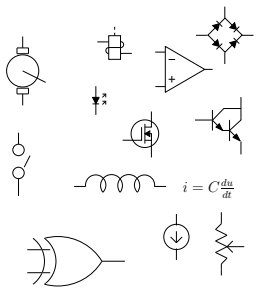
Zaawansowane metody symulacji układów elektronicznych

Metody i algorytmy obliczeniowe

Paweł Plaskura



ISBN 978-83-937245-0-5

Recenzent: dr inż. Andrzej Góralski

Dokument złożono systemem składu LaTeX
Wykorzystano system operacyjny Linux (Debian) 
Wersja dokumentu: 1.3

Zaawansowane metody symulacji układów elektronicznych

Paweł Plaskura

http://epub.aiva.pl?isbn=978-83-937245-0-5

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

ISBN 978-83-937245-0-5

Recenzja

Książka przedstawia zaawansowane metody symulacji układów elektrycznych i elektronicznych w zakresie metod i algorytmów obliczeniowych, zastosowanych także do symulacji mikrosystemów. Pełne przedstawienie zagadnień jak i samej poruszanej tematyki czyni niniejszą książkę wyjątkową, a właściwie unikalną na rynku polskim w postaci pozycji literaturowej nieodzownej dla osób zajmujących się analizą i symulacją układów, w tym układów elektrycznych i elektronicznych.

Zakres wiedzy jak i jej złożoność przedstawiona jest w sposób jasny, umożliwiający czytelnikowi dogłębne zrozumienie zagadnień analizy i symulacji układów. Należy podkreślić, że szczegóły implementacyjne algorytmów w programach komercyjnych są objęte tajemnicą firmy. Dzieje się to z racji ich złożoność, czasu potrzebnego na implementację i testowanie, co wiąże się z wysokimi kosztami, a także koniecznością posiadania odpowiedniej kadry z dużym doświadczeniem interdyscyplinarnym.

Opisane w publikacji metody zostały zweryfikowane, dopracowane oraz zaimplementowane w autorskim symulatorze *Dero*. Było to możliwe dzięki specjalistycznej wiedzy teoretycznej i praktycznej autora obejmującej kilka dziedzin oraz wieloletniemu doświadczeniu w implementacji systemów symulacji. Stanowi to o dużej wartości publikacji.

dr inż. Andrzej Góralski

Spis treści

Rozdział 1

Wstęp

Rozwój metod analizy i projektowania układów elektronicznych wymusiło pojawienie się na rynku w końcu lat 60-tych XX wieku układów scalonych wielkiej skali integracji (VLSI). Metody obliczeniowe są stale rozwijane i udoskonalane.

Rozwój układów scalonych doprowadził do szybkiego rozwoju oprogramowania do symulacji. W chwili obecnej na rynku dostępnych jest wiele środowisk do symulacji, jednak szczegóły implementacji algorytmów nie są najczęściej publikowane. Ze względu na duże skomplikowanie problemów w dziedzinie elektroniki, opracowano szereg języków opisujących układy elektroniczne (np. opis układu w postaci listy elementów), które były stosunkowo łatwe do nauczenia się i wykorzystania. Języki opisu ewoluowały, co doprowadziło do stworzenia uogólnionych języków opisu (np. VHDL), które mogły być stosowane w różnych dziedzinach. Algorytmy i metody obliczeniowe z dziedziny elektroniki oraz odpowiednie oprogramowanie zaczęto stosować w innych dziedzinach. Pojawiło się pojęcie mikrosystemów. Zostało to wymuszone przez rynek, ze względu na szerokie wykorzystywanie np. w układach sterowania układów elektronicznych. Doprowadziło to do powstania nowej dziedziny - mechatroniki. Układy mechaniczno-elektroniczne są obecnie szeroko stosowane. Problemem jest projektowanie i symulacja takich układów za pomocą symulatorów mikrosystemów. Umożliwiają one opis matematyczny i symulację układów złożonych, których części pochodzą z różnych środowisk.

Celem niniejszej pracy jest omówienie wybranych bardziej zaawansowanych i mających praktyczne znaczenie zagadnień związanych z symulacją układów elektronicznych oraz mikrosystemów. W pracy podano szereg odnośników do prac omawiających szczegółowo dane zagadnienie. Ze względu na złożoność tematyki, nie jest możliwe przestudiowanie omawianych zagadnień bez studiowania literatury dla danego zagadnienia.

Skoncentrowanie się na symulacji układów elektronicznych ma swoje uzasadnienie. W tej dziedzinie nastąpił i następuje najszybszy postęp.

Szereg metod i algorytmów autor zastosował w praktyce w symulatorze *Dero*.

1.1 Zawartość pracy

Praca składa się z 12 rozdziałów. W każdym rozdziale starano się omówić jedno ważne zagadnienie. Na końcu każdego rozdziału zamieszczono literaturę problemu.

W rozdziale 1 omówiono podstawowe zagadnienia związane z symulacją układów elektronicznych i mikrosystemów. Omówiono obecny stan wiedzy oraz przedstawiono kierunki dalszego rozwoju. Zamieszczono krótkie wprowadzenie w tematykę.

W rozdziale 2 przedstawiono podstawowe zagadnienia związane z wykonywaniem obliczeń numerycznych. Omówiono sposoby zapisu liczb w komputerach i wykonywania obliczeń. Omówiono zagadnienia poprawności numerycznej algorytmów oraz podstawowe problemy numeryczne. Z punktu widzenia programów zagadnienia te są krytyczne i należy je uwzględniać przy projektowaniu i implementacji algorytmów. Dotyczy to także użytkowników systemów, którzy także powinni posiadać odpowiednią wiedzę z zakresu metod numerycznych.

W rozdziale 3 zamieszczono podstawowe pojęcia z dziedziny matematyki, które są wykorzystywane w pracy: normy wektorów, normy macierzy, podział macierzy, wskaźnik uwarunkowania macierzy.

W rozdziale 4 omówiono metody rozwiązywania równań liniowych i układów równań liniowych: metody klasyczne (eliminacja Gaussa, metoda rozkładu LU) wykorzystujące macierze pełne i rzadkie oraz metody iteracyjne. Podano szereg metod podnoszących dokładność obliczeń i redukujących nakłady obliczeniowe: przestawienia kolumn i wierszy ze względu na strukturę, wybór elementów podstawowych macierzy na przekątnej, skalowanie elementów macierzy. W rozdziale omówiono także szereg metod iteracyjnego rozwiązywania układów równań i ich mutacje. Omówiono warunki konieczne zbieżności oraz podano warunki zakończenia iteracji.

W rozdziale 5 przedstawiono zagadnienia analizy punktu pracy w programach symulacji. Omówiono metody rozwiązywania równań nieliniowych. Szczególne znaczenie ma tutaj metoda Newtona-Raphsona, która jest szeroko stosowana w programach symulacji do znajdowania pojedynczego rozwiązania. Omówiono szereg rozszerzeń metody poprawiających jej niezawodność i redukujących nakłady obliczeniowe. Omówiono zagadnienia analizy węzłów odosobnionych oraz metody zabezpieczania modeli elementów przed problemami numerycznymi. Omówiono wyznaczanie charakterystyk stałoprądowych. Zajęto się zagadnieniem wyznaczania wszystkich wszystkich punktów pracy. Omówiono szereg metod kontynuacji, w tym metody: podstawowe startujące z punktu 0, stopniowania wartości wymuszeń, stopniowego zwiększania wpływu nieliniowości. Omówiono metody homotopii i odcinkowo-liniowe.

Zajęto się także zagadnieniem wykorzystania algorytmów ewolucyjnych do wyznaczania wszystkich punktów pracy. Omówiono różne strategie ewolucyjne i algorytmu, w tym najskuteczniejszy *Hybrydowy Algorytm Ewolucyjny*.

Omówiono także metody relaksacyjne zastosowane do poszukiwania rozwiązań, które w określonych zastosowaniach są niezwykle wydajne. Omówiono technikę wirtualnych pojemności oraz metodę nad- i podrelaksacyjną.

W rozdziale 6 omówiono najważniejsze metody rozwiązywania równań różniczkowych zwyczajnych wykorzystywanych w algorytmach omawianych w pracy. Omówiono pojęcie schematu różnicowego, metody doboru rzędu i kroku schematu różnicowego oraz zagadnienia dokładności. Skupiono się tylko na wybranych najważniejszych i mających praktyczne znaczenie zagadnieniach.

W rozdziale 7 przedstawiono metody i algorytmy klasycznej analizy czasowej. Omówiono także metody poszukiwania stanu ustalonego okresowego dla układów nieliniowych.

W rozdziale 8 przedstawiono bardziej zaawansowane metody symulacji czasowej kierowanej zdarzeniami wykorzystujące iteracyjne metody rozwiązywania układów równań nieliniowych. Dla niektórych klas układów są one bardzo efektywne. Omówiono szereg stosowanych technik oraz warunki, jakie muszą być spełnione aby metody te można było stosować. Omówiono także szereg programów, które implementują omawiane techniki wraz z zastosowanymi modyfikacjami.

W rozdziale 9 omówiono zagadnienia związane z małosygnałową analizą częstotliwościową, która jest powszechnie stosowana w symulatorach.

W rozdziale 10 przedstawiono podstawowe metody optymalizacji. Zdefiniowane podstawowe pojęcia oraz algorytmy optymalizacji deterministycznej. W rozdziale 10.2 zajęto się metodami wyznaczania obszarów sprawności dla układów liniowych. Techniki te użyteczne są np. przy projektowaniu filtrów. Są podstawą do maksymalizacji uzysku produkcyjnego.

W rozdziale 11 przedstawiono zagadnienia związane z wyznaczaniem i obliczaniem pochodnych na potrzeby systemów symulacji. Omówiono opracowany przez autora i zaimplementowany w symulatorze *Dero*algorytm różniczkowania symbolicznego. Działanie algorytmu można przetestować w Internecie pod adresem: http://deriv.aiva.pl.

Rozdział 12 poświęcony jest automatycznemu układaniu równań sieci dla poszczególnych rodzajów analiz. Skoncentrowano się na zmodyfikowanej metodzie potencjałów węzłowych(ZMPW), która jest szeroko stosowana. Podano podstawowe równania sieci elektrycznych, ich szablony ZMPW oraz transformacje równań na inne dziedziny. Dzięki transformacjom możliwa jest analiza mikrosystemów.

W pracy starano się zająć najważniejszymi metodami i technikami wykorzystywanymi w praktyce oraz takimi, które są obiecujące. Celowo pominięto przykłady, które ułatwiają zrozumienie omawianych technik, aby nie rozpraszać uwagi czytelnika. Przykłady można znaleźć w literaturze oraz w serwisie poświęconym symulatorowi *Dero*pod adres http://dero.aiva.pl. Ograniczono także niezbędną liczbę wzorów i przekształceń do minimum. Starano się nie pomijać ważnych przekształceń, które dla wąskiego grona specjalistów (w tym matematyków) są zrozumiałe, lecz mogą stanowić trudność dla czytelnika. Podano szereg algorytmów także w postaci diagramów. W praktycznych implementacjach stosunkowo proste algorytmy stają się bardzo rozbudowane. Z tego względu szczegóły implementacyjne zostały pominięte.

Praca w żaden sposób nie wyczerpuje omawianych zagadnień, ze względu na złożoność i skomplikowanie tematyki oraz stały postęp. Szereg przykładów oraz zagadnienia bardziej szczegółowe można znaleźć w literaturze, głównie w specjalistycznych publikacjach pokonferencyjnych. Należy jasno powiedzieć, że szczegóły implementacyjne algorytmów w komercyjnych programów najczęściej są objęte tajemnicą firmy.

Omówione w pracy algorytmy zostały w większości zaimplementowane w oprogramowaniu stworzonym przez autora i przetestowane w praktyce.

1.2 Techniki analizy czasowej

Obecnie stosowane techniki analizy czasowej można podzielić na trzy podstawowe grupy, dostosowane do specyfiki symulowanych układów. Należą do nich:

symulacja dużych sieci (*ang. circuit simulation*) to analiza odpowiedzi układu analogowego wykorzystująca metody bezpośrednie - oparte na rozwiązywaniu układów równań. Wadą jest mała szybkość działania. Można ją zwiększyć stosując następujące techniki: podziału sieci (blokowe), usuwania nadmiarów obliczeniowych (omijanie, zamrażanie), techniki macierzy rzadkich. Metody te nadają się do symulacji fragmentów dużych układów, a szczególnie do bloków wymagających większej dokładności analizy. Nie nadają się natomiast do symulacji układów z czasem dyskretnym, takich jak układy cyfrowe VLSI.

symulacja logiczna sieci (*ang. logic simulation*) umożliwiająca ocenę opóźnień sygnałów logicznych w układzie. Wykorzystuje się tu metodę rozdzielania węzłowego

(*ang. decoupling*). Do tej grupy należą także techniki stosowane w symulacji układów analogowych takie jak: "timing", relaksacji przebiegów czasowych (*ang. waveform relaxation*), techniki "timing" kierowane zdarzeniami.

metody mieszane (*ang. mixed mode***)** Metody są połączeniem obu wymienionych wcześniej technik. Umożliwiają użytkownikowi wyspecyfikowanie bloków, które mają być analizowane jedną z powyższych technik.

W symulacji układów analogowych (z czasem ciągłym) najczęściej stosuje się metody oparte na rozwiązywaniu równań różniczkowych opisujących sieć. Opracowano metody oparte na analizie lokalnych interakcji pomiędzy elementami sieci [JCL94], które w ogóle nie rozwiązują równań różniczkowych. Wykorzystują one specyficzną reprezentację sieci. Odpowiedzi układu są wyznaczane jako ciąg prostych podstawowych operacji matematycznych.

1.3 Równania sieci

Nieliniową sieć można opisać za pomocą układu równań postaci (1.1).

$$f(x, \dot{x}, t) = 0 \tag{1.1}$$

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 (1.1) należy zdyskretyzować celem eliminacji pochodnej $\dot{x}$ za pomocą schematu różnicowego [J.O95] postaci (6.2)

$$\dot{x}_n = \gamma\ x_n - d_{xn} \tag{1.2}$$

gdzie: d_x jest wektorem przeszłości, γ współczynnikiem zależnym od rodzaju schematu różnicowego. Otrzymany w ten sposób układ nieliniowych równań algebraicznych (1.3).

$$f(x_n, \gamma\ x_n - d_{xn}, t_n) = 0 \tag{1.3}$$

Można go rozwiązać np. metodą Newtona-Raphsona (zobacz 5.2) przez linearyzację równań - rozwinięcie w szereg Taylora [J.O95] do wyrazów rzędu pierwszego. Otrzymamy wtedy układ równań liniowych.

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

$$= -\underbrace{f(x_n^{(p-1)}, \gamma\ x_n^{(p-1)} - d_{xn}, t_n)}_{f^{(p-1)}} \tag{1.4}$$

Po pogrupowaniu czynników i wprowadzeniu oznaczeń przyjmuje on postać (1.5).

$$\underbrace{\frac{df^{(p-1)}}{dx_n}}_{Y} \cdot {x_n}^{(p)} = \underbrace{-f^{(p-1)} + \frac{df^{(p-1)}}{dx_n} \cdot x_n^{(p-1)}}_{B} \tag{1.5}$$

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 4.1.3) np.:

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

Algorytm rozwiązywania równań sieci Proces rozwiązywania równań sieci można zapisać w postaci algorytmu 1.

Algorytm 1 Rozwiązywanie równań różniczkowych

1: $f(x, \dot{x}, t) = 0$ ▷ *równania różniczkowe zwyczajne opisujące system/układ (1.1)*
2: inicjalizacja wartości początkowych
3: **for** $t_m = t_{min} \dots t_{max}$ **do** ▷ pętla czasu
4: dyskretyzacja równań za pomocą schematu różnicowego
5: $f(x_m, \gamma\ x_m - d_{xm}, t_m) = 0$ ▷ *układ równań algebraicznych nieliniowych (1.3)*
6: **repeat** ▷ pętla iteracji NR
7: linearyzacja równań (algorytm Newtona-Raphsona)
8: $\underbrace{\frac{df^{(p-1)}}{dx_m}}_{Y} \cdot {x_m}^{(p)} = \underbrace{-f^{(p-1)} + \frac{df^{(p-1)}}{dx_m} \cdot x_m^{(p-1)}}_{B}$ ▷ *układ równań liniowych (1.5)*
9: iteracyjne rozwiązanie układu równań liniowych postaci ${x_m}^{(p)} = Y^{-1} \cdot B$
10: **until** ${x_m}^{(p)} - {x_m}^{(p-1)} < \epsilon$ ▷ test stopu pętli NR
11: wyprowadzenie wyników analizy w danym punkcie czasu t_m
12: **end for**
13: KONIEC

W procesie rozwiązywania można wydzielić trzy podstawowe poziomy równań:

1. poziom równań różniczkowych,
2. poziom równań zdyskretyzowanych,
3. poziom równań zlinearyzowanych.

Algorytm rozwiązywania równań sieci W zależności od sposobu współdziałania równań w procesie rozwiązywania można podać klasyfikację technik symulacji (tab. 1.1). W grupie metod bezpośrednich układ równań traktowany jest nierozłącznie. Jeżeli układ równań zapisany macierzowo ma szczególną strukturę, tzw. blokowo diagonalną z obrzeżem[1], to można go podzielić na bloki. Każdy blok opisuje wtedy pewien fragment układu.

W grupie technik relaksacyjnych równania traktowane są w sposób rozdzielony. Każde równanie rozwiązywane jest niezależnie od pozostałych, korzystając z poprzednich rozwiązań pozostałych równań. Podział ten zwany jest węzłowym, gdyż pojedyncze równanie opisuje bilans prądów w węźle. Drugim podziałem jest podział blokowy na grupy węzłów analizowanych wspólnie.

Dla tak sklasyfikowanych metod analizy czasowej można podać zalety i wady dobrze charakteryzujące poszczególne grupy (tab. 1.2).

[1] BBD - macierz blokowo diagonalna z obrzeżem

Tab. 1.1: Klasyfikacja technik symulacji opartych na rozwiązywaniu równań różniczkowych

Poz. r-ń	metody bezpośrednie		metody relaksacyjne	
	nie rozdzielone	podział blokowy	rozdzielenie węzłowe	rozdzielenie blokowe
1	układ równań różniczkowych zwyczajnych, metody dyskretyzacji ze zmiennym krokiem	układ równań różniczkowych zwyczajnych zagnieżdżonych, bezpośrednia technika przebiegów	równania różniczkowe bilansu w węzłach, całkowanie równania każdego węzła - technika relaksacji przebiegów	równania różniczkowe bilansu w węzłach, całkowanie bloków - blokowa relaksacja przebiegów
2	układ równań algebraicznych nieliniowych, metody quasi-Newtona-Raphsona	układ równań algebraicznych nieliniowych zagnieżdżonych, analiza wielopoziomowa Newtona-Raphsona	układ równań algebraicznych bilansu w węzłach, rozwiązywanie niezależne każdego węzła - techniki timing	układ równań algebraicznych nieliniowych, blokowe iteracje nieliniowe - np. GSN, techniki mieszane
3	układ równań algebraicznych liniowych, metoda LU, techniki macierzy rzadkich	układ równań algebraicznych liniowych o macierzy BBD[1] , analiza przez podział	układ równań algebraicznych liniowych, rozwiązanie metodami iteracyjnymi (np.GS)	układ równań algebraicznych liniowych o macierzy BBD[1], metody blokowe iteracyjne (np.GS)

W większości stosowanych symulatorów układów elektronicznych stosuje się bezpośrednie metody analizy czasowej (programy: SABER[Ana], SPICE[J.P96], OPTIMA[D.B95]), Dero[Pla13]), co umożliwia analizę szerokiej klasy sieci przy stosunkowo małej szybkości. Do analizy układów złożonych z tranzystorów MOS stosuje się specjalizowane programy wykorzystujące techniki relaksacyjne, w tym także kierowane zdarzeniami: ELDO[Ana97], iSPLICE[R.A89a], MOTIS [D.O77], RELAX[J.W83], OPTIMA-R[P.P99]). Symulatory te są o rząd wielkości szybsze, jednak ich stosowalność jest ograniczona. Trudności z analizą szerokiej klasy sieci związane są ze słabą zbieżnością technik iteracyjnych, co uniemożliwia otrzymanie dokładnych rozwiązań. Aby temu chociaż częściowo zaradzić stosuje się techniki dodatkowe - omówione, co jednak znacznie komplikuje działanie symulatora i obniża jego wydajność.

Programy symulacji analogowych układów elektronicznych Programy symulacji analogowych układów elektronicznych są jedynymi z ważniejszych narzędzi wykorzysty-

Tab. 1.2: Zalety i wady technik symulacji opartych na rozwiązywaniu równań różniczkowych

Zalety	Wady
metody bezpośrednie	
• szeroka klasa analizowanych sieci • duża dokładność - ograniczona dokładnością arytmetyki komputera[J.O95]	• układ analizowany jest z krokiem czasowym wynikającym z najszybciej zmieniającego się sygnału w układzie • duży nakład obliczeń na rozwiązanie układu równań całego układu (nawet przy zastosowaniu techniki macierzy rzadkich) rzędu $O(\ n^{1.1})$ • dla dłuższych czasów analiz występuje efekt kumulacji lokalnego błędu obcięcia, co prowadzi do pogorszenia dokładności wyników analizy
metody relaksacyjne	
• krok czasowy analizy indywidualny dla każdego węzła sieci • mały nakład obliczeniowy (rozdzielenie równań) • możliwość wykorzystania techniki usypiania (*ang. latency*) bloków/węzłów celem redukcji nakładów obliczeniowych	• ograniczona klasa analizowanych sieci do układów z elementami o jednokierunkowym przepływie sygnału (unilateralnych) lub słabymi sprzężeniami pomiędzy węzłami • nakład obliczeń uwarunkowany jest rodzajem układu i zależy od szybkości zbieżności technik iteracyjnych rozwiązywania równań

wanych do symulacji nie tylko układów elektronicznych, lecz po zastosowaniu metod omówionych w rozdziale 12, także do symulacji mikrosystemów. Najbardziej nakładochłonną analizą z punktu widzenia wykorzystania czasu pracy komputera jest analiza czasowa [J.O94]. Podstawowymi ograniczeniami są tu:

- krok analizy układu zdeterminowany przez najszybciej zmieniający się sygnał w układzie,
- konieczność rozwiązywania dużych układów równań.

Drastyczny wzrost liczby elementów (zarówno tranzystorów MOS jak i bipolarnych) spowodował, że techniki klasyczne stały się mało efektywne ze względu na wymienione wyżej ograniczenia. Aby temu zaradzić opracowano szereg technik przyśpieszających analizę [J.O94], do których należą:

- usuwanie nadmiarów obliczeń przez zastosowanie: metody macierzy rzadkich [Agu90, J.O94], metody zamrażania bloków [RA89b, T.S88], metody omijania [J.O95, M.N91],
- dopuszczanie obliczeń zgrubnych [R.K94],
- stosowanie uproszczeń w modelach elementów i podukładów [J.D99, B.R75, D.O77, Y.H89] - modele odcinkowo-liniowe [R.K94, J.D99], modele ze stabelaryzowanymi wartościami wyjść (*ang. lookup table models*) [J.D99, B.R75, D.O77],
- wykorzystywanie informacji o kształcie sygnałów sterujących [R.A96].

Szybko okazało się, że dla potrzeb analizy dużych sieci wymienione metody stały się niewystarczające. W latach 80-tych opracowano nową klasę algorytmów zwanych relaksacyjnymi [ARN83, P.S93] używających iteracyjnych technik rozwiązywania równań różniczkowych [C.T95, Owe94], zamiast metod bezpośrednich wykorzystujących zmodyfikowaną metodę potencjałów węzłowych i techniki macierzy rzadkich [J.O94, Lin81, J.O95]. Metody te są o rząd wielkości szybsze od metod tradycyjnych. Są one dobrze przystosowane do analizy układów z tranzystorami MOS, w których występuje jednokierunkowy przepływ sygnału i słabe sprzężenia pomiędzy węzłami. Dzięki temu możliwe jest rozdzielenie węzłowe równań i rozwiązanie ich metodami iteracyjnymi [C.T95]. Zapisując macierzowo równania dla tego typu układów otrzymujemy macierz admitancyjną z silną dominacją głównej przekątnej, co zapewnia szybką zbieżność metod iteracyjnych [M.D82, C.T95]. Nakład obliczeń na rozwiązanie jest rzędu $O(n^2)$ działań (pod warunkiem, że zbieżność zostanie osiągnięta w $k < n$ iteracjach, gdzie n jest liczbą równań) zamiast $O(n^3)$ dla metod bezpośrednich [J.O94].

Analiza relaksacyjna układów złożonych z elementów o dwukierunkowym przepływie sygnału (np. z tranzystorów bipolarnych) nastręcza wiele trudności, ze względu na słabą zbieżność metod iteracyjnego rozwiązywania równań. Pomimo tych trudności metody takie zostały opracowane dla układów cyfrowych z tranzystorami bipolarnych [G.M85, RW91] [M.N91, RA89b]. Wykorzystują one techniki podziału układu na bloki analizowane metodami klasycznymi (macierze pełne - *ang. direct matrix*). Pomiędzy blokami stosuje się analizę relaksacyjną. Podstawowym problemem jest tu odpowiedni podział układu na bloki. Bloki powinny być jak najmniejsze aby nie zmniejszać szybkości analizy, ale z drugiej strony muszą być zapewnione warunki zbieżności technik iteracyjnych. W literaturze opisywane są następujące metody podziału układu:

- statyczny wykonywany przez użytkownika przed analizą [P.O90, Ana97],
- wykorzystujący wyniki analizy punktu pracy [M.N91, D$^+$87],
- na podstawie topologii układu [M.N88] (np. podział hierarchiczny wykonywany na poziomie: bloków, przerzutników, bramek, tranzystorów [P.S88]),
- na podstawie znajomości modeli elementów (ich właściwości) [G.M85, M.N91],
- dynamiczny wykonywany w trakcie analizy [G.M85, M.N93].

Wiele symulatorów układów analogowych posiada możliwość tworzenia modeli behawioralnych (definiowanych przez użytkownika) opisywanych dowolnymi wyrażeniami matematycznymi. W literaturze nie jest znany przykład zastosowania technik relaksacyjnych jako jedynej analizy czasowej w tego typu symulatorach ze względu na występujące trudności, do których należą:

- szeroka klasa analizowanych sieci nieakceptowana przez metodę,
- nieprzewidywalny *a priori* opis układu,
- brak pewności co do poprawności wyników analizy, ze względu na popełniane błędy rozwiązania równań metodami iteracyjnymi.

Rozdział 2

Metody numeryczne

Obliczenia numeryczne przeprowadzane za pomocą komputera wymagają znajomości zagadnień związanych z reprezentacją binarnych liczb zmiennoprzecinkowych (rzeczywistych) w systemach komputerowych. Należy zdawać sobie sprawę, że nie jest możliwe przedstawienie nieskończonego zbioru liczb rzeczywistych w postaci binarnych liczb zmiennoprzecinkowym dla skończonej liczby bitów. W systemach komputerowych zostały ustalone zasady przyporządkowywania liczb rzeczywistych odpowiedniej kombinacji bitów (w systemie dwójkowym) zgodnie z przyjętą reprezentacją. Obecnie w systemach komputerowych obliczenia najczęściej wykonywane są zgodnie z normą IEEE-754 [IEEa] - standard zapisu i działań arytmetycznych na liczbach zmiennoprzecinkowych stworzony w 1985 roku. Obecnie obowiązującą normą jest norma IEEE-754-2008 [IEEb] opublikowana w 2008 roku, która zawiera poprzednie normy IEEE-754-1987 i IEEE-854-1987. Jest ona zgodna z normą ISO/IEC/IEEE 60559:2011 [ISO].

W przypadku tworzenia oprogramowania wymagana jest wiedza z zakresu języków programowania oraz zagadnień reprezentacji liczb w poszczególnych językach. Implementacja niektórych języków programowania zależna jest od sprzętu. Istnieje także szereg wersji języków programowania, które w różny sposób implementują standardy zapisu i reprezentacji liczb zmiennoprzecinkowych.

Podstawowymi ważnymi zagadnieniami szczegółowymi omówionymi w kolejnych rozdziałach są:

- własności arytmetyki zmiennoprzecinkowej,
- zakresy reprezentowanych liczb,
- błędy reprezentacji liczb,
- błędy obliczeniowe,
- zagadnienia poprawności numerycznej algorytmów (nie omawiane w pracy),
- znajomość normy zapisu liczb IEEE-754 (rozdział 2.4),
- sposoby rozłożenia danych w pamięci komputera w przypadku tablic (nie omawiane w pracy).

W zależności od wykorzystanego sprzętu komputerowego oraz języka programowania wyniki obliczeń mogą się różnić. Stanowi to duży problem związany z implementacją i wykorzystaniem systemów CAD/CAM.

Omawiany w rozdziale binarny zapis liczb zmiennoprzecinkowych po raz pierwszy zastosował Konrad Zuse w mechanicznym komputerze Z1.

2.1 Reprezentacja liczb w komputerze

Wartość liczby zmiennoprzecinkowej x można zapisać w postaci binarnej (2.1).

$$x = S \cdot M \cdot B^E \tag{2.1}$$

gdzie:

S (*ang. sign*) - znak liczby, 1 lub -1,

M (*ang. mantissa*) - znormalizowana mantysa (liczba ułamkowa),

B (*ang. base*) - podstawa systemu liczbowego (2 dla systemów komputerowych),

E (*ang. exponent*) - wykładnik (liczba całkowita).

Istnieje kilka konwencji, według których normalizowana jest mantysa M. Często stosuje się normalizację $M \in [0.1, 1)$ (przedział prawostronnie otwarty). W systemach komputerowych mantysa jest znormalizowana i należy do przedziału $[1, BIAS)$, gdzie $BIAS = 2$ zgodnie z najczęściej stosowaną normą IEEE-754 [IEEa]. Nazwa reprezentacji pochodzi stąd, że jeśli M jest stałe, a E zmienia się, wówczas przesunięciu ulega przecinek.

Liczba cyfr dla mantysy M oraz wykładnika E jest z góry ustalona. Oznacza to, że dana liczba jest reprezentowana z pewną skończoną dokładnością i należy do skończonego zbioru wartości.

Powstaje problem co zrobić z liczbami, które nie mogą być reprezentowane. Możliwe są dwa podejścia:

- pominięcie bitów, które nie mogą być reprezentowane, co odpowiada zaokrągleniu w dół,
- zaokrąglenie wartości do najbliższej reprezentowalnej wartości w góre lub w dół - ten sposób jest stosowany w standardzie IEEE-754 [IEEa],

2.1.1 Zakres reprezentowanych liczb

Załóżmy, że m oznacza liczbę cyfr przeznaczonych na mantysę, natomiast $n+1$ to liczba cyfr przeznaczonych na wykładnik. n cyfr przeznaczona jest do reprezentowania wartości i 1 cyfra do reprezentowania znaku wykładnika. Przyjmuje się, że jedna dodatkowa pozycja (najstarsza) zarezerwowana jest dla zapisu znaku całej liczby. Dla tak zdefiniowanych zakresów reprezentacji wykładnika E i mantysy M, wartości maksymalne i minimalne określone są następująco:

- Wykładnik E:

  $E_{\min} = -B^n + 1$

  $E_{\max} = B^n - 1$

- Mantysa M:

  $M_{\min} = 1$

  $M_{\max} = B - B^{-(m-1)}$

Z powyższych zależności wynikają wartości minimalne i maksymalne reprezentowanych liczb rzeczywistych:

$$x_{\min} = M_{\min} \cdot B^{E_{\min}} = 1 \cdot B^{E_{min}} \tag{2.2}$$

$$x_{\max} = M_{\max} \cdot B^{E_{\max}} = \left(B - B^{-(m-1)}\right) \cdot B^{E_{max}} < B^{E_{max}+1}. \tag{2.3}$$

Zakres reprezentacji liczb zmiennoprzecinkowych w danym zapisie wynosi:

$$[-x_{\max}, -x_{\min}] \cup \{0\} \cup [x_{\min}, x_{\max}] \tag{2.4}$$

Zero jest wartością specjalną, która nie może zostać bezpośrednio reprezentowana w tym formacie.

2.1.2 Błędy reprezentacji liczb

Błąd względny reprezentacji dany jest zależnością (2.5).

$$\epsilon = \frac{1}{B^{m-1}} \tag{2.5}$$

Wartość błędu odpowiada wartości dla najmniej znaczącej cyfry mantysy. Błędów bezwzględnych nie podaje się. W przypadku operacji matematycznych mogą zostać wygenerowane liczby, które traktowane są jako błędy:

- liczba o wartości $|x| < B^{E_{\min}}$ traktowana jest jako niedomiar (*ang. underflow*),
- liczba o wartości $|x| > M_{\max} \cdot B^{E_{\max}}$ traktowana jest jako nadmiar wykładniczy (*ang. overflow*).

2.1.3 Przykład reprezentacji

Przyjmijmy, że $B = 10$, liczba cyfr dziesiętnych przeznaczonych na mantysę wynosi 4, natomiast na wykładnik 2. Chcemy zapisać wartość 60,12546.

Liczba większa od 10 Liczba $30,12546$ odpowiada $M = 30,12546$, $E = 0$. Ponieważ mantysa nie należy do przedziału $[1, 10)$, to należy przesunąć przecinek aby ją znormalizować i zwiększyć wykładnik.

$$M = 3,012546, E = 1$$

Możliwe są dwa przypadki uzyskania wyniku

- odcięcie: $3,012$,
- zaokrąglenie: $3,013$.

W wyniku zaokrąglenia otrzymamy: $3,013o101 = 3,013E1$.

Liczba mniejsza od 1 Dla liczby mniejszej od 1: $0,0000125$.

$$M = 0,0000125, E = 0$$

Po normalizacji

$$M = 1,25, E = -5$$

Liczba cyfr znaczących jest mniejsza od dostępnej, więc nie jest potrzebne zaokrąglanie.

$$1,250o10 - 5 = 1,25E - 5$$

2.2 Własności arytmetyki zmiennoprzecinkowej

Arytmetyka zmiennoprzecinkowa nie jest łączna i rozdzielna. Oznacza to, że dla x, y i z mogą zachodzić różności:

$$(x + y) + z \neq x + (y + z) \tag{2.6}$$
$$(x \cdot y) \cdot z \neq x \cdot (y \cdot z) \tag{2.7}$$
$$x \cdot (y + z) \neq (x \cdot y) + (x \cdot z) \tag{2.8}$$

W związki z tym kolejność wykonywanych operacji wpływa na wynik końcowy. Jest to spowodowane tym, że podczas wykonywania operacji zmiennoprzecinkowych mogą występować:

- zaokrąglenia,
- nieprawidłowe operacje,
- przepełnienie,
- niedomiar.

2.3 Podstawowe operacje numeryczne

Załóżmy, że operacje numeryczne zostaną przeprowadzone na dwóch liczbach x_1 i x_2, które można zapisać w poniższej postaci.

$$x_1 = M_1 \cdot B^{E_1} \tag{2.9}$$
$$x_2 = M_2 \cdot B^{E_2} \tag{2.10}$$

2.3.1 Dodawanie i odejmowanie

Załóżmy, że wykonujemy operacje dodawania lub odejmowania dwóch liczb zmiennoprzecinkowych x_1 i x_2, które spełniają zależność $x_1 \geq x_2$. Założenia to można spełnić dla dowolnych liczb manipulując ich kolejnością, znakiem wyniku oraz rodzajem wykonywanej operacji, zgodnie z poniższym schematem działań.

$$\begin{array}{llll} (+x_1)+(+x_2), & (+x_1)-(-x_2), & (+x_2)+(+x_1), & (+x_2)-(-x_1) \quad \rightarrow \quad +(x_1+x_2) \\ (+x_1)+(-x_2), & (+x_1)-(+x_2), & (-x_2)+(+x_1), & (-x_2)-(-x_1) \quad \rightarrow \quad +(x_1-x_2) \\ (-x_1)+(-x_2), & (-x_1)-(+x_2), & (-x_2)+(-x_1), & (-x_2)-(+x_1) \quad \rightarrow \quad -(x_1+x_2) \\ (-x_1)+(+x_2), & (-x_1)-(-x_2), & (+x_2)+(-x_1), & (+x_2)-(+x_1) \quad \rightarrow \quad -(x_1-x_2) \end{array} \tag{2.11}$$

Operacja, która zostanie wykonana zapisana jest wzorem (2.12).

$$x_1 \pm x_2 = (M_1 \pm M_2 \cdot B^{E_2-E_1}) \cdot B^{E_1} \tag{2.12}$$

W przypadku, gdy liczby mają różne wykładniki, to podczas dodawania mantysa liczby o mniejszym wykładniku M_2 musi zostać zdenormalizowana przez przemnożenie M_2 przez czynnik $B^{E_2-E_1}$.

W szczególnym przypadku, jeśli $E_1 - E_2$ jest większe niż liczba cyfr mantysy m, to po denormalizacji mantysa będzie miała wartość 0, a liczba o mniejszym wykładniku nie wpłynie na wynik dodawania bądź odejmowania.

Odejmowanie liczb zmiennoprzecinkowych o takim samym wykładniku E i niewiele różniącej się mantysie M powoduje, że wynikowa mantysa jest znacznie zdenormalizowana.

Ponowna normalizacja mantysy M powoduje wprowadzenie wielu nieznaczących zer na końcu mantysy. Obliczenia muszą być tak zaprojektowane, aby nie wprowadzać nieznaczących cyfr na końcu mantysu.

