdl"
10 .LIB "rlc/c.mdl"
11 .LIB "bjt/bjt2.mdl"
12 # MODEL LINES
13 .MODEL NPN BJT2 (TYPE= 1,IS=1E-15,BF=50,BR=10,TF=0.2U,TR=0.2U,CJC=50P,CJE=50P,CJS=50P)
14 .MODEL PNP BJT2 (TYPE=-1,IS=1E-15,BF=50,BR=10,TF=0.2U,TR=0.2U,CJC=50P,CJE=50P,CJS=50P)
15 .MODEL VIN V (V1=0,V2=2,TD=1E-9,TR=1E-9,TF=1E-9,PW=20E-9,TDSI=1E-2,VSI=0.7,FREQ=0.5K)
16 .MODEL VCC V (DC=0)
17 .MODEL C C
18 .MODEL R R
19 # INPUT
20 VIN.VBE 1 0 (DC=0)
21 # C B E
22 # NPN
23 NPN.QN CN BN 0 0
24 R.RN 2 BN 1kOhm
25 R.RCN 4 CN 1kOhm
26 EN 2 0 V(1) 1
27 # PNP
28 PNP.QP CP BP 0 0
29 R.RP 3 BP 1kOhm
```

```
R.RCP 5 CP 1kOhm
EP 3 0 V(1) -1
# SRC
VCC.VCN 4 0 5
VCC.VCP 5 0 -5
# MODELE
.CMD
.OPTI PIV=0,FIXS=0,ITL1=100
.OP
.PROBE DC QN.ICOL QP.ICOL QN.IBAS QP.IBAS
.PROBE TRAN V(BN) V(CN) V(BP) V(CP) V(1)
.DC VAR1=VBE.DC START1=0.5 STOP1=1 STEP1=5m
.TRAN TMIN=0.5nS TMAX=60nS TITLE="Transient analysis (TRAN)"
.END
```

16.4 Tranzystor MOS

Na rys.16.3 przedstawiono schemat układu testowego modelu tranzystora MOS opisanego modelem Meyera - rozdział 10.14. Testowane są dwa tranzystory z kanałem typu n i p. Zbiór wejściowy symulatora przedstawiono na wydruku 16.4.



Rys. 16.3: Układ testowy tranzystorów MOS (plik: test-models/mos-n-p.cir).

Wydruk 16.4: Tranzystor MOS.

```
# Revision : 1.2 (C) AIVA 2002-2010
.TASK "POJEDYNCZY INWERTER MOS"
.LIB "types.mdl"
.LIB "units.mdl"
.LIB "pnjlim.mdl"
.LIB "math/pulse.mdl"
.LIB "src/v.mdl"
.LIB "rlc/r.mdl"
.LIB "rlc/c.mdl"
.LIB "mos/mos2mey.mdl"

.MODEL R R
.MODEL C C
.MODEL VDD V (V1=0,V2=0,TD=1,TR=1E-5,PW=20E-5,TF=2E-5)
.MODEL V V (V1=0,V2=5,TD=4E-5,TR=1E-5,PW=20E-5,TF=2E-5)

.MODEL NMOS MOS2MEY (TYPE= 1)
#VTO=0.5,KP=1.0E-3) #GAMMA=1.83,LAMBDA=0.115,LEVEL=2,
#CGSO=1e-12,CGDO=1e-12,CBD=5e-12,CBS=5e-12)#PHI=0.9,
.MODEL PMOS MOS2MEY (TYPE=-1)
#VTO=-0.5,KP=1.0E-3) #GAMMA=1.83,LAMBDA=0.115,LEVEL=2,
#CGSO=1,CGDO=1e-12,CBD=5e-12,CBS=5e-12) #PHI=0.9,

## STEROWANIE NAPIECIECIOWE JAKO ZRODLO PRADOWE
V.VBN WEN 0 DC=1
R.RWEN WEN 0 R=10kOhm
VDD.VDD VDDPLUS 0 DC=5
R.RDDN VDDN 0 R=100
## N-MOS - VDS>0
# kanal wzbogacany Up>0: VGS>0
# kanal wzbogacany Up<0: VGS<0
NMOS.M12N VDDN WEN 0 0 (W=25e-6,L=5e-6)

## P-MOS - VD<0V VG<0
V.VBP WEP 0 DC=-1
R.RWEP WEP 0 R=10kOhm

VDD.VDDM VDDMINUS 0 DC=-5
R.RDDP VDDP 0 R=100
```

```
40 ## N-MOS
41 PMOS.M12P VDDP WEP 0 0 (W=25e-6,L=5e-6)
42
43 .CMD
44 .OPTI PIV=2, REXG=MIN, RELT=1e-3, WDCT=3, WNRI=1
45 .OPTI ITL1=30, DEBU_MOD=0, DEBU_MDL=0
46 .PRINT ADD OP("W*")
47 .OPTI DEBU_MDL=4
48 .OPTI DEBU_MDL=0
49 .OP
50 .PRINT ADD TR(IN) TR("VDDP") TR("WDDN") PAR(HN) PAR(NR)
51 .PROBE ADD TR(IN) TR("WE") TR("WY") PAR(HN)
52 .OPTI SR=GEAR
53 .OPTI SR=TRAP
54 .OPTI POST=1,EPTR=1e-2,ALLP=1,ITL4=50
55 .OPTI HMIN=10nS, HINI=10nS, FIXS=0, RELT=1e-4, ERTR=1e-3
56 .TRAN STEP=0.1mS TMIN=0 TMAX=40mS HMAX=0.1mS TITLE="Transient analysis (TRAN)"
57 .END
```

Bibliografia