2.3.2 Mnożenie i dzielenie

Operacje mnożenia i dzielenia można zapisać w postaci (2.13) i (2.14).

$$x_1 \cdot x_2 = (S_1 \cdot S_2) \cdot (M_1 \cdot M_2) \cdot B^{E_1+E_2} \tag{2.13}$$

$$\frac{x_1}{x_2} = (S_1 \cdot S_2) \cdot (\frac{M_1}{M_2}) \cdot B^{E_1-E_2} \tag{2.14}$$

2.3.3 Błędy operacji elementarnych

W celu wykonania analizy błędów elementarnych operacji liczbę zmiennoprzecinkową można przedstawić jako wartość dokładną zaburzoną pewnym błędem reprezentacji ε:

$$\bar{x} = x + x * \varepsilon_x == x(1+\varepsilon_x)$$

Błąd względny poszczególnych operacji elementarnych wykonywanych na liczbach a i b

$$\bar{a} = a(1+\varepsilon_a) \tag{2.15}$$

$$\bar{b} = b(1+\varepsilon_b) \tag{2.16}$$

Operacja	Błąd
dodawanie/odejmowanie	$\varepsilon_{a\pm b} = \frac{a\varepsilon_a \pm b\varepsilon_b}{a \pm b} + \varepsilon_\pm$
mnożenie	$\varepsilon_{a\cdot b} = \varepsilon_a + \varepsilon_b + \varepsilon_.$
dzielenie	$\varepsilon_{a/b} = \varepsilon_a - \varepsilon_b + \varepsilon_/$

można oszacować następująco:

gdzie: $\varepsilon_\pm$, $\varepsilon_.$ i $\varepsilon_/$ są błędami wprowadzanymi przez poszczególne operacje arytmetyczne.

Rozbijając każde wyrażenie arytmetyczne na operacje elementarne można za pomocą tych zależności oszacować powstałe błędy. Istnieją jednak lepsze i szybsze metody modelowania błędów [FZ09, V.L05] - zobacz także rozdział 3.2.3.

2.4 Implementacje sprzętowe IEEE-754

W implementacjach sprzętowych liczby zmiennoprzecinkowe wyraża się liczbami dwójkowymi ($B = 2$), co ma następujące zalety:

- Mantysa należy do przedziału $[1, 2)$, jest więc postaci $1.xxxxx...$. (x - bit o dowolnej wartości). Część całkowita jest znana, $i = 1$ (zawsze), więc nie jest zapamiętywana, co daje dodatkowy bit na część ułamkową.
- Znak liczby jest zapamiętywany na jednym bicie, więc . otrzymanie modułu i wartości przeciwnej wymaga, odpowiednio, wyzerowania tego bitu (logiczna operacja AND), lub zmiany na wartość przeciwną (logiczna operacja XOR).

2.4.1 Reprezentacja zmiennoprzecinkowa IEEE-754-2008

W celu ujednolicenia zasad operacji na liczbach zmiennoprzecinkowych na różnych platformach sprzętowych, opracowano standard IEEE-754, w oparciu o który realizuje się obecnie wszystkie implementacje sprzętowe liczb zmiennoprzecinkowych. W standardzie zdefiniowano następujące klasy liczb:

- połowie precyzji (*ang. half precision* lub binary16),
- pojedynczej precyzji (*ang. single precision* lub binary32),
- podwójnej precyzji (*ang. double precision* lub binary64),
- poczwórnej precyzji (*ang. quadruple precision* lub binary128).

W praktyce spotykane są inne sposoby zapisu, różniące się jedynie liczbą bitów przeznaczoną na poszczególne pola. Np. koprocesor w procesorach x86, oprócz typów standardowych, wspiera liczby 10-bajtowe. Liczby zgodne ze standardem IEEE-754 mają dokładnie określoną semantykę, jak na przykład: dokładność operacji elementarnych, kierunki zaokrągleń, czy obsługa sytuacji wyjątkowych - są to cechy bardzo pożądane w zastosowaniach naukowych i inżynieryjnych, a również ułatwiają przenoszenie kodu programu na inny sprzęt.

Nazwa	Format	Znak [bity]	Wykładnik [bity]	Mantysa [bity]	Szerokość słowa [bity]	Typy w językach programowania
binary16	IEEE-754 half	1	5	10	16	float (C), single (Pascal), real*4 (Fortran)
binary32	IEEE-754 single	1	8	23	32	float (C), single (Pascal), real*4 (Fortran)
binary64	IEEE-754 double	1	11	52	64	double (C), real lub double (Pascal), real*8 (Fortran)
binary128	IEEE-754 quadruple	1	15	112	128	long double/__float128 (C/C++), real*16 (Fortran)
	IEEE-754 extended/x87	1	15	64	80	long double (C99), extended (Pascal)

Tab. 2.1: Formaty reprezentacji zmiennoprzecinkowej IEEE-754.

Przesunięcie wykładnika

Wykładnik będący liczbą całkowitą jest zapisywany w kodzie spolaryzowanym, co można interpretować jako wartość przesuniętą o pewną stałą (ang. biased exponent). Właściwą wartość wykładnika uzyskuje się odejmując od zakodowanego wykładnika wartość przesunięcia (ang. bias). Wartość liczby zmiennoprzecinkowej oblicza się ze wzoru:

$$x = (-1)^S \cdot M \cdot 2^{E-\text{bias}}$$

gdzie S to bit znaku - liczba jest ujemna, gdy bit znaku jest równy 1, w przeciwnej sytuacji ma on wartość 0.

Typowe wartości przesunięcia dla koprocesora x87 (występującego w procesorach x86) wynoszą:

$127(7FH)$ w formacie 32-bitowym

$1023(3FFH)$ w formacie 64-bitowym

$16383(3FFFH)$ w formacie 80-bitowym

Wartości specjalne

Oprócz zwykłych liczb zdefiniowano następujące wartości specjalne:

NaN - nie-liczba (*ang. Not-a-Number*), to symbol, który nie reprezentuje wartości liczbowej, powstały zazwyczaj w wyniku niedozwolonej operacji (np. pierwiastkowanie liczby ujemnej)

sNaN - sygnalizujące NaN (*ang. signalling NaN*) - rozróżnienie wprowadzone w procesorach z rodziny x86; dla większości operacji wykonanie liczba operacja sNaN spowoduje zgłoszenie wyjątku.

qNaN - ciche *NaN* (*ang. quiet NaN*) - przekazanie tej wartości jako argumentu operacji nie powoduje zgłoszenia wyjątku; w operacjach SSE można ustalić, że liczba operacja $qNaN- > 0$.

Zero - rozróżnia się $+0,0$ i $-0,0$.

Nieskończoność - jest wynikiem operacji w przypadku wystąpienia nadmiaru

(przepełnienia) , przy dzieleniu przez 0, itp.; może być dodatnia lub ujemna. Liczba nieznormalizowana - pojawia się, gdy występuje niedomiar (*ang. underflow*), ale wynik operacji jeszcze można zapisać denormalizując mantysę (w takim przypadku mantysa reprezentuje liczbę w postaci $0, xxx...xxx$, a nie $1, xxx...xxxx$).

Standard definiuje również liczby podwójnej precyzji, których zapis składa się z 64 bitów, przy czym na wykładnik przypada 11 bitów (BIAS=1023), a na mantysę 52 bity. Liczby podwójnej precyzji reprezentują około 16 dziesiętnych miejsc znaczących, a ich zakres stosowalności rozciąga się od około $2.2 \cdot 10^{308}$ do około $1.8 \cdot 10^{308}$.

Istnieją też inne formaty liczb zmiennoprzecinkowych, jak np 10-bajtowe (obsługiwane sprzętowo przez popularne procesory kompatybilne z x86), 16-bajtowe (procesory przeznaczone na rynek superkomputerów), jak też i formaty o mniejszej precyzji, używane m.in. przez procesory graficzne.

Wartość specjalna	Bit znaku	Bity wykładnika	Bity mantysy	Uwagi
NaN	x	111..111	$xxx...xxx$	wszystkie bity wykładnika są równe 1, natomiast mantysa ma niezerową wartość
QNaN	x	111..111	$1xxx...xxx$	uwagi jak dla NaN, ale pierwszy bit mantysy zawsze równy 1
SNaN	x	111..111	$0xxx...xxx$	uwagi jak dla NaN, ale pierwszy bit mantysy zawsze równy 0
± Zero	x	000..000	0000...000	wszystkie bity mantysy i wykładnika równe 0, bit znaku decyduje o znaku wartości
± Nieskończoność	x	111..111	0000...000	wszystkie bity mantysy równe 0, wszystkie bity wykładnika równe 1, bit znaku decyduje o znaku wartości
Nieznormalizowana	x	000..000	$xxx...xxx$	mantysa różna od 0, wszystkie bity wykładnika równe 0, bit znaku decyduje o znaku wartości

Tab. 2.2: Wartości specjalne reprezentacji zmiennoprzecinkowej IEEE-754.

2.5 Uwarunkowanie numeryczne algorytmu

Wskaźnik uwarunkowania (WU) określa wpływ błędu reprezentacji danych wejściowych (liczb zmiennoprzecinkowych) danego problemu na błąd danych wyjściowych (błąd wyniku).

WU definiuje się jako maksymalny stosunek błędu względnego rozwiązania do błędu względnego danych.

Jeżeli WU jest mały, to mówimy o dobrym uwarunkowaniu. Jeżeli WU jest duży to mówimy o złym uwarunkowaniu. W przypadku obliczeń w arytmetyce zmiennoprzecinkowej o ograniczonej reprezentacji liczb, zadania źle uwarunkowane nie nadają się do numerycznego rozwiązywania, gdyż błąd wynikający z numerycznej reprezentacji liczb powoduje relatywnie duży błąd danych wyjściowych.

Wskaźnik uwarunkowania jest cechą problemu i nie zależy od numerycznych właściwości algorytmów. Wskaźnik uwarunkowania stanowi informację o błędzie przeniesionym z danych.

W przypadku wykonywania obliczeń w arytmetyce zmiennoprzecinkowej występują dodatkowo błędy zaokrągleń wprowadzone przez algorytm (związane z samym algoryt-

mem). Nie są one powiązane i nie mają wpływu na wskaźnik uwarunkowania problemu.

W praktyce stosowane są wskaźniki uwarunkowania do:

- algorytmów,
- zadań,
- macierzy (rozdział 3.2.3).

Rozdział 3

Podstawowe pojęcia matematyczne

Ze względu na specyfikę dziedziny wymagana jest dobra znajomość matematyki, a w szczególności algebry liniowej oraz analizy matematycznej. W pracy używanych jest szereg pojęć, które wykorzystywane są w kolejnych rozdziałach. Podane tutaj zagadnienia można znaleźć w literaturze [FZ09, Wiki, Wikh, Wikc].

3.1 Normy wektorów

W praktycznych zastosowaniach wykorzystywane są normy wektorów. W przestrzeni R^n, której elementami są wektory:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \tag{3.1}$$

można zdefiniować następujące normy wektorów:

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \ldots + |x_n| \tag{3.2}$$

$$\|\mathbf{x}\|_2 = \sqrt{{x_1}^2 + {x_n}^2 + \ldots + {x_n}^2} \tag{3.3}$$

$$\|\mathbf{x}\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\} \tag{3.4}$$

Należy zauważyć, że dla dowolnego $\mathbf{x} \in R^n$ obowiązuje zależność (3.5).

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n}\|\mathbf{x}\|_2 \leq n\|\mathbf{x}\|_\infty \tag{3.5}$$

Więcej informacji można znaleźć w literaturze [Wikh].

3.2 Macierze

Macierze wykorzystywane są w algorytmach rozwiązywania układów równań liniowych do przechowywania mnożników $a_{1,1} \ldots a_{n,n}$ niewiadomych $x_1 \ldots x_n$, gdzie n jest liczbą równań i niewiadomych. W przypadku rozwiązywania układów równań, duże znaczenie mają odpowiednie normy macierzy oraz zagadnienia obliczania wektorów własnych macierzy, co zostanie omówione w kolejnych rozdziałach.

3.2.1 Normy macierzy

Macierz $A_{m\times n}$ [Wikd] można traktować jako operator liniowy przekształcający przestrzeń R^n w R^m [Wikc, Wikf]. Można określić tzw. *indukowaną* normę macierzy daną (3.6).

$$\|\mathbf{A}\|_{pq} = \max_{x\in R^n, x\neq 0} \frac{\|\mathbf{Ax}\|_q}{\|\mathbf{x}\|_p} \tag{3.6}$$

gdzie p i q są odpowiednimi normami wektorów w przestrzeniach R^n i R^m. Podstawowe normy dane są zależnościami:

$$\|\mathbf{A}\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| \tag{3.7}$$

$$\|\mathbf{A}\|_2 = \sqrt{\max\{\lambda(\mathbf{A})\}} \tag{3.8}$$

$$\|\mathbf{A}\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}| \tag{3.9}$$

$$\|\mathbf{A}\|_{1\infty} = \max_{i,j} |a_{ij}| \tag{3.10}$$

Należy zwrócić uwagę, że normy:

- $\|\mathbf{A}\|_1$ - maksymalna suma modułów w kolumnie,
- $\|\mathbf{A}\|_\infty$ - maksymalna suma modułów w wierszu,
- $\|\mathbf{A}\|_{1\infty}$ - maksymalna moduł elementu,

można łatwo obliczy, natomiast wartość normy $\|\mathbf{A}\|_2$ - największa wartość własna macierzy $(\mathbf{A}^T\mathbf{A})^{1/2}$ jest trudna do obliczenia. Norma $\|\mathbf{A}\|_2$ ma jednak wiele zastosowań. Dlatego tez często używa się tzw. *normy euklidesowej (normy Shura[Wikg], normy Fobeniusza[Wikf])*:

$$\|\mathbf{A}\|_E = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} \tag{3.11}$$

Norma ta nie jest indukowana przez żadną normę wektorów, spełnia jednak z normą $\|\cdot\|_2$ warunek *zgodności*:

$$\|\mathbf{Ax}\|_2 \leq \|\mathbf{A}\|_E \|\mathbf{x}\|_2 \tag{3.12}$$

dla każdego $x \in R^n$. Norma euklidesowa jest zatem na ogół *zbyt duża*. Ogólnie dla dowolnej macierzy $\mathbf{A}$ zachodzi zależność (3.13).

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_E \leq \sqrt{n}\|\mathbf{A}\|_2 \tag{3.13}$$

Dla dowolnej macierzy kwadratowej zgodnej z normą wektorów obowiązuje zależność:

$$|\lambda_i| \leq \|\mathbf{A}\| \tag{3.14}$$

gdzie λ_i jest dowolną wartością własną macierzy $\mathbf{A}$.

Norma euklidesowa przestrzeni indukuje normę spektralną przekształcenia. Ponieważ macierz A^hA jest hermitowska, istnieje diagonalizująca ją transformacja bazy. Po przekształceniach otrzymamy zależność na normę spektralną macierzy (3.15).

$$\|\mathbf{A}\|_S = \sqrt{\max_{i=1}^{m} \lambda_i(A^hA)} \tag{3.15}$$

Norma spektralna A jest pierwiastkiem z największej wartości własnej macierzy $A^h A$. Dla symetrycznych A jest równa promieniowi spektralnemu $\rho(A)$ macierzy A.

$$\|\mathbf{A}\|_S = \rho(A) \tag{3.16}$$

Dla niesymetrycznych A promień spektralny $\rho(A)$ macierzy A dany jest nierównością (3.17).

$$\|\mathbf{A}\|_S \geq \rho(A) \tag{3.17}$$

3.2.2 Podział macierzy

W pracy używany jest podział macierzy A postaci (3.18), który jest wykorzystywany w kolejnych rozdziałach[Wike].

$$A = D + U + L \tag{3.18}$$

gdzie:

D macierz elementów diagonalnych,

L macierz elementów spod przekątnej,

U macierz elementów znad przekątnej macierzy A.

Na rys.3.1 przedstawiono podział macierzy w postaci graficznej.

$$\underbrace{\begin{bmatrix} a_{1,1} & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & a_{n,n} \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & a_{1,2} & \dots & \dots \\ 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & a_{n,n} \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{U} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 \\ \dots & \dots & 0 & 0 \\ a_{n,1} & \dots & a_{n,n-1} & 0 \end{bmatrix}}_{L}$$

Rys. 3.1: Podział macierzy A.

3.2.3 Wskaźnik uwarunkowania macierzy

Wskaźnik uwarunkowania κ macierzy A w równaniu (3.19)

$$Ax = b \tag{3.19}$$

jest charakterystyczną własnością macierzy opisującą wpływ zmiany normy macierzy A na normę rozwiązania x.

Wskaźnik uwarunkowania macierzy definiuje się bardziej precyzyjnie jako maksymalny stosunek błędu względnego wektora rozwiązania x, do błędu względnego b. Jeżeli założymy, że ϵ jest błędem b, to równanie przyjmie postać:

$$Ax = b + \epsilon \tag{3.20}$$

i błąd ϵ będzie propagował się na rozwiązanie:

$$x = A^{-1}(b + \epsilon) \tag{3.21}$$

Stąd $\kappa(A)$ jako stosunek relatywnego błędu rozwiązania do relatywnego błędu w b wynosi:

$$\kappa(A) = \frac{\frac{\|A^{-1}\epsilon\|}{\|A^{-1}b\|}}{\frac{\|\epsilon\|}{\|b\|}} = \frac{\|A^{-1}\epsilon\|}{\|\epsilon\|} \cdot \frac{\|b\|}{\|A^{-1}b\|} = \|A^{-1}\| \cdot \|A\| \tag{3.22}$$

Maksymalna wartość (dla niezerowych b i ϵ) będzie iloczynem dwóch norm (definiowanych w różny sposób, np. często jako normę traktuje się maksymalną sumę wartości bezwzględnych wierszy):

$$\kappa(A) = \|A^{-1}\| \cdot \|A\| \tag{3.23}$$

Definicja ta jest taka sama dla każdej zwartej normy. Liczba $\kappa(A)$ oznacza wskaźnik uwarunkowania macierzy. κ pozwala na oszacowanie, z jaką (maksymalnie) dokładnością (do ilu miejsc po przecinku) można podać wynik. Dokładność jest zależna od iloczynu dokładności obliczeń zmiennoprzecinkowych ϵ (zobacz rozdział 2.1.2 i 2.3.3) i wskaźnika uwarunkowania κ.

Przykład

Załóżmy, że mamy macierz $\mathbf{A}$ oraz $\mathbf{A^{-1}}$.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{3.24}$$

$$\mathbf{A^{-1}} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix} \tag{3.25}$$

Stosując normę (3.9) możemy obliczyć wskaźnik uwarunkowania

$$\kappa(A) = \|A^{-1}\| \cdot \|A\| = (2+4) \cdot (2+1.5) = 21.$$

Załóżmy, że obliczenia będziemy przeprowadzać na maszynie, która przechowuje liczby rzeczywiste używając 24 bitowej mantysy (zobacz rozdział 2.1.2), wtedy błąd reprezentacji liczb wynosi

$$\epsilon = 2^{1-24} = 0.119209 \cdot 10^{-6}.$$

Po pomnożeniu tych wartości możemy oszacować do ilu miejsc po przecinku otrzymany wynik będzie istotny na podstawie poniższej równości:

$$5 \cdot 10^{-m} = 2.5 \cdot 10^{-6}.$$

Ponieważ wartość 2.5 mieści się w zakresie $[1, 10]$ to $m = 6$ (zobacz rozdział 2.1.3). Możemy wnioskować, że w tym przypadku dokładność wyniesie 6 miejsc po przecinku.

Inne przykłady wpływu błędów reprezentacji i współczynnika uwarunkowania macierzy na wyniki można znaleźć w literaturze, np. w [Wiki, J.O95].

Rozdział 4

Metody rozwiązywania układów równań liniowych

W rozdziale omówiono podstawowe metody rozwiązywania układów równań liniowych, które podzielono na dwie grupy:

- metody macierzowe, w tym metody wykorzystujące rzadkość macierzy,
- metody iteracyjne.

W praktyce stosuje się obie grupy metod. Metody macierzowe należą do grupy metod dających dokładne rozwiązania (o ile równania są dobrze uwarunkowane 3.2.3). Wymagają one dużych nakładów obliczeniowych. W przypadku wykorzystania techniki macierzy rzadkich (zobacz rozdział 4.1.3), ich efektywność wzrasta wraz z liczbą równań i mniejszym wypełnieniem macierzy. Efektywność metod silnie zależy od struktury macierzy. Jeżeli struktura jest optymalna, to w trakcie rozkładu macierzy (np. rozkładu LU) pojawia się niewiele wypełnień i metoda jest bardzo efektywna.

Druga grupa wykorzystuje metody iteracyjnego rozwiązywania równań rozdzielonych (metoda Gaussa-Seidela). Metody te mają ograniczony zakres zastosowań do pewnej klasy równań. Efektywność metod zależy od kolejności rozwiązywania równań oraz od dominacji przekątnej macierzy - im większa tym zbieżność szybsza. Otrzymane rozwiązanie jest jednak najczęściej obarczone pewnym błędem.

Rozpatrzmy układ równań (4.1).

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n &= b_2 \\ \vdots &= \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n &= b_n \end{aligned} \tag{4.1}$$

Układ ten można zapisać macierzowo w postaci (4.2).

$$Ax = b \tag{4.2}$$

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. Równanie (4.2) można zapisać macierzowo

w postaci (4.3).

$$\begin{bmatrix} a_{1,1} & \dots & \dots & a_{1,n} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{n,1} & \dots & \dots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \dots \\ \dots \\ b_n \end{bmatrix} \tag{4.3}$$

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 wyznacznik macierzy $detA \neq 0$, to rozwiązanie układu równań jest jednoznaczne dla każdego b, przy czym dla $b = 0$ rozwiązanie jest zerowe. Rozwiązanie układu równań (4.2) można zapisać w postaci (4.4).

$$x = A^{-1}b \tag{4.4}$$

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).

4.1 Metody macierzowe

W rozdziale zostaną omówione dwie podstawowe metody rozwiązywania układów równań liniowych: metodę eliminacji Gaussa i metodę rozkładu LU. W praktyce częściej stosowana jest metoda rozkładu LU.

4.1.1 Metoda eliminacji Gaussa

Metoda eliminacji Gaussa [Wika] jest podstawową metodą rozwiązywania układów równań liniowych w skończonej liczbie operacji elementarnych obliczający $A^{-1}b$ - równanie (4.4). Algorytm eliminacji Gaussa jest pewnym przekształceniem liniowym M obu stron równania (4.2).

$$U = MA \tag{4.5}$$

$$b' = MB \tag{4.6}$$

Z definicji $A = M^{-1}U = LU$, gdzie macierz L jest macierzą dolną trójkątną, a U macierzą górną trójkątną macierzy A. Algorytm eliminacji Gaussa jest równoważny rozkładowi na czynniki L, U. W praktycznych zastosowaniach stosowany jest algorytm rozkładu LU - rozdział 4.1.2. Z tego też powodu algorytm nie będzie omawiany w szczegółach - zainteresowani znajdą odpowiednie informacje w literaturze [J.O95]

4.1.2 Metoda rozkładu LU

Metoda rozkładu LU [J.O94, J.O95, Wikd] jest jedną z najczęściej stosowanych technik rozwiązywania układów równań postaci (4.2). Metoda ta polega na znalezieniu takich macierzy L i U, aby

$$A = LU \tag{4.7}$$

Zatem równania postaci (4.2) można zapisać w postaci(4.8)

$$LUx = b \tag{4.8}$$

Prowadzi to do układu równań (4.9).

$$\begin{aligned} Ly &= b \\ Ux &= y \end{aligned} \tag{4.9}$$

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 (4.2) dla różnych wektorów b.

W praktyce stosowane są pewne modyfikacje mające na celu zwiększenie dokładności, ograniczenie liczby działań i liczby 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.

Implementacja rozkładu LU 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 [Lin81], w którym przez element podstawowy (z przekątnej macierzy) dzielony jest wiersz macierzy. Analogicznym algorytmem operujący na kolumnach jest algorytm Doolittle'a [Lin81]. Niestety podejścia te są niekorzystne z punktu widzenia wyboru elementu podstawowego [A.E86], dlatego też w praktyce najlepszym jest algorytm eliminacji Gaussa (alg. 2). Umożliwia on wybór elementu podstawowego z całej podmacierzy (*ang. complete pivoting*), a nie tylko z wiersza czy kolumny macierzy (*ang. partial pivoting*).

4.1.3 Metoda rozkładu LU macierzy rzadkich

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 tutaj:

  - informacje o postaci macierzy - należy odpowiednio ustawić równania węzłowe sieci minimalizując w ten sposób liczbę nowych wypełnień [Lin81] w trakcie rozkładu LU,

  - informacje o właściwościach elementów niezerowych (typach elementów niezerowych).

Algorytm 2 Podstawowy algorytm rozkładu LU z normalizacją macierzy U.

```
1:  for k = 1...n-1 do                                   ▷ dekompozycja lu
2:      for j = 1...n do
3:          a_kj = a_kj / a_kj
4:          for i = k+1...n do
5:              a_ij = a_ij - a_kj · a_ik
6:          end for
7:      end for
8:  end for
9:  for i = 1...n do                                     ▷ podstawienie w przód
10:     b_i = b_i / a_ii
11:     s = b_i
12:     for k = i+1...n do
13:         b(k) = b_j - a_ik · s
14:     end for
15: end for
16: for i = n...1 do                                     ▷ podstawienie wstecz
17:     for j = 1...i-1 do
18:         b_j = b_j - a_ij · b_i
19:     end for
20: end for
```

- 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 [1] (*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].

Na etapie przygotowania macierzy ważne jest odpowiednie ustawienie kolejności równań, jeżeli znana jest struktura macierzy. Techniki te zostaną omówione w dalszej części.

Klasyfikacja elementów niezerowych

Przedstawiona klasyfikacja elementów macierzy została stworzona pod kątem analizy układów elektronicznych, wykorzystującej algorytm Newtona-Raphsona 5.2 do iteracyjnego poszukiwania rozwiązania równań nieliniowych. W macierzy admitancyjnej $Y = A$ układanej metodą ZMPW (rozdział 12.1) dla potrzeb analiz OP/TRAN/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 - w iteracjach NR. (np. konduktancje wnoszone przez liniowe rezystancje),

[1] Element podstawowy macierzy to element z głównej przekątnej, przez który dzielony jest wiersz lub kolumna w trakcie rozkładu macierzy.

\+ - elementy zmienne, których wartość zmienia się w trakcie analiz:

OP, DC w iteracjach Newtona-Raphsona,

TRAN, 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.

Klasyfikacja elementów niezerowych umożliwia lepsze działanie algorytmów zapewniających poprawność numeryczną i redukujących nakłady obliczeniowe.

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 symulatorach najczęściej stosuje się różne strategie przestawień, które można zmieniać opcjami programu: wyłączenie przestawień, eliminacja zer z przekątnej, minimalizacja liczby elementów niezerowych. Z punktu widzenia strategii przestawień możliwe są przestawienia:

- wierszy,
- kolumn,
- wierszy i kolumn.

Przestawienie wierszy nie powoduje zmiany kolejności zmiennych w wektorze rozwiązań x, natomiast przestawienie kolumn wymaga zmiany kolejności zmiennych w wektorze rozwiązań x, co należy uwzględnić.

Z punktu widzenia poprawności numerycznej stosuje się odpowiednie strategie przestawień wierszy:

przyciąganie 1L strukturalnych - 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.

eliminajca zer z przekątnej - 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.

   Można to pokazać na przykładzie macierzy zawierającej 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 metodą ZMPW (rozdział 12.1) może działać w oparciu o miary Markowitza, tzn. wszystkie elementy mające miarę 1 mają elementy symetryczne [A.E86].

niesymetryczne przyciąganie 1 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 próbuje się zastąpić elementy zmienne na przekątnej jedynkami symetrycznymi (jeśli jest to możliwe).

W symulatorze *Dero* zastosowano algorytm 3 realizujący wstępne przestawienia niesymetryczne i symetryczne. Sposób przestawień zależy od opcji REXG.

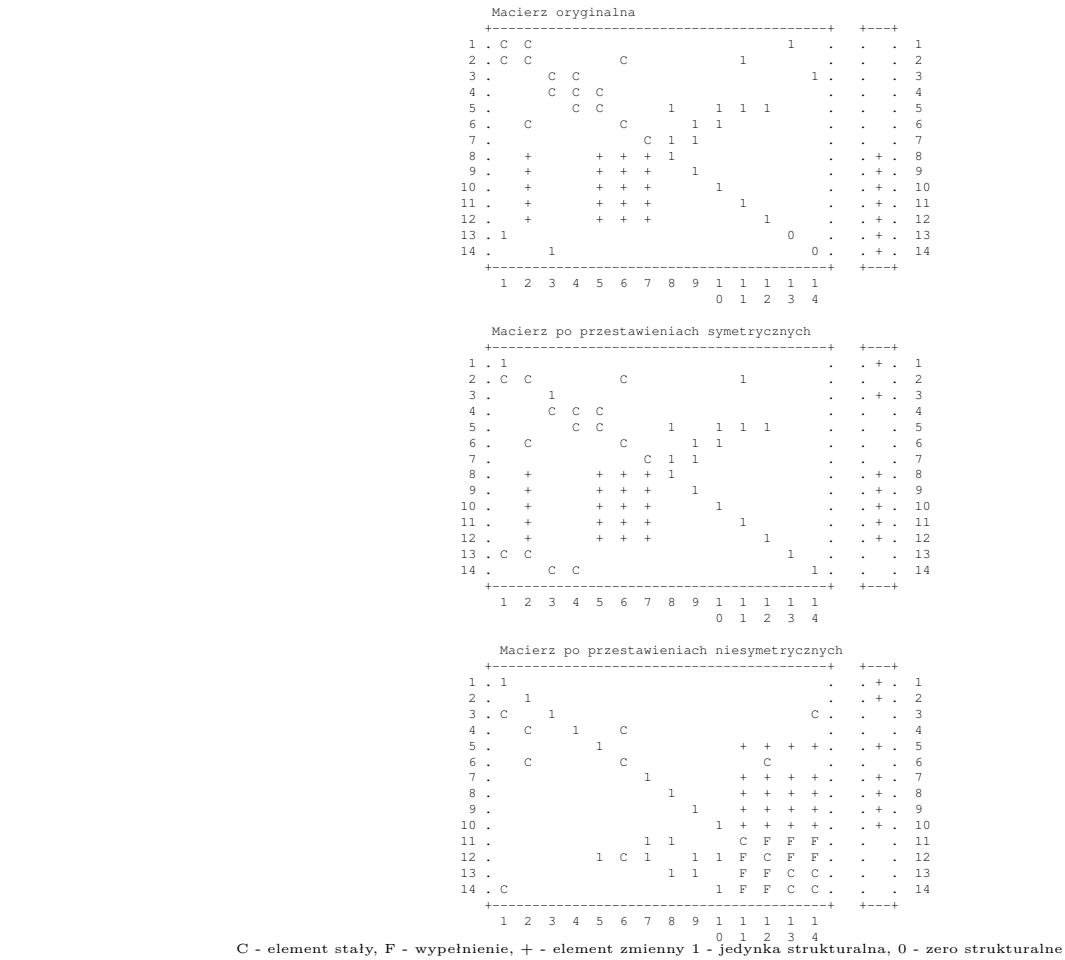
Wynik działania algorytmu przedstawiony jest na rys. 4.1. W pierwszym etapie zostają wyeliminowane zera z pozycji (13,13) i (14,14) przez przestawienia symetryczne jedynek. Następnie wykonywane zostają przestawienia elementów na przekątnej, tak aby na początku znalazły się elementy o najmniejszych miarach Markowitza. W trakcie przestawień następuje generacja brakujących wypełnień (F), np. na pozycji (14,11).

Wstępnie uporządkowana macierz nadaje się do dekompozycji, nie posiada zer na przekątnej i umożliwia efektywne zaimplementowanie algorytmu kontroli poprawności

Algorytm 3 Pełny algorytm wstępnego przenumerowania zastosowany w programie *Dero*.

Require: REXG ▷ Opcja sterująca przestawieniami
utwórz listy elementów niezerowych macierzy
wykonaj wstępne przestawienia symetryczne wierszy i kolumn Poniższe operacje wykonywane są iteracyjnie aż do momentu, gdy wszystkie zera z przekątnej zostaną wyeliminowane.
if REXG==MIN **then** ▷ minimalizacji liczby wypełnień
    przyciągnij 1L strukturalne
    eliminuj zera z przekątnej
    przyciągnij niesymetryczne 1
end if
if REXG==ZERO **then** ▷ normalne przestawienia i eliminacja zer
    przyciągnij niesymetryczne 1
    if nie udało się przestawić **then** ▷ zmień strategię przestawień
        przyciągnij 1L strukturalne
        eliminuj zera z przekątnej
        przyciągnij niesymetryczne 1
    end if
end if
if REXG==OFF **then** ▷ normalne przestawienia
    przyciągnij 1L strukturalne
    if są zera na przekątnej **then** ▷ zmień strategię przestawień
        eliminuj zera z przekątnej
        przyciągnij niesymetryczne 1
    end if
end if
sprawdź poprawność elementów na głównej przekątnej
if brak elementu na przekątnej lub zero strukturalne **then**
    BŁĄD - macierz osobliwa
else
    usuń elementy zerowe spoza głównej przekątnej
    oblicz liczbę elementów pojedynczych (*ang. singletons*)
    ustaw elementy pojedyncze (*ang. singletons*) na początku od największej wartości modułu
    oblicz miary Markowitza dla elementów z przekątnej
    określ kolejność elementów przekątnej na podstawie miary Markowitza ▷ przestawienia niesymetryczne
    usuń elementy zerowe spoza przekątnej
    przestaw elementy symetryczne spoza przekątnej
    ustaw elementy na przekątnej w kolejności rosnącej miary Markowitza
    usuń elementy zerowe spoza głównej przekątnej
    SUKCES - udało się wykonano przestawienia
end if
KONIEC

numerycznej dekompozycji. Na początkowe miejsca powinny zostać wstawione elementy pojedyncze (*ang. singletons*, dla których nie ma potrzeby wykonywania dekompozycji.



Rys. 4.1: Struktura macierzy układu w trakcie wstępnego przestawienia wierszy i kolumn.

Kontrola dokładności dekompozycji

Prawidłowy dobór elementów podstawowych macierzy jest bardzo ważny z punktu widzenia poprawności numerycznej algorytmu dekompozycji macierzy. Jest to realizowane przez kontrolę poziomu wartości elementów podstawowych. Dla arytmetyki 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

$$\|a_{rs}^{(k)}\| = \max_{i,j=k\ldots n} \|a_{ij}^{(k)}\| \tag{4.10}$$

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 wybierany jest taki r-ty element w kolumnie k, który spełnia poniższą zależność.

$$\|a_{rk}^{(k)}\| = \max_{i=k\ldots n} \|a_{ik}^{(k)}\| \tag{4.11}$$

Następnie przestawiane są 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.

$$\|a_{ii}\| \geq \sum_{i=1, i\neq j}^{n} \|a_{ij}\| \tag{4.12}$$

to wybór elementu podstawowego nie jest konieczny[2].

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[3]. Może to spowodować znaczne zafałszowanie wyników[4] albo błędy nadmiaru (*ang. overflow*) lub niedomiaru (*ang. underflow*) - rozdział 2.1.2. 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. W trakcie rozwiązania należy przeskalować odpowiednio wartość rozwiązania[5].

Skalowanie 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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\, a_{kj}^{(k)} \tag{4.13}$$

[2] Strukturę symetryczną z silnie dominującą przekątną posiadają np. macierze admitancyjne układów pasywnych.

[3] Duży wpływ ma tzw. *numeryczne uwarunkowanie* równań. Układ równań jest *źle uwarunkowany*, jeśli wskutek błędów reprezentacji (zaokrągleń) liczb rzeczywistych oryginalne równania zostaną zaburzone z sposób uniemożliwiający dekompozycję.

[4] Jest to efekt maskowania elementów na skutek błędów zaokrągleń dla dużych różnic w wartościach elementów.

[5] 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

$$a_{ij}^{(k+1)} = \beta_i\ \alpha_j\ a_{ij}^{(k)} - \frac{\beta_i\ \alpha_k\ a_{ik}^{(k)}}{\beta_k\ \alpha_k\ a_{kk}^{(k)}}\ \beta_k\ \alpha_j\ a_{kj}^{(k)} \tag{4.14}$$

Po uproszczeniach otrzymamy

$$a_{ij}^{(k+1)} = \beta_i\ \alpha_j\ \left[a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\ a_{kj}^{(k)}\right] \tag{4.15}$$

Jak widać oba składniki są jednakowo skalowane i w przypadku użycia arytmetyki zmiennoprzecinkowej poprawa dokładności nie następuje.

Mnożnik powinien mieć wartość całkowitą i w przypadku systemów komputerowych używających arytmetyki binarnej, powinien mieć wartość będącą potęgą liczby 2*, co zapobiega błędom zaokrągleń.* Można zastosować np. współczynnik skalowania r dany (4.18).

$$x = \frac{log10(10 \cdot \gamma / a_{ii})}{log10(2.0)} \tag{4.16}$$

$$s = \begin{cases} x + 0.5 & dla \quad x > 0 \\ x - 0.5 & dla \quad x \leq 0 \end{cases} \tag{4.17}$$

$$r = \beta_i\ \alpha_j\ = 2^{RoundInteger(s)} \tag{4.18}$$

gdzie: γ jest minimalną wartością elementu z przekątnej poniżej której zaczyna się skalowanie macierzy. Skalowanie jest wykonywane jeśli $a_{ii} < \gamma$. Współczynnik γ można wyliczyć np. z (4.19).

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 W praktycznych zastosowaniach należy uwzględnić różne podejścia do kontroli dokładności dekompozycji macierzy. W symulatorze *Dero* wprowadzono możliwość kontroli dokładności dekompozycji macierzy. Zastosowano różne podejścia zależne od wartości opcji ustawionej opcji *PIV* symulatora *Dero*:

BASIC powoduje pominięcie kontroli poziomu elementów podstawowych (alg.2). 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.

SCALE i REORD stosuje się empiryczne kryterium doboru elementu podstawowego. Dla i-tego elementu z przekątnej a_{ii} obliczana jest miara

$$\gamma = \epsilon_{PIV} \max_{j=1\ldots i} \|a_{jj}\| + \delta_{PIV} \tag{4.19}$$

gdzie: ϵ_{PIV} jest dopuszczalną względną różnicą wartości elementów podstawowych (wartość ustawiana przez użytkownika), δ_{PIV} - dopuszczalną bezwzględną różnicą (wartość ustawiana przez użytkownika), $\max_{j=1\ldots i} \|a_{jj}\|$ - maksymalnym dotychczasowym modułem wartości elementu podstawowego. Jeśli wartość modułu elementu podstawowego jest mała, tzn.

$$\|a_{ii}\| < \gamma \tag{4.20}$$

to dla:

SCALE skalowane są wszystkie elementy i-tej kolumny, tzn. mnożone są one przez taką wartość τ, że

$$\tau \cdot a_{ii} > \gamma \tag{4.21}$$

Realizuje to alg. 4. Skalowanie kolumn macierzy opisuje wzór (4.18).

REORD następuje wybór nowego elementu podstawowego z podmacierzy[6]. Realizuje to alg. 5.

4.2 Metody iteracyjne

Metody iteracyjnego rozwiązywania układów równań liniowych postaci (4.1) należą do metod przybliżonych.

Zajęcie się ww. metodami jest uzasadnione ich praktycznym wykorzystaniem w systemach symulacji. W praktyce obliczeniowej systemy symulacji rozwiązują coraz większe układy równań, ze względu na symulację coraz większych układów opisanych równaniami o kilku milionach niewiadomych (kilku milionach równań). Metody bezpośrednie (wykorzystujące na przykład rozkład LU, metodą eliminacji Gaussa lub inne rozkłady macierzy, np. QR) mogą być zbyt kosztowne lub mniej wygodne w użyciu. Dodatkowo nakład obliczeniowy jest rzędu $O(n^3)$, gdzie n jest liczbą niewiadomych. W przypadku, jeśli niewiadomych jest kilka milionów, a macierz nie ma żadnej specjalnej struktury, którą można wykorzystać do minimalizacji liczby działań, to korzystając z eliminacji Gaussa należy wykonać rzędu 10^{19} działań arytmetycznych. Należy zarezerwować rzędu 10^{13} bajtów (10000 GB) pamięci na czynniki rozkładu macierzy A. Przy założeniu, że komputer jest w stanie wykonać 10^{10} działań na sekundę (około $10 Gflops/s$), czas obliczeń wyniesie $10^9 s = 32 lata$. Alternatywą mogą być metody iteracyjnego rozwiązania układu równań w sposób przybliżony. Wymaga to mniejszej mocy obliczeniowej, a co za tym idzie krótszego czasu. Metody te są szczególnie użyteczne, gdy A jest macierzą rozrzedzoną. Oczywiście metody iteracyjnego rozwiązywania równań mają swoje ograniczenia i nie zawsze mogą być stosowane. W praktyce wykorzystywane są także

[6] Wybierany jest nowy element tylko z przekątnej podmacierzy. Może to powodować generację nowych wypełnień.

Algorytm 4 Algorytm rozkładu LU - algorytm Gaussa ze skalowaniem kolumn macierzy.

```
1: d_max = 0
2: for k = 1 ... n − 1 do
3:     d_max = max(d_max, ||d_kk||)   ▷ skalowanie - max moduł elementu podstawowego
4:     γ = ε_PIV d_max + δ_PIV                                   ▷ poziom skalowania
5:     if ||d_kk|| < γ then
6:         if d_kk == 0 then                                     ▷ zero na przekatnej
7:             d_kk = pivot_min
8:             d_max = max(d_max, ||d_kk||)
9:         end if
10:        γ = ε_PIV d_max + δ_PIV                               ▷ poziom skalowania
11:        if γ > 0 then                                         ▷ skalowanie kolumny
12:            y = 10 · γ/||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 ... n do                                ▷ skaluj kolumnę k
21:                a_ik = r · a_ik
22:            end for
23:        end if
24:    end if
25: end for
26: for j = 1 ... n do                                           ▷ dekompozycja
27:    a_kj = a_kj/a_kj
28:    for i = k + 1 ... n do
29:        a_ij = a_ij − a_kj · a_ik
30:    end for
31: end for
32: for i = 1 ... n do                                ▷ algorytm podstawienia w przód
33:    b_i = b_i/a_ii
34:    s = b_i
35:    for k = i + 1 ... n do
36:        b_k = b_j − a_ik · s
37:    end for
38: end for
39: for i = n ... 1 do                                ▷ algorytm podstawienia wstecz
40:    for j = 1 ... i − 1 do
41:        b_j = b_j − a_ij · b_i
42:    end for
43: end for
44: for i = 1 ... n do                                ▷ skalowanie rozwiązania
45:    b_i = r_i · b_i
46: end for
```

Algorytm 5 Algorytm rozkładu LU z przestawieniem elementów na przekątnej.

1: DEKOMPOZYCJA LU
Require: ϵ_{PIV}, δ_{PIV} ▷ dokładności
2: $d_{max} = 0$
3: oblicz liczbę elementów pojedynczych $singletons_{num}$
4: ustaw elementy pojedyncze ($singletons$)na początku przekątnej
5: **for** $k = singletons_{num} \dots n-1$ **do** ▷ pomiń elementy pojedyncze
6: $d_{max} = max(d_{max}, \|d_{kk}\|)$ ▷ maksymalny moduł elementu podstawowego
7: $\gamma = \epsilon_{PIV}\ d_{max} + \delta_{PIV}$ ▷ poziom przestawienia
8: **if** $\|d_{kk}\| < \gamma$ **then** ▷ zbyt mała wartość *pivota* - przestawienia
9: oblicz miary Markowitza dla elementów $k \dots n$
10: sortuj *pivoty* w kolejności rosnącej miary Markowitza
11: indeksy znajdują się w wektorze m_s o długości n
12: **for** $i = k \dots n$ **do** ▷ znajdź pierwszy *pivot* o najmniejszej miarze Markowitza
13: **if** $d_{m_i} > \gamma$ **then** ▷ znalazłem pierwszy dobry *pivot*
14: przestaw k-ty z m_i-tym *pivotem* ▷ przestaw symetrycznie wiersze i kolumny
15: idz do 18
16: **end if**
17: **end for**
18:
19: **end if**
20: **for** $j = 1 \dots n$ **do** ▷ dekompozycja
21: $a_{kj} = a_{kj}/a_{kj}$
22: **for** $i = k+1 \dots n$ **do**
23: $a_{ij} = a_{ij} - a_{kj} \cdot a_{ik}$
24: **end for**
25: **end for**
26: **end for**
27: **for** $i = 1 \dots n$ **do** ▷ podstawienie w przód
28: $b_i = b_i/a_{ii}$
29: $s = b_i$
30: **for** $k = i+1 \dots n$ **do**
31: $b(k) = b_j - a_{ik} \cdot s$
32: **end for**
33: **end for**
34: **for** $i = n \dots 1$ **do** ▷ podstawienie wstecz
35: **for** $j = 1 \dots i-1$ **do**
36: $b_j = b_j - a_{ij} \cdot b_i$
37: **end for**
38: **end for**
39: ustaw kolejność zmiennych w wektorze rozwiązań

metody mieszane - iteracje na poziomie bloków równań, a bloki rozwiązywane metodami bezpośrednimi.

W dalszej części omówiono szereg metod:

- algorytm iteracji prostej,

- metodę Gaussa-Seidela,
- metodę Gaussa-Jacobiego,
- metody pod- i nadrelaksacyjną,
- metodę Gaussa-Seidela-Newtona dla układów równań nieliniowych.
- metody siatkowe (multigrid).

Podano warunki stosowania ww. metod, warunki konieczne zbieżności oraz warunki zakończenia iteracji. Omawiane algorytmy są podstawą działania analizy czasowej kierowanej zdarzeniami (ED-ITA - rozdział 8.5).

4.2.1 Algorytm iteracji prostej

Przyjmijmy dowolną nieosobliwą macierz B. Odejmijmy stronami iloczyn Bx w równaniu (4.2). Po przekształceniach równania $Bx = b - (A - B)x$ otrzymamy formułę iteracji prostej (4.22).

$$Bx^{k+1} = b - (A - B)x^k \tag{4.22}$$

Formuła ta umożliwia określenie x^{k+1} na podstawie x^k, jeżeli układ (4.22) jest rozwiązywalny i ciąg $x^k \mid_{k=0}^{\infty}$ jest zbieżny (twierdzenie 4.2.1). Przepiszmy równanie (4.22) do postaci:

$$x^{k+1} = \underbrace{(1 - B^{-1}A)}_{C} x^k + \underbrace{B^{-1}b}_{E} \tag{4.23}$$

Twierdzenie 4.2.1. *Załóżmy, że $b \in R^n$ i $A \in R^{n\times n}$ jest nieosobliwa. Jeżeli B jest nieosobliwa i promień spektralny $C = (1-B^{-1}A)$, oznaczony jako $\rho(C)$, spełnia warunek $\rho(C) < 1$, to ciąg iteracji $x^{(k)}$ zdefiniowany jako $x^{k+1} = Cx^k + E$ jest zbieżny do rozwiązania $x = A^{-1}b$ startując z dowolnie wybranego punktu startowego x^0.*

Z równania (4.23) widać, że szybkość zbieżności zależy od promienia spektralnego $\rho(C)$ (rozdział 3.2.1) macierzy charakterystycznej C. Im promień ten jest mniejszy, tym szybkość zbieżności jest większa. Z tego powodu wszystkie metody iteracyjne dążą do minimalizacji promienia spektralnego. Jeżeli zapiszemy równania w postaci macierzowej, to szybkość zbieżności zależy od dominacji głównej przekątnej. Jeżeli suma elementów spoza przekątnej w każdym wierszu jest dużo mniejsza od elementu diagonalnego, to promień spektralny jest mały (4.24).

$$\sum_{j=i,j\neq i}^{n} |a_{ij}| < |a_{ii}| \tag{4.24}$$

gdzie $1 \leq i \leq n$.

4.2.2 Metoda Gaussa-Seidela

Metoda Gaussa-Seidela (GS) jest iteracyjną metodą numeryczną rozwiązywania układów równań liniowych. Stosowana jest głównie do rozwiązywania dużych układów równań postaci $Ax = b$, dla których A jest macierzą z silnie dominującą przekątną. Stosowana jest w metodach numerycznego rozwiązania eliptycznych równań cząstkowych.

Obecnie dla małych układów równań dużo szybsze są metody bezpośrednie, np. metoda eliminacji Gaussa, rozkładu LU. Dla dużych układów stosowane metody iteracyjne zapewniające lepszą zbieżność - metody nadrelaksacyjne oraz wielosiatkowe (ang. multigrid) [J.M] - rozdział 4.2.5.

Metoda GS jest metodą relaksacyjną. Idea metody polega na poszukiwaniu rozwiązania z dowolnie wybranego rozwiązania próbnego x_0. W kolejnych krokach (iteracjach) składowe rozwiązania są przybliżane do rozwiązania. Metoda Gaussa-Seidela bazuje na metodzie Jacobiego, jednak w metodzie GS korzysta się ze wszystkich aktualnie dostępnych przybliżonych składowych rozwiązania. Pozwala to zaoszczędzić pamięć i zmniejszyć około dwukrotnie nakład obliczeniowy niezbędny do osiągnięcia zadanej dokładności rozwiązania.

Dla ustalenia uwagi ponownie zapiszmy układ równań postaci (4.2).

$$Ax = b$$

przyjmując podział macierzy A taki jak w rozdziale 3.2.2) $A = D+L+U$, gdzie D - nieosobliwa macierz diagonalna, L - macierz dolna trójkątna, U - macierz górna trójkątna macierzy A Pojedynczą iterację metody Gaussa-Seidela można zapisać algebraicznie w postaci (4.25).

$$x^{(k+1)} = (D + L)^{-1} \left(-Ux^{(k)} + b \right), \tag{4.25}$$

gdzie (rozdział 3.2.2): $k = 0, 1, 2, \ldots$ oznacza numer iteracji Gaussa-Seidela. Po rozpisaniu na składowe wzór ten przyjmuje postać używaną w implementacjach numerycznych:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \; i = 1, 2, \ldots, n. \tag{4.26}$$

Uwaga!

W powyższych wzorach zakłada się, że w razie potrzeby kolejność równań została zmieniona tak, by dominujące (tj. największe co do modułu w danym równaniu) współczynniki równania znajdowały się na głównej przekątnej macierzy A.

Jeżeli A jest macierzą nieosobliwą, to zawsze można tak przestawić jej wiersze i kolumny, by macierz D też była nieosobliwa.

Metodę Gaussa-Seidela stosuje się niemal wyłącznie do układów z macierzą z dominującą przekątną, gdyż w wielu praktycznych zastosowaniach (np. przy rozwiązywaniu eliptycznych równań różniczkowych cząstkowych) jest to łatwy do spełnienia warunek gwarantujący zbieżność metody.

Metodę Gaussa-Seidela można stosować także do układów równań liniowych, w których macierz nie posiada dominującej przekątnej, ale poza nielicznymi wyjątkami zwykle nie ma gwarancji, że w tym przypadku metoda będzie zbieżna.

Kryteria zbieżności metody GS

Kryterium silnej dominacji w wierszach Metoda Gaussa-Seidela jest zbieżna dla każdej macierzy A spełniającej warunek ścisłej dominacji przekątniowej w rzędach [JS02].

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad \text{dla wszystkich } i = 1, 2, \ldots, n \tag{4.27}$$

Kryterium silnej dominacji w kolumnach Metoda Gaussa-Seidela jest zbieżna dla każdej macierzy A spełniającej warunek ścisłej dominacji przekątniowej w kolumnach [JS02].

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ji}| \qquad \text{dla wszystkich } i = 1, 2, \ldots, n \tag{4.28}$$

Kryterium słabej dominacji w wierszach Kolejne kryterium dotyczy nieredukowalnych układów równań liniowych, tj. takich układów, których nie można uporządkować przez permutacje wierszy i kolumn w ten sposób, by niektóre niewiadome można było wyznaczyć poprzez rozwiązanie mniejszej liczby równań niż n.

Jeżeli wszystkie wyrazy diagonalne macierzy nieredukowalnej A dominują rzędami w sensie słabym

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad \text{dla wszystkich } i = 1, 2, \ldots, n \tag{4.29}$$

oraz jeżeli dla co najmniej jednego wiersza $i \in \{1, 2, \ldots, n\}$ zachodzi dominacja silna:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \tag{4.30}$$

to ciąg iteracji Gaussa-Seidela jest zbieżny [JS02, JCT97].

Macierz A jest nieredukowalna, jeżeli poprzez przestawienie wierszy i kolumn nie można jej sprowadzić do postaci blokowej górnej trójkątnej. Nieredukowalność macierzy kwadratowej A o rozmiarze $n \times n$ można sprawdzić za pomocą grafu skierowanego $G(A)$ mającego n węzłów $P_1, P_2, \ldots, P_n$, w którym para (P_i, P_j) jest połączona (skierowanym) łukiem biegnącym od P_i do P_j wtedy i tylko wtedy, gdy $A_{ij} \neq 0$. Macierz A jest nieredukowalna wtedy i tylko wtedy, gdy między dowolnymi dwoma różnymi węzłami P_i, P_j w grafie $G(A)$ istnieje połączenie łukami skierowanymi (połączenie to nie musi być bezpośrednie).

Kryterium słabej dominacji w kolumnach Jeżeli wszystkie wyrazy diagonalne macierzy nieredukowalnej A dominują kolumnami w sensie słabym

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ji}| \qquad \text{dla wszystkich } i = 1, 2, \ldots, n \tag{4.31}$$

oraz jeżeli dla co najmniej jednej kolumny $i \in \{1, 2, \ldots, n\}$ zachodzi dominacja silna:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ji}|, \tag{4.32}$$

to ciąg iteracji Gaussa-Seidela jest zbieżny [JS02].

Kryterium dodatniej określoności Jeżeli macierz A jest dodatnio określona, to metoda Gaussa-Seidela jest zbieżna dla dowolnego wektora początkowego [JS83, JS93, JS02].

Warunek konieczny zbieżności

Niech

$$B = -(D + L)^{-1}U \tag{4.33}$$

Metoda Gaussa-Seidela jest zbieżna wtedy i tylko wtedy, gdy moduły wszystkich wartości własnych B są mniejsze od 1 [JS83].

Powyższe kryterium jest niepraktyczne i nie jest wykorzystywane w obliczeniach numerycznych.

Warunek zakończenia iteracji

W praktyce iteracje Gaussa-Seidela kończy się wtedy, gdy dla iteracji o numerze $k > 1$ maksymalna względna zmiana składowej przybliżonego rozwiązania nie przekracza pewnego z góry zadanego małego parametru ε (np. $\varepsilon = 10^{-10}$):

$$\max_i(|x_i^k - x_i^{k-1}|) < \varepsilon \max_i(|x_i^k|). \tag{4.34}$$

Alternatywny sposób polega na śledzeniu wektora reszt

$$r = b - Ax$$

. Obliczenia przerywa się, gdy

$$\max_i(|r_i|) < \varepsilon$$

osiągnie wartość mniejszą od pewnego z góry ustalonego małego parametru ε.

W metodzie Gaussa-Seidela w każdym kroku modyfikuje się pewną składową rozwiązania (x_j), tak by wyzerować odpowiadającą mu składową wektora reszt (r_j). Sukcesywne zerowanie jednej lub kilku składowych wektora reszt stanowi istotę wszystkich metod relaksacyjnych. Aktualizacja wektora reszt w kolejnych krokach może być przeprowadzona stosunkowo niewielkim nakładem obliczeń.

4.2.3 Metoda Gaussa-Jacobiego

W metodzie Gaussa-Jacobiego (GJ) [C.T95, M.D82, Z.F93] dokonano innego wyboru macierzy B (4.35).

$$B = L + U \tag{4.35}$$

Zapisując równanie postaci (4.2) za pomocą macierzy D,L,U otrzymamy (4.36).

$$(D + L + U)x = b \tag{4.36}$$

Przekształcając równanie (4.36) i stosując wybór macierzy B zgodnie z (4.35) otrzymamy (4.37).

$$Dx = b - (L + U)x \tag{4.37}$$

Po przekształceniach otrzymujemy równanie iteracji prostej postaci (4.38).

$$Dx^{k+1} = b - (L + U)x^k \tag{4.38}$$

Macierz charakterystyczna dana jest (4.39).

$$C_J = -D^{-1}(L + U) \tag{4.39}$$

Metoda ta jest zbieżna w jednej iteracji, jeśli $L+U=0$. Ponadto, jeżeli metoda GJ jest zbieżna dla danej macierzy A i macierz C_J ma nieujemne elementy, to metoda Gaussa-Seidela także jest zbieżna. Jeżeli $0<\rho(C_J)<1$, to $\rho(C_{GS})<\rho(C_J)$, a więc metoda GS jest asymptotycznie szybciej zbieżna [R.S65].

4.2.4 SOR - metoda nadrelaksacyjna

Metoda relaksacyjna [C.T95, M.D82, Z.F93] jest udoskonaloną wersją metody Gaussa-Seidela mającą na celu poprawę zbieżności poprzez zmniejszenie modułu największej wartości własnej macierzy charakterystycznej. W metodzie SOR macierz B ma postać (4.40).

$$B(\omega)=D/\omega+L \tag{4.40}$$

Macierz charakterystyczna C, decydująca o zbieżności, przyjmuje postać (4.41).

$$C=B(\omega)^{-1}[B(\omega)-A]=(D+\omega L)^{-1}[(1-\omega)D-\omega U] \tag{4.41}$$

Algorytm iteracyjny dla i-tego równania można zapisać w postaci (4.42).

$$x_i^{k+1}=(1-\omega)x_i^{k+1}+\frac{\omega}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{k+1}-\sum_{j=i+1}^{n}a_{ij}x_j^k\right) \tag{4.42}$$

Zbieżność jest możliwa dla $\omega\in(0,2)$. Dla $\omega=1$ metoda sprowadza się do metody Gaussa-Seidela. W praktyce stosuje się $\omega>1$ i metoda taka nazywana jest metodą nadrelaksacyjną (*ang. Simple Over-Relaxatiom method*). Optymalna wartość ω_{opt} dana jest (4.43).

$$\omega_{opt}=\frac{2}{1+\sqrt{1+\lambda}} \tag{4.43}$$

gdzie: λ jest największą wartością własną macierzy C, dla której promień spektralny dany jest (4.44).

$$\rho[C(\omega_{opt})]=\omega-1=\left[\frac{\sqrt{cond(A)}-1}{\sqrt{cond(A)}+1}\right]^2 \tag{4.44}$$

W praktyce trudno jest wyznaczyć największą wartość własną λ. Dlatego wartość ω z szacunków jest nieco większa niż optymalna. Przyjęcie nieco za dużej wartości powoduje jednak niewielkie zmniejszenie szybkości zbieżności (zobacz [M.D82],rys.6.3,str.65).

Metoda nadrelaksacyjna jest stabilna numerycznie, ale nie jest numerycznie poprawna, ze względu na wprowadzony współczynnik ω.

4.2.5 Metody wielosiatkowe - multigrid

Metody wielosiatkowe (*ang. multigrid*) są odmianą metod iteracyjnego rozwiązywania równań polegającą na wykorzystaniu faktu, że rozwiązaniem układu równań posiada składowe, które szybciej i wolniej są wytłumiane w procesie dochodzenia do rozwiązania. W zapisie macierzowym odpowiada to istnieniu wartości własnych macierzy, gdzie:

- wartości własne o dużej wartości odpowiadają składowym o wysokich częstotliwościach

- wartości własne o małych wartości odpowiadają składowym o niskich częstotliwościach.

Stosując różne kroki (siatki) przyrostów wartości zmiennych można przyśpieszyć proces dochodzenia do rozwiązania przez wstępne *wygładzenie* rozwiązania (*ang. smooting*).

Algorytm 6 przedstawia przypadek dla dwóch siatek.

Algorytm 6 Multigrid dla przypadku dwóch siatek

1: Na gęstej siatce przeprowadź kilka lub kilkanaście iteracji, które zapewnią gładki przestrzenny rozkład błędów.
2: Oblicz resztę r dla gęstej siatki.
3: Oszacuj resztę dla siatki rzadkiej R.
4: Przeprowadź kilka iteracji równania korekcji na rzadkiej siatce.
5: Przenieś rozwiązanie na siatkę gęstą za pomocą interpolacji.
6: Skoryguj rozwiązanie na siatce gęstej.
7: Idź do 1

Efektywność działania metod wielosiatkowych (liczba iteracji potrzebnych do otrzymania rozwiązania) można podnieść przez stosowanie wielu siatek.

Podstawowymi parametrami metod wielosiatkowych są:

- metoda wygładzania reszty przy przejściu od siatki gęstej do rzadkiej,
- liczba iteracji wykonywanych na każdej siatce,
- kolejność wykonywania obliczeń na poszczególnych siatkach,
- metody interpolacji przy zmianie siatki z rzadkiej na gęstą.

4.2.6 Metoda Gaussa-Seidela-Newtona

Metoda Gaussa-Seidela-Newtona (GSN) służy do rozwiązywaniu układów równań nieliniowych z n niewiadomymi opisującymi dowolną sieć. Dla wygody zapiszmy równania sieci w postaci (4.45).

$$f(x) = 0 \tag{4.45}$$

Rozwiązując (4.45) techniką relaksacyjną (rozdzielania węzłowego) należy rozwiązać n równań biorąc do rozwiązania i-tego równania względem x_i zmienne $x_1, \dots x_{i-1}$ z danej relaksacji i $x_i, \dots x_n$ z poprzedniej relaksacji o postaci (4.46) względem x_i^k dla $i = 1, n$.

$$f_i^k(x_1^k, \dots, x_i^k, x_{i+1}^{k-1}, \dots, x_n^{k-1}) = 0 \tag{4.46}$$

gdzie: k - numer relaksacji, p - numer iteracji NR. Równanie (4.46) należy zlinearyzować względem niewiadomej x_i^k w otoczeniu rozwiązania z poprzedniej relaksacji x_i^{k-1}. Dla pojedynczego równania równanie iteracji ma postać (4.47)

$$x_i^{(k)(p)} = x_i^{(k)(p-1)} - \frac{f_p^k(x_1^{(k)(p)}, \dots, x_i^{(k)}, x_{i+1}^{(k-1)(p-1)}, \dots, x_n^{k-1})}{\delta f_i(\dots)/\delta x_i} \tag{4.47}$$

Metoda nosi nazwę Nieliniowej metody Gaussa-Seidela-Newtona[C.T95] (NGSN). Jeżeli wykonana zostanie tylko jedna iteracja NR ($p = 1$), to otrzymamy metodę GSN.

Rozdział 5

Analiza punktu pracy i charakterystyk

Analiza punktu pracy układu umożliwia wyznaczenie wartości zmiennych, jakie ustalą się w układzie po włączeniu zasilania i zaniku stanu nieustalonego (stanu przejściowego po włączeniu układu). Istnienie stanu nieustalonego związane jest z istnieniem w układzie elementów gromadzących energię (np. L, C). W zależności od wartości parametrów układu stan nieustalony może trwać krótko lub nieskończenie długo z punktu widzenia systemu symulacji. Wyznaczenie punktu pracy odpowiada wykonaniu analizy czasowej układu startującej z punktu $t = 0$ z czasem dążącym do nieskończoności $t = \infty$, gdzie zanika stan nieustalony. Z punktu widzenia systemu symulacji podejście takie jest nieakceptowane i nierealizowalne w praktyce ze względu na czas obliczeń i poważne trudności z zapewnieniem odpowiedniej dokładności obliczeń. Jest to szczególnie ważne w przypadku analizy układów nieliniowych, które mogą posiadać kilka możliwych punktów pracy. W zależności od wartości startowej algorytmu analizy czasowej i popełnianych w trakcie analizy błędów algorytm może wyznaczać rozwiązania lokalne.

Najbardziej popularną i szeroko stosowaną w praktyce metodą poszukiwania punktu pracy układów nieliniowych jest metoda Newtona-Raphsona (NR) - rozdział 5.2. Umożliwia ona znalezienie tylko jednego punktu pracy (rozwiązania), zależnego jednak od wartości startowej. W przypadku konieczności wyznaczenia wszystkich rozwiązań (punktów pracy) stosuje się różne metody, które wyznaczają punkt startowy dla metody NR, np.:

- metody kontynuacji,
- metody ewolucyjne,
- metody odcinkowo-liniowe.

Metody te zostaną omówione w kolejnych rozdziałach.

Innym ważnym problemem w analizie punktu pracy jest wyznaczenie wartości sygnałów w węzłach odosobnionych[1]. Zamiast wykonywać wspomnianą wcześniej analizę czasową z czasem $t \to \infty$, elementy takie zastępuje się konduktancjami o odpowiednio dużej wartości, proporcjonalnej do wartości pojemności. Podejście takie umożliwia szybkie wyznaczenie wartości sygnałów w węzłach odosobnionych przez analizę sieci konduktancyjnej - rozdział 5.2.4.

[1] Węzłem odosobnionym jest np. węzeł połączenia dwóch połączonych szeregowo kondensatorów.

5.1 Wyznaczenie pojedynczego punktu pracy układu

W analizie punktu pracy układu wykorzystywana jest szeroko metoda Newtona-Raphsona (NR) rozwiązywania układów równań nieliniowych opisana w rozdziale 5.2. Opiera się ona na rozwinięciu funkcji układowej (1.1)

$$f(x) = f(x, 0, 0)$$

w szereg Taylora do wyrazów rzędu drugiego. Metoda gwarantuje zbieżność w przypadku, gdy punkt startowy zostanie wybrany dostatecznie blisko rozwiązania (leży w obszarze zbieżności). W przypadku układów, które mają kilka punktów pracy, metoda NR nie gwarantuje znalezienia wszystkich punktów, a tylko najbliższego. Znaleziony punkt zależy od wartości startowych algorytmu NR. W praktycznych zastosowaniach algorytm NR powoduje najczęściej:

- problemy ze zbieżnością (źle dobrany punkt startowy, silne oscylacje),
- problemy numeryczne związane z przekroczeniem zakresu reprezentacji liczb.

Dlatego też implementowane są techniki umożliwiające poprawę właściwości algorytmu NR:

- Wybranie wartości startowych w obszarze zbieżności. Niestety nie ma pewności, czy wybrany punkt leży w obszarze zbieżności dla danego problemu. Dodatkowo wybór punktu startowego poza obszarem zbieżności powoduje rozbieżność metody NR.
- Metody ograniczania (tłumienia) zmian wartości zmiennych sterujących w kolejnych iteracjach. Na początku tłumienie jest większe i zmniejsza się wraz ze wzrostem liczny iteracji (rozdział 5.2.2).
- Aproksymacja przebiegów funkcji (najczęściej wykładniczych) zapobiegająca wyjściu algorytmu poza zakres reprezentowanych liczb (rozdział 5.2.2).
- Uważna implementacja algorytmu oraz modeli elementów pod kątem minimalizacji błędów obliczeniowych (rozdział 2.3.3).

W praktycznych zastosowaniach stosuje się kilka technik podnoszących niezawodność metody i redukujących nakłady obliczeniowe. Należą do nich:

- Metoda kontynuacji (rozdział 5.4.3), w której wartości startowe wymuszeń (wektor B) w procesie iteracyjnym są sukcesywnie zwiększane, aż do osiągnięcia wartości nominalnych. W każdym punkcie (dla B_i) stosowana jest metoda NR do uzyskania rozwiązania. Następnie zwiększane są wartości wymuszeń i obliczane metodą NR kolejne przybliżenia rozwiązania.
- Wstępne przewidywanie wartości startowej przez wykorzystanie informacji o właściwościach fizycznych elementów wchodzących w skład układu, np. dobrą wartością startową dla złącz p-n diod jest wartość około $0.7V$ (rozdział 5.2.2).
- Wprowadzenie dodatkowych elementów zmiennych do obwodu (konduktancje w każdym węźle sieci w obwodach elektronicznych) celem ograniczenia oscylacji w początkowych iteracjach NR i ich zwiększanie lub usuwanie w kolejnych iteracjach.

- Zastosowanie technik relaksacyjnych np. techniki wirtualnych pojemności. Odpowiada to wykonaniu analizy czasowej układu z zerowymi warunkami początkowymi z czasem dążącym do nieskończoności.
- Omijanie, czyli ponowne wykorzystywanie pochodnych $f'(x)$ z poprzedniej iteracji, jeśli x niewiele się zmienia - tzn. Δx spełnia test stopu NR - rozdział 5.2.2.

Analizę punktu pracy umożliwiającej wyznaczenie pojedynczego punktu pracy zapisano w postaci algorytmu 7.

Algorytm 7 Algorytm analizy punktu pracy układu.

Require: x_0 ▷ warunki początkowe (startowe)
Require: p=0 ▷ licznik iteracji NR
1: $x^{(0)} = x_0$
2: **for** $p = 1 \dots p_{max}$ **do**
3: oblicz wartości funkcji układowych $f(x^{(p-1)})$
4: ułóż równania sieci $Ax^{(p)} = B$ postaci (5.3)
5: rozwiąż $x^{(p)} = A^{-1}B$
6: **if** test stopu NR spełniony, np. (5.6) **then**
7: idź do 10
8: **end if**
9: **end for**
10: KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

W przypadku konieczności wyznaczenia punktu pracy układów A-stabilnych[2] stosowane są różne metody znane z literatury, do których należą:

- homotopia - ciągłe przejście między dwoma przekształceniami ciągłymi przestrzeni topologicznych, tj. takie, za pomocą którego można w jednostce czasu w wyniku ciągłej deformacji z jednego przekształcenia otrzymać drugie (rozdział 5.4.5).
- metoda przeszukiwania regionów *ang. multigrid*,
- metody ewolucyjne (rozdział 5.5).

5.2 Algorytm Newtona-Raphsona

Metoda Newtona-Raphsona [J.O95] jest klasyczną metodą rozwiązywania nieliniowych równań algebraicznych[3]. W metodzie Newtona przyjmuje się następujące założenia dla funkcji f:

1. W przedziale [a,b] znajduje się dokładnie jeden pierwiastek.
2. Funkcja ma różne znaki na krańcach przedziału, tj.
$$f(a) \cdot f(b) < 0.$$
3. Pierwsza i druga pochodna funkcji mają stały znak w tym przedziale.

[2] Układy A-stabilne to układy mające kilka punktów pracy.
[3] W nowoczesnych symulatorach mikrosystemów wykorzystywane są także inne metody np. metoda Powell'a [P+07]

4. Funkcja $f(x)$ jest różniczkowalna w punkcie x_0.

Jeżeli funkcja $f(x)$ jest różniczkowalna w punkcie x_0, to można ją rozwinąć w szereg Taylora do wyrazów rzędu drugiego w otoczeniu punktu x_0.

$$f(x) = f(x_0) + \frac{\delta f(x_0)}{\delta x}(x - x_0) \tag{5.1}$$

Równanie można rozwiązać iteracyjnie przyrównując wartość f do zera.

$$f(x) = 0$$

Po przekształceniach (5.1) ciąg kolejnych przybliżeń rozwiązania można zapisać w postaci (5.2).

$$\frac{\delta f(x_0)}{\delta x}(x - x_0) = -f(x_0) \tag{5.2}$$

Ciąg kolejnych przybliżeń rozwiązań

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

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:

$$x^{(p)} = x,$$
$$x^{(p-1)} = x_0.$$

Proces iteracji NR odpowiada prowadzeniu stycznej w $f(x_0)$, której przecięcie z osią OX umożliwia znalezienie kolejnego przybliżenia x_1. Jeżeli przybliżenie x_1 nie jest satysfakcjonujące, wówczas punkt x_1 wybierany jest jako kolejny punkt startowy. W punkcie $f(x_1)$ prowadzona jest styczna, której przecięcie z osią OX daje kolejne przybliżenie x_2. Proces kończy się po spełnieniu warunków zakończenia iteracji NR - rozdział 5.2.1. Odpowiednie przyrosty wartości 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[4]. Z punktu widzenia symulatora wartość Δx i tak należy wyznaczyć np. na potrzeby testu stopu lub w algorytmie obliczania przyrostów.

W praktyce stosowana jest także inna wersja algorytmu NR, gdzie niewiadomymi są wartości bezwzględne x, a nie Δx. Równanie przyjmuje wtedy postać (5.3), z której należy wyznaczyć $x^{(p)}$.

$$\frac{\delta f(x^{(p-1)})}{\delta x}\ x^{(p)} = -f(x^{(p-1)}) + \frac{\delta f(x^{(p-1)})}{\delta x}\ x^{(p-1)} \tag{5.3}$$

[4]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

Błąd przybliżenia Błąd p-tego przybliżenia wartości x w p-tej iteracji NR można oszacować z (5.4) na podstawie reszt z rozwinięcia w szereg Taylora, znając pierwszą i drugą pochodną.

$$|x - x^{(p)}| \leq \frac{f(x^{(p)})}{m} \tag{5.4}$$

Po przekształceniach otrzymujemy (5.5)

$$|x - x^{(p)}| \leq \frac{M}{2m}(x - x^{(p-1)})^2 \tag{5.5}$$

Odpowiednie stałe opisane są zależnościami:

$$\begin{aligned} m &= \min_{x\in[a,b]} |f'(x)| \\ M &= \max_{x\in[a,b]} |f''(x)| \end{aligned}$$

5.2.1 Warunek zakończenia iteracji

Proces iteracji może trwać przez wiele iteracji zbliżając się coraz bardziej do rozwiązania. W praktyce obliczenia wykonywane są z określoną wymaganą dokładnością. Z tego względu stosuje się kryteria zakończenia iteracji NR, które dodatkową umożliwiają ograniczenie nakładów obliczeniowych.

W praktyce stosowanych jest kilka kryteriów zakończenia iteracji NR (Δ to przyjętą dokładnością obliczeń):

1. Wartość funkcji w wyznaczonym punkcie $x^{(p)}$ jest bliska 0.
$$\left|f(x^{(p)})\right| \leq \Delta$$
2. Odległość pomiędzy kolejnymi przybliżeniami jest dość mała.
$$\left|x^{(p)} - x^{(p-1)}\right| \leq \Delta$$
3. Szacowany błąd jest dostatecznie mały.
$$\frac{M}{2m}(x^{(p)} - x^{(p-1)})^2 \leq \Delta$$
4. Kryterium mieszane - jednoczesne wykorzystanie kryterium 1 i 2.