[A+81] A.Vladimirescu u. a. ; Department of Electrical Engineering and Computer Science University of California (Hrsg.): *SPICE version 2G. User's guide.* Berkeley: Department of Electrical Engineering and Computer Science University of California, 1981

[A.E86] A.E.Ruehli: *Advanced in CAD for VLSI.* Bd. III: *Circuit Analysis, Simulation and Design.* Amsterdam, The Netherlands : Elsevier Science Publishers B.V., 1986

[ARN83] Arthur Richard Newton, Alberto L. Sangiovanni-Vincentelli: Relaxation-Based electrical simulation. In: *IEEE Transactions On Electron Devices* ED-30 (1983), Sep, Nr. 9, S. 1184–1207

[B+92] B.Folkman, H.L.Offereins u. a.: A simulation tool for mechanical sensor design (SENSOR). In: *Sensors and Actuators* A32 (1992), S. 521–524

[C+] Costa, A. u. a.: *Debian Project - dpkg.* WWW page, . – http:/andothersmanpages.debian.net/cgi-bin/man.cgi?query=dpkg

[D+90] D.J.Erdman u. a. ; MCNC Center for Microelectronics and Duke University (Hrsg.): *CAzM - Circuit AnalyZer with Macromodeling Version Realize 4.1.* USA: MCNC Center for Microelectronics and Duke University, 1990

[Duy94] Duyn, D.C. van: Modeling and simulation of solid-state transducers: the thermal and electrical energy domain. In: *Sensors and Actuators* A41 (1994), Jan, S. 268–274

[E.S98a] E.Senturia, Stephen: CAD challenges for microsensors, microactuators and microsystems. In: *PROCEEDINGS OF THE IEEE* 86 (1998), S. 1611–1626

[E.S98b] E.Senturia, Stephen: Simulation and design of microsystems: a 10-year perspective. In: *Sensors and Actuators* A67 (1998), S. 1–7

[Fot97] Foty, D.: *Mosfet Modeling with Spice.* Upper Saddle River, NJ : Prentice Hall, 1997

[Gaw93] Gawryś, Michał: *UNIX: Narzędzia programistyczne yacc i lex.* Warszawa : Wydawnictwo PLJ, 1993

[GD74] G. Dahlquist, A. B.: *Numerical methods.* Prientice-Hall, 1974

[GEF67] George E. Forsythe, Cleve B. M.: *Computer Solution of Linear Algebraic Systems*. Prientice-Hall, 1967

[J.O94] J.Ogrodzki: *Circuit simulation methods and algorithms*. Boca Raton,Florida,USA : CRC Press, 1994

[J.O95] J.Ogrodzki: *Komputerowa analiza układów elektronicznych*. Warsaw : PWN, 1995

[K.M04] K.Madsen, O.Tingleff H.B.Nielsen: *Methods for non-linear least aquares problems (2nd Edistion)*. Informatics and Mathematical Modelling Technical Univerity of Denmark, 2004

[LD98] Lynn D.Gabbay, Stephen E.: Automatic generation of dynamic macromodels using quasistatic simulations with modal analysis. In: *in Tech. Dig. IEEE Solid-State Sensor and Actuator Workshop* (June 1998), S. 197–200

[Lin92] Lin, Leon O.Chua Pen-Min: *Komputerowa analiza układów elektronicznych*. Warszawa : WNT, 1992

[LNC00] Lillian N. Cassel, Richard H. A.: *Computer Networks and Open Systems: An Application Development Perspective*. Jones & Bartlett Publisher, 2000. – ISBN 0–7637–1122–5

[L.W75] L.W.Nagel ; Electronics Research Laboratory (Hrsg.): *SPICE2 A computer program to simulate semiconductor circuits*. Berkeley: Electronics Research Laboratory, May 1975