W praktycznych rozwiązaniach stosuje się kryterium zakończnia iteracji NR wykorzystujące błąd względny i bezwzględny.

$$|\ \Delta x^{(p)}\ | < \epsilon\ \max(|\ x^{(p)}\ |, |\ x^{(p-1)}\ |) + \delta \tag{5.6}$$

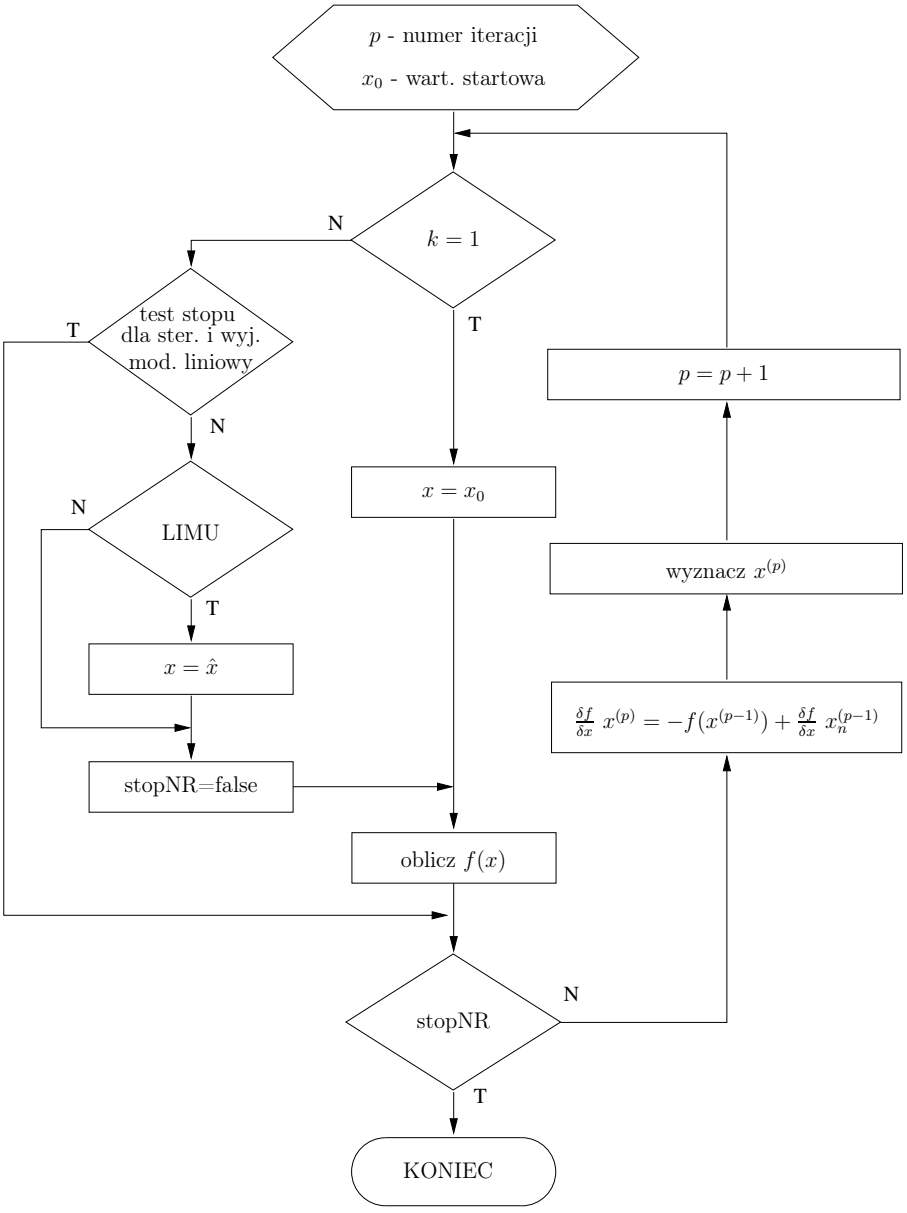
gdzie: ϵ - wartość błędu bezwzględnego, δ - wartość błędu względnego zmiennej x.

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 12.1). Czasami w obliczeniach łatwiej jest posługiwać się miarą błędu w danej iteracji NR (5.7).

$$M^{(p)} = \frac{|\ \Delta x^{(p)}\ |}{\epsilon\ \max(|\ x^{(p)}\ |, |\ x^{(p-1)}\ |) + \delta} \tag{5.7}$$

Wartość $M < 1$ oznacza, że spełniono kryterium zakończenia iteracji NR. Dla $M > 1$ błąd narasta i kryterium zakończenia iteracji NR nie jest spełnione. Proces iteracyjnego rozwiązywania równania (5.3) można zapisać w postaci algorytmu 8.

Algorytm 8 Algorytm Newtona-Raphsona



5.2.2 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.

$$expo(x) = \begin{cases} exp(A)(x - A - 1) & x > A \\ exp(x) & 0 \le x \le A \\ \frac{1}{1-x^2} & x < 0 \end{cases} \tag{5.8}$$

$$dxpo(x) = \begin{cases} exp(A) & x > A \\ exp(x) & 0 \le x \le A \\ \begin{cases} 1 & \text{wersja liniowa} \\ \frac{1}{1-x^2} & \text{wersja hiperboliczna} \end{cases} & x < 0 \end{cases} \tag{5.9}$$

$$cpx1am(x, m) = \begin{cases} (1 - x)^{-m} & x \le B \\ 10^m[m(10 \cdot x - 9) + 1] & x > B \end{cases} \tag{5.10}$$

Wartości stałych A i B zależą od zakresu reprezentacji liczb. Dla liczb typu *float* wartości mogą wynosić odpowiednio $A = 40$ i $B = 0.9$.

Transformacje zmiennych sterujących

Tłumienie przyrostów zmiennych sterujących x jest bardzo skuteczną metodą przyśpieszania zbieżności algorytmu NR. Zmienne sterujące można poddać transformacji zgodnie z zależnością (5.11).

$$\hat{x}^{(p)} = \hat{x}^{(p-1)} + \frac{1}{k} \cdot sign(x^{(p)} - \hat{x}^{(p-1)}) \cdot ln(1 + k\|x^{(p)} - \hat{x}^{(p-1)}\|) \tag{5.11}$$

gdzie: $\hat{x}$ - wartość ograniczona, $k = max(CLLB, CLUB - p)$ decyduje o sile ograniczenia, $CLUB$ i $CLLB$ są odpowiednimi parametrami zadanymi przez użytkownika, które decydują o sile ograniczania dla dodatnich i ujemnych przyrostów, p - numer iteracji NR.

Algorytm NR (5.3) w wersji tłumionej (5.12) został zastosowany w symulatorze *Dero* [Pla13].

$$\frac{\delta f(\hat{x}^{(p-1)})}{\delta x} x^{(p)} = -f(\hat{x}^{(p-1)}) + \frac{\delta f(\hat{x}^{(p-1)})}{\delta x} x_n^{(p-1)} \tag{5.12}$$

Przewidywanie wartości startowych

Metoda opiera się na znajomości właściwości fizycznych elementów opisanych analizowanymi równaniami. W przypadku elementów półprzewodnikowych, np. diod lub tranzystorów bipolarnych, dobrym punktem startowym jest wartość zmiennej sterującej x równa około $0.7V$. Wystartowanie z takiej wartości i zastosowanie algorytmu tłumienia (transformacje zmiennych sterujących 5.2.2), gwarantuje szybką zbieżność algorytmu NR.

Metoda omijania Metoda omijania (*ang. bypassing*) jest efektywną metodą podnoszącą wydajność metody NR nawet o kilkadziesiąt procent. Działanie metody opiera się na wykorzystaniu testu stopu iteracji NR dla elementów wchodzących w skład sieci. Jeżeli zmienne sterujące danego elementu spełniają test stopu NR, to nie wyznacza się nowych pochodnych cząstkowych funkcji wyjściowych po sterowaniach (zmiennych sterujących), lecz wykorzystuje sie pochodne obliczone w poprzedniej iteracji NR. W przypadku wystąpienia omijania danego elementu, nie ma sensu dalsze liczenie testu stopu dla zmiennych sterujących.

5.2.3 Quasi-newtonowska trzystopniowa metoda tłumiona

Trzystopniowa metoda tłumiona [D.82] jest bardzo efektywną metodą dynamicznego sterowania tłumieniem. Metoda składa się z trzech poziomów tłumienia:

1. Ograniczenie przyrostów zmiennych sterujących za pomocą ograniczenia, które zależy od przyrostu w poprzedniej iteracji. W przypadku, gdy znaki obu przerostów są zgodne, to stosunek przyjmowany jest jako 1.5, w przeciwnym przypadku 0.5. Przyrost wartości x w kolejnej iteracji wynosi:

$$\begin{aligned} x^{(p)} &= x^{(p-1)} + \Delta x^{(p)} \\ \Delta x^{(p)} &= sign(x^{(p)}) \cdot min(\Delta|x^{(p)}|, K|\Delta x^{(p-1)}|) \\ K &= \begin{cases} 1.5 & dla \quad sign(x^{(p)}) \cdot sign(x^{(p-1)}) > 0 \\ 0.5 & dla \quad sign(x^{(p)}) \cdot sign(x^{(p-1)}) > 0 \end{cases} \end{aligned} \tag{5.13}$$

2. Ograniczenie przyrostów macierzy pochodnych, które zabezpiecza przed wejściem algorytmu NR w obszar zbyt dużych nachyleń. Powoduje to powolną zbieżność algorytmu i grozi przekroczenie zakresu liczb (błąd nadmiaru). Metoda ta wykorzystuje przybliżoną macierz pochodnych J. Iteracje wykonywane są tak, jak w metodzie tłumionej, lecz wprowadzono współczynnik tłumienia macierzy pochodnych α.

$$\begin{aligned} x^{(p+1)} &= x^{(p)} + \alpha \Delta x^{(p+1)} \\ J^{(p)} \Delta x^{(p+1)} &= f(x^{(p)}) \end{aligned} \tag{5.14}$$

   W bieżącej iteracji używana jest macierz pochodnych pośrednia pomiędzy macierzą z poprzedniej iteracji, a bieżącą macierzą pochodnych. Metoda NR nie wymaga dokładnej wartości pochodnych w macierzy J. Wartości te decydują o sposobie dochodzenia do rozwiązania.

$$J^{(p)} = J^{(p-1)} + \alpha \left[\frac{f(x^{(p)}}{dx}) - J^{(p-1)} \right] \tag{5.15}$$

Działanie algorytmu rozpoczyna się od przyjęcia $\alpha = 1$ oraz $J^{(p)}$ równe macierzy pochodnych. W kolejnej iteracji J będzie już różne. Wadą metody jest konieczność pamiętania macierzy z poprzedniej iteracji $J^{(p-1)}$. Aktualizację wpsółczynnika α wykonuje się, zgodnie z algorytmem 9, po każdej iteracji NR, tj. rozwiązaniu układu równań liniowych. W algorytmie przejęto miarę błędu (5.7), która jest

Algorytm 9 Algorytm modyfikacji współczynnika tłumienia.

1: **if** $M^{(p)} > M^{(p-1)}$ **then**
2: $\alpha = \alpha/4$ ▷ zmniejsz tłumienia
3: $M^{(p)} = 1.5M^{(p-1)}$
4: **else**
5: $\alpha = min(2\alpha, 0.32 + 0.7\alpha, 1)$ ▷ zwiększenie tłumienia
6: **end if**

odpowiednio przekształconym testem stopu iteracji (5.6). W przypadku, gdy błąd wzrasta tłumienie jest wzmacniane, a gdy maleje jest osłabiane. W ostatniej iteracji miara błędu wzrasta (brak poprawy), co spowodowane jest większą odpornością na lokalne minima, bez pogarszania zbieżności.

3. Metoda kontynuacji (uciąglania zagadnienia) - rozdział 5.4.3. Polega ona na zastosowaniu ciągu metod NR, przy czym każda z nich startuje z punktu rozwiązania uzyskanego z poprzedniego wywołania metody NR. Wprowadza się współczynnik skalujący wymuszenia zewnętrzne $\alpha \in [0, 1]$ (wektor prawych stron B). Jako punkt startowy przyjmuje się często $\alpha = 0$ - zerowe warunki początkowe. Po osiągnięciu zbieżności zwiększa się α i startuje się z poprzednio obliczonego punktu startowego. Odpowiada to przesuwaniu się punktu rozwiązania coraz bliżej końcowego punktu (dla $\alpha = 1$) na charakterystyce elementów nieliniowych.

5.2.4 Wyznaczenie wartości w węzłach odosobnionych

W analizie punktu pracy występuje problem analizy wartości sygnałów w tzw. węzłach odosobnionych. Są to węzły, w których z definicji nie można wyznaczyć wartości w analizie punktu pracy, bez odpowiedniej modyfikacji równań. Węzłami odosobnionymi są np.:

- węzły bramki tranzystora MOS, gdzie bramka podłączona jest do drenu i źródła tylko poprzez pojemności,
- węzeł środkowy pomiędzy połączonymi szeregowo pojemnościami.

W rzeczywistości w węzłach odniesienia ustalają się wartości sygnałów, które są wynikiem zaniku stanu nieustalonego dla $t \to \infty$. W analizie komputerowej wyznaczane są wartości graniczne bez uwzględnienia stanu nieustalonego. Jest to spowodowane tym, że analizie poddawana jest sieci rezystancyjna, która jest modelem układu dla stanu granicznego, po zaniku stanu nieustalonego dla $t = \infty$.

Rozwiązaniem problemu jest wprowadzenie do procesu iteracyjnego mechanizmów imitujących zbieżny do 0 proces ustalania się prądów pojemności, co odpowiada zanikowi stanu nieustalonego. Nie ma znaczenia jak proces ten przebiega, ale musi prowadzić do ustalenia odpowiednich napięć na pojemnościach, a co za tym idzie w węzłach odosobnionych. Należy zmienić sposób modelowania pojemności za pomocą konduktancji

przez wprowadzenie odpowiedniego współczynnika Δ_p dążącego szybko do ∞ w kolejnych iteracjach p (5.16).

$$i^{(p+1)} = \frac{C}{\Delta_p} v^{(p+1)} \tag{5.16}$$

W wyniku eksperymentów [OJ70] wyznaczono eksperymentalnie wartość Δ_p.

$$\Delta_p = \begin{cases} 1 & dla \quad p = 1 \dots k, \ \epsilon_p \leq 20 \cdot \epsilon \\ min(10 \cdot \Delta_{p-1}, 10^{12}) & dla \quad p = k \dots p_{max} \end{cases} \tag{5.17}$$

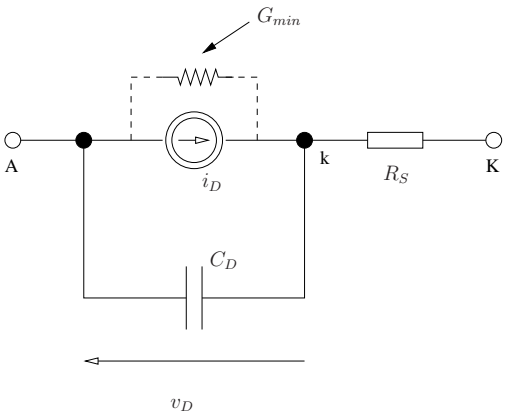
Algorytm 10 Algorytm wyznaczania sygnałów w węzłach odosobnionych.

Require: x_0 ▷ warunki początkowe (startowe)
Require: $p = 0$ ▷ licznik iteracji NR
1: $x^{(0)} = x_0$
2: $\Delta_0 = 1$
3: **for** $p = 1 \dots p_{max}$ **do**
4: **if** $\epsilon_p \leq 10 \cdot \epsilon$ **then** ▷ Δ_p dla konduktancji zastępczych
5: **if** $\Delta_p >= 10^{12}$ **then**
6: $\Delta_p = 10^{12}$
7: **else**
8: $\Delta_p = 10 \cdot \Delta_{p-1}$
9: **end if**
10: **end if**
11: oblicz wartości funkcji układowych $f(x^{(p-1)}, \Delta_p)$
12: ułóż równania sieci $Ax^{(p)} = B$ postaci (5.3)
13: rozwiąż $x^{(p)} = A^{-1}B$
14: **if** test stopu NR spełniony np. (5.6) **then**
15: idź do 18
16: **end if**
17: **end for**
18: KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

5.2.5 Zabezpieczenia w elementach silnie nieliniowych

W rzeczywistych modelach elementów nieliniowych (szczególnie półprzewodnikowych) występują zakresy pracy, w których nachylenia charakterystyk elementów stają się bardzo małe. Powoduje to problemy numeryczne, ze względu na pojawianie się bardzo małych wartości w macierzy układu - (rozdział 12). Zabezpieczeniem jest wprowadzenie do macierzy układu możliwie dużych wartości, które nie zmieniają punktu pracy elementów nieliniowych.

Dla układów elektronicznych wprowadza się minimalną konduktancję $G_{min} = 10^{-10}$, która normalnie nie występuje w układach i nie zmienia punktów pracy. Konduktancje wprowadzane są równolegle do nieliniowych źródeł prądowych elementów nieliniowych na etapie tworzenia modelu elementu. Na rys. 5.1 przedstawiono wielkosygnałowy model diody z zaznaczoną konduktancją G_{min} dodaną w celu zabezpieczenia modelu [Pla13]. W przypadku tranzystorów FET zabezpieczeniem przed osobliwością macierzy (szczególnie, gdy pojemności są małe) jest wprowadzenie równoległych rezystancji między



Rys. 5.1: Model wielkosygnałowy diody z uwzględnieniem G_{min}.

źródłem a masą i między drenem a masą [D.82]. Rezystancje te zależą od numeru p iteracji NR zgodnie z 5.18.

$$R(p) = \begin{cases} 1k\Omega & dla \quad p = 1 \ldots 3 \\ 1G\Omega & dla \quad p > 3 \end{cases} \tag{5.18}$$

Omawiane podejścia są szeroko stosowane w programach symulacji układów elektronicznych.

5.3 Analiza charakterystyk stałoprądowych

Analiza charakterystyk stałoprądowych (*ang. Direct Current* - DC) jest metodą wyznaczania zależności odpowiedzi stałoprądowej sieci na zmianę zmiennej niezależnej. Zmienną niezależną może być:

1. wartość wymuszenia,
2. wartość rezystancji,
3. parametr modelu elementu.

Analizę charakterystyk sprowadza się do szeregu analiz punktu pracy układu z pewnym krokiem zmiany wartości zmiennej niezależnej. W każdym punkcie stosowany jest algorytm NR, który startuje z warunków początkowych (predykowanych, ekstrapolowanych). Wartości początkowe można otrzymać na różne sposoby:

- przez przyjęcie warunków startowych otrzymanych w poprzednim punkcie analizy DC (predykcja rzędu 0),
- przez wykonanie ekstrapolacji wielomianowej - takiej jak w analizie czasowej omówionej w rozdziale 7.

Algorytm 11 przedstawia zmiennokrokowy algorytm analizy charakterystyk DC układu. Dla uproszczenia w algorytmie nie uwzględniono rastra zmian parametru niezależnego

Algorytm 11 Algorytm zmiennokrokowy analizy charakterystyk DC układu.

Require: P_{min}, P_{max} ▷ zakres zmienności parametrów analizy
Require: Δ_P ▷ początkowy krok zmienności parametru analizy DC
Require: x_0 ▷ warunki startowe algorytmu NR
Require: $p = 0$ ▷ licznik iteracji NR
1: $x^{(0)} = x_0$
2: **for** $P = P_{min} \dots P_{max}$ **do** ▷ pętla wartości parametrów
3: **for** $p = 1 \dots p_{max}$ **do** ▷ pętla NR
4: oblicz wartości funkcji układowych $f(x^{(p-1)})$
5: ułóż równania sieci $Ax^{(p)} = B$ postaci (5.3)
6: rozwiąż $x^{(p)} = A^{-1}B$
7: **if** test stopu NR spełniony, np. (5.6) **then**
8: idź do 11
9: **end if**
10: **end for**
11: KONIEC PĘTLI NR ▷ rozwiązanie wyznaczone
12: **if** test stopu NR nie spełniony **then** ▷ brak zbieżności - skróć krok
13: $P = P - \Delta_P$ ▷ cofnij algorytm
14: $\Delta_P = \Delta_P/20$ ▷ skróć krok
15: idź do 20
16: **else**
17: Dobierz krok Δ_P na podstawie LTE ▷ rozdział 6.3.2
18: $x^{(0)} = x_0$ ▷ wyznacz nowe warunki początkowe
19: **end if**
20: $P = P + \Delta_P$ ▷ zwiększ wartość parametru
21: **end for**

(najczęściej zadawanego przez użytkownika), z jakim wykonywana jest analiza charakterystyk. W praktycznych zastosowaniach algorytm ten należy uzupełnić o wpadanie w punkty rastra, co jest zadaniem dosyć prostym. Kluczową kwestią w algorytmie jest dobór kroku analizy Δ_P, który jest dobierany na dwa sposoby:

- poprzez kontrolę lokalnego błędu obcięcia (*ang. LTE* - rozdział 6.3.2),
- na podstawie liczby iteracji NR.

5.4 Wyznaczanie wszystkich punktów pracy

Podstawowym problemem analizy układów nieliniowych jest wyznaczenie wielu, często lokalnych rozwiązań, gdyż nieliniowe równania algebraiczne mogą posiadać ich wiele. Jeżeli nie sformułuje się założeń dotyczących klasy rozpatrywanych funkcji, zagadnienie znalezienia wszystkich rozwiązań może być nierozwiązalne. Dlatego też w praktyce stosowane są metody dostosowane do klas funkcji spotykanych w praktyce. Metody wyznaczania wszystkich rozwiązań można podzielić na kilka podstawowych grup:

- podstawowe metody kontynuacji:
  - metoda kontynuacji z ograniczaniem przyrostów (rozdział 5.4.2)
  - metoda stopniowania wartości wymuszeń (rozdział 5.4.3)
  - metoda stopniowego zwiększania wpływu nieliniowości (rozdział 5.4.4)
- zaawansowane metody kontynuacji
  - metoda homotopii
  - metoda parametryzacji względem długości trajektorii
  - metoda przełączania parametryzacji
- metody odcinkowo-liniowe
  - hybrydowe równania sieci zawierające dwójniki rezystancyjne
  - metoda Chuy i Yinga
  - metoda Huanga i Liu
  - metody wstępnej lokalizacji obszaru rozwiązań
- metody ewolucyjne

Wadą podstawowych metod kontynuacji jest słaba odporność na gwałtowne zakrzywienia charakterystyk elementów nieliniowych. Dlatego też w ostatnich 10-ciu latach prowadzono intensywne prace nad metodami bardziej zaawansowanymi: metodami opartymi na równaniach różniczkowych i metodach homotopii. Umożliwiają one znalezienie wszystkich rozwiązań sieci. Przy pewnych założeniach są również zbieżne z dowolnego punktu startowego.

Metody ewolucyjne, umożliwiają wyznaczenie punktu startowego dla algorytmu NR, dokonując mutacji na zbiorach wektorów startowych. Jak wspomniano wcześniej rozwiązanie uzyskiwane przez algorytm NR zależny jest od punktu startowego, co umożliwia wyznaczenie wielu rozwiązań lokalnych.

5.4.1 Podstawowe metody kontynuacji startujące z punktu zerowego

Idea zastosowania metod kontynuacji do rozwiązywania nieliniowych równań algebraicznych polega na zastąpieniu zagadnienia algebraicznego zagadnieniem dynamicznym, którego rozwiązanie dąży do rozwiązania algebraicznego. Stosowane tutaj podejście rozwiązania równania

$$f(x) = 0$$

sprowadzane jest do rozwiązania równania różniczkowego

$$\frac{df(x(t))}{dt} = f(x(t))$$

Rozpoczynając rozwiązanie równania z warunkiem początkowym

$$f(x(0)) = f(0)$$

można znaleźć rozwiązanie równania dla stanu ustalonego, dla $t \to \infty$. Jedynym możliwym stanem ustalonym funkcji $f()$ jest zero, osiągane jeśli $x(t)_{t\to\infty} = x$ jest rozwiązaniem równania algebraicznego.

$$f(x(t))_{t\to\infty} = f(x) = 0$$

5.4.2 Metoda kontynuacji z ograniczaniem przyrostów

Metoda ta została zaprojektowana i wyjątkowo dobrze sprawdza się dla układów elektronicznych zawierających elementy bipolarne, w których występują nieliniowości typu wykładniczego. Działanie metody polega na ograniczaniu w kolejnych iteracjach przyrostu δ zmiennych niezależnych sterujących modelem x. Przyrost nie większy niż δ powoduje, że rozwiązanie $\hat{x}$ przesuwa się w kolejnych iteracjach NR w kierunku rozwiązania. Po osiągnięciu wartości mniejszej niż $x^{(p-1)} + \delta$ ograniczenie przestaje działać. Wzór 5.19 opisuje działanie algorytmu ograniczania.

$$x^{(k)} = max(x^{(p-1)} - \delta, min(x^{(p-1)} + \delta, \hat{x}^{(p)})) \tag{5.19}$$

gdzie: δ jest wartością zadaną przez użytkownika, $\hat{x}^{(p)}$ jest wartością rozwiązania (nie ograniczoną) w p-tej iteracji NR. Postać wzoru (5.19) wynika z tego, że rozwiązanie $\hat{x}$ może wypadać z lewej bądź z prawej strony rozwiązania z poprzedniej iteracji.

Wadą metody jest rosnąca liczba iteracji NR w przypadku wybrania zbyt małego δ i powolna zbieżność. Po ustaniu ograniczania zbieżność będzie jednak bardzo szybka. Istnieje możliwość przyśpieszenia zbieżności przez uzależnienie przyrostu $\delta(p)$ od numeru iteracji p (5.20).

$$\delta(p) = \frac{\delta}{k} \tag{5.20}$$

gdzie: $k = max(PLB, PUP - p)$ decyduje o sile ograniczenia, PLB i PUB są parametrami zadanymi przez użytkownika.

5.4.3 Metoda stopniowania wartości wymuszeń

Metoda ta polega na równomiernym zwiększaniu wartości stałych wymuszeń w sieci od 0 do wartości nominalnych. Dla sieci o ciągłych i monotonicznych nieliniowościach można przyjąć, że mała zmiana wymuszenia spowoduje małą zmianę odpowiedzi. Można skonstruować ciąg kolejnych rozwiązań NR dla kolejnych wartości wymuszeń s przemnożonych przez współczynnik skalujący $\alpha \in [0, 1]$. Metoda startuje z zerowych warunków początkowych ($\alpha \cdot s = 0$). Metoda stopniowania wartości wymuszeń opisuje algorytm 12.

Algorytm 12 Metoda stopniowania wymuszeń.

Require: $\alpha \in [0, 1]$ ▷ współczynnik skalujący wymuszenia
Require: $x_0 = 0$ ▷ warunki początkowe (startowe)
Require: $p = 0$ ▷ licznik iteracji NR
1: $x^{(0)} = 0$
2: **for** $\alpha = 0 \dots 1$ **do**
3: Rozwiąż $f(x) = \alpha \cdot s$ np. metodą NR NR p ▷ dobierz krok na podst. liczby iteracji
4: **if** test stopu NR niespełniony **then** ▷ brak zbieżności NR
5: $\Delta\alpha = \Delta\alpha/10$ ▷ skróć krok
6: **end if**
7: **if** $p > 8$ **then** ▷ zbieżności w dużej liczbie iteracji
8: $\Delta\alpha = \Delta\alpha/2$ ▷ skróć krok
9: **end if**
10: **if** $p \in [3 \cdots 8]$ **then**
11: pozostaw krok $\Delta\alpha$ ▷ krok bez zmian
12: **end if**
13: **if** $p < 3$ **then**
14: $\Delta\alpha = 1.5 \cdot \Delta\alpha$ ▷ wydłuż krok
15: **end if**
16: $\Delta\alpha = min(\Delta\alpha, 1)$ ▷ ogranicz krok do 1
17: $\alpha = \alpha + \Delta\alpha$ ▷ zwiększ wartość α
18: **end for**

5.4.4 Metoda stopniowego zwiększania wpływu nieliniowości

W metodzie stosowana jest modyfikacja równań zlinearyzowanych sieci iteracyjnej poprzez wprowadzenie niezerowej konduktancji $G^{(p)}$ pomiędzy każdym węzłem sieci oraz równolegle do gałęzi nieliniowych. Konduktancje takie wchodzą do macierzy na głównej przekątnej i przykrywają wartości zlinearyzowane (efekty nieliniowe). Powoduje to dobre uwarunkowanie macierzy. W algorytmie konstruuje się ciąg rozwiązań dla kolejnych wartości $G^{(p)}$. Wartości początkowe dla kolejnego rozwiązania metodą NR równań sieci przyjmuje się jako wartości obliczone w poprzednim kroku kontynuacji. Po każdym kroku kontynuacji wartość G zmniejszana jest γ-krotnie. Przy wyborze γ algorytm 13 bierze pod uwagę liczbę iteracji NR, w których uzyskano zbieżność. W przypadku braku zbieżności wartość γ jest zmniejszana dwukrotnie.

Algorytm 13 Metoda stopniowego zwiększania wpływu nieliniowości.

Require: $m = 0$ ▷ licznik kroku met. kontynuacji
Require: $\gamma = 10$ ▷ współczynnik skalujący
Require: $G_0 = 10^3$ ▷ warunki początkowe (startowe)
Require: $p = 0$ ▷ licznik iteracji NR
1: $x^{(0)} = 0$
2: **for** $m = 0 \dots m_{max}$ **do**
3: Rozwiąż $f(x, \gamma) = 0$ ▷ metoda NR ze wsp. skalującym nieliniowości
4: **if** test stopu NR nie spełniony **then** ▷ brak zbieżności
5: $\gamma = \gamma/2$ ▷ zmniejsz γ
6: **else** ▷ dobór γ na podst. liczby iteracji NR p
7: **if** $p > 8$ **then** ▷ zbieżności w dużej liczbie iteracji
8: $\gamma = \gamma/1.5$ ▷ zmniejsz γ
9: **end if**
10: **if** $p < 3$ **then**
11: $\gamma = 2\gamma$ ▷ zwiększ γ
12: **end if**
13: **end if**
14: **end for**

5.4.5 Metoda homotopii

Metoda homotopii należy do grupy metod kontynuacji. Metody te są od wielu lat rozwijane. Pojęcie homotopii zawiera definicja 5.4.1[Wikb, HVl+11, He03].

Definicja 5.4.1. *Homotopia - ciągłe przejście między dwoma przekształceniami ciągłymi przestrzeni topologicznych, tj. takie, za pomocą którego można w jednostce czasu w wyniku ciągłej deformacji z jednego przekształcenia otrzymać drugie.*

Od strony matematycznej idea metody polega na wprowadzeniu do rozwiązania $f(x) = 0$ ($f : R^n \to R^n$) funkcji

$$H(x, \lambda)$$

gdzie $H : R^{n+1} \to R^n$.
Funkcja H spełnia następujące założenia:

- H posiada rozwiązanie a,
- $H(x, 0) = 0$,
- $H(x, \lambda) = f(x)$ dla wybranych wartości λ.

Rozwiązaniem homotopii jest trajektoria $x(\lambda)$, wychodząca z punktu rozwiązania a i przechodząca przez wszystkie zera funkcji $f(x)$. Algorytm 14 przedstawia metodę homotopii.

Wyznaczenie trajektorii $x(\lambda)$ wymaga rozwiązania iteracyjnego równania (5.21) metodą NR.

$$H(x, \lambda_k) = 0 \tag{5.21}$$

Algorytm 14 Metoda homotopii.

Require: $m = 0$ ▷ licznik kroku met. kontynuacji
Require: $\lambda = 0$ ▷ współczynnik skalujący
1: $x^{(0)} = 0$ ▷ warunki startowe iteracji NR
2: **for** $m = 0 \dots m_{max}$ **do**
3: Rozwiąż $f(x, \lambda_m) = 0$ metodą NR
4: Dobierz λ_{m+1} ▷ λ w następnym kroku
5: Wyznacz $x^{(m+1)}$ ▷ warunki startowe iteracji NR w następnym kroku
6: **end for**

Wartości współczynnika λ powinny przybierać wartości rosnące zgodnie z zależnością (5.22).

$$0 = \lambda_1 < \lambda_2 < \lambda_3 \cdots < \lambda_k \tag{5.22}$$

Proces iteracji NR startuje z warunków początkowych, które mogą zostać wyznaczone na różne sposoby:

- wartości wyznaczone z punktu poprzedniego dla λ_{p-1},
- wartości wyznaczonych z przewidywania punktu startowego (predykcja) - rozdział 6.3.1.

Dobór kolejnych wartość współczynnika λ może odbywać się na podstawie:

- liczby iteracji NR potrzebnych do uzyskania zbieżności w poprzednim kroku (rozdział 6.3.3),
- wartości wyznaczonych z przewidywania punktu startowego (predykcja) - rozdział 6.3.1 (algorytm taki jak w analizie czasowej).

Metoda wyznaczania całej trajektorii rozwiązania jest skomplikowana. Jest to spowodowane samą naturą trajektorii $x(\lambda)$, która może posiadać rozwidlenia (bifurkacje [Wik16][5]) w punktach osobliwych. Macierz pochodnych $\delta H/\delta x$ w rozwidleniach staje się osobliwa i nie ma możliwości znalezienia wszystkich gałęzi rozwidlenia.

Znalezienie rozwiązania w danej gałęzi możliwe jest po wykonaniu iteracji NR z punktu startowego b dobranego tak aby leżał poza bifurkacją. Umożliwia to znalezienie tylko jednej gałęzi i jednego rozwiązania. Wymaga to jednak wprowadzenia założeń dotyczących funkcji homotopii.

Istnieją metody znajdowania wszystkich odgałęzień, ale dla tylko pewnych wąskich klas sieci. Można znaleźć wtedy punkt startowy b obliczeń dla danej gałęzi (dla przejścia przez gałąź).

W praktyce stosuje się wiele homotopii:

- Homotopia zmiennego pobudzenia (*ang. variable stimulus*),

$$H(x, \lambda) = \underbrace{(1 - \lambda)G(x - a)}_{A} + \underbrace{f(x, \lambda)}_{B} \tag{5.23}$$

[5]Bifurkacja to skokowa zmiana własności jakościowych lub topologicznych układu przy drobnej ciągłej zmianie jego parametrów. Np. równanie kwadratowe $ax^2 + bx + c = 0$ posiada bifurkację dla $a = 0$ - zmienia się wtedy w równanie liniowe postaci $bx + c = 0$.

dla $\lambda \in [0,1]$. λ jest mnożnikiem zmiennych niezależnych gałęzi nieliniowych B, oraz składnika A zależnego od rozwiązania początkowego a umożliwiającego ominięcie bifurkacji. W sieciach elektrycznych, których równania układane są metodą ZMPW (rozdział 12.1), G stanowi dodatkową gałąź dołączoną do każdego węzła sieci. Gałąź taka składa się z połączonych w szereg konduktancji i źródła napięciowego. Dla $\lambda = 1$ składnik A znika i rozwiązanie zbliża się do rozwiązania sieci.

W praktyce stosuje się także homotopię (5.24), w której przez λ mnoży się wszystkie zmienne sieciowe f.

$$H(x,\lambda) = (1-\lambda)G(x-a) + f(\lambda x) \tag{5.24}$$

- Homotopia zmiennego wzmocnienia (*ang. variable gain*), dla której λ skaluje współczynniki wzmocnienia tranzystorów.

Obecnie rozwijane metody homotopii można znaleźć w literaturze [AESD12, Ami12, A+02, WMM02, RM06, BA09, He09, VL12, HVL+12a, HVL+12b, LTS98, AD+99]

5.4.6 Homotopia parametryzowana względem długości trajektorii rozwiązania

Metoda parametryzacji względem długości trajektorii [UA84] polega na wprowadzeniu λ jako zmiennej parametryzującej wartość wymuszeń (5.25).

$$f(\lambda x) = \lambda s \tag{5.25}$$

Załóżmy, że trajektoria rozwiązania układu (x,λ) jest jednobieżna i można ją opisać równaniami parametrycznymi ze zmienną s. Zmienna s jest długością krzywej rozpoczynającą się w pewnym punkcie początkowym. Długość krzywej można otrzymać z całkowania różniczki łuku ds (5.26).

$$(ds)^2 = (d\lambda)^2 + \sum_i (dx_i)^2 \tag{5.26}$$

Daje to układ równań algebraiczno-różniczkowych postaci (5.27)

$$\begin{array}{rcl} f(x,\lambda) & = & 0 \\ (ds)^2 & = & (d\lambda)^2 + \sum_i (\frac{dx_i}{ds})^2 = 1 \end{array} \tag{5.27}$$

Układ (5.27) można rozwiązać przez dyskretyzację równań w każdym punkcie s_n za pomocą schematu różnicowego (6.2), otrzymujemy (5.28).

$$\dot{\lambda}_n = \gamma\lambda_n + d_\lambda \tag{5.28}$$

Po podstawieniach otrzymamy układ równań algebraicznych nieliniowych (5.29), który rozwiązuje się metodą NR (rozdział 5.2).

$$\begin{array}{rcl} f(x,\lambda_n) & = & 0 \\ (\gamma\lambda_n + d_\lambda)^2 + \sum_i (\gamma\lambda_n + d_\lambda)^2 & = & 1 \end{array} \tag{5.29}$$

Start metody NR odbywa się tak, jak w klasycznej metodzie czasowej - wartości startowe przewidywane są za pomocą otwartego SR (rozdział 6.3.1). Tak jak w analizie czasowej krytyczny jest odpowiedni dobór kroku, który można zrealizować na podstawie:

- miary krzywizny łuku,
- lokalnego błędu obcięcia (rozdział 6.3.3).

Zmiana wartości λ prowadzona jest do momentu osiągnięcia przez wymuszenie s wartości nominalnej.

Metoda umożliwia znalezienie jednej trajektorii, jeżeli punkt początkowy zostanie wybrany poza punktem osobliwym. W przypadku sieci rezystancyjnych można znaleźć liczbę gałęzi i punkty początkowe poszczególnych gałęzi.

W pracy [L.V87] podano uogólnienie metody. Idea polegała na dodaniu do zmiennych układu zmiennej parametryzującej λ jako $n+1$ zmiennej $H(x, \lambda)$. Równanie (5.30) trzeba zróżniczkować.

$$H(x, \lambda) = 0 \tag{5.30}$$

Po zróżniczkowaniu otrzymamy równanie (5.31).

$$\frac{\delta H(x, \lambda)}{\delta x} \frac{dx}{d\lambda} = 0 \tag{5.31}$$

Idea metody polega na doborze zmiennych x, które są parametrem metody kontynuacji. W przypadku dojścia do punktu bifurkacji macierz H staje się osobliwa. Należy wtedy wybrać spośród x inną zmienną jako λ, a x będący dotychczasowym parametrem kontynuacji zaliczyć do składowych x.

5.4.7 Metody odcinkowo-liniowe

Metody odcinkowo-liniowe *ang. piecewise linear (PWL)* bardzo silnie rozwijały się w latach 80-tych i 90-tych XX wieku. Zaletą metod jest możliwość opracowania algorytmów, które umożliwiają znalezienie wszystkich rozwiązań (globalnych). Podstawą metod jest modelowanie wszystkich elementów nieliniowych sieci za pomocą modeli odcinkowo-liniowych. Funkcje nieliniowe zastępuje się ich przybliżeniem odcinkowo-liniowym, złożonym z m (wielu) odcinków. Tak więc funkcja odcinkowo-liniowa składa się z m odcinków i m warunków określających przedziały obowiązywania odcinków.

$$y = a_j x + b_j, \ dla \ x_a \leq x < x_b, \ j = 1, \ldots, m \tag{5.32}$$

Dla sieci, w których występuje wiele elementów nieliniowych istnieje problem reprezentacji dużej ilości odcinkowo-liniowej funkcji (prostopadłościennych regionów). Sposoby reprezentacji funkcji odcinkowo-liniowych i regionów np. metodą Katzenelsona można znaleźć w [Lin81]. Istnieje także bardzo efektywna metoda reprezentacji w postaci sumy funkcji przesuniętych o segment wynikający z punktów sklejenia odcinków (5.33).

$$y = px + q + \sum_{k=1}^{m-1} r_k |x - t_k| \tag{5.33}$$

gdzie: t_k są kolejnymi punktami sklejenia odcinków, p, q, r_k są współczynnikami określającymi początek odcinka i jego nachylenie.

Analiza sieci Algorytm analizy sieci odcinkowo-liniowej wymaga tzw. równań hybrydowych. Z założenia wszystkie elementy sieci są konduktancjami $i = f(u)$ lub rezystancjami $u = f(i)$. Dla źródeł nieliniowych stosuje się modele zastępcze złożone z rezystancji, konduktancji nieliniowej oraz liniowych źródeł sterowanych. W ogólności wielowrotnik opisuje się macierzą hybrydową H. Równanie hybrydowe przyjmuje postać.

$$f(x) + Hx + s = 0 \tag{5.34}$$

gdzie s jest wektorem wymuszeń wewnątrz wielowrotnika, a $f(x)$ jest funkcją wektorową opisującą elementy nieliniowe.

Macierz hybrydową można wyznaczyć poprzez wykonanie trzech analizy sieci [Lin81]. Analiza stałoprądowa wymaga jednokrotnego obliczenia parametrów zastępczych H oraz s. Poszukiwanie rozwiązań sieci można zrealizować poprzez przeszukiwanie kombinacji regionów, co jest bardzo nakładochłonne i prymitywne. Lepszym rozwiązaniem jest śledzenie trajektorii odcinkowo-liniowej [Lin81]. Opracowano także szereg udoskonaleń metod opisanych w [CL82] oraz [HQ89] (analiza smug - zbiorów regionów zawierających rozwiązanie). Istnieją także metody wstępnej lokalizacji obszaru rozwiązań przez wyznaczanie obszarów zawierających rozwiązania [TM91] dla sieci nieliniowych.

Osoby zainteresowane opisanymi powyżej metodami powinni przestudiować zamieszczoną literaturę.

5.5 Metody ewolucyjne

Idea metod ewolucyjnych wzięła się z obserwacji środowiska naturalnego i metod doboru osobników. W środowisku istnieje pewna populacja osobników. Każdy z osobników ma przypisany pewien zbiór informacji stanowiących jego genotyp, a będących podstawą do utworzenia fenotypu. Fenotyp to zbiór cech podlegających ocenie funkcji przystosowania modelującej środowisko. Genotyp opisuje proponowane rozwiązanie problemu, a funkcja przystosowania ocenia, jak dobre jest to rozwiązanie.

Genotyp składa się z chromosomów, gdzie zakodowany jest fenotyp i ewentualnie pewne informacje pomocnicze dla algorytmu genetycznego. Chromosom składa się z genów.

Wspólnymi cechami algorytmów ewolucyjnych, odróżniającymi je od innych, tradycyjnych metod optymalizacji, są:

- stosowanie operatorów genetycznych, które dostosowane są do postaci rozwiązań,
- przetwarzanie populacji rozwiązań, prowadzące do równoległego przeszukiwania przestrzeni rozwiązań z różnych punktów, w celu ukierunkowania procesu przeszukiwania wystarczającą informacją jest jakość aktualnych rozwiązań,
- celowe wprowadzenie elementów losowych.

Działanie algorytmu ewolucyjnego przedstawiono w postaci algorytmu 15.

Działanie algorytmu genetycznego obejmuje kilka zagadnień potrzebnych do ustalenia:

- ustalenie genomu jako reprezentanta wyniku,
- ustalenie funkcji przystosowania/dopasowania,
- ustalenie operatorów przeszukiwania.

Algorytm 15 Algorytm ewolucyjny.

1: Losuj pewną populację początkową
2: Poddaj populację ocenie ▷ selekcja
3: Najlepiej przystosowane osobniki biorą udział w procesie reprodukcji
4: Zastosuj operatory ewolucyjne do genotypów wybranych osobników
5: - wykonaj krzyżowanie ▷ kojarzenie poprzez złączanie genotypów rodziców
6: - dokonaj mutacji ▷ wprowadzenie drobnych losowych zmian
7: Rodzi się kolejne pokolenie
8: Utrzymuj stałą liczbę osobników w populacji
9: - powiel najlepsze osobniki w populacji (według funkcji oceniającej fenotyp)
10: - usuń najsłabsze osobniki w populacji
11: **if** nie znaleziono dostatecznie dobrego rozwiązania **then**
12: idź do 2
13: **end if**
14: Wybierz najlepszego osobnika z populacji
15: KONIEC

Kodowanie Kodowanie jest bardzo istotnym etapem projektowania algorytmu. Sposób zakodowania w chromosomie informacji o proponowanym rozwiązaniu wydatnie wpływa na szybkość i jakość znajdowanych wyników. Przyczyną takiego zjawiska jest wpływ kodowania na sposób w jaki przeszukiwana jest przestrzeń rozwiązań [6]. Najczęściej stosowane kodowania chromosomu:

- wektorem genów, z których każdy z nich może być jedno- lub wielobitową liczbą całkowitą,
- liczbą rzeczywistą za pomocą drzewiastych struktur danych.

Funkcja przystosowania Proces wyboru osobników poddanych ocenie według określonych kryteriów. Kryteria te zapisuje się w postaci funkcji oceny albo inaczej funkcji przystosowania. Algorytm genetyczny dąży zwykle do jej minimalizacji (maksymalizacji). Jako kryterium często przyjmowana jest funkcja celu rozważanego problemu optymalizacji.

Funkcja oceny Funkcja oceny to miara jakości dowolnego osobnika (fenotypu, rozwiązania) w populacji. Dla każdego osobnika jest ona obliczana na podstawie pewnego modelu rozwiązywanego problemu. Załóżmy dla przykładu, że chcemy zaprojektować obwód elektryczny o pewnej charakterystyce. Funkcja oceny będzie premiowała rozwiązania najbardziej zbliżonej do tej charakterystyki, zbudowane z najmniejszej liczby elementów. W procesie selekcji faworyzowane będą najlepiej przystosowane osobniki. One staną się *rodzicami* dla następnej populacji.

Metody selekcji Istnieje wiele metod selekcji. Dla przykładu można przedstawić tzw. metodę ruletki. Budujemy wirtualnie koło, którego wycinki odpowiadają poszczególnym osobnikom. Im lepszy osobnik, tym większy wycinek koła zajmuje. Rozmiar wycinków może zależeć od wartości funkcji oceny, jeśli wysoka wartość oceny oznacza wysokie

[6] Złe kodowanie może spowodować, że nigdy nie zostanie przeszukany fragment przestrzeni, w którym znajdują się najlepsze rozwiązania.

przystosowanie. W takim układzie prawdopodobieństwo, że lepszy osobnik zostanie wybrany jako rodzic, jest większe. Niestety ewolucja przy takim algorytmie z każdym krokiem zwalnia. Jeżeli osobniki są podobne, to każdy dostaje równy wycinek koła fortuny i presja selekcyjna spada. Algorytm słabiej rozróżnia osobniki dobre od słabszych.

Pozbawiona tej wady jest metoda rankingowa. Obliczamy dla każdego osobnika funkcję oceny i ustawiamy je w szeregu najlepszy-najgorszy. Pierwsi na liście dostają prawo do rozmnażania, a reszta usuwana jest z populacji. Wadą metody jest niewrażliwość na różnice pomiędzy kolejnymi osobnikami w kolejce. Może się okazać, że sąsiadujące na liście rozwiązania mają różne wartości funkcji oceny, ale dostają prawie taką samą ilość potomstwa.

Istnieją także metody selekcji wielokryterialnej. Tworzymy kilka różnych funkcji oceny (oceniających pewne wybrane cechy osobników osobno). Dla przykładu osobniki mogą być ułożone nie w jednym, ale w kilku szeregach najlepszy-najgorszy, a proces selekcji jest bardziej złożony.

Jak widać, selekcja daje większe prawdopodobieństwo reprodukcji osobnikom o dużym przystosowaniu, więc kolejne pokolenia są coraz lepiej przystosowane. Spada jednak różnorodność genotypu populacji - populacja z czasem zostaje zmonopolizowana przez nieznacznie różniące się (lub wręcz identyczne) odmiany tego samego osobnika. Objawia się to zbieżnością kolejnych, najlepszych rozwiązań do pewnej granicy. Czasami zbieżność jest przedwczesna, a ewolucja utyka i uzyskane rozwiązania przedstawiają pewne ekstrema lokalne. Mogą być one dalekie od oczekiwanych rozwiązań globalnych, czyli tych najlepszych w całej przeszukiwanej przestrzeni. Częściowym rozwiązaniem tego problemu jest losowa mutacja genotypu osobników.

Operatory przeszukiwania W każdym cyklu (każde pokolenie) poddawane są *obróbce* za pomocą operatorów ewolucyjnych. Celem tego etapu jest wygenerowanie nowego pokolenia, na podstawie poprzedniego, które być może będzie lepiej dopasowane do założonego środowiska.

Operator krzyżowania ma za zadanie łączyć w różnych kombinacjach cechy pochodzące z różnych osobników populacji, zaś operator mutacji ma za zadanie zwiększać różnorodność tych osobników. O przynależności dowolnego algorytmu do klasy algorytmów genetycznych decyduje głównie zastosowanie operatora krzyżowania i praca z całymi populacjami osobników (idea łączenia w przypadkowy sposób genotypów nieprzypadkowo wybranych osobników). Równie ważny jest operator mutacji. Jeśli krzyżowanie traktować jako sposób eksploatacji przestrzeni rozwiązań, to mutacja jest sposobem na jej eksplorację. Może się jednak zdarzyć, że dla niektórych zagadnień jej zastosowanie nie jest krytyczne.

Krzyżowanie Krzyżowanie polega na połączeniu niektórych (wybierane losowo) genotypów w jeden. Kojarzenie ma sprawić, że potomek dwóch osobników rodzicielskich ma zespół cech, który jest kombinacją ich cech (może się zdarzyć, że tych najlepszych).

Sposób krzyżowania jest zależny od kodowania chromosomów i specyfiki problemu. Jednak można wskazać kilka standardowych metod krzyżowania:

- rozcięcie dwóch chromosomów i stworzenie nowego poprzez sklejenie lewej części jednego rodzica z prawą częścią drugiego rodzica (dla chromosomów z kodowaniem binarnym i całkowitoliczbowym),

- stosowanie operacji logicznych (kodowanie binarne),

- obliczenie wartości średniej genów (kodowanie liczbami rzeczywistymi).

Mutacja Mutacja wprowadza do genotypu losowe zmiany. Jej zadaniem jest wprowadzanie różnorodności w populacji, czyli zapobieganie (przynajmniej częściowe) przedwczesnej zbieżności algorytmu. Mutacja zachodzi z pewnym przyjętym prawdopodobieństwem - zazwyczaj rzędu 1%. Jest ono niskie, ponieważ zbyt silna mutacja przynosi efekt odwrotny do zamierzonego: zamiast subtelnie różnicować dobre rozwiązania - niszczy je. Stąd w procesie ewolucji mutacja ma znaczenie drugorzędne, szczególnie w przypadku długich chromosomów.

W przypadku chromosomów kodowanych binarnie losuje się zazwyczaj dwa geny i zamienia się je miejscami bądź np. neguje pewien wylosowany gen. W przypadku genotypów zakodowanych liczbami całkowitymi stosuje się permutacje. W przypadku genotypów zakodowanych liczbami rzeczywistymi wprowadza się do przypadkowych genów losowe zmiany o danym rozkładzie - najczęściej normalnym.

5.5.1 Zastosowanie metod ewolucyjnych w analizie punktu pracy

Metody ewolucyjne w analizie punktu pracy rozwijane są od końca lat 80-tych XX wieku. Tak jak w metodach omówionych wcześniej, do znajdowania rozwiązań równań stosuje się metodę Newtona-Raphsona (NR-rozdział 5.2), która startuje z punktu początkowego x. Metoda NR (rozdział 5.2) trzy podstawowe wady:

- w każdej iteracji należy obliczyć Jakobian,
- metoda możę być niezbieżna lub może wpadać w oscylacje,
- znalezione rozwiązanie zależy od punktu startowego w przypadku układów o wielu rozwiązaniach.

Do obliczenia punktu pracy można zastosować np. analizę czasową - jako analizę stanu ustalonego po zaniku pewnego stanu przejściowego (rozdział 5.4.1) oraz algorytmy ewolucyjne (AE)[ZMY00].

Algorytmu ewolucyjne mają szereg zalet w stosunku do zwykłego algorytmu NR:

- lepszą zbieżność,
- możliwość znalezienia wielu rozwiązań,
- możliwość wykorzystania obliczeń równoległych do poszukiwania rozwiązań na zestawach wektorów startowych.

Podstawowym zadaniem wszystkich AE jest znalezienie wartości startowych dla algorytmu NR, które mogą prowadzić do różnych rozwiązań. Jest to podstawowy problem w poszukiwaniu wielu rozwiązań.

W algorytmach ewolucyjnych wektor rozwiązań x_l^p w p-tej iteracji można potraktować jako l-tego członka pewnej populacji wartości (kandydata) rozwiązań. W takim przypadku funkcja $f(x) \in R^n$ reprezentuje równania charakterystyczne elementów nieliniowych układu. W algorytmach ewolucyjnych funkcja celu f [7] opisuje dopasowanie

[7] Pojęcie *funkcji celu* (FC) używane jest w zagadnieniach optymalizacji. W zależności od parametrów opisuje ona przebieg pewnego procesu jak pewna liczba. Funkcję celu f można minimalizować lub maksymalizować. W omawianym przypadku funkcję f należy minimalizować

do rozwiązania pewnego wektora x (kandydata). Dobór wektorów (kandydatów) jest zagadnieniem kluczowym.

W praktyce spotyka się wiele rodzajów algorytmów ewolucyjnych [Rec65, Sch65, SP95, Fog00, Gol89]:

- *ang. differential evolution*,
- *ang. evolutionary programming*,
- *ang. genetic programming*.

Wszystkie algorytmy ewolucyjne używają podobnych technik, lecz różnią się sposobem reprezentacji wektora kandydatów wartości rozwiązania. Metody ewolucyjne posługują się pojęciami genotypu i fenotypu. Głównym problemem jest dobranie odpowiedniego wektora wartości startowych metody NR. Na gruncie metod ewolucyjnych należy taki wektor skonstruować. W przypadku analizy DC, wektor kandydatów x powinien mieć (posługując się terminologią stosowaną w metodach ewolucyjnych):

- genotyp[8], który stanowi wektor binarny wartości całkowitych,
- fenotyp[9] opisany wektorem wartości rzeczywistych określających wartości sygnałów w węzłach (*ang. nodes*) - np. napięć (rozdział 1.3).

Algorytm genetyczny [Gol89] operuje na wektorze binarnym wykorzystując bity mutacji oraz bity opisu do odpowiedniego mapowania wartości fenotypu (wartości opisujących sygnały w węzłach). Inaczej jest w przypadku strategii ewolucyjnych i podejścia różnicowego, które operują tylko na poziomie fenotypu (wartości rzeczywistych sygnałów).

Podejście genetyczne operuje na zbiorze populacji wektorów kandydatów x. Wykorzystuje się obliczenia równoległe prowadzone na populacji x zamiast przeszukiwania sekwencyjnego jak w metodzie NR.

Można zdefiniować pojedynczy element l-tej populacji (5.35).

$$P^l = (x_1^l, \ldots, x_N^l) \tag{5.35}$$

gdzie: x_i^k reprezentuje wektor kandydatów wartości rzeczywistych sygnałów (np. napięć węzłowych) dla $i = 1, \ldots, L$ w l-tej generacji, L jest rozmiarem populacji. Liczba generacji jest najczęściej ograniczona do L_{MAX}.

Należy zdefiniować wektor celu (5.36), który będzie minimalizowany.

$$y_i^l = (y_{i,1}^l, \ldots, y_{i,L}^l)^T \tag{5.36}$$

Ze względu na fakt, że wektor x może zawierać tylko napięcia węzłowe, wektor y będzie zawierał wartości prądów w węzłach. Zgodnie z prawem Kirchoffa [10] w przypadku populacji x zbliżającej się do rozwiązania, y będzie dążyć do 0. Po osiągnięciu

[8] Genotyp (ród, pochodzenie + odbicie) - zespół genów danego osobnika warunkujących jego właściwości dziedziczne. Jest to sparowany układ alleli.

[9] Fenotyp (gr. phainomai - przejawiać; typos - wzór, norma) - zespół cech organizmu, włączając w to nie tylko morfologię, lecz również np. właściwości fizjologiczne, płodność, zachowanie się, ekologię, cykl życiowy, zmiany biologiczne, wpływ środowiska na organizm. Fenotyp jest ściśle związany z genotypem, bowiem to właśnie oddziaływanie między genotypem a środowiskiem daje fenotyp. Dlatego ten sam genotyp może dać różne fenotypy w różnych środowiskach (tzw. plastyczność fenotypowa), lub odwrotnie - mimo odmiennych genotypów uzyskać podobny fenotyp.

[10] Prawo Kirchoffa - suma prądów w węźle jest równa zero $\sum i = 0$.

rozwiązania osiągnie 0. Tak więc zadanie sprowadza się do minimalizacji y jako najlepszego dopasowania do rozwiązania. Każdemu rozwiązaniu y odpowiada wektor x. Zapamiętując wektory kandydatów x, dla których otrzymywane są coraz lepsze dopasowania y_i w kolejnych generacjach, można minimalizować funkcję celu f (zbliżać się do rozwiązania).

Strategie ewolucyjne

Strategie ewolucyjne zostały niezależnie rozwinięte w 1965 przez Rechenberga [Rec65] i Schwefela [Sch65]. Należą one do grupy metod probabilistycznych, heurystycznych[11], wykorzystujących techniki optymalizacji.

W technikach ewolucyjnych nie stosuje się przełączania lub inwersji. AE wykorzystują dyskretną rekombinację, gdzie dwa wektory (rodzice) są rekombinowane przez wymieszanie elementów wektorów x. Elementy są dzielone na część zwaną rodzicami oraz część zwaną potomstwem. Generacja nowego wektora kończy się przyjęciem najlepszych wartości z wektora rodziców oraz potomstwa jako rodziców następnej generacji. W ogólności algorytm ES używa operatora mutacji z algorytmem adaptacyjnym i operatorem rekombinacji.

AE używają także operatora selekcji (*ang. crossover*). Jest to rodzaj selekcji dyskretnej, dla której rekombinacji podlegają oboje rodziców. Zamieniane są losowo wybrane części wektorów. Populacja wektorów podzielona jest na rodziców i dzieci. Następnie wybierane są najlepsze wektory ze zbioru rodziców i zbioru potomków. Wektory te traktowane są jako rodzice dla następnego pokolenia. Ogólnie można stwierdzić, że w strategiach ewolucyjnych stosowane są tylko operatory mutacji. Algorytmy adaptacyjne (ESA) używają także operatorów rekombinacji.

Selekcja turniejowa

Metoda ta jest alternatywną metodą selekcji. Na początku nowej generacji, porównuje się parami każdego członka populacji z członkiem wybranym losowo. Oszacowuje się dopasowanie, czy jest lepsze niż obecne. W przypadku, kiedy wymagany jest rodzic, jest on wybierany losowo z populacji i losowo akceptowany, bądź odrzucany na podstawie dopasowania.

Ewolucja różnicowa

Storn i Price opisali adaptacyjne algorytmy ewolucyjne, które są proste i wystarczająco efektywne. Nazywane są algorytmami ewolucji różnicowej *ang. Differential Evolution* (DE) [SP95]. Idea polega na generacji dla każdego wektora kandydata w populacji wektora pośredniego. W pracy [Sto96] zaproponowano kilka podejść generacji wektora pośredniego.

Pierwsze podejście (DE1) polega na wyborze pośredniego wektora kandydata zgodnie z (5.37).

$$v_i^l = x_{r1}^l + \tau(x_{r2}^l + x_{r3}^l) \tag{5.37}$$

[11] Heurystyka (gr. heuresis - odnaleźć, odkryć, heureka - znalazłem) - w informatyce metoda znajdowania rozwiązań, dla której nie ma gwarancji znalezienia rozwiązania optymalnego, a często nawet prawidłowego.

gdzie: τ jest dodatnią wartością rzeczywistą - współczynnikiem skalowania ustalanym przez użytkownika, r_1, r_2, r_3 są wybieranymi losowo różnymi wartościami całkowitymi z przedziału $[1, N]$.

Wektor pośredni v_i^l oraz x_i^l są wykorzystywane w procedurze doboru do wygenerowania nowego potomka $\dot{x}_i^l$. Jeżeli nowy potomek jest lepiej dopasowany, to pozostaje. W przeciwnym przypadku jest usuwany. Oznacza to, że w metodzie nie jest potrzebny zbiór potomków.

Drugie podejście polega na innym sposobie generacji wektora pośredniego zgodnie z (5.38).

$$v_i^l = x_i^l + \tau'(x_{best}^l + x_i^l) + \tau(x_{r1}^l + x_{r2}^l) \tag{5.38}$$

W podejściu tym wykorzystywane są tylko dwie losowo wybierane wartości r_1 i r_2 oraz dwa dodatnie współczynniki skalujące zadane przez użytkownika τ i τ'.

Hybrydowe algorytmy ewolucyjne

Główną różnicą w stosunku do algorytmu DE inicjalizującego algorytm Newtona-Raphsona (DENR) [CU76] jest to, że po kilku generacjach algorytm EA zaprzestaje zmian populacji. Zamiast tego stara się poprawić populację wektorów aż do uzyskania rozwiązania. Punkt zmiany wybierany jest w chwili, gdy pojawi się gradient(lepsze dopasowanie). Następuje zmiana na szybszą metodę optymalizacji - poszukiwania rozwiązania. W omawianym przypadku jest do punkt startowy dla metody NR.

W punkcie zmiany algorytmu należy zastosować algorytm filtrowania zwielokrotnionych rozwiązań. Każdy z wektorów kandydatów jest wykorzystywany jako punkt startowy dla algorytmu NR.

Zaletą algorytmu jest wykorzystanie szybkości metody NR oraz algorytmu ewolucyjnego do wyboru wartości startowych dla metody NR. Omawiany algorytm jest szybki i dokładny.

5.5.2 Podsumowanie właściwości algorytmów ewolucyjnych

Algorytmy DE1, DE2 i ES są zwykle najlepszymi algorytmami automatycznego znajdowania wielu rozwiązań. Algorytm adaptacyjny (ESA) jest najgorszym z dobrych algorytmów znajdowania wielu rozwiązań, nawet w przypadku stosowania restartu algorytmu i zmiany parametrów.

Najlepszym algorytmem ewolucyjnym do znajdowania wielu rozwiązań jest algorytm hybrydowy (DENR) - algorytm ewolucyjny sprzężony z algorytmem NR.

5.6 Metody relaksacyjne w analizie DC

Metoda iteracyjnego rozwiązywania równań może zostać z powodzeniem użyta w analizie stałoprądowej DC sieci nieliniowych. Standardowym sposobem rozwiązywania równań nieliniowych jest zastosowanie metody NR. Sieć nieliniowa zostaje przekształcona w konduktancyjną sieć liniową, której równania rozwiązywane są metodami tradycyjnymi (np. rozkładu LU). Zamiast metod tradycyjnych można użyć metod iteracyjnego rozwiązywania równań rozdzielonych (np. GS), jeśli zostaną spełnione warunki zbieżności dla tych metod (rozdział 4.2.1, 4.2.2). Ze względu na trudności ze zbieżnością, stosowane są liczne modyfikacje metod relaksacyjnych przystosowane do konkretnych zastosowań.

5.6.1 Technika pojemności wirtualnych

Metoda pojemności wirtualnych[HA89] (*ang. Virtual State relaxation technique*) jest jedną z metod zapewniających zbieżność metody GS (rozdział 4.2.2) przez zapewnienie dominacji głównej przekątnej macierzy układu. Jeżeli do każdego węzła sieci liniowej (po linearyzacji) dołączona zostanie pojemność wirtualna pomiędzy każdym węzłem układu i węzłem odniesienia (masa), to na głównej przekątnej macierzy pojawią się dodatkowe wartości związane z tymi pojemnościami (rozdział 12.1). Załóżmy, że sieć opisana jest równaniem (1.1). Jeżeli zastosujemy metodę NR, to otrzymamy równanie iteracji (5.39).

$$\begin{aligned} J(x^{(p)})x &= -f(x^{(p)}) \\ x^{(p+1)} &= x^{(p)} + x \end{aligned} \tag{5.39}$$

gdzie: $J(x^{(p)})$ jest Jakobianem o wymiarach $N \times N$ w p-tej iteracji NR. Równanie (5.39) można dla x zapisać w postaci (5.40).

$$Ax = b \tag{5.40}$$

Jeżeli do każdego węzła układu dołączymy pojemność wirtualną (co odpowiada wprowadzeniu niezerowych wartości tylko na przekątną macierzy A), to równanie (5.40) można przekształcić do postaci (5.41).

$$Ax + C\dot{x} = b \tag{5.41}$$

gdzie: C jest macierzą diagonalną pojemności wirtualnych. Jeżeli założymy, że czas t dąży do nieskończoności, to można otrzymać rozwiązanie równania (5.40). Równanie (5.41) może być rozwiązane numerycznie przez zastosowanie formuły całkowania numerycznego Eulera. Wtedy (5.41) można zapisać jako (5.42).

$$\left(\frac{C}{h} + A\right) x^{t+1} = b + \frac{C}{h}x^{t} \tag{5.42}$$

Macierz $\left(\frac{C}{h} + A\right)$ jest diagonalnie dominująca dla odpowiednio małego kroku czasowego h, co zapewnia zbieżność metody. Jeżeli do rozwiązania równania (5.42) zastosujemy metodę GS, to otrzymamy równanie iteracyjne postaci (5.43).

$$(C + hD)x^{t+1,k+1} = Cx^{t} + hb - h(Lx^{t+1,k+1} + Ux^{t+1,k}) \tag{5.43}$$

Jeżeli tylko jedna iteracja GS wykonywana jest w każdym kroku czasowym i wartość początkowa iteracji GS $x^{k=0} = x^{t}$ (x^{t} jest predykowana ze wzoru Eulera), to równanie (5.43) można przepisać do postaci (5.44), zastępując dla uproszczenia x^{t+1} przez x.

$$(C + hD)x^{k+1} = Cx^{k} + hb - h(Lx^{k+1} + Ux^{k}) \tag{5.44}$$

Aby zapewnić silną dominację głównej przekątnej macierzy $(C+hA)$, współczynniki c_i przyjmowane są według (5.45).

$$c_i = \begin{cases} 0 & d_{ii} \geq \sum_{j\neq i}^{N} \mid a_{ij} \mid \\ h_i \sum_{j\neq i}^{N} \mid a_{ij} \mid -h_i d_{ii} & d_{ii} \geq \sum_{j\neq i}^{N} \mid a_{ij} \mid \end{cases} \tag{5.45}$$

5.6.2 Metoda podrelaksacyjna

Metoda podrelaksacyjna [HA89] (*ang. Under Relaxation technique*) jest rozwinięciem techniki VSR [HA89]. Zastosowano tu kombinację techniki nadrelaksacyjnej SOR (roz. 4.2.4) i technikę wirtualnych pojemności (roz. 5.6.1) dla sieci liniowej RC o równaniu (5.46).

$$C\dot{x} + Gx = b \tag{5.46}$$

gdzie: C - macierz diagonalna pojemności dołączonych do masy, G - macierz konduktancji, b - wektor wymuszeń.
Jeżeli do rozwiązania użyjemy schematu różnicowego otwartego, to otrzymamy

$$Ax = b \tag{5.47}$$

gdzie:

$$\begin{aligned} A &= C + \Delta t G \\ b &= Cx^t \\ x &= x^{t+1} \end{aligned} \tag{5.48}$$

Do rozwiązania równania (5.47) zastosujemy metodę SOR. Otrzymamy rozwiązanie w postaci (5.49).

$$Dx^{k+1} = (1-\omega)Dx^k - \omega b - \omega(Lx^{k+1} + Ux^k) \tag{5.49}$$

Jeżeli założymy $h = \omega$ i $C = (1-\omega)D$, to otrzymamy równanie (5.50).

$$(C + hD)x^{k+1} = Cx^k - hb - h(Lx^{k+1} + Ux^k) \tag{5.50}$$

Dla układu zawierającego sprzężenie między węzłami j oraz i zależność na współczynnik ω_i dana jest równaniem (5.51).

$$\omega_i = \min\left(\frac{\mid d_{ii} \mid}{\sum_{j\neq i}^{N} \mid a_{ij} \mid}, 1\right) \tag{5.51}$$

Dla pozostałych węzłów $\omega_i = 1$ i metoda ta sprowadza się do metody GSN.

Metoda podrelaksacyjna dla układów równań nieliniowych

Jeżeli do rozwiązania równania (1.1) zastosujemy metodę GSN, to otrzymamy równanie (5.52).

$$\begin{aligned} (\delta f_i(x^k/\delta z_i)x_i &= -f_i(x^k) \\ x^k &= [z_1^k, \ldots, z_{i-1}^k, z_i^{k-1}, \ldots, z_N^{k-1}]^T \end{aligned} \tag{5.52}$$

W metodzie SOR NR rozwiązanie w $k+1$ relaksacji dane jest równaniem (5.53).

$$z_i^{k+1} = z_i^k + \omega x_i \tag{5.53}$$

W metodzie SOR NR współczynnik $\omega_i = 1$. W metodzie UR-NR współczynnik ω_i przyjmowany zgodnie ze wzorem (5.54).

$$\omega_i = \min\left(\frac{\mid \delta f_i(x)/\delta z_i \mid}{\sum_{j\neq i}^{N} \mid f_i(x)/\delta z_j \mid}, 1\right) \tag{5.54}$$

Rozdział 6

Metody rozwiązywania równań różniczkowych

6.1 Schemat różnicowy

Rozwiązywanie równań algebraiczno-różniczkowych sieci postaci (1.1) wymaga zastosowania algorytmów numerycznego całkowania. Algorytmy te opierają się na przybliżeniu równania różniczkowego równaniem różnicowym (6.1).

$$\dot{x} = \frac{dx}{dt} \approx \frac{\Delta x}{\Delta t} \tag{6.1}$$

Ogólna postać schematu różnicowego (SR) dla n-tego punktu czasowego dana jest (6.2).

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

gdzie: γ - współczynnik dyskretyzacji SR, d_{xn} - wektor przeszłości.
Schemat różnicowy umożliwia obliczenie pochodnej czasowej zmiennej sieciowej ($\dot{x}$).
Współczynniki SR zależne są od postaci SR. Wyróżnia się:

- schematy różnicowe zamknięte, dla których wartość $\dot{x}_n$ nie zależy od bieżącej chwili czasowej,
- schematy różnicowe otwarte, dla których wartość $\dot{x}_n$ zależy od bieżącej chwili czasowej.

Podstawowymi cechami SR są:

- rząd SR,
- krotność SR,
- stabilność,
- dokładność,
- długość kroku h, jaki można uzyskać przy danej dokładności.

Szczegółowe rozważania dotyczące SR można znaleźć np. w [J.O94, J.O95] i wielu innych pozycjach literaturowych.

6.2 Stabilność schematów różnicowych

Zagadnienia stabilności i zbieżności SR mają podstawowe znaczenie w zapewnieniu poprawności numerycznej.

Twierdzenie 6.2.1. *Schemat różnicowy jest zbieżny, jeśli błąd w ustalonym punkcie czasu dąży do* 0*, gdy krok czasowy* $ht \to 0$.

$$lim_{h\to 0,t_n} E_n = 0 \tag{6.3}$$

gdzie: E_n = (rozwiązanie dokładne - rozwiązanie przybliżone).

Rząd dokładności każdej zbieżnej metody jest o jeden niższy niż lokalny błąd obcięcia. Ponadto jeżeli stworzymy taki SR, że $LBO = 0$, to SR nie będzie zbieżny.

Twierdzenie 6.2.2. *Liniowy schemat różnicowy jest zbieżny wtedy i tylko wtedy, gdy jest spójny i stabilny.*

Twierdzenie 6.2.3. *Liniowy schemat wielokrokowy jest spójny, jeśli rząd dokładności* ≥ 1.

Zachowanie się numerycznego rozwiązania równania różniczkowego zależy zarówno od samego równania jak i postaci SR. Właściwości SR należy testować pod kątem klasy równań, do których mają być stosowane. Najważniejszym zadaniem testowym jest zadanie liniowe postaci (6.4).

$$\dot{x} = \lambda x \tag{6.4}$$

gdzie: $x(0) = 1$, $x, \lambda \in Z$, $Re\lambda \leq 0$.

Rozwiązaniem zadania testowego dane jest (6.5).

$$x(t) = \exp(\lambda t) = \exp(Re\lambda t)\left(\cos(Im\lambda t) + j\sin(Im\lambda t)\right) \tag{6.5}$$

Twierdzenie 6.2.4. *Liniowy schemat wielokrokowy jest stabilny wtedy i tylko wtedy, gdy każde jednokrotne miejsca zerowe (pierwiastek) równania charakterystycznego* $p(z) = a(z)+h\lambda b(z)$ *spełnia warunek* $|z| \leq 1$*, a każde wieloktrotne miejsce zerowe (pierwiastek) spełnia warunek* $|z| < 1$.

Stabilność SR dobrze jest przedstawić w postaci obszaru stabilności na płaszczyźnie zespolonej $h\lambda$, w zależności od największej wartości pierwiastka równania charakterystycznego $\max|z_n|$. Można wyróżnić kilka rodzajów stabilności SR:

- 0-stabilne,
- A-stabilne,
- S-stabilne,

w zależności od długości kroku jaki zapewnia stabilność danego schematu różnicowego.

0-stabilność

Twierdzenie 6.2.5. *Schemat różnicowy jest* 0*-stabilny, jeśli jest stabilny dla* $h = 0$.

Oznacza to, że dla $h \to 0$ pierwiastki równania charakterystycznego $p(z)$ zbliżają się do rozwiązania równania $a(z) = 0$. Nie oznacza to jednak stabilności algorytmu numerycznego (zobacz rozdział 2.5).

A-stabilność

Twierdzenie 6.2.6. *Schemat różnicowy jest A-stabilny dla każdego kroku $h > 0$, jeżeli jest stabilny w całej lewej półpłaszczyźnie (domkniętej) $h\lambda$.*

A- stabilne są schematy różnicowe rzędu $l \leq 2$.

S-stabilność Pojęcie S-stabilności odnosi się do analizy równań sztywnych, o znacznie różniących się stałych czasowych. Rozwiązanie takich równań może wymagać różnych kroków. Odpowiedź może być także oscylacyjna, o bardzo różniących się okresach oscylacji, co wymaga stosowania np. bardzo krótkiego kroku do analizy przebiegu o krótkim okresie i długiego kroku po zaniku odpowiedzi o krótkim okresie.

Twierdzenie 6.2.7. *Schemat różnicowy jest S-stabilny, jeżeli jest stabilny na pewnej części lewej domkniętej półpłaszczyzny obszaru $h\lambda$, gdzie leżą rozwiązania $h\lambda$ równań sztywnych.*

W zależności od rodzaju odpowiedzi obszary stabilności będą różnie wyglądały. Na podstawie szczegółowej analizy omawianych przypadków opracowano SR Gear'a rzędu $l \in< 1, 6 >$, które są schematami S-stabilnymi. Schematami S-stabilnymi są także SR wykorzystujące interpolację Newtona. Szczegółową analizę omawianych zagadnień można znaleźć np. w [J.O95].

SR jako odwzorowanie zwężające W praktycznych zastosowaniach na SR nakłada się silniejsze wymagania, aby rozwiązanie pewnego zadania w kolejnych punktach czasowych z_n bazujące na rozwiązaniach z poprzednich p punktów czasowych było zwężające w sensie wybranej normy, tzn. aby zachodziła zależność $||z_n|| \leq ||z_{n-1}||$ dla każdego n. W ogólności, jeżeli SR spełnia wymagania na rozwiązanie zwężające, to jest także stabilny.