[L.W80] L.W.Nagel: *SPICE2 a computer program to simulate semiconductor circuits*. Berkeley 94720 : Electronic Research Laboratory College Engineering University of California, 1980

[P+07] Press, William H. u. a.: *Numerical Recipes - The Art of Scientific Computing*. Cambridge Uniwersity Press, 2007. – ISBN 978–0–521–88407–5

[pag] pages, Linux man: *vim*. WWW page and system manpages, . – http://andothersmanpages.debian.net/cgi-bin/man.cgi?query=vim

[Ped89] Pedro, Pierre P. d.: Mecanique Vibratoire, Systemes discrets lineaires. In: *Presses Polytechniques Romandes,Laussane* (1989)

[Pla12] Plaskura, P.: *Quela 2 - podręcznik użytkownika*. 2012. – niepublikowany

[R.K77] R.K.Brayton, G.D.Hachtel F.G.Gustavson: A new efficient algorithm for solving differential-algebraic systems using implicit backward-differentiation formulas. In: *Proc. of the IEEE* 60 (1977), Apr, Nr. 1, S. 399–402

[S.C91] S.Crary, Y.Zhand O.Juma: Software tools for designers of sensor and actuator CAE systems. In: *TRANSDUCERS'91. 1991 International Conference on Solid-State Sensors and Actuators. Digest of Technical Papers* (1991), Jun, S. 498–501

[Tel11] Tell, Stephen: *http://gwave.sourceforge.net/*. WWW page, 2011

[Vla94] VLADIMIRESCU, A.: *The Spice Book.* New York NY : John Wiley and Sons, 1994

[Wil63] WILKINSON, J.H.: *Rounding Errors in Algebraic Processes.* Prientice-Hall, 1963

[Wil65] WILKINSON, J.H.: *The algebraic Eigenvalue Problem.* Oxford : Clarendon Press, 1965

[Wir04] WIRTH, Niklaus: *Algorytmy + struktury danych = programy.* Warszawa : Wydawnictwo Naukowo-Techniczne, 2004

Spis tablic

3.1 Operatory języka MDL 34
3.2 Predefiniowane funkcje wbudowane 35
3.3 Stałe wbudowane w program *Dero* 36
3.4 Predefiniowane zmienne symulatora dostępne z poziomu modelu. 36
3.5 Funkcje do komunikacji z kernelem z poziomu modelu 37

4.1 Lista opcji 53

7.1 Uogólnione zmienne dla kilku wybranych środowisk. 112
7.2 Środowiska zdefiniowane w języku HDL-A. 113
7.3 Podwójne analogie systemów elektryczno-mechanicznych 113
7.4 Analogie pomiędzy środowiskiem elektrycznym, a mechanicznym 1 i 2. . 123
7.5 Analogie równań pomiędzy środowiskiem elektrycznym, a mechanicznym 1 i 2. 124

10.1 Lista parametrów modelu rezystancji RT. 159
10.2 Lista parametrów modelu pojemności CT. 161
10.3 Lista parametrów modelu indukcyjności LT. 163
10.4 Lista parametrów źródła napięciowego VT 165
10.5 Lista parametrów modelu źródła prądowego IT 168
10.6 Lista parametrów diody 171
10.7 Lista parametrów modelu tranzystora bipolarnego BJT. 176
10.8 Lista parametrów modelu tranzystora MOS. 187

Spis rysunków

1.1 Wejście wyjście w programie 14

2.1 Struktura zadania 17
2.2 Przykładowy schemat analizowanego układu z dziedziny elektrycznej . . 24

4.1 Optymalizowany dwójnik RR 52

5.1 Interfejs systemu DeroWWW na platformie Quela. 60
5.2 Katalog zadań i wyników dla symulatora Dero. 60
5.3 Katalog dokumentacji. 60
5.4 Okna systemu pomocy *manpages*. 61