Tłumienie SR w zadaniach oscylacyjnych Tłumienie to cecha SR, która odpowiada za powstawanie błędów numerycznych przy analizie układów oscylacyjnych, których odpowiedź nie zanika i jest stała (nietłumiona). Analiza takich układów prowadzi do powstania szeregu błędów, co prowadzi do sytuacji, gdy obliczona amplituda odpowiedzi nie jest stała, lecz ulega wzmocnieniu lub tłumieniu.

Do testowania SR można użyć zadania testowego postaci (6.6), które generuje odpowiedź o stałej amplitudzie.

$$\dot{x} = j\omega x \tag{6.6}$$

gdzie $\omega \in R^n$. Rozwiązaniem jest odpowiedź postaci (6.7).

$$x(t) = \cos(\omega t) + j\sin(\omega t) \tag{6.7}$$

Chwilowa amplituda rozwiązania $x(t)$ powinna być stała i równa 1. Na skutek błędów może jednak ulegać wzmocnieniu lub tłumieniu.

6.3 Schemat różnicowy Gear'a oparty na różnicach skalowanych

Największe znaczenie w symulatorach ma schemat różnicowy Gear'a. Omawiany tutaj SR Gear'a oparty jest na różnicach skalowanych ze zmiennym rzędem ($l = 1 \dots 6$). Dobór rzędu odbywa się automatycznie. Krok algorytmu jest zmienny, dobierany na podstawie lokalnego błędu obcięcia (*ang. LTE*). Będziemy posługiwać się schematem różnicowym postaci (6.2).

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 (6.8).

$$\underbrace{\dot{x}_n}_{\dot{x}_n} = \underbrace{a_0\ x_n}_{\gamma\ x_n} \underbrace{- \sum_{k=0}^{l-1} (c_k\ a_k\ d_{n-1}^k\ x_{n-1})}_{d_{xn}} \tag{6.8}$$

Współczynnikami schematu różnicowego są wielkości a_k (6.9) i c_k(6.10). Wartości współczynników przechowuje się w tablicach AK i CK o długościach równych rzędowi SR (l).

$$a_k = \sum_{j=k+1}^{l} \frac{1}{t_n - t_{n-j}} = \sum_{j=k+1}^{l} \frac{1}{\underbrace{t_n - t_{n-1}}_{h_n} + \underbrace{t_{n-1} - t_{n-j}}_{hs(j-1)}} = \sum_{j=k+1}^{l} \frac{1}{h_n + hs_{n-1}(j-1)} \tag{6.9}$$

$$c_k = \begin{cases} 1 & \text{dla } k = 0 \\ \prod_{j=1}^{k} \frac{t_n - t_{n-j}}{t_{n-1} - t_{n-1-j}} = \prod_{j=1}^{k} \frac{\overbrace{t_n - t_{n-1}}^{h_n} + \overbrace{t_{n-1} - t_{n-j}}^{hs_{n-1}(j-1)}}{\underbrace{t_{n-1} - t_{n-1-j}}_{hs_{n-1}(j)}} = \prod_{j=1}^{k} \frac{h_n + hs_{n-1}(j-1)}{hs_{n-1}(j)} & \text{dla } k = 1, \dots, l \end{cases} \tag{6.10}$$

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}[1] 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 (6.10), 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

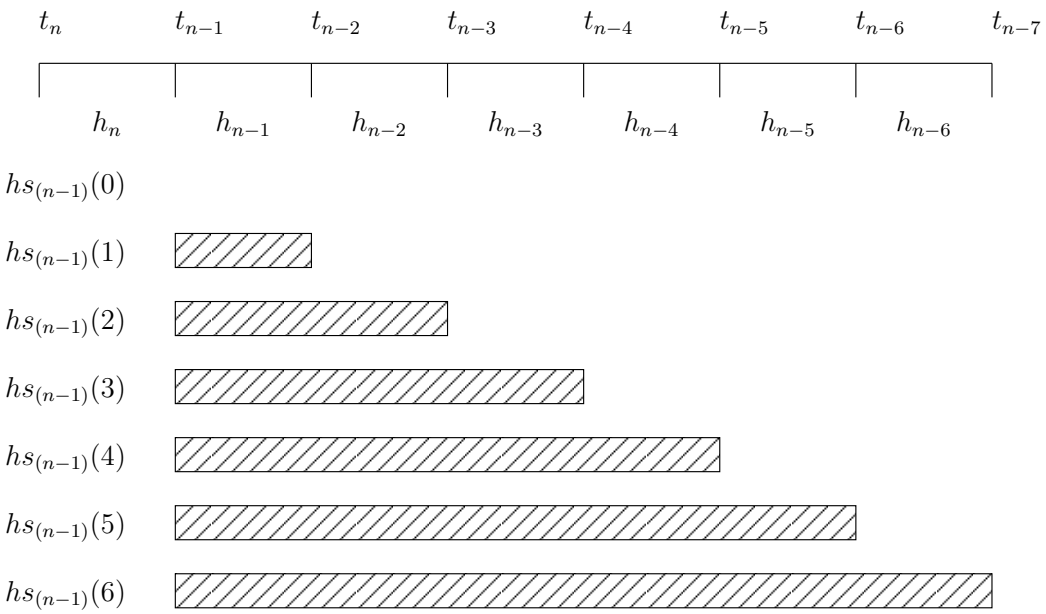
$$h_n = t_n - t_{n-1}$$

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

w n-tym kroku w punkcie t_n. W punkcie t_{n-1} zawartość poszczególnych komórek tablicy (6.11) będzie sumą odpowiednich kroków czasowych h_n (6.11).

$$\begin{aligned}
hs_{n-1}[0] &= t_{n-1} - t_{n-1-0} = 0 \\
hs_{n-1}[1] &= t_{n-1} - t_{n-1-1} = h_{n-1} \\
hs_{n-1}[2] &= t_{n-1} - t_{n-1-2} = t_{n-1} - t_{n-2} + t_{n-2} - t_{n-3} = h_{n-1} + h_{n-2} \\
hs_{n-1}[3] &= t_{n-1} - t_{n-1-3} = t_{n-1} - t_{n-2} + t_{n-2} - t_{n-3} + t_{n-3} - t_{n-4} = h_{n-1} + h_{n-2} + h_{n-3} \\
&\dots \\
hs_{n-1}[l] &= t_{n-1} - t_{n-1-l} = h_{n-1} + \dots + h_{n-l}
\end{aligned} \tag{6.11}$$

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



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

należy uaktualniać w każdym kroku czasowym. Zauważmy, że w punkcie t_n tablicę należy zaktualizować zgodnie z (6.12).

$$hs_n(j) = \begin{cases} hs_{n-1}[j-1] & \text{dla } j = l, 2 \\ h_n & \text{dla } j = 1 \\ 0 & \text{dla } j = 0 \end{cases} \tag{6.12}$$

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 $t_{n=3}$:

$$\begin{aligned} hs_{n-1}[0] &= 0 \\ hs_{n-1}[1] &= h_2 \\ hs_{n-1}[2] &= h_2 + h_1 \\ hs_{n-1}[3] &= 0 \\ hs_{n-1}[4] &= 0 \\ hs_{n-1}[5] &= 0 \\ hs_{n-1}[6] &= 0 \end{aligned} \tag{6.13}$$

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 następnego punktu czasowego t_{n+1}.

Aktualizacja różnic skalowanych Aktualizacja różnic skalowanych, wymagająca niewielkich nakładów obliczeniowych, może zostać wykonana zgodnie z (6.14).

$$\begin{aligned} d_n^0\ x_{n-1} &= x_n \\ d_n^k\ x_n &= x_n - \sum_{j=0}^{k-1}(c_j\ d_{n-1}^j\ x_{n-1}) \\ & \quad k = 1, l \end{aligned} \tag{6.14}$$

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 (6.15) 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 ${x_1}^2$ można wyznaczyć z (6.14)

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

i następny krok wykonać SR rzędu 2.

6.3.1 Przewidywanie wartości startowej

Przewidywanie wartości startowej SR (predykcja) oparte jest tych samych punktach dyskretyzacji. SR dla predykcji różnią się tylko liczbą współczynników, a nie ich warto-

[2] Wartość x_1 wyznacza się przez analizę układu w punkcie t_1.

ściami. Predykcja ([J.O95, wzór (8.40) str. 248]) dla SR Gear'a realizowana jest zgodnie ze schematem do predykcji rzędu l (6.15).

$$x_n^{(pred,l)} = \underbrace{\sum_{k=0}^{l} (c_k \; d_{n-1}^k \; x_{n-1})}_{dx} \tag{6.15}$$

6.3.2 Lokalny błąd obcięcia

Do kontroli lokalnego błędu obcięcia (LBO) (*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 (6.16) na podstawie znajomości δ i ϵ. Wartość lokalnego błędu obcięcia oblicza się z zależności (6.17).

$$LTE_{user} = \epsilon \; \max(\mid x_n^{(pred,l)} \mid, \mid x_n \mid) + \delta \tag{6.16}$$

$$LTE = \mid x_n^{(pred,l)} - x_n \mid \tag{6.17}$$

gdzie: $x_n^{(pred,l)}$ jest wartością przewidywaną z (6.15) rzędem l, x_n jest wartością obliczoną (dokładną) w n-tym punkcie czasowym.

Test dokładności analizy Analiza jest dokładna, jeśli w każdym punkcie analizy spełniona jest zależność

$$LTE <= LTE_{user},$$

która w n-tym punkcie czasowym przyjmuje postać (6.18).

$$\mid x_n^{(pred,l)} - x_n \mid <= \epsilon \; \max(\mid x_n^{(pred,l)} \mid, \mid x_n \mid) + \delta \tag{6.18}$$

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 (6.19).

$$r_{LTE} = \frac{\mid LTE_{user} \mid}{\mid LTE \mid} = \left[\frac{h_{n+1}}{h_n}\right]^{l+1} \tag{6.19}$$

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} (6.20) gwarantuje utrzymanie zadanego poziomu lokalnego błędu obcięcia LTE_{user}.

$$h_{n+1} = r_{LTE}\ h_n \tag{6.20}$$

Jeśli $r_{LTE} > 1$ oznacza to, że wartość lokalnego błędu obcięcia jest mniejsza niż założona i otrzymane rozwiązanie jest prawidłowe - krok można wydłużyć. W praktyce wprowadza się także dodatkowy mnożnik wydłużenia kroku R.

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

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 (6.21).

$$h_{n+1} = \begin{cases} h_n\ \max(s_l, r_{LTE}) & r_{LTE} < 1 \\ h_n & 1 \le r_{LTE} \le \alpha \\ h_n\ \min(s_u, \beta\ r_{LTE}) & r_{LTE} > \alpha \end{cases} \tag{6.21}$$

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 (6.20). 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} (6.22) na podstawie wartości LTE i LTE_{user} wyznaczanych dla każdej zmiennej sieciowej.

$$r_{LTE} = \min_i \left| \frac{LTE_{user}}{LTE} \right|^{\frac{1}{l+1}} \tag{6.22}$$

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:

$$h_{n+1} = \begin{cases} h_n / s_1 & p > p_{max} \\ h_n & p_{min} \le p \le p_{max} \\ h_n \cdot s_2 & p < p_{min} \end{cases} \tag{6.23}$$

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

Schemat różnicowy trapezowy (1-krotny, 2-go rzędu) ma dobre właściwości numeryczne. Jest najdokładniejszą metodą A-stabilną spośród liniowych metod wielokrokowych. Schemat różnicowy trapezowy można zapisać dla n-tego punktu czasowego w postaci (6.24).

$$x_n = \frac{1}{2} h_n \dot{x}_n + \frac{1}{2} h_n \dot{x}_{n-1} + x_{n-1} \tag{6.24}$$

Równanie (6.24) można przepisać do postaci (6.2) otrzymując (6.25).

$$\dot{x}_n = \underbrace{\frac{2}{h_n}}_{\gamma} x_n + \underbrace{\left(-\dot{x}_{n-1} - \frac{2}{h_n} x_{n-1} \right)}_{d_x} \tag{6.25}$$

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 (6.26).

$$x_n^0 = \sum_{k=1}^{l+1} \alpha_k^l x_{n-k} \tag{6.26}$$

Współczynniki α dane są równaniem (6.27).

$$\alpha_k^l = \prod_{i=1, i \neq k}^{l+1} \frac{s_i}{s_i - s_k} = \prod_{i=1, i \neq k}^{l+1} \frac{h s_{n-1}(i-1)}{h s_{n-1}(i-1) - h s_{n-1}(k-1)} \tag{6.27}$$

Współczynniki s są kolejnymi sumami kroków czasowych danych równaniem (6.28), które można zapisać z wykorzystaniem tablicy sum kroków[3] hs_{n-1} z poprzedniej chwili czasowej t_{n-1}.

$$s_i = t_n - t_{n-i} = \sum_{k=0}^{i} h_{n-k} = h_n + hs_{n-1}(i-1) \tag{6.28}$$

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 (6.26) otrzymamy wzór predykcji postaci (6.29).

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

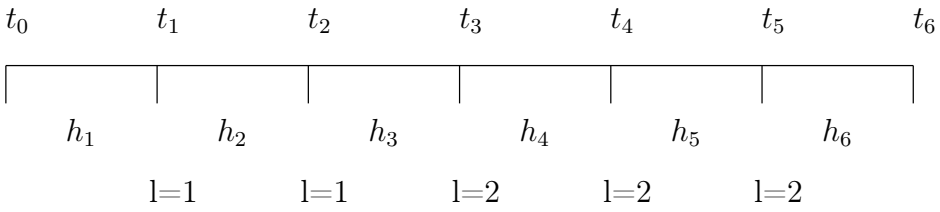
Współczynniki α można zapis jawnie w postaci (6.29).

$$\begin{array}{rcl} \alpha_1^2 & = & \frac{s_2 s_3}{(s_2 - s_1)(s_3 - s_1)} \\ \alpha_2^2 & = & \frac{s_1 s_3}{(s_1 - s_2)(s_3 - s_2)} \\ \alpha_3^2 & = & \frac{s_1 s_2}{(s_1 - s_3)(s_2 - s_3)} \end{array} \tag{6.29}$$

Współczynniki s można jawnie zapisać jako (6.30).

$$\begin{array}{rcllcl} s_1 & = & t_n - t_{n-1} & = & h_n & = & h_n + hs_{n-1}(0) \\ s_2 & = & t_n - t_{n-2} & = & h_n + h_{n-1} & = & h_n + hs_{n-1}(1) \\ s_3 & = & t_n - t_{n-3} & = & \underbrace{h_n}_{(t_n - t_{n-1})} + \underbrace{h_{n-1}}_{(t_{n-1} - t_{n-2})} + \underbrace{h_{n-2}}_{(t_{n-2} - t_{n-3})} & = & h_n + hs_{n-1}(2) \end{array} \tag{6.30}$$

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.



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

[3] zobacz tablica sum kroków (6.11)

Rozdział 7

Klasyczna analiza czasowa

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

$$f(x, \dot{x}, t) = 0. \tag{7.1}$$

Proces rozwiązywania rozpoczyna się od obliczenia pochodnej $\dot{x}$, czyli tzw. dyskretyzacji za pomocą schematu różnicowego (rozdział 6.1) postaci (6.2), który ponownie przepiszemy poniżej.

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

gdzie: d_x jest wektorem przeszłości, γ jest współczynnikiem zależnym od rodzaju schematu różnicowego. Otrzymamy w ten sposób układ nieliniowych równań algebraicznych (7.2).

$$f(x_n, \gamma\ x_n + d_{xn}, t_n) = 0 \tag{7.2}$$

Układ równań (7.2) można rozwiązać metodą Newtona-Raphsona (rozdział 5.2) przez linearyzację równań, tzn. rozwinięcie w szereg Taylora do wyrazów rzędu pierwszego.

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

$$= -\underbrace{f(x_n^{(p-1)}, \gamma\ x_n^{(p-1)} - d_{xn}, t_n)}_{f^{(p-1)}} \tag{7.3}$$

Po pogrupowaniu czynników w (7.3) i wprowadzeniu oznaczeń przyjmuje on bardziej przejrzystą postać (7.4).

$$\underbrace{\frac{\delta f^{(p-1)}}{\delta x_n}}_{Y}\ {x_n}^{(p)} = \underbrace{-f^{(p-1)} + \frac{\delta f^{(p-1)}}{\delta x_n}\ x_n^{(p-1)}}_{B} \tag{7.4}$$

Układ równań (7.4) jest liniowy i można go rozwiązać jedną z metod rozwiązywania układów równań liniowych - np. metodą rozkładu LU (rozdział 4.1.2).

Podstawowy algorytm analizy analizy czasowej został przedstawiony w postaci algorytmu 16.

Stosowany w praktyce algorytm analizy czasowej 17 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 Gea'ra (rozdział 6.3) z doborem rzędu SR.

Algorytm 17 jest uproszczoną wersją algorytmu zimplementowanego w symulatorze *Dero*[Pla13]. Szczegóły implementacyjne zostały pominięte dla większej czytelności algorytmu.

7.1 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 istnieje stan nieustalony.

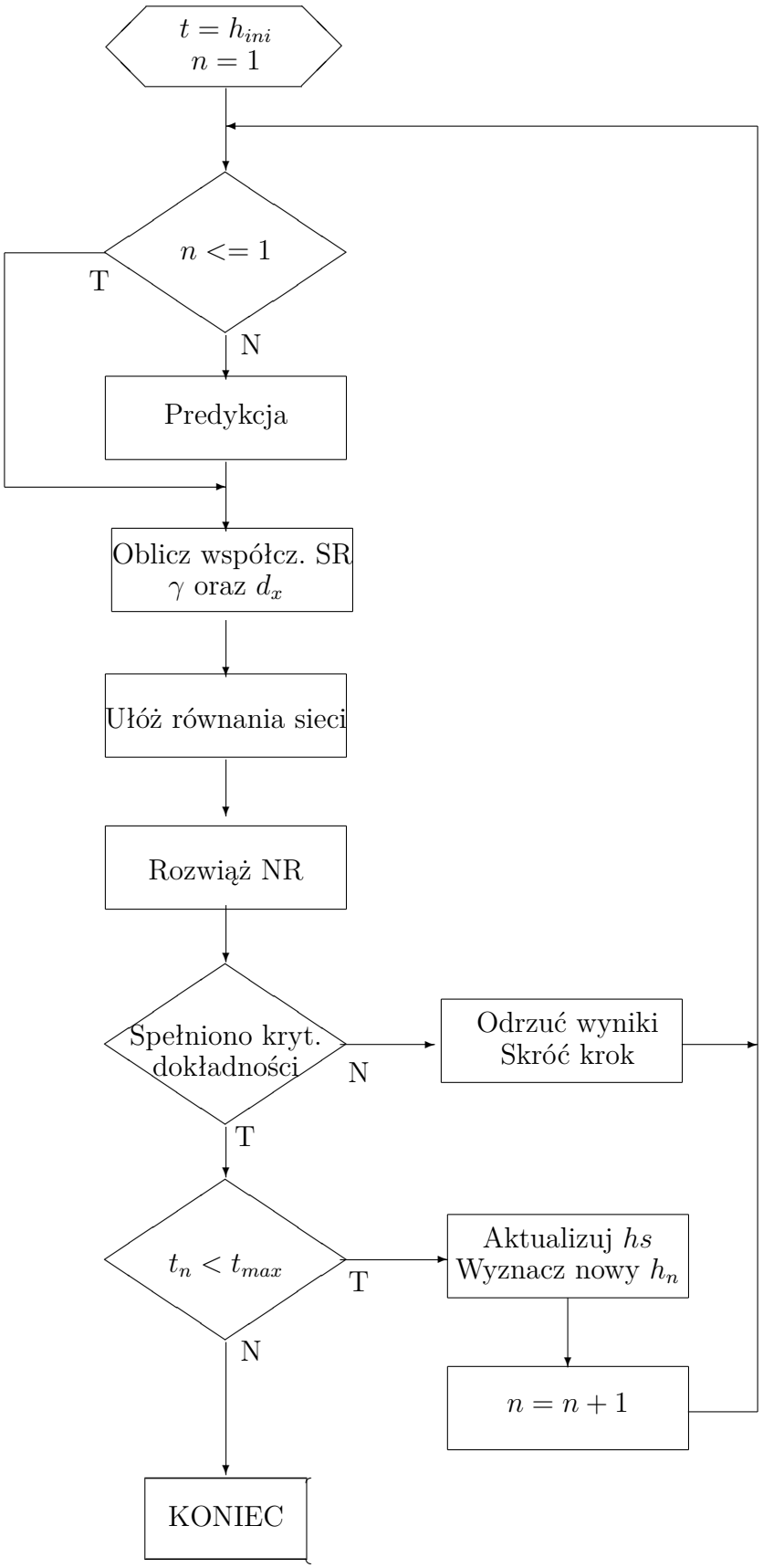
Każdym warunkom początkowym $x_0 = x(t_0)$ w pewnej chwili t_0 odpowiada pewien przebieg rozwiązania $x(t)$, przybierający po okresie wartość $x(T)$. Wartość $x(T)$ można potraktować jako kolejne przybliżenie warunków początkowych. Odwzorowanie to nazwiemy odwzorowaniem podstawowym i oznaczymy symbolem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

$$x(T) = \Phi(x_0) \tag{7.5}$$

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

$$x_0^* = \Phi(x_0^*) \tag{7.6}$$

Algorytm 16 Algorytm analizy czasowej



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

$$x(T) = A(x_0 - x_0^*) + x_0^* \tag{7.7}$$

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

$$\begin{aligned} x^{(k)}(T) &= A(x_0^{(k)} - x_0^*) + x_0^* \\ x^{(k+1)}(T) &= A(x_0^{(k+1)} - x_0^*) + x_0^* \end{aligned} \tag{7.8}$$

na różnice odpowiednich wyników po k-tym okresie:

$$x^{(k)}(T) - x^{(k+1)}(T) = A(x_0^{(k)} - x_0^{(k+1)}) \tag{7.9}$$

Po zastosowaniu liniowej aproksymacji przekształcenia podstawowego (7.5) w otoczeniu punktu stałego i zakładając ciągłość jego pochodnej otrzymamy postać podobną do (7.7).

$$\Phi(x_0^{(k)}) = x(T) = x_0^* + \frac{\delta\Phi(x_0^*)}{\delta x_0}(x_0^{(k)} - x_0^*) \tag{7.10}$$

Macierz pochodnych $\frac{\delta\Phi(x_0^*)}{\delta x_0}$ decyduje o sposobie dochodzenia do stanu ustalonego. Po odjęciu od obu stron równania (7.10) 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 największą wartość własną bliską 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 metodą rzędu pierwszego.

W praktyce stosuje się raczej rząd 2, tj. metodę Newtona-Raphsona (zobacz 5.2). Polega ona na linearyzacji funkcji $f(x_0)$ w otoczeniu przybliżenia $x_0^{(k)}$ warunków początkowych:

$$f(x_0) = \Phi(x_0^{(k)}) - x_0^{(k)} + [\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1](x_0^{(k+1)} - x_0^{(k)}) \tag{7.11}$$

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

$$[\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1]\ x_0^{(k+1)} = -(\Phi(x_0^{(k)}) - x_0^{(k)}) + [\frac{\delta\Phi(x_0^{(k)})}{\delta x_0} - 1]x_0^{(k)} \tag{7.12}$$

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 (7.5) oraz obliczenie stanu ustalonego jako punktu stałego (7.6).

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.

Metoda Bukowskiego Metoda [J.O95] polega na estymacji macierzy pochodnych $\frac{\delta\Phi(x_0^*)}{\delta x_0}$ odwzorowania podstawowego (7.10) dla każdej iteracji w sposób odpowiadający wielowymiarowej metodzie siecznych[1].

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 $(< 0, T >)$.

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

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

[1] 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.

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

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

Jeżeli dysponujemy n takimi związkami, to możliwe jest połączenie przez wprowadzenie macierzy przyrostów $Y^{(k)}$, $Z^{(k)}$ i zapisanie układu n równań (znak $\|$ oznacza ustawienie kolumn obok siebie).

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

Wyznaczenie macierzy przejścia wymaga obliczenia $Y^{(k)}$ 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[2] otrzymamy:

- po pierwszym okresie - warunki początkowe $x_0^{(1)}$,
- 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 otrzymamy: $x_0^{(2)}$, $x_0^{(3)}$, $\Phi(x_0^{(2)}) = x_0^{(3)}$, $z^{(2)} = \Phi(x_0^{(2)}) - \Phi(x_0^{(1)})$.

Po zakończeniu analiz wstępnych (po trzecim okresie) z kolejnych wektorów y i z konstruowane są przybliżone macierze Y i Z. Służą one do wyznaczenia pierwszego przybliżenia macierzy przejścia $A^{(0)}$ $(k = 0)$ odwzorowania podstawowego (7.9).

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

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 (7.11) na miejsce macierzy pochodnych $\frac{\delta \Phi(x_0^{(k)})}{\delta x_0}$, (7.11) stanie się lokalną aproksymacją liniową przechodzącą przez wprowadzone punkty. Po przekształceniach otrzymamy układ dwóch równań, który umożliwia wyznaczenie stanu ustalonego dla zidentyfikowanego liniowego przekształcenia podstawowego.

$$\begin{aligned} (Y^{(k)} - Z^{(k)}) w &= x_0^{(k)} - \Phi(x_0^{(k)}) \\ x_0^{(k+1)} = x_0^{(k)} + Y^{(k)} w &= \Phi(x_0^{(k)}) + Z^{(k)} w \end{aligned} \tag{7.13}$$

Iteracje zaczynają się od wyznaczenia nowych warunków początkowych $x_0^{(k+1)}$ z (7.13). Z pierwszego równania oblicza się w

[2] Pierwsze 3 okresy analizy muszą być wykonane 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.

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

a następnie podstawia do drugiego, z którego obliczana jest wartość $x_0^{(k+1)}$

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

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 (7.14) i ewentualnie przerywa obliczenia, gdy wartość ta jest mniejsza od wartości zadanej ϵ_r.

$$\frac{\|\phi(x_0^{(k+1)}) - x_0^{(k+1)}\|}{\max_k(\|x_0\|)} < \epsilon_r \tag{7.14}$$

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 względną nie gorszą niż ϵ_r (7.14).

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[3].

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

Algorytm wymaga wykonania w jednej iteracji quasi-newtonowskiej jednej analizy czasowej za okres. Nakład na znalezienie nowych warunków początkowych, czyli rozwiązanie układu równań (7.13) wymaga wykonania

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

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

[3]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 17 Uproszczony algorytm analizy czasowej zastosowany w symulatorze *Dero*.

Require: x_0 ▷ wartości początkowe x
$t_n = 0$ ▷ inicjalizacja czasu
$h_n = h_{ini}$ ▷ inicjalizacja kroku
$f(x, \dot{x}, t) = 0$ ▷ równania różniczkowe zwyczajne
while $t_n < t_{max}$ **do** ▷ pętla czasu
  if $t_n == t_{break}$ **then** ▷ pułapka czasowa
    $h_n = h_{min}$
  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 (ZMPW)
    $x^{(p)} = Y^{-1}B$ ▷ rozwiąż układ równań liniowych (LU)
  until zbieżność NR osiągnięta ▷ test stopu NR

  if włączona kontrola LTE po analizie **then**
    oblicz LTE_{POST} ▷ oblicz LTE po analizie
    if $LTE_{POST} > LTE_{USR}$ **then**
      odrzuć wyniki analizy w punkcie t_n
      cofnij czas $t_n = t_n - h_n$
      wyznacz nowy krok h_n na podstawie wyznaczonego LTE_{POST}
      wyznacz skorygowany punkt czasowy $t_n = t_n + h_n$
      Idź do 5
    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 tablice przeszłości d_x dla rzędów $k = 1 \ldots min(l_{max}, l + 1)$
    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 $h_{n+1} = r_{LTE} h_n$ na podstawie r_{LTE} (6.21)
    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 pułapkę czasową $h_{n+1} = t_{break} - t_n$
    end if
  end if
end while

Algorytm 18 Algorytm analizy czasowej stanu ustalonego.

Require: s ▷ liczba zmiennych stanu
Require: T ▷ okres analizy
Require: K_{max} ▷ maksymalna liczba okresów analizy stanu ustalonego (iteracji STDS)
1: **for** $k = 1 \dots s$ **do** ▷ inicjalizacja algorytmu
2: wykonaj analizę czasową TRAN za okres $(0...T)$
3: inicjalizuj tablice robocze Y,Z
4: **end for**
5: **for** $k = s \dots K_{max}$ **do** ▷ pętla iteracji STDS
6: oblicz $x_0^{(k+1)}$ z (7.13) ▷ warunki początkowe dla analizy czasowej
7: wykonaj analizę czasową TRAN za okres $(0...T)$ ▷ klasyczna analiza czasowa
8: **if** $\frac{\|\phi(x_0^{(k+1)}) - x_0^{(k+1)}\|}{\max_k(\|x_0\|)} < \epsilon_r$ **then** ▷ test zakończenia iteracje STDS
9: idź do 16
10: **else** ▷ aktualizuj wektory y i z
11: $y^{(k+1)} = x_0^{(k+1)} - x_0^{(k)}$
12: $z^{(k+1)} = \phi(x^{(k+1)}) - \phi(x^{(k)})$
13: oblicz $Y^{(k+1)}$, $Z^{(k+1)}$
14: **end if**
15: **end for**
16: KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

Algorytm 19 Algorytm analizy czasowej stanu ustalonego metodą Bukowskiego.

Require: N_p ▷ liczba punktow czasowych w okresie T
Require: M ▷ liczba zmiennych stanu
Require: T ▷ okres analizy
Require: h ▷ krok analizy czasowej (stały)
Require: K_{max} ▷ maksymalna liczba iteracji STDS
Require: ϵ_r ▷ dokładność wyznaczania stanu ustalonego
for $k = 1 \dots K_{max}$ **do** ▷ pętla iteracji STDS ▷ inicjalizacja algorytmu
  $pe = N_p$ ▷ liczba punktów czasowych do końca okresu
  for $p = 1 \dots N_p$ **do**
    $t_p = t_{min} + p \cdot h$ ▷ punkt czasowy analizy
    wykonaj analizę czasową TRAN w okresie do punktu t_p
    if pe <= M **then**
      if k == 1 **then** ▷ pierwszy okres
        zapamiętaj warunki początkowe $x_0^{(1)}$ w punkcie t_p
      end if
      if k == 2 **then** ▷ drugi okres - zapamiętaj i oblicz w punkcie t_p
        $x_0^{(1)}$, $x_0^{(2)}$
        $\Phi(x_0^{(1)}) = x_0^{(2)}$
        $y^{(2)} = x_0^{(2)}$ - $x_0^{(1)}$
      end if
      if k == 3 **then** ▷ trzeci okres - zapamiętaj i oblicz w punkcie t_p
        $x_0^{(2)}$, $x_0^{(3)}$
        $\Phi(x_0^{(2)}) = x_0^{(3)}$
        $z^{(2)} = \Phi(x_0^{(2)}) - \Phi(x_0^{(1)})$.
      end if
      if k > 3 **then** ▷ następne okresy - zapamiętaj i oblicz w punkcie t_p
        if $\frac{|x_0^{(k)} - x_0^{(k-1)}|}{\max_k(|x_0|)} < \epsilon_r$ **then** ▷ test stopu STDS
          stan ustalony osiągnięty - idź do 38
        else ▷ stan ustalony nieosiągnięty - test stopu niespełniony
          oblicz zmienne
          $y^{(k+1)} = x_0^{(k+1)} - x_0^{(k)}$
          $z^{(k+1)} = \Phi(x_0^{(k+1)}) - \Phi(x_0^{(k)})$
          aktualizuj tablice Y, Z
          - zapomnij najstarsze przyrosty $y^{(k-n)}$, $z^{(k-n)}$
          - dopisz nowo obliczone przyrosty $y^{(k+1)}$, $z^{(k+1)}$
          wyznacz nowe warunki początkowe $x_0^{(k+1)}$ (7.13)
        end if
      end if
      zapamiętaj wektor maksymalnych modułów zm. sieciowych $|x_{max}|$
    end if
    $p_e = p_e - 1$ ▷ zmniejsz licznik punktów w okresie analizy
  end for
end for
KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

Rozdział 8

Relaksacyjna analiza czasowa

Wraz z rozwojem układów scalonych wielkiej skali integracji (VLSI) tradycyjne metody symulacji okazały się niewystarczające ze względu na ich małą efektywność (długie czasy analiz, duża zajętość pamięci). Doprowadziło to do opracowania relaksacyjnych metod symulacji układów elektronicznych, które z powodzeniem zastosowano do symulacji układów cyfrowych VLSI. Metody relaksacyjne mają swoje ograniczenia. Bloki układów o silnie sprzężonych węzłach, w których występują sprzężenia zwrotne i sprzężenia dwukierunkowe muszą być symulowane za pomocą klasycznej analizy czasowej układów analogowych.

Dzięki opracowaniu metod relaksacyjnych, nastąpił istotny postęp także w analizie układów analogowych. Metody relaksacyjne wykorzystują metody iteracyjnego rozwiązywania równań (patrz rozdział 4.2). Zamiast układu równań rozwiązywane są pojedyncze rozdzielone równania w kolejności odpowiadającej propagacji sygnału. Wyróżnia się dwie metody podziału równań:

- podział węzłowy [ARN83, E.L82], gdzie każdy węzeł sieci traktowany jest osobno i równanie każdego węzła rozwiązywane jest oddzielnie,
- podział blokowy [M.N88, D.J92, MP89], gdzie zamiast pojedynczych węzłów występują grupy węzłów, których układy równań rozwiązywane są metodami bezpośrednimi (np.LU) na poziomie bloków, natomiast pomiędzy blokami stosuje się algorytm relaksacyjny.

Należy zaznaczyć, że podział blokowy umożliwia analizę szerszej klasy sieci niż metody relaksacyjne z podziałem węzłowym.

Różnice pomiędzy metodami relaksacyjnymi wynikają z poziomu równań, na którym wykonywane jest rozdzielanie zgodnie z tym, co napisano w rozdziale 1.2 i podano w tab. 1.1. Jeżeli równania zostają rozdzielone przed dyskretyzacją i zdyskretyzujemy każde równanie osobno, to otrzymamy tzw. metodę relaksacji przebiegów (*ang. waveform-relaxation*) omówioną w rozdziale 8.4. Jeśli najpierw zdyskretyzowane będą równania całego układu, a potem zostaną one rozdzielone, to otrzymamy metody jednopunktowe (*ang. one-point relaxation*) opisane w rozdziale 8.3. Jeżeli rozdzielanie węzłowe zostanie wykonane na poziomie nieliniowych równań algebraicznych, to otrzymamy metody relaksacyjne nieliniowe. Jeśli równania zostaną zlinearyzowane i dopiero po linearyzacji zostanie wykonany rozdział, to otrzymamy metody relaksacyjne liniowe.

W praktyce większe znaczenie mają liniowe metody relaksacyjne, w których wykonywana jest tylko jedna relaksacja. Zbieżność metod jest liniowa, lecz redukowany jest

nakład obliczeń związany ze sprawdzaniem testu stopu pętli iteracji. Szczególnie rozpowszechnioną metodą jest OSR (*ang. one-step relaxation*) [Hun80, Hag81, Owe94], która może być wykorzystywana także w nieliniowych metodach relaksacyjnych, co prowadzi do analizy "timing". W nowoczesnych symulatorach metoda OSR została wyparta przez metody iteracyjne (*ang. iterated relaxation methods*), które lepiej radzą sobie z rzeczywistymi układami, także z tymi, w których występują silne sprzężenia. Obecnie można wyróżnić cztery grupy metod mających znaczenie praktyczne:

1. *ang. timing analysis* znana jako *ang. relaxation-based integration method*,
2. *ang. iterated timing analysis* (GSN ITA, Event-Driven ITA),
3. *ang. one-step relaxation analysis* (*ang. one-point relaxation methods*),
4. *ang. waveform relaxation analysis.*

Metody relaksacyjne mają dwie podstawowe zalety decydujące o ich przydatności, a mianowicie:

- kosztowne układanie i rozwiązywanie układów równań metodami bezpośrednimi jest zastąpione przez mniej kosztowne rozwiązywanie pojedynczych równań rozdzielonych;
- ze względu na rozdzielenie węzłowe lub blokowe można łatwo wykorzystać niezależność równań i zastosować techniki zamrażania (*ang. latency*).

Metody relaksacyjne są o rząd wielkości szybsze od metod bezpośrednich, jednak ich stosowalność jest ograniczona do pewnej klasy sieci. Szybkość działania zależy od topologii układu i istnienia w układzie elementów bilateralnych.

W latach 90-tych XX wieku opracowano technikę symulacji - EDLICS (*ang. Event-Driven Local Iteractive Circuit Simulation*) [JCL94] wykorzystującą zachowanie sieci elektrycznej (rozdział 8.7).

W metodach bezpośrednich istnieje także możliwość zastąpienia rozwiązywania układów równań liniowych w pętli NR metodami iteracyjnymi, lecz zyski są małe ze względu na potrzebę wielokrotnego układania macierzy pochodnych (Jakobianu). Dodatkowo dokładność wewnętrznej pętli iteracji jest mała co powoduje, że zbieżność pętli NR nie jest kwadratowa, lecz staje się liniowa.

8.1 Równania sieci w analizie relaksacyjnej

Analiza relaksacyjna wykorzystuje techniki iteracyjnego rozwiązywania rozdzielonych równań sieci. Zastosowane rozdzielenie węzłowe równań powoduje, że klasa akceptowanych sieci zostaje zawężona do układów zbudowanych tylko z dwóch rodzajów dwójników[1]:

- nieliniowych pojemności sterowanych co najwyżej napięciowo

$$i_a = \dot{q}(v_a, v_b, t),$$

[1] Niezależne źródła napięciowe uziemione można zastosować (pomimo, że nie są akceptowane) przez zastosowanie triku polegającego na zwiększeniu napięcia w węźle o wartość napięcia tego źródła.

- nieliniowych przewodności sterowanych co najwyżej napięciowo

$$i_a = f_i(v_a, v_b, t).$$

Równania rozdzielone rozwiązywane są metodami iteracyjnymi omówionymi w rozdziale 4.2.

W wektorze zmiennych sieciowych x (12.1) mogą występować tylko napięcia węzłowe v. Wektor uogólnionych zmiennych sieciowych x można podzielić na dwa wektory (8.1).

$$x = [x_A^T, x_N^T]^T \tag{8.1}$$

gdzie: $x_A = [v]^T$ - wektor zmiennych akceptowanych, $x_N = [i, q, \psi]^T$ - wektor zmiennych nieakceptowanych.

Równanie opisujące uogólnioną nieliniową sieć elektryczną (1.1) można przepisać do postaci (8.2).

$$\dot{q}(v) + f(v, t) = 0 \tag{8.2}$$

Po dyskretyzacji za pomocą schematu różnicowego (6.2) dla chwili t_m otrzymamy:

$$\gamma\left[q(v_{t_m}) - d_q\right] + f(v_{t_m}) = 0$$

Po linearyzacji równań zdyskretyzowanych i zastosowaniu rozdzielenia węzłowego otrzymamy n równań zależnych od $(i-1)$ rozwiązań z obecnej chwili, oraz $(n-(i-1))$ rozwiązań z poprzedniej chwili czasowej (8.3).

$$\begin{aligned} v_{t_m}^{k-1,p-1} &= [v_{1,t_m}^{k,p-1}, \ldots, v_{i-1,t_m}^{k,p-1}, v_{i,t_m}^{k-1,p-1}, \ldots, v_{n,t_m}^{k-1,p-1}, t_m]^T \\ v_{i,t_m}^{k,p} &= v_{i,t_m}^{k-1,p-1} - \frac{\gamma[q_{i,t_m}(v_{t_m}^{k-1,p-1}) - d_{qi}] + f_i(v_{t_m}^{k-1,p-1})}{\gamma C_i(v_{i,t_m}^{k-1,p-1}) + \dfrac{df_i(v_{t_m}^{k-1,p-1})}{dv_i}} \end{aligned} \tag{8.3}$$

gdzie: m jest indeksem chwili czasowej, i jest indeksem zmiennej.

Równania (8.3) można zapisać w ogólnej postaci (8.4).

$$v_{i,t_m}^{k} = v_{i,t_m}^{k-1} - \frac{\sum I}{\sum \dfrac{dI}{dv}} \tag{8.4}$$

Licznik w ilorazie jest sumą prądów pochodzących od elementów dołączonych do węzła i, natomiast mianownik jest sumą konduktancji pochodzących od tych elementów. Zgodnie z prądowym prawem Kirchoffa, prądy wpływające do węzła sumowane są ze znakiem ujemnym, natomiast wypływające ze znakiem dodatnim. Równanie (8.4) jest wykorzystywane w algorytmie automatycznego układaniu równań sieci. Równanie (8.3) można rozwiązać jedną z technik iteracyjnych podanych w rozdziale 4.2.

8.2 Analiza czasowa timing

W analizie *timing* równania sieci (8.2) po linearyzacji i dyskretyzacji za pomocą schematu różnicowego (6.2) otrzymamy równania sieci w postaci (8.3). Ponieważ wykonywana jest tylko jedna relaksacja i jedna iteracja NR, więc indeksy relaksacji i iteracji NR można opuścić i równania przyjmują postać (8.5).

$$\begin{aligned} v_{t_m} &= [v_{1,t_m}, \ldots, v_{i-1,t_m}, v_{i,t_{m-1}}, \ldots, v_{n,t_{m-1}}, t_m]^T \\ v_{i,t_m} &= v_{i,t_{m-1}} - \frac{\gamma[q_{i,t_m}(v_{t_m}) - d_{qi}] + f_i(v_{t_m})}{\gamma C_i(v_{i,t_{m-1}}) + df_i(v_{t_m})/dv_i} \end{aligned} \tag{8.5}$$

Algorytm 20 przedstawia sposób działania analizatora. W każdym punkcie czasowym analizy rozwiązywane są równania rozdzielone.

Algorytm 20 Analiza czasowa *timing*

```
1: while t_m < t_max do                    ▷ pętla czasu
2:     for i = 1, n do                     ▷ pętla węzłów
3:         równanie (8.5)
4:     end for
5: end while
```

Jeżeli układ zbudowany jest z elementów o jednokierunkowym przepływie sygnału i równania rozwiążemy w takiej kolejności jak propaguje się sygnał, to otrzymamy szukane rozwiązanie. Ogólnie można stwierdzić, że *dokładne* rozwiązanie może być otrzymane tylko w przypadku silnej dominacji głównej przekątnej macierzy. Jest to spełnione dla układów, w których nie ma pojemności pływających[2] i do każdego węzła sieci dołączono uziemioną pojemność. W takim przypadku dominację głównej przekątnej można zapewnić dobierając odpowiednio krótki krok czasowy h. W przypadku sieci rezystancyjnych silna dominacja głównej przekątnej nie jest z reguły zapewniona i otrzymane rozwiązania obarczone są dużymi błędami. Jeżeli krok analizy jest zbyt długi, to analiza może dawać odpowiedź oscylacyjną, pomimo zastosowania A-stabilnego schematu różnicowego [J.O95](rozdział 6.2). Bliższe informacje na temat właściwości numerycznych metod *timing* można znaleźć w pracach [GDM82, GM83].

8.2.1 Symetryczna analiza timing

W celu poprawienia zbieżności i dokładności analizy *timing* wprowadzono analizę symetryczną z punktem pośrednim pomiędzy kolejnymi punktami analizy. Jeżeli założymy, że analiza wykonywana jest w punktach t_m oraz t_{m+1} to w punkcie $t_{m+1/2} = t_m + h/2$ wykonuje się dodatkową analizę.

Załóżmy, że nieliniowa sieć elektryczna opisana jest równaniem (8.5). Załóżmy, że znane są rozwiązania w k-tej relaksacji z chwili czasowej t_m tzn. $v_{1,t_m}, \ldots, v_{k-1,t_m}$ i część rozwiązań z poprzedniej chwili czasowej $v_{k,t_{m-1}}, \ldots, v_{n,t_{m-1}}$. Metoda GS daje dokładne rozwiązania tylko dla macierzy dolnej trójkątnej. W innym przypadku tylko wtedy, gdy występuje silna dominacja głównej przekątnej macierzy. Pierwszemu przypadkowi odpowiada występowanie w układzie elementów unilateralnych o numerach węzłów wejściowych niższych niż wyjściowych, natomiast w drugim - elementów bilateralnych. Równania symetrycznej analizy GSN przyjmują wtedy postać (8.6) (8.7).

[2] Pojemność pływająca jest to kondensator, którego żadna z okładek nie jest dołączona do masy.

$$
\begin{aligned}
v_{t_m} &= [v_{1,t_m}, \ldots, v_{i-1,t_m}, v_{i,t_{m-1}}, \ldots, v_{n,t_{m-1}}, t_m, h]^T \\
v_{i,t_{m-1/2}} &= v_{i,t_{m-1}} - \frac{\gamma[q_{i,t_{m-1/2}}(v_{t_{m-1/2}}) - d_{qi}] + f_i(v_{t_{m-1/2}})}{\gamma C_i(v_{i,t_{m-1}}) + df_i(v_{t_m})/dv_i} \qquad (8.6) \\
& \quad i = 1, \ldots, n \\
v_{i,t_m} &= v_{i,t_{m-1/2}} - \frac{\gamma[q_{i,t_m}(v_{t_m}) - d_{qi}] + f_i(v_{t_m})}{\gamma C_i(v_{i,t_{m-1/2}}) + df_i(v_{t_m})/dv_i} \qquad (8.7) \\
& \quad i = n, \ldots, 1
\end{aligned}
$$

Równania te można rozwiązać algorytmem 21

Algorytm 21 Symetryczna analiza *timing*.

```
1: while t_m < t_max do                          ▷ pętla czasu
2:     for i = 1, n do                           ▷ pętla węzłów - w kolejności
3:         równanie (8.6)
4:     end for
5:     for i = n, 1 do                           ▷ pętla węzłów - odwrotna
6:         równanie (8.7)
7:     end for
8: end while
```

Metody *timing*, w tym także metoda symetryczna, mają dwa słabe punkty co powoduje, że mają one małe znaczenie praktyczne. Należą do nich:

1. analiza układów z pojemnościami pływającymi obarczona jest dużymi błędami. Pojemności elementów wprowadzają poza główną przekątną macierzy elementy niezerowe, których wartości są proporcjonalne do $1/h$. Dla małych h przekątna macierzy układu staje się słabo dominująca, co prowadzi do bardzo powolnej zbieżności lub braku zbieżności metody iteracyjnego rozwiązywania równań. Dodatkowo przewidywany krok analizy jest i tak zbyt długi ze względu na stabilność metod całkowania.

2. bardzo trudno jest efektywnie przewidzieć krok czasowy na podstawie lokalnego błędu obcięcia, gdyż zależy on od błędu obcięcia schematu różnicowego i od dokładności otrzymanego rozwiązania, co jest trudne do oszacowania.

8.2.2 Iteracyjna analiza Gaussa-Seidela-Newtona

Lepszą dokładność i zbieżność niż analiza *timing* uzyskuje się w metodzie iteracyjnej Gaussa-Seidela-Newtona [R.A83](ITA). Osiąga się to dzięki wprowadzeniu wielu relaksacji w każdej chwili czasowej. Jeżeli macierz układu ma silnie dominującą przekątną, to liczba relaksacji jest mała. Po dyskretyzacji i linearyzacji równań względem v^k_{i,t_m} wokół rozwiązania z poprzedniej relaksacji v^{k-1}_{i,t_m} otrzymamy pełny iteracyjny algorytm analizy czasowej - algorytm 22).

$$
\begin{aligned}
v_{t_m} &= [v^k_{1,t_m}, \ldots, v^k_{i-1,t_m}, v^{k-1}_{i,t_m}, \ldots, v_{n,t_{m-1}}, t_m]^T \\
v^k_{i,t_m} &= v^{k-1}_{i,t_m} - \frac{\gamma[q_{i,t_m}(v^{k-1}_{t_m}) - d_{qi}] + f_i(v^{k-1}_{t_m})}{\gamma C_i(v^{k-1}_{i,t_m}) + df_i(v^{k-1}_{t_m})/dv_i} \qquad (8.8)
\end{aligned}
$$

Należy zauważyć, że w metodzie tej wykonuje się tylko jedną iterację NR w każdej relaksacji.

Algorytm 22 Iteracyjna analiza Gaussa-Seidela-Newtona

```
1:  while t_m < t_max do                                   ▷ pętla czasu
2:      przewidywanie (predykcja) v^(0)_{1,t_m}, ..., v^(0)_{n,t_m}   ▷ punkt startowy relaksacji
3:      for k = 1, k_max do                                ▷ pętla relaksacji
4:          for i = 1, n do                                ▷ pętla węzłów rozdzielonych
5:              równanie (8.8)                             ▷ wyznacz rozwiązanie
6:          end for
7:          if Test Stopu Relaksacji Spełniony then
8:              zakończ pętle relaksacji - skocz do 1
9:          end if
10:     end for
11: end while
```

Wykonując relaksacje aż do osiągnięcia zbieżności i stosując stabilny schemat różnicowy do dyskretyzacji można udowodnić, że algorytm ten jest stabilny, dokładny i zbieżny [R.A83].

8.3 One-Step Relaxation

Metoda OSR (*ang. One-Step Relaxation*) [B.H85b, B.H85a] różni się od ITA wprowadzeniem pętli iteracji NR wewnątrz pętli relaksacji i wykonaniem kilku iteracji NR. Równania sieci po zdyskretyzowaniu i zlinearyzowaniu (8.3) rozwiązywane są algorytmem 23. Algorytm ten wymaga większych nakładów obliczeniowych ze względu na wewnętrzną pętlę NR. Częściowe usunięcie tego problemu można uzyskać przez zastosowanie efektywnych metod predykcji i efektywnych iteracji NR. Ponadto test stopu pętli NR powinien być sprawdzany dla każdego węzła przez porównanie wartości napięcia w kolejnych relaksacjach.

8.4 Metoda relaksacji przebiegów

Metoda relaksacji przebiegów WR (*ang. Waveform Relaxation*) różni się od metody OSR tym, że rozdzielenie węzłów następuje na poziomie równań różniczkowych. Równania sieci zdyskretyzowane i zlinearyzowane (8.3) rozwiązywane są algorytmem 24.

Pętla relaksacji jest najbardziej zewnętrzną pętlą algorytmu, w której następuje rozdzielenie węzłów (pętla węzłów). Analiza czasowa wykonywana jest dla każdego węzła (linia2) w pełnym zakresie analizy (pętla czasu - linia 3) $(0, t_{max})$. W przypadku układów nieliniowych równania zlinearyzowane rozwiązuje się metodą NR. Warunki konieczne zbieżności metody podano w [E.L82, C.A86, MP89]. Zbieżność metody gwarantują uziemione pojemności dołączone do każdego węzła układu i przyjęcie odpowiednio małego kroku h, co gwarantuje dominację głównej przekątnej. Warunki te są szczególnymi przypadkami ogólnej teorii zbieżności omówionej w rozdziale 8.8. Wydajność metody jest podobna do metody OSR - rozdział 8.3. Metoda ta może być w prosty sposób rozszerzona na metodę blokową przez wprowadzenie zamiast rozwiązywania równania, układu równań w pętli NR. Jeżeli macierz pochodnych ładunków względem sterowań

Algorytm 23 One-Step Relaxation

1: **while** $t_m < t_{max}$ **do** ▷ pętla czasu
2: przewidywanie (predykcja) $v^{(0)}_{1,t_m}, \ldots, v^{(0)}_{n,t_m}$
3: **for** $k = 1, k_{max}$ **do** ▷ pętla relaksacji
4: **for** $i = 1, n$ **do** ▷ pętla węzłów
5: **for** $p = 1, p_{max}$ **do** ▷ pętla NR
6: równanie (8.3)
7: **if** Test Stopu NR Spełniony **then**
8: zakończ pętle NR - skocz do 4
9: **end if**
10: **end for**
11: **end for**
12: **if** Test Stopu Relaksacji Spełniony **then**
13: zakończ pętle relaksacji - skocz do 1
14: **end if**
15: **end for**
16: **end while**

Algorytm 24 Waveform Relaxation Analysis

1: **for** $k = 1, k_{max}$ **do** ▷ pętla relaksacji
2: **for** $i = 1, n$ **do** ▷ pętla węzłów
3: **while** $t_m < t_{max}$ **do** ▷ pętla czasu
4: przewidywanie (predykcja) $v^{(0)}_{i,t_m}$
5: **for** $p = 1, p_{max}$ **do** ▷ pętla NR
6: równanie (8.3)
7: **if** Test Stopu NR Spełniony **then**
8: zakończ pętle NR - skocz do 3
9: **end if**
10: **end for**
11: **end while**
12: **end for**
13: **if** Test Stopu Relaksacji Spełniony **then**
14: zakończ pętle relaksacji - skocz do 17
15: **end if**
16: **end for**
17: KONIEC

$dq(v)/dv$ jest ciągła, $f(v,t)$ jest różniczkowalna i istnieje ścieżka pojemności z każdego węzła sprzęgającego do masy, to metoda ta jest zbieżna do rozwiązania z jakiegokolwiek punktu początkowego.

8.4.1 Windowing

Teoria zbieżności metod WR[J.W85a] gwarantuje zbieżność, lecz nic nie mówi o szybkości zbieżności. Zbieżność osiągana jest zwykle w kilku relaksacjach. W trakcie badań zaobserwowano, że dla układów z silnymi pętlami sprzężenia zwrotnego liczba relaksacji potrzebnych do uzyskania zbieżności jest proporcjonalna do interwału czasowego analizy $[0,T]$ [J.W83]. W celu przyśpieszenia analizy zakres $[0,T]$ podzielono na mniejsze odcinki czasu $[0,T_1],[T_1,T_2],\ldots,[T_{m-1},T_m]$, zwane oknami czasowymi (*ang. windows*). Analiza WR wykonywana jest w poszczególnych oknach czasowych. Początkowo wybierane jest okno $[0,T_1]$ i analiza przebiega aż do uzyskania zbieżności w tym oknie. Następnie wybiera się kolejne okno $[T_1,T_2]$ i wykonuje się analizę w tym oknie aż do uzyskania zbieżności. Proces ten powtarza się do momentu przeanalizowania wszystkich okien czasowych z zadanego przedziału $[0,T]$. Zbieżność analizy jest tym szybsza im okna są mniejsze. Dlatego długość okna czasowego jest ważnym czynnikiem decydującym o szybkości analizy. Technika ta jest obecnie standardową metodą przyśpieszania zbieżności metod WR.

8.4.2 Waveform-Relaxation-Newton Analysis

Działanie metody Waveform-Relaxation-Newton (WRN) [RJ90][A.R85, LG64] jest podobne do metody WR tyle, że nieliniowe równania różniczkowe są po rozdzieleniu linearyzowane (f_i względem z_i) i wprowadzana jest pętla iteracji NR. Każde równanie zlinearyzowane rozwiązywane jest osobno z indywidualnym krokiem czasowym. Rozwiązania poszczególnych równań używane są przez pozostałe równania w trakcie analizy w obrębie okna czasowego. Dokładne rozwiązanie poszczególnych równań wymaga różnej liczby kroków czasowych. Proces rozwiązania powtarza się do momentu uzyskania zbieżności dla wszystkich równań (ustaleniu się poszczególnych przebiegów czasowych *ang. waveforms*).

Algorytm 25 próbuje zredukować nakłady obliczeniowe przez mniej dokładne obliczenia w obrębie okna czasowego w kilku pierwszych iteracjach NR. W tym celu najpierw równania sieci (1.1) są rozdzielane, a potem linearyzowane. Otrzymane równania liniowe, z czasem jako zmienną, rozwiązywane są metodą WR (*ang. waveform-Gauss-Seidel* lub *ang. waveform-Jacobi*). Ze względu na to, że większość układów jest nieliniowa, takie rozwiązanie równań zlinearyzowanych prowadzi tylko do aproksymacji rozwiązań przebiegów czasowych w każdej relaksacji.

8.4.3 Newton-Waveform-Relaxation Analysis

W odróżnieniu od metody WR lub WRN, algorytm NWR rozwiązuje metodą WR zlinearyzowane metodą NR równania sieci (1.1). Najbardziej zewnętrzną pętlą jest pętla NR. Zlinearyzowane równanie, dla każdego węzła sieci, rozwiązywane jest w pełnym zakresie czasu analizy metodą WR. Po zlinearyzowaniu równania sieci oryginalnej (8.2) otrzymamy:

Algorytm 25 Waveform Relaxation Newton

```
1:  for k = 1, k_max do                                   ▷ pętla relaksacji
2:      for i = 1, n do                                   ▷ pętla węzłów
3:          for p = 1, p_max do                           ▷ pętla NR
4:              while t_m < t_max do                      ▷ pętla czasu
5:                  linearyzacja
6:                  przewidywanie (predykcja) v_{i,t_m}^{(0)}
7:                  równanie (8.8)
8:              end while
9:          end for
10:     end for
11:     if Test Stopu Spełniony then
12:         zakończ pętle relaksacji - skocz do 15
13:     end if
14: end for
15: KONIEC
```

$$\frac{d}{dt}\left[q(v_{i,t_m}^{(p-1)},v)+\underbrace{\frac{\delta q(v_{i,t_m}^{(p-1)},v)}{\delta v_{i,t_m}^{(p-1)}}(v_{i,t_m}^{(p)}-v_{i,t_m}^{(p-1)})}_{Q(v_{i,t_m}^{(p-1)})\cdot x_{i,t_m}^{(p-1)}}\right] + \underbrace{\frac{\delta f(v_{i,t_m}^{(p-1)},v,t_m)}{\delta v_{i,t_m}^{(p-1)}}(v_{,t_m}^{(p)}-v_{i,t_m}^{(p-1)})}_{F(v_{i,t_m}^{(p-1)})\cdot x_{i,t_m}^{(p-1)}}+$$
$$+\dot{q}(v_{i,t_m}^{(p-1)},v)+f(v_{i,t_m}^{(p-1)},v,t_m) = 0 \tag{8.9}$$

Równania zlinearyzowane tworzą zastępczą dynamiczną liniową sieć elektryczną, której elementy zależne są od czasu poprzez $v_{i,t_m}^{(p-1)}$. Jeżeli wprowadzimy oznaczenie $x_{i,t_m}^{(p-1)} = v_{i,t_m}^{(p)} - v_{i,t_m}^{(p-1)}$, to równanie (8.9) można przepisać do postaci:

$$\begin{aligned}\frac{d}{dt}\left[Q(v_{i,t_m}^{(p-1)})\cdot x_{i,t_m}^{(p-1)}\right]+F(v_{i,t_m}^{(p-1)})\cdot x_{i,t_m}^{(p-1)} &= -\left[\dot{q}(v_{i,t_m}^{(p-1)},v)+f(v_{i,t_m}^{(p-1)},v,t_m)\right]\\ v_{i,t_m}^{(p)} &= v_{i,t_m}^{(p-1)}+x_{i,t_m}^{(p-1)}\end{aligned} \tag{8.10}$$

Z pierwszego równania należy obliczyć taką wartość kroku Newtona $x_{i,t_m}^{(p-1)}$, który przesunie wartość $v_{i,t_m}^{(p-1)}$ bliżej rozwiązania. Do wyznaczenia $x_{i,t_m}^{(p-1)}$ można użyć metody iteracyjnej (4.23) (rozdział 4.2) ogólnej postaci $x^{k,(p-1)} = C\cdot x^{k-1,(p-1)}+E$. Pojedyncze równanie sieci przyjmuje postać (8.11).

$$\begin{aligned}v_{i,t_m}^{k,(p)} &= v_{i,t_m}^{k-1,(p-1)} - \frac{\dot{q}(v_{i,t_m}^{(p-1)},v)+f(v_{i,t_m}^{(p-1)},v,t_m)}{\dot{Q}(v_{i,t_m}^{(p-1)})+F(v_{i,t_m}^{(p-1)})}\\ v_{tm}^{k-1,(p-1)} &= \left[v_{1,t_m}^{k-1,(p-1)},\ldots,v_{i,t_m}^{k-1,(p-1)},v_{i+1,t_m-1}^{k-1,(p-1)},\ldots,v_{n,t_m-1}^{k-1,(p-1)}\right]\end{aligned} \tag{8.11}$$

Algorytm 26 NWR oblicza rozwiązanie poprzez wyznaczenie przybliżeń wartości przebiegów w oknie czasowym zdeterminowanych kolejnymi krokami Newtona $x_{i,t_m}^{(p-1)}$.

Algorytm 26 Newton Waveform Relaxation

1: **for** $p = 1, p_{max}$ **do** ▷ pętla NR
2: **for** $k = 1, k_{max}$ **do** ▷ pętla relaksacji
3: **for** $i = 1, n$ **do** ▷ pętla węzłów
4: linearyzacja i-tego równania (8.2) względem v i $\dot{v}$
5: **while** $t_m < t_{max}$ **do** ▷ pętla czasu
6: przewidywanie (predykcja) $v_{i,t_m}^{(0)}$
7: rozwiązanie równania liniowego postaci (8.11)
8: **end while**
9: **end for**
10: **if** Test Stopu Relaksacji Spełniony **then**
11: zakończ pętle relaksacji - skocz do 1
12: **end if**
13: **end for**
14: **end for**

Jeśli metoda ta jest dobrze zaimplementowana, to redukcja nakładów obliczeniowych osiągana jest przez zmniejszenie liczby wykonywanych obliczeń funkcji układowych, gdyż z reguły szybciej wykonywana jest wewnętrzna pętla relaksacji (równania liniowe) niż pętla zewnętrzna wymagająca obliczenia modelu elementu (1.1). Z tego powodu, dla układów z tranzystorami MOS, algorytm NWR jest dużo szybszy od WRN.

Wadą metody NWR są duże wymagania na zajętość pamięci ze względu na konieczność zapamiętywania pochodnych cząstkowych elementów dla każdego obliczanego punktu czasowego.

8.5 Analiza czasowa kierowana zdarzeniami

Metoda analizy czasowej kierowanej zdarzeniami ED ITA (*ang. Event-Driven Iterated Timing Analysis*) jest jedną z najbardziej efektywnych. Jej autorami są: A.R.Newton, A.L.Sangiovanni-Vincentelli. Opublikowana została w pracy [ARN83]. Dalszy rozwój metody został wykonany przez J.Ogrodzkiego [J.O95] oraz autora [Pla01a]. W metodzie węzły układu analizowane są z indywidualnym krokiem czasowym. Wykorzystane są techniki usypiania węzłów. W trakcie procesu analizy generowane są zdarzenia (definicja 8.5.1) charakteryzowane dwoma wielkościami [indeks węzła, czas analizy].

Definicja 8.5.1. *Zdarzeniem nazywamy proces wyznaczenia sygnału określonego węzła w określonym punkcie czasowym.*

Zdarzenia opisują zarówno propagację sygnału w układzie jak i dynamiczne zachowanie układu w trakcie analizy. Działanie oryginalnego algorytmu 27 opiera się na dwóch listach:

$E_A(t_m)$ jest listą węzłów, na których napięcie osiągnęło zbieżność w danej chwili czasowej,

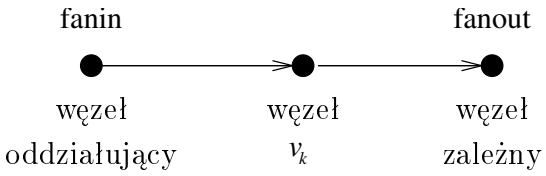
$E_B(t_m)$ jest listą węzłów, które nie osiągnęły zbieżności w danej relaksacji i czekają na następną relaksację.

Listy te umożliwiają wybranie do analizy tylko tych węzłów, które w danej chwili czasowej biorą udział w propagacji sygnału.

Algorytm 27 Event-Driven Iterated Timing Analysis

```
1: t_m = 0
2: while t_m < t_stop do                                   ▷ pętla czasu
3:     while E_A(t_m) nie pusta do                         ▷ pętla relaksacji
4:         for i = 1, n do                                 ▷ pętla węzłów
5:             oblicz napięcia metodą GSN
6:             if Test Stopu Relaksacji Spełniony then
7:                 dla i-tego węzła
8:                 wyznacz krok h_i oraz czas t_s = t_m + h_i
9:                 wstaw i-ty węzeł na listę E_A(t_s)
10:            else
11:                wstaw i-ty węzeł na listę E_B(t_m)
12:                wstaw węzły zależne od i na listę E_A(t_m)
13:            end if
14:        end for
15:        kopiuj E_B do E_A(t_m), wyczyść E_B)
16:    end while
17: end while
```

Działanie algorytmu jest następujące. W pierwszej chwili t_1 wszystkie węzły znajdują się na liście $E_A(t_1)$. W pętli relaksacji wyznaczane są wartości sygnałów w węzłach z listy E_A. Do obliczeń stosuje się np. algorytm GSN (opisany w dodatku 4.2.6) z wieloma iteracjami NR. Jeżeli osiągnięto zbieżność, to po wyznaczeniu indywidualnego kroku czasowego h_i oraz czasu następnej analizy $t_s = t_m + h_i$, węzeł zostaje umieszczony na liście $E_A(t_s)$. Jeżeli jednak nie osiągnięto zbieżności, to jest on wstawiany na listę E_B i czeka na następną relaksację. Na listę E_A wstawiane są węzły zależne (*ang. fanouts* zgodnie z definicją 8.5.2 i rys. 8.1).



Rys. 8.1: Graficzna interpretacja węzłów oddziałujących i zależnych

Definicja 8.5.2. *Węzeł v_k nazywamy węzłem oddziałującym (ang. fanin) na węzeł v_i, określonym jako $v_k \in Fanin(v_i)$, jeżeli v_k bezpośrednio wpływa na v_i. Węzeł v_j nazywany węzłem zależnym (ang. fanout) od węzła v_i, określonym jako $v_j \in Fanout(v_i)$, jeżeli na węzeł ten bezpośrednio oddziałuje v_i.*

Po przeanalizowaniu wszystkich węzłów z listy E_A węzły oczekujące na liście E_B na kolejną relaksację, są kopiowane na listę E_A. Proces powtarza się do momentu, gdy lista E_A będzie pusta. Można wtedy przejść do analizy następnego zdarzenia z $E_A(t_{m+1})$. Indywidualny krok czasowy i analiza tylko tych węzłów, które odczuły zmiany sygnału istotnie przyśpiesza proces analizy. Do krytycznych zagadnień metody należą:

- trudności z efektywnym doborem kroku czasowego,

- efektywna gospodarka zdarzeniami, w tym także ograniczanie liczby generowanych zdarzeń,
- eliminacja zjawiska wzajemnego wstawiania się węzłów sprzężonych na listę E_A.

8.5.1 Zmodyfikowany algorytm ED ITA

Modyfikacje oryginalnego algorytmu ED ITA zostały zaproponowane przez J.Ogrodzkiego [J.O94]. Modyfikacje polegają na wprowadzeniu klasyfikacji węzłów układu i zdarzeń w sposób umożliwiający łatwiejsze określenie zależności pomiędzy węzłami i zdarzeniami. Działanie zmodyfikowanego algorytmu 28 oparte jest na czterech listach:

AL lista tymczasowa węzłów czekających na następną relaksację,

CEL lista węzłów, które mają być przeanalizowane w danej chwili czasowej,

NEL lista zdarzeń następnych,

EEL lista zdarzeń zewnętrznych, na której znajdują się węzły dołączone do zewnętrznych źródeł sygnału.

Algorytm 28 Zmodyfikowany algorytm Event-Driven ITA

```
1: Inicjalizuj NEL i AL jako puste
2: Węzły źródeł niezależnych wstaw chronologicznie na EEL t_m = 0
3: while t_m < t_stop do                                   ▷ pętla czasu
4:     wybierz najbliższe zdarzenie z list NEL i EEL (czas t_m)
5:     wstaw węzły oczekujące na analizę w chwili t_m na CEL
6:     while CEL nie pusta do                              ▷ pętla relaksacji
7:         for każdego węzła na CEL do                     ▷ pętla węzłów
8:             oblicz napięcie węzła metodą ITA lub OSR
9:             if Test Stopu Relaksacji Spełniony then
10:                dla i-tego węzła wyznacz krok h_i oraz czas t_i = t_m + h_i
11:                wstaw i-ty węzeł na NEL z czasem t_i
12:            else
13:                wstaw i-ty węzeł na listę AL
14:                wstaw węzły zależne od i na CEL
15:            end if
16:        end for
17:        wyczyść CEL, kopiuj AL na CEL, wyczyść AL
18:    end while
19: end while
```

Przedstawiony tutaj algorytm (z niewielkimi modyfikacjami) został praktycznie zbadany [T.C97]. Krytycznym punktem tego algorytmu okazała się efektywna gospodarka zdarzeniami i sposób doboru kroku czasowego analizy. Dla układów CMOS analiza czasowa wykazywała przyśpieszenie w granicach od kilku do kilkuset %. W przypadku analizy układów z tranzystorami MOS o dużych wartościach pojemności *pasożytniczych* algorytm okazał się nieefektywny. Otrzymywano niepoprawne wyniki analizy ze względu na brak mechanizmów cofania w czasie.

8.6 Analiza czasowa kierowana zdarzeniami układów analogowych

Algorytm ED-ITA umożliwia bardzo efektywną symulację układów z elementami unilateralnymi, takimi jak układy cyfrowe, układy MOS. W ogólnym przypadku układów analogowych jest trudny do zastosowania, ze względu na słabą zbieżność dla układów z elementami bilateralnymi. Elementy taki muszą być odpowiednio grupowane w bloki i ich równania rozwiązywane metodami klasycznymi, jako układ równań. Rozszerzenie metody na układy analogowe zostało zaproponowane w [Pla01a] i zaimplementowane w symulatorze OPTIMA v4 [Pla01b] oraz w jeszcze bardziej udoskonalonej formie w symulatorze *Dero* [Pla13]. Przedstawiany w pracy algorytm został zaimplementowany w symulatorze OPTIMA. Algorytm zmodyfikowany umożliwia 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 rozgrupowania 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 27 opiera się na dwóch listach:

Do działania algorytmu 29 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.

Algorytm 29 Zmodyfikowany 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
  skopiowanie węzłów z NEL(t) na EQ ▷ pobranie zdarzenia o czasie t
  dla każdego węzła w z EQ wyznacz wartość startową 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 brak zbieżności dla węzła w **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 it 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 zbieżnosci, odrzuć, krok wstecz
      wstaw węzły *niezbieżne* na NEL(t)
      wykonaj krok wstecz
    end if ▷ koniec pętli relaksacji
  end for
  przewidywanie nowych kroków analizy dla węzłów z listy EQ
  redukcja liczby zdarzeń i synchronizacja czasów analizy grup SCC i T
  generacja zdarzeń
  wydruk wyników analizy
  zerowanie listy EQ i pobranie czasu t najbliższego zdarzenia z NEL
until $t < t_{stop}$ & NEL niepusta ▷ koniec pętli zdarzeń
KONIEC

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 inicjalizacji 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 4 rozpoczyna się główna pętla czasowa algorytmu. Z wektora zdarzeń NEL 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 8. 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 4.1. 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 maksymalnej liczbie relaksacji k_{max} węzły, które nie uzyskały zbieżności są grupowane. Dla grupy ustawiany jest znacznik czasu życia grupy. Znacznik ten określa liczbę punktów czasowych analizy, po których grupę należy próbować podzielić 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 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. Algorytm ograniczania liczby zdarzeń redukuje zdarzenia nadmiarowe i zabezpiecza algorytm przed nadmiernym skracaniem kroku analizy.

- 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 wyprowadzenie wyników analizy.

- 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 wypadku algorytm kończy działanie.

8.7 Event-Driven Local Interactive Circuit Simulation

W latach 90-tych XX wieku opracowano nową technikę analizy EDLICS (*ang. Event-Driven Local Interactive Circuit Simulation*)[JCL94] opartą nie na rozwiązywaniu równań różniczkowych, lecz wykorzystującą zachowanie sieci elektrycznej. Realizuje się to przez propagację zmian (przyrostów) sygnałów w układzie (napięcia, prądu, ładunku) jako zdarzeń, podobnie jak w kierowanej zdarzeniami symulacji logicznej. Element sieci otrzymuje zdarzenie (zmianę sygnału), konwertuje je i rozsyła do swoich węzłów. Technika ta została zastosowana w symulatorze EDsim przeznaczonym do symulacji układów MOS.

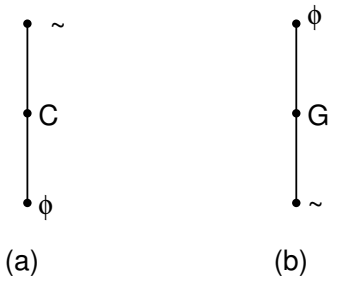
Zdarzenia węzłowe

W metodzie tej obwód elektryczny traktowany jest zgodnie z definicja 8.7.1.

Definicja 8.7.1. *Obwód elektryczny jest to graf $G = \{N, B\}$, gdzie N jest zbiorem węzłów a B jest zbiorem gałęzi elementów dołączonych do par węzłów. Każda gałąź sieci $b \in B$ posiada siłę $s \in S_B = \{\phi, G, C, \infty\}$, która wiąże przewodność z prądem i ładunkiem. Źródła prądowe mają wagę ϕ, gałęzie konduktancyjne mają wagę G, pojemności mają wagę C, masa ma specjalną wagę ∞.*

Relacje pomiędzy poszczególnymi wagami gałęzi przedstawiono poglądowo na rys. 8.2. Węzeł masy posiada nieskończenie duży prąd i ładunek; źródło prądu posiada zerową konduktancję; źródło napięciowe posiada nieskończony ładunek i przewodność. W przeciwieństwie do masy, źródło napięciowe ogranicza napięcia przez grupowanie

węzłów w specjalną strukturę zwaną super węzłem (*ang. supernode*). Jeżeli różnice pomiędzy węzłami w super węźle nie są rozpatrywane, wtedy super węzeł zachowuje się jak węzeł.



Rys. 8.2: Interpretacja wag dla ładunków (a) i prądów (b)

Zdarzenia będące zmianą napięcia w węźle są propagowane do wszystkich węzłów incydentnych z danym węzłem. Zdarzenia napięciowego nie potrzeba przesyłać do węzła połączonego z masą. Źródła prądowe ignorują zdarzenia napięciowe. Zdarzenie napięciowe jest bezpośrednio propagowane przez wszystkie węzły super węzła. Gałąź konduktancyjna przesyła zdarzenia do przyległych węzłów. W ten sam sposób kondensator obsługuje zmianę ładunku.

Ponieważ konduktancja nie może obsługiwać zdarzenia ładunkowego, zdarzenie ładunkowe jest obsługiwane przez kondensator, źródło prądowe lub masę. Najsilniejszy element *absorbuje* cały ładunek. Jeżeli elementy mają jednakową *siłę*, to ładunek jest dzielony pomiędzy nimi. Jeśli węzeł ma siłę ∞ (np. węzeł jest uziemiony), wtedy cały ładunek jest *absorbowany* i nie jest podejmowana żadna dalsza akcja. Jeżeli węzeł nie jest uziemiony, wtedy zmiana napięcia węzła jest wyznaczana na podstawie ładunku dzielonego w stosunku do wartości odpowiednich pojemności dołączonych do węzłów.