5.5 Aplet wyników analizy przykładowego układu. 61

6.1 Zawartość tablicy sum kroków hs_{n-1} w chwili t_{n-1}. 72
6.2 Rząd l SR przy starcie w kolejnych punktach czasowych t_n. 78
6.3 Biliniowa funkcja układowa. 105
6.4 Przebieg kwadratu modułu biliniowej funkcji układowej. 106
6.5 Graficzna ilustracja ograniczeń fazowych na płaszczyźnie zespolonej. . . 107
6.6 Aproksymacja odcinkowo-liniowa obszaru sprawności. 108
6.7 Trójwymiarowa aproksymacja odcinkowo-liniowa obszaru sprawności. . . 109

7.1 Gałąź typu napięciowego 114
7.2 Gałąź typu prądowego 115
7.3 Szablon ZMPW dla gałęzi typu prądowego 115
7.4 Szablon ZMPW dla gałęzi typu prądowego z prądem jako niewiadomą . 115
7.5 Szablon ZMPW dla gałęzi typu napięciowego 115
7.6 Szablon ZMPW dla gałęzi opisanej równaniem ładunkowym 116
7.7 Szablon ZMPW dla gałęzi opisanej równaniem strumieniowym 116

8.1 Drzewo parsowania 125
8.2 Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych 127
8.3 Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych 128
8.4 Redukcja drzewa 129
8.5 Generacja postaci symbolicznej dla operacji mnożenia 129
8.6 Przykład drzewa parsowania dla wyrażenia (8.1). 131
8.7 Drzewo pochodnej wyrażenia (8.1) obliczonej względem a. 131

9.1 Struktura macierzy układu w trakcie wstępnego przestawienia wierszy i kolumn. 139

10.1 Model rezystancji (plik biblioteczny rlc/rt.mdl). 159
10.2 Model pojemności (plik biblioteczny rlc/ct.mdl). 161
10.3 Model indukcyjności (plik biblioteczny rlc/lt.mdl). 163
10.4 Model źródła napięciowego (plik biblioteczny src/vt.mdl) 165
10.5 Model źródła prądowego (plik biblioteczny src/it.mdl) 168
10.6 Model wielkosygnałowy diody (plik biblioteczny d/ds.mdl) 171
10.7 Model tranzystora BJT (plik biblioteczny bjt/bjt2.mdl). 176
10.8 Model wielkosygnałowy tranzystora MOS. 186

11.1 Układ RLC (plik: linear-filters/rlc.cir). 211
11.2 Układ ze sterowanym ładunkiem (plik: linear-filters/charge.cir) 212
11.3 Układ ze sterowanym strumieniem magnetycznym (plik: linear-filters/flux.cir). 213

12.1 Wzmacniacz z tranzystorem pracującym w układzie wspólnego emitera. 215
12.2 Wzmacniacz mocy (plik: nonlin-amp/powamp.cir). 217
12.3 Schemat ze wzmacniaczem różnicowym (plik: nonlin-amp/diff-pair.cir). . 218
12.4 Schemat układu idealnego wzmacniacza operacyjnego (plik: oper-amp-ideal.cir). 220
12.5 Schemat układu ua741 (plik: nonlin-oper/ua741x1.cir). 221

13.1 Schemat układu do symulacji bramek (plik: nonlin-digital/gates.cir). . . 224
13.2 Inwerter zbudowany z tranzystorów MOS. 226
13.3 Trzy inwertery MOS. 227

14.1 Schemat optymalizowanego układu RR. 229

15.1 Schemat układu RC. 231
15.2 Schemat układu RC2. 232

16.1 Układ testowy diody (plik: test-models/d.cir). 237
16.2 Układ testowy tranzystorów bipolarnych (plik: /test-models/bjtnp.cir). 237
16.3 Układ testowy tranzystorów MOS (plik: test-models/mos-n-p.cir). . . . 238

Lista algorytmów

1 Algorytm Newtona-Raphsona 69
2 Algorytm analizy czasowej 80
3 Uproszczony algorytm analizy czasowej zastosowany w symulatorze *Dero* . 81
4 Algorytm analizy czasowej stanu ustalonego. 86
5 Algorytm analizy czasowej stanu ustalonego metodą Bukowskiego. . . . 87
6 Opracowanego algorytmu analizy czasowej kierowanej zdarzeniami . . . 90
7 Pełny algorytm optymalizacji 97
8 Algorytm optymalizacji metodą Lavenberga-Marquardta 97
9 Algorytm optymalizacji zastosowany w programie. 98
10 Algorytm obliczania 7CX 104
11 Przekształcanie drzewa parsowania 126
12 Redukcja gałęzi drzewa 130
13 Generacja pochodnej w postaci symbolicznej 130
14 Podstawowy algorytm rozkładu LU z normalizacją macierzy U. 135
15 Pełny algorytm wstępnego przenumerowania zastosowany w programie *Dero* 138
16 Algorytm rozkładu LU - algorytm Gaussa ze skalowaniem kolumn macierzy. 143
17 Algorytm rozkładu LU z przestawieniem elementów na przekątnej. . . . 144

Oznaczenia

Θ Kat. 112

ω Zmienna przyspieszenia katowowego. 112, 113

ψ Strumień. 112, 114, 116, 121, 122

τ Zmienna momentu obrotowego. 112, 113

a Wymuszenie uogólnione. 113

a Przepływ uogólniony. 113

B Tarcie. 123, 124

C Pojemnosc. 123, 124

d Zmienna przesuniecia. 113

e Wymuszenie. 112

f Przeplyw. 112

f Sila. 112, 123

f Zmienna sily. 113

fr flow rate. 113

hfr heat flow rate. 113

i Prad. 112–124

i Zmienna pradowa. 113

K Sprezystosc. 123, 124

L Indukcyjnosc. 123, 124

L Smarnosc. 123

m Masa. 113, 123, 124

P Moc. 112

P Cisnienie. 112

p Ped. 112

p Polozenie. 112

p Zmienna cisnienia. 113

q Ladunek. 112, 114, 116, 121

R Rezystancja. 123

t Temperatura. 113

V Przyspieszenie. 112, 113, 123

V Objetosc. 112

v Napiecie. 112, 114–124

v Zmienna napieciowa. 112, 113

x Zmienna, Przesuniecie. 112, 114

Słownik

AC *ang. AC Analysis* - analiza małosygnałowa (w funkcji częstotliwości).. 230

CENT centrowanie deterministyczne.. 230

DC *ang. Direct Current Analysis* - analiza charakterystyk stałoprądowych.. 230

ED-ITA analiza czasowa kierowana zdarzeniami.. 231, 232

MDL język opisu modeli programu *Dero* - *ang. Model Definition Language*.. 25, 223, 225, 226

MOS *ang. Metal Oxide Semiconductor*.. 225, 226, 248

OP *ang. Operating Point Analysis* - analiza punktu pracy.. 230

Praca poświęcona jest symulatorowi Dero, który dedykowany jest przede wszystkim do symulacji układów elektronicznych. Program posiada język opisu modeli elementów umożliwiający użytkownikowi tworzenie własnych modeli elementów. W pracy opisano język wejściowy, język modelowania elementów oraz komendy analiz. Przedstawiono metody i algorytmy obliczeniowe wykorzystane w symulatorze. Zamieszczono przykłady użycia oraz wydruki plików bibliotek modeli elementów. Na potrzeby symulatora został stworzony system symulacji DeroWWW poprzez stronę WWW. Obecnie jest on zintegrowany z platformą e-learningową Quela. Systemy dostępne są w sieci:

DeroWWW: http://dero.aiva.pl
Platforma e-learningowa Quela: http://quela.aiva.pl

ISBN 978-83-937245-1-2