Dla zdarzeń *prądowych* węzły dołączone do pojemności powinny być traktowane jako węzły, które tworzą super węzeł pojemnościowy. Zdarzenie *prądowe* może być obsługiwane przez konduktancję, źródło prądowe lub masę zgodnie z rysunkiem 8.2.

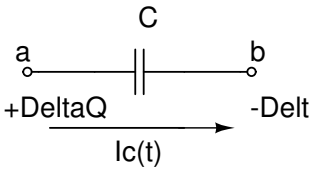
Zdarzenia gałęziowe

Przyrosty napięć lub prądów gałęziowych są obsługiwane jako zdarzenia gałęziowe. Jeżeli prąd gałęzi (źródła prądowego) zmieni się o ΔI, to zmiana ta jest propagowana do węzła *dodatniego* jako $+\Delta I$, natomiast do węzła *ujemnego* jako $-\Delta I$. Jeżeli jeden z węzłów ma wagę ∞ to nie ma potrzeby przesyłania przyrostu do tego węzła. Gałęziowe zdarzenie *napięciowe* jest dzielone pomiędzy poszczególne węzły super węzła sformowanego przez gałęzie napięciowe. Elementy dołączone do węzłów źródła determinują sposób podziału napięcia gałęzi pomiędzy węzły elementu.

Algorytm EDLICS

W celu znalezienia odpowiedzi układu, zmiany ładunków pojemności muszą być obliczane wraz z upływem czasu. Rozpatrzmy kondensator z rys. 8.3 analizowany z krokiem czasowym h.

Ładunek pojemności jest obliczany przez różniczkowanie płynącego prądu. Tak obliczony ładunek gromadzony ΔQ przesyłany jest jako zdarzenie do węzłów będących



Rys. 8.3: Gałąź pojemnościowa

zaciskami kondensatora: do węzła a jako $+\Delta Q$, natomiast do b jako $-\Delta Q$. Jeżeli jeden z węzłów ma wagę ∞ (jest uziemiony), to nie ma potrzeby obsługi tego zdarzenia, gdyż węzeł ten w całości *absorbuje* zmianę ładunku.

Jeżeli założymy, że do dyskretyzacji użyjemy schematu różnicowego zamkniętego Eulera, wtedy przyrosty napięć w węzłach a i b kondensatora z rys. 8.3 dane będą zależnościami (8.12) i (8.13).

$$\Delta v_a = I_C \frac{(\sum G_b + \frac{\sum C_b}{h}) - \frac{C}{h}}{(\sum G_a + \frac{\sum C_a}{h})(\sum G_b + \frac{\sum C_b}{h}) - \frac{C^2}{h^2}} \tag{8.12}$$

$$\Delta v_b = I_C \frac{(\sum G_a + \frac{\sum C_a}{h}) - \frac{C}{h}}{(\sum G_a + \frac{\sum C_a}{h})(\sum G_b + \frac{\sum C_b}{h}) - \frac{C^2}{h^2}} \tag{8.13}$$

gdzie: G_x i C_x są sumami konduktancji i pojemności widzianych na zacisku x. Jeżeli jeden z węzłów jest uziemiony, to można mu przypisać nieskończoną konduktancję i pojemność. W innych metodach symulacji wartości napięć są obliczane dla uziemionych pojemności. Wymagane jest połączenie węzłów z masą przez pojemność lub ścieżkę pojemności. Proponowany tutaj metoda nie wymaga dołączenia do węzła uziemionych pojemności. Jeżeli do węzła dołączonych jest kilka pojemności, to zmiana napięcia węzła jest sumą poszczególnych przyrostów wnoszonych przez wszystkie dołączone pojemności.

Symulator EDsim, w którym zaimplementowano tą technikę, jest około 10 razy szybszy od programu iSPLICE3 i około 20 razy szybszy od SPICE3 wykorzystanych do symulacji układów CMOS.

8.8 Ogólna teoria zbieżności technik relaksacyjnych

W metodach relaksacyjnych wykorzystywane są techniki iteracyjnego rozwiązywania równań. Metody te zostały opisane w rozdziale 4, gdzie podano także kryteria zbieżności (twierdzenie 4.2.1). Podstawowym kryterium zbieżności jest zagwarantowanie odpowiednio silnej dominacji głównej przekątnej macierzy układu, co z matematycznego punktu widzenia oznacza minimalizację promienia spektralnego macierzy charakterystycznej $\rho(C) < 1$ (warunek konieczny zbieżności). Im promień ten jest mniejszy, tym przekątna macierzy układu jest silniej dominująca i zbieżność jest szybsza. Pierwsza praca nad warunkami zbieżności ukazała się w 1982 [E.L82]. Pokazano w niej, że metoda relaksacji przebiegów (*ang. waveform relaxation*) jest zbieżna w dziedzinie czasu ciągłego startując z arbitralnie wybranego punktu, jeżeli do każdego węzła sieci dołączono uziemiony kondensator. W 1986 pokazano [C.A86], że metoda WR jest zbieżna

w dziedzinie czasu ciągłego startując z arbitralnie wybranego punktu, jeżeli każdy węzeł połączony jest z masą przez ścieżkę kondensatorów. W 1989 roku Desai [MP89] wykazał, że dla danego podziału blokowego każdy węzeł sprzęgający musi być dołączony do masy przez ścieżkę kondensatorów. W dziedzinie czasu *dyskretnego* zbieżność była gwarantowana, jeżeli warunki początkowe znajdowały się odpowiednio blisko rozwiązania. W 1998 roku opublikowano ogólną teorię [G.D98] dotyczącą zbieżności technik relaksacji przebiegów i wyprowadzono nowe - mniej rygorystyczne warunki zbieżności. Uwzględnia ona wszystkie opublikowane wcześniej warunki jako przypadki szczególne. Jej podstawę stanowi badanie trajektorii, po której *porusza* się rozwiązanie w procesie iteracyjnym.

Równania sieci

Załóżmy, że zastosujemy Zmodyfikowaną Metodę Potencjałów Węzłowych (ZMPW) [J.O95](rozdział 12.1) do układania równań sieci. Niech $x(t)$ będzie wektorem zmiennych sieciowych ZMPW dla $0 < t < T$, który podzielimy na dwa wektory

$$x(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \tag{8.14}$$

gdzie: $x(t)$ jest wektorem zmiennych, których pochodne występują w równaniach układu, $y(t)$ jest wektorem pozostałych zmiennych. Po takim podziale można przedstawić wektor zmiennych sieciowych w postaci (8.15).

$$w(t) = \begin{bmatrix} \dot{x}(t) \\ x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ x(t) \end{bmatrix} \tag{8.15}$$

Tak więc nieliniowe równania różniczkowe układu można zapisać w postaci rozdzielonej (8.16).

$$\begin{aligned} h(w(t),t) &= \begin{bmatrix} f(w(t),t) \\ g(x(t),t) \end{bmatrix} = 0 \\ x(0) &= x_0 \end{aligned} \tag{8.16}$$

gdzie: f zawiera tylko równania zależne od pochodnych zmiennych sieciowych, a g pozostałe równania.

Jeżeli podzielimy układ na podobwody (np. s i d) i do rozwiązania zastosujemy metodę relaksacji przebiegów czasowych (WR), to równania dla i-tego podobwodu mogą zostać zapisane w postaci (8.17).

$$\begin{aligned} h_i(w_{d,i}^{k+1}(t), w_{s,i}^{k+1}(t), w_{s,i}^{k}(t), t) &= 0 \\ x_{s,i}^{0}(t) &= x_{0,s,i}(t) \\ x_{d,i}^{k+1}(0) &= x_{d,i}^{k}(0) = x_{0,d,i}(0) \\ \\ x_{s,i}^{k+1}(0) &= x_{s,i}^{k}(0) = x_{0,s,i}(0) \end{aligned} \tag{8.17}$$

gdzie: argumenty wektora h są podwektorami $w(t)$, podobwód s reprezentowany jest przez niezależne źródła dołączone do wejść i-tego podobwodu. Po jednej iteracji (8.17)

dla wszystkich podobwodów zostanie wprowadzony współczynnik mapowania $x^k(t) \rightarrow x^{k+1}(t)$. Przejście to można zapisać ogólnie dla wszystkich podobwodów w formie układu równań (8.18).

$$\begin{aligned} h_{wr}(w^{k+1}(t), w^{k+1}(t), t) &= \begin{bmatrix} f_{wr}(w^{k+1}(t), w^k(t), t) \\ g(x^{k+1}(t), x^k(t), t) \end{bmatrix} = 0 \\ x^0(t) &= x_0(t) \\ x^{k+1}(0) &= x^k(0) = x_0(0) \end{aligned} \tag{8.18}$$

gdzie k jest numerem iteracji.

WR operator mapowania

Załóżmy n elementowy wektor $x(t)$ w pewnym punkcie t. Oznaczmy przestrzeń, do której należy wektor $x(t)$ jako R^n. Załóżmy, że wektor $x(\cdot)$ oznacza wektor wszystkich wartości $x(t)$ w przedziale $x \in [0, T]$. Przestrzeń tego wektora oznaczmy przez $R^{n \times t}$. Wtedy operator przeprowadzający $x^k(t) \rightarrow x^{k+1}(t)$ zgodnie z (8.18) można zapisać w postaci (8.19)

$$\begin{aligned} x^{k+1}(\cdot) &= \wedge(x^k(\cdot)) \\ x^0(\cdot) &= x_0(\cdot) \\ x^{k+1}(0) &= x^k(0) = x_0(0) \end{aligned} \tag{8.19}$$

gdzie $\wedge : D \subset R^{n \times t} \rightarrow R^{n \times t}$ jest operatorem funkcyjnym. Jeżeli $x^*(t)$ jest jedynym rozwiązaniem równania (8.16), to mamy zależność

$$x^*(\cdot) = \wedge(x^*(\cdot)) \tag{8.20}$$

Rozważmy teraz p i takie $\wedge$, oznaczone przez $\wedge^{(p)}$, że dla $p > 1$ otrzymamy

$$x^{k+p}(\cdot) = \wedge^{(p)}(x^k(\cdot)) \tag{8.21}$$

Rozpatrzmy teraz dwie trajektorie $x_a(\cdot)$ oraz $x_b(\cdot)$ w przestrzeni $D \subset R^{n \times t}$, dla których $x_a(0) = x_b(0)$ (te same warunki początkowe). Dla tych trajektorii można otrzymać trajektorie (8.22,8.23).

$$x_a^p(\cdot) = \wedge^{(p)}(x_a(\cdot)) \tag{8.22}$$

$$x_b^p(\cdot) = \wedge^{(p)}(x_b(\cdot)) \tag{8.23}$$

Różnica trajektorii, po których porusza się rozwiązanie ma postać:

$$\begin{aligned} \delta x(t) &= x_a(t) - x_b(t) \\ \delta x^{(p)}(t) &= x_a^{(p)}(t) - x_b^{(p)}(t) \end{aligned}$$

Jeżeli istnieje takie $c < 1$, że spełniony jest warunek (8.24), to ciąg iteracji (8.21) jest zbieżny do $x^*(\cdot)$ z jakiegokolwiek punktu startowego $x^0(\cdot) \in D$.

$$\| \delta x^{(p)}(t) \| \leq c \| \delta x(t) \| \tag{8.24}$$

Zbieżność liniowych równań różniczkowych zależnych od czasu

W tym rozdziale zostaną wyprowadzone warunki zbieżności dla układów równań różniczkowych liniowych zależnych od czasu. Układ równań zostanie wyprowadzony z równań nieliniowych postaci (8.17). W tym celu przetransformujemy każde równanie $h_{wr,i}$ (jako funkcję zależną od wektorów $w^{k+1}(t)$ i $w^k(t)$) do funkcji zależnej od współczynników $\gamma_i(t)$, gdzie $0 < \gamma_i(t) < 1$. Później zróżniczkujemy każde równanie $h_{wr,i}$ względem γ_i i zastosujemy twierdzenie Rolle'a [I.D85], aby otrzymać żądany układ równań. Korzystając ze zdefiniowanych wcześniej trajektorii (8.22) i (8.23) dla $p = 1$, otrzymamy po transformacji wektory (8.25).

$$\begin{aligned} w_i(\gamma_i(t)) &= \gamma_i(t)w_a(t) + (1-\gamma_i(t))w_b(t) \\ w_i^{(1)}(\gamma_i(t)) &= \gamma_i(t)w_a^{(1)}(t) + (1-\gamma_i(t))w_b^{(1)}(t) \end{aligned} \tag{8.25}$$

Równania te można zastąpić równaniami układu podzielonego na podukłady po zastosowaniu WR $h_{wr,i}$.

$$h_{wr,i}\left(\gamma_i(t), t\right) = h_{wr,i}\left(w_i^{(1)}(t), w_i(t), t\right) \tag{8.26}$$

Jeżeli przyjmiemy, że każda funkcja $h_{wr,i}$ jest ciągła i różniczkowalna względem $w^{(1)}(t)$ i $w(t)$, to każda $h_{wr,i}$ będzie ciągła i różniczkowalna względem $\gamma_i(t)$ ponieważ $w_i(t)$ i $w^{(1)}(t)$ są ciągłe i różniczkowalne względem $\gamma_i(t)$. Na końcach przedziału funkcja ma takie same wartości $h_{wr,i}\left(0,t\right) = h_{wr,i}\left(1,t\right)$, więc stosując twierdzenie Rolle'a [I.D85] otrzymamy:

$$\frac{\delta h_{wr,i}\left(\gamma_i(t), t\right)}{\gamma_i(t)}|_{\gamma_i(t)=\theta_i(t)} = 0 \tag{8.27}$$

dla $0 < \theta_i(t) < 1$. Na podstawie (8.27) otrzymamy szukane równania różniczkowe dla każdego i

$$h_{k+1,i,j}(t)\delta w^{(1)}(t) + h_{k,i,j}(t)\delta w(t) = 0 \tag{8.28}$$

$$h_{k+1,i,j}(t) = \frac{\delta h_{wr,i}\left(w_i^{(1)}(\gamma_i(t)), w_i(\gamma_i(t)), t\right)}{\delta w_j^{(1)}(t)}|_{\gamma_i(t)=\theta_i(t)} \tag{8.29}$$

$$h_{k,i,j}(t) = \frac{\delta h_{wr,i}\left(w_i^{(1)}(\gamma_i(t)), w_i(\gamma_i(t)), t\right)}{\delta w_j(t)}|_{\gamma_i(t)=\theta_i(t)} \tag{8.30}$$

$$\delta w(t) = w_a(t) - w_b(t) = \begin{bmatrix} \dot{x}_a(t) \\ x_a(t) \\ y_a(t) \end{bmatrix} - \begin{bmatrix} \dot{x}_b(t) \\ x_b(t) \\ y_b(t) \end{bmatrix} = \begin{bmatrix} \delta\dot{x}(t) \\ \delta x(t) \end{bmatrix} \tag{8.31}$$

$$\delta w^{(1)}(t) = w_a^{(1)}(t) - w_b^{(1)}(t) = \begin{bmatrix} \dot{x}_a^{(1)}(t) \\ x_a^{(1)}(t) \\ y_a^{(1)}(t) \end{bmatrix} - \begin{bmatrix} \dot{x}_b^{(1)}(t) \\ x_b^{(1)}(t) \\ y_b^{(1)}(t) \end{bmatrix} = \begin{bmatrix} \delta\dot{x}^{(1)}(t) \\ \delta x^{(1)}(t) \end{bmatrix} \tag{8.32}$$

$$\delta w^{(1)}(0) = \delta w(0) = 0 \tag{8.33}$$

Równania (8.28) można zapisać w formie blokowej

$$\begin{bmatrix} C_{f,x,k+1}(t) & G_{f,x,k+1}(t) & G_{f,y,k+1}(t) \\ 0 & G_{g,x,k+1}(t) & G_{g,y,k+1}(t) \end{bmatrix} \begin{bmatrix} \delta\dot{x}^{(1)}(t) \\ \delta x^{(1)}(t) \\ \delta y^{(1)}(t) \end{bmatrix} + \\ + \begin{bmatrix} C_{f,x,k}(t) & G_{f,x,k}(t) & G_{f,y,k}(t) \\ 0 & G_{g,x,k}(t) & G_{g,y,k}(t) \end{bmatrix} \begin{bmatrix} \delta\dot{x}(t) \\ \delta x(t) \\ \delta y(t) \end{bmatrix} = 0 \tag{8.34}$$

Równanie to może być rozwiązane dla $\delta x^{(1)}(t)$ względem $\delta x(t)$. Otrzymujemy wtedy

$$\delta x^{(1)}(t) = E(t)\delta x(t) + \int_0^t F(t,\tau)\delta x(\tau)d\tau \tag{8.35}$$

gdzie:

$$\begin{aligned} E(t) &= \begin{bmatrix} B(t) & 0 \\ M(t) & N(t) \end{bmatrix} \\ B(t) &= -C^{-1}_{f,x,k+1}(t)C_{f,x,k}(t) \\ N(t) &= -G^{-1}_{g,y,k+1}(t)G_{g,y,k}(t) \\ M(t) &= G^{-1}_{g,y,k+1}(t)G_{g,x,k+1}(t)C^{-1}_{f,x,k+1}(t)C_{f,x,k}(t) - G^{-1}_{g,y,k+1}(t)G_{g,x,k}(t) \end{aligned}$$

$F(t,\tau)$ jest macierzą odpowiedzi impulsowej układu (8.35). O macierzach $C_{f,x,k+1}(t)$ i $G_{g,y,k+1}(t)$ zakłada się, że są odwracalne i mają ograniczoną normę. Macierz $E(t)$ odpowiada za następujące zjawiska:

1. sprzężenia typu napięcie-ładunek pływających kondensatorów w podukładzie,
2. sprzężenia stałoprądowe pomiędzy węzłami i gałęziami sieci, które nie zawierają elementów magazynujących energię.

Macierz $F(t,\tau)$ odpowiada za wymianę energii pomiędzy elementami magazynującymi energię i elementami nie magazynującymi energii w dziedzinie DC. Jeżeli do każdego węzła sieci dołączony jest kondensator, to $E(t) = B(t)$. Jeżeli w sieci nie ma elementów magazynujących energię, to $E(t) = N(t)$ i $F(t,\tau) = 0$. Macierz $M(t)$ odpowiada za napięcia węzłowe i prądy gałęziowe w dziedzinie DC wektorów $x(t)$ i $y(t)$. Wprowadźmy teraz rozwiązanie równania (8.35) dla $\wedge^{(p)}$ i $p > 1$

$$\delta x^{(p)}(t^{(p)}) = \left[\prod_{i=p}^{1} E^{(p)}(t^{(p)})\right] \delta x(t^{(p)}) + \ldots + \int_0^{t^{(p)}} F^{(p)}(t^{(p)}, t^{(p-1)}) \ldots \\ \times \int_0^{t^{(1)}} F^{(1)}(t^{(1)}, t^{(0)})\delta x(t^{(0)})dt^{(0)} \ldots dt^{(p-1)} \tag{8.36}$$

Teraz rozważania pójdą w kierunku identyfikacji ograniczeń w topologii układu tak, aby współczynnik c występującego w równaniu (8.24) spełniał warunek $c < 1$. W tym celu zdefiniujmy normy

$$\begin{aligned} \| x(\cdot) \|_e &= \max(e^{-\lambda t} \| x(t) \|) \\ \| x(\cdot) \|_m &= \max_t(\| x(t) \|) \\ \| F(\cdot,\cdot) \|_m &= \max_{t,\tau}(\| F(t,\tau) \|) \end{aligned} \tag{8.37}$$

Skorzystamy teraz z zależności

$$e^{-\lambda t} \int_0^t e^{\lambda \tau} d\tau \leq \frac{1}{\lambda} \tag{8.38}$$

i otrzymamy

$$\| \delta x^{(1)}(\cdot) \|_e \leq c \| \delta x(\cdot) \|_e \tag{8.39}$$

gdzie:

$$c = \| E(\cdot) \|_m + \frac{1}{\lambda} \| F(\cdot,\cdot) \|_m \tag{8.40}$$

Tak długo jak macierze funkcji $E(t)$ i $F(t,\tau)$ są ograniczone i $\| E(t) \| < 1$, dla wszystkich t, τ można wybrać takie λ, że $c < 1$ i $\wedge$ jest odwzorowaniem zwężającym (*ang. contraction*). Taka sama analiza dla $\wedge^{(p)}$ i równania (8.36) daje

$$c = \| \prod_{i=p}^{1} E^{(i)}(\cdot) \|_m + \frac{1}{\lambda} \left[\ldots + \prod_{i=p}^{1} \| F^{(i)}(\cdot,\cdot) \|_m \right] \tag{8.41}$$

Wybranie odpowiednio dużego współczynnika λ gwarantuje, że $c < 1$.

Warunki zbieżności

Z wyprowadzonych wcześniej zależności wynika, że kluczem do wyprowadzenia warunków zbieżności jest macierz $E(t)$. W rozwiązaniu równania (8.35) zostały poczynione założenia, że macierz odwrotna $G_{g,y,k+1}$ istnieje i ma ograniczoną normę - co jest słuszne w rzeczywistych układach. Założenie o istnieniu macierzy odwrotnej $C_{f,y,k+1}$ o ograniczonej normie zostanie dowiedzione na podstawie topologii układu. Jeżeli podobwody tworzone są zgodnie z wcześniejszymi założeniami dla sieci z kondensatorami i indukcyjnościami[3], prądy indukcyjności nie są prądami sterującymi podobwodu i każdy węzeł dołączony do kondensatora jest dołączony do ścieżki kondensatorów połączonych z masą, to będzie istniała macierz odwrotna $C_{f,y,k+1}$ o ograniczonej normie. Można zatem zapisać warunki zbieżności w postaci twierdzenia.

Twierdzenie 8.8.1. *Dla równania (8.17) zakładamy co następuje.*

1. *Skalarny składnik h_{wr} ma ciągłe i ograniczone pierwsze pochodne po skalarnych komponentach $w^{k+1}(t)$ i $w^k(t)$.*

2. *Każdy węzeł dołączony do kondensatora jest dołączony do ścieżki kondensatorów połączonych z masą.*

[3]Przez podobwody z kondensatorami i indukcyjnościami rozumiane są podobwody powstałe przez usunięcie wszystkich innych elementów.

3. Istnieje macierz odwrotna $G_{g,y,k+1}(t)$ o ograniczonej normie.

4. Podział układu wykonywany jest zgodnie z poniższymi założeniami:

(a) nie ma funkcjonalnej zależności pomiędzy żadnymi składnikami wektora $y^{k+1}(t)$ i $y^k(t)$. Tak więc zależności zawierające $y^k(t)$ i $y^{k+1}(t)$ nie mogą występować w tych samych składowych h_{wr}, czyli $G_{g,y,k}(t) = 0$.

(b) nie ma żadnych zależności zawierających prądy indukcyjności w $x^k(t)$ w żadnej składowej h_{wr}.

Przy takich założeniach można znaleźć dolne ograniczenie p_{lower} współczynnika p takiego, że $\wedge^{(p)}$ jest odwzorowaniem zwężającym dla wszystkich $p \geq p_{lower}$.

Dowód:

$$\prod_{i=p}^{1} E^{(i)}(t) \quad = \quad \begin{bmatrix} \prod_{i=p}^{1} B^{(i)}(t) & 0 \\ M^{(i)}(t) \prod_{i=p}^{1} B^{(i)}(t) + N^{(i)}(t) \dots & \prod_{i=p}^{1} N^{(i)}(t) \end{bmatrix} \tag{8.42}$$

Jeżeli układ jest podzielony zgodnie z warunkiem 4 twierdzenia 8.8.1, to $N(t) = 0$ i

$$\prod_{i=p}^{1} E^{(i)}(t) \quad = \quad \begin{bmatrix} \prod_{i=p}^{1} B^{(i)}(t) & 0 \\ M^{(p)}(t) \prod_{i=p-1}^{1} B^{(i)}(t) & 0 \end{bmatrix} \tag{8.43}$$

Biorąc $\| \cdot \|$ jako normę L_∞ otrzymamy

$$\| \prod_{i=p}^{1} E^{(i)}(t) \| \quad \leq \quad \max \left(\| \prod_{i=p}^{1} B^{(i)}(t) \|_\infty, \| M^{(p)}(t) \|_\infty \| \prod_{i=p-1}^{1} B^{(i)}(t) \|_\infty \right) \tag{8.44}$$

Jeżeli zdefiniujemy $\beta = \min \left(1, \frac{1}{\|M^{(p)}(t)\|_\infty}\right)$, to należy znaleźć taką wartość p, która spełnia zależność:

$$\| \prod_{i=p}^{1} B^{(i)}(t) \|_\infty \leq \alpha \leq \beta \tag{8.45}$$

gdzie: $\alpha > 0$.

Jeżeli każdy węzeł dołączony do kondensatora jest dołączony do masy przez ścieżkę kondensatorów, to macierz $C(t)$ będzie miała silnie dominującą przekątną dla przynajmniej i odnoszącego się do węzła dołączonego przez ścieżkę pojemności do masy jak i przez gałąź prądową cewki. Jeżeli macierz $C(t)$ przedstawimy w postaci macierzy blokowo-diagonalnej $PC(t)P^T$, gdzie każdy węzeł w bloku dołączony jest do masy przez ścieżkę kondensatorów lub cewkę z prądem $x_j(t)$, to będzie istniała C_j^{-1} i konsekwencji $C_{f,x,k+1,j}^{-1}$. Ponieważ $C(t)$ jest macierzą blokowo-diagonalną, więc istnieje C^{-1} i $C_s f, x, k+1^{-1}$. W ten sam sposób możemy przedstawić iloczyn $\prod_{i=p}^{1} B^{(i)}(t)$ jako $P \prod_{i=p}^{1} B^{(i)}(t) P^T$, gdzie macierz $B^{(i)}(t) = -C_{f,x,k+1,j}^{(i)-1}(t) C_{f,x,k,j}^{(i)}(t)$. Można pokazać, że

liczba p_{min} jest równa lub większa ilości kondensatorów w najdłuższej ścieżce kondensatorów z jakiegokolwiek węzła do masy w układzie. Otrzymamy wtedy dla każdego j

$$\| \prod_{i=p}^{1} B^{(i)}(t) \|_{\infty} \leq \gamma < 1 \tag{8.46}$$

gdzie:

$$\gamma = 1 - \min_{i,j,l,t} \left(|\frac{c_{i,j}^{(l)}(t)}{c_{i,i}^{(l)}(t)}| : c_{i,j}^{(l)}(t) \neq 0 \right)^{p_{min}-1} \times \min_{i,j,l,t} \left(1 - \sum_{j \neq i} |\frac{c_{i,j}^{(l)}(t)}{c_{i,i}^{(l)}(t)}| : |c_{i,i}^{(l)}(t)| > \sum_{j \neq i} |c_{i,j}^{(l)}(t)| \right) \tag{8.47}$$

i $1 \leq l \leq p_{min}$.
Korzystając z

$$\| \prod_{i=l \times p}^{1} B^{(i)}(t) \| \leq \prod_{j=l}^{1} \| \prod_{i=p_{min}}^{1} B^{(i)}(t) \| \leq \gamma^{l} \tag{8.48}$$

można wybrać takie p_{lower} jako najmniejszą liczbę całkowitą, dla której

$$\alpha = \gamma^{p_{lower}} < \beta \tag{8.49}$$

Tak więc dla każdego $p > p_{lower}$ otrzymamy

$$\alpha = \gamma^{p} < \beta \tag{8.50}$$

co kończy dowód.

Warunki 2 i 4b twierdzenia 8.8.1 gwarantują istnienie macierzy odwrotnej $C_{f,x,k+1}(t)$. Warunek 4a gwarantuje eliminację mapowania $y^k(t) \rightarrow y^{k+1}(t)$ opisanego macierzą $E(t)$.

8.9 Rozszerzenie klasy sieci

Jak wspomniano w rozdziale 8.1 klasa sieci akceptowanych przez relaksacyjne metody analizy ograniczona jest do układów zbudowanych z:

1. nieliniowych pojemności sterowanych co najwyżej napięciowo,
2. nieliniowych konduktancji sterowanych co najwyżej napięciowo,
3. przez zastosowanie *triku* możliwe jest zastosowanie uziemionych źródeł napięciowych.

Występowanie silnych sprzężeń międzywęzłowych, na skutek wprowadzenia do układu elementów bilateralnych, powoduje słabą zbieżność metod relaksacyjnych. Aby temu zaradzić i umożliwić analizę sieci w skład których wchodzą elementy nieakceptowane, stosuje się blokowe metody analizy. Węzły silnie sprzężone i węzły elementów nieakceptowanych typu MNA (*ang. Modified Nodal Analysis*, dla których układ równań musi być ułożone zmodyfikowaną metodą potencjałów węzłowych) umieszczane są

w blokach analizowanych metodami tradycyjnymi. Pomiędzy blokami stosowany jest algorytm relaksacyjny. Podejście to przyśpiesza proces analizy redukując liczbę relaksacji i nakłady na rozwiązanie układów równań (rozwiązywane są układu o mniejszym rozmiarze), o ile układ został prawidłowo podzielony. Istnieje kilka metod podziału blokowego układu, które zostaną omówione w następnych rozdziałach:

1. Podział statyczny wykonywany przez użytkownika przed analizą (rozdział 8.9.1).
2. Podział konduktancyjny (rozdział 8.9.3).
3. Podział na podstawie topologii układu:

   podział hierarchiczny konduktancyjny (rozdział 8.9.3),

   podział logiczny (rozdział 8.9.5).
4. Ma podstawie znajomości modeli elementów (rozdział 8.9.6 oraz 8.9.7).
5. Podział dynamiczny wykonywany w trakcie analizy (rozdział 8.9.8)

8.9.1 Podział statyczny

Podział statyczny jest najprostszym sposobem podziału układu. Wykonywany jest przed rozpoczęciem analizy w czasie przetwarzania opisu układu. Wymaga on ingerencji użytkownika. Przykładem zastosowania omawianej techniki jest symulator ELDO-XL [Ana97]. Opis układu może zawierać podobwody, które zostały wykorzystane przez twórców do wydzielenia podobwodów (podukładów) *analogowych*, które analizowane są metodami tradycyjnymi. Deklaracja typu podobwodu realizowana jest przez ustawienie opcji w definicji podobwodu. Do analizy układów z tranzystorami MOS i pomiędzy blokami wykorzystano metodę OSR[B.H85b, B.H85a], natomiast na poziomie bloków *analogowych* stosuje się algorytm NR.

Na użytkownika spada odpowiedzialność za prawidłowe zadeklarowanie podobwodów *analogowych*, gwarantujące poprawność działania. Szczegóły realizacji powyższej techniki zostały opisane w rozdziale 8.13.1.

8.9.2 Automatyczne grupowanie węzłów

Przedstawiony w tym rozdziale algorytm 30 automatycznego grupowania[R.A96] wykorzystuje informacje o topologii układu w celu zapewnienia warunków zbieżności metod relaksacyjnych na poziomie blokowym. Grupowanie wykonywane jest na poziomach numeracji warstw węzłów (*ang. levelizing*). Celem algorytmu jest utworzenie bloków zawierających węzły dołączone do elementów MNA (rozdział 8.9) i taka ich modyfikacja aby były spełnione warunki zbieżności. Same grupy tworzone są za pomocą algorytmu grupowania konduktancyjnego omówionego w rozdziale 8.9.3 oraz przez zgrupowanie węzłów, o których wiadomo, że dołączone są do elementów MNA. Algorytm 30 bada węzły zewnętrzne każdego bloku w warstwie i sprawdza czy do każdego wyjściowego węzła sterującego (*ang. feedforward*) (definicja 8.9.1) i wejściowego węzła sterowanego (*ang. feedforward*)[4] (definicja 8.9.2) dołączony jest uziemiony kondensator. Jeżeli nie, to dołączane są kolejne węzły do czasu, gdy warunek ten zostanie spełniony.

[4]Nie ma prostego i krótkiego tłumaczenia tego pojęcia na język polski.

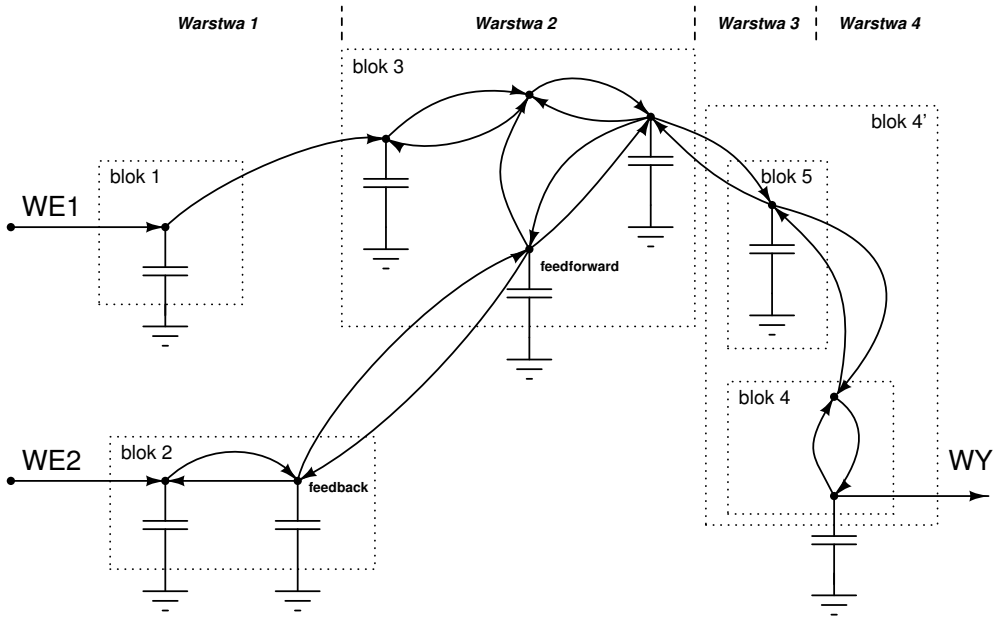
Definicja 8.9.1. *Węzeł ang. feedback jest to węzeł brzegowy bloku sprzężony z węzłami danego bloku, którego równanie jest funkcją sygnałów w węzłach w bloku o wyższym numerze warstwy.*

Definicja 8.9.2. *Węzeł ang. feedforward jest węzłem niższej warstwy, który sterowany jest przez węzeł feedback z warstwy wyższej.*

Algorytm 30 Grupowanie elementów MNA

```
lista węzłów ← wszystkie węzły
for każdego i-tego elementu MNA do
    zgrupuj węzły i-tego elementu MNA
end for
grupowanie_G(lista węzłów)                    ▷ grupowanie konduktancyjne
grupowanie_C(lista węzłów)                    ▷ grupowanie pojemnościowe
ustawienia danych grupy
numerowanie grup
for każdego poziomu i = n, ..., 1 do
    for każdej grupy j z poziomu i do
        numerowanie węzłów w grupie j
    end for
end for      ▷ sprawdzenie czy istnieją pojemności w węzłach feedback i feedforward
for każdego poziomu i grupy do
    for każdej grupy j z poziomu i do
        numerowanie węzłów w grupie
    end for
end for
for poziom i = n, ..., 1 do
    for każdej grupy j z poziomu i do
        for każdego węzła k w grupie j do
            if węzeł k nie był odwiedzany then
                ustaw węzeł k jako odwiedzony
                for każdego nieodwiedzanego węzła zależnego z grupy j do
                    if brak pojemności w węzłach feedback i feedforward then
                        dołącz węzeł z uziemioną pojemnością/połącz grupy
                        aktualizuj grupę
                    end if
                end for
            end if
        end for
    end for
end for
```

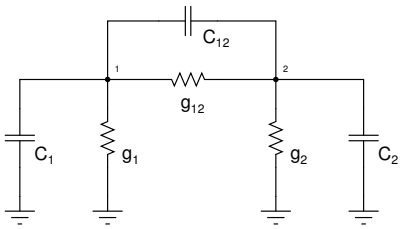
Na rys. 8.4 przedstawiono wynik podziału hipotetycznego układu powstały w wyniku działania algorytmu 30. Układ został podzielony na pięć bloków. Każdy węzeł wyjściowy posiada dołączony uziemiony kondensator. W bloku 4 węzeł sterujący nie posiada takiej pojemności i bloki 4 i 5 muszą zostać połączone tworząc jeden blok 4'.



Rys. 8.4: Schematyczne przedstawienie wyniku podziału hipotetycznego układu

8.9.3 Podział typu G i C

Podział konduktancyjny [RS94] gwarantuje zbieżność dla sieci konduktancyjnej. Oparty jest on na Nortonowskim źródle zastępczym. Na rys. 8.5 został przedstawiony schemat układu liniowego używanego w rozważaniach.



Rys. 8.5: Zastępczy układ liniowy używany w algorytmie podziału C i G

Szybkość zbieżności metod iteracyjnego rozwiązywania równań zależna jest od współczynnika γ_∞ we wzorze (8.51).

$$\| x^{k+1} - x^k \| \leq \gamma_\infty \| x^k - x^{k-1} \| \tag{8.51}$$

Dla sieci czysto konduktancyjnej (dla schematu z rys. 8.5 po usunięciu kondensatorów) γ_∞ dana jest wzorem 8.52.

$$\gamma_\infty = \frac{g_{12}}{(g_2 + g_{12})} \frac{g_{12}}{(g_1 + g_{12})} \tag{8.52}$$

gdzie: g_1, g_2 są Nortonowskimi konduktancjami widzianymi z zacisków w stronę reszty układu. Algorytm 31 grupowania konduktancyjnego wyznacza γ_∞ i porównuję z wartością zadaną α, która wyznacza granicę rozpoczęcia grupowania.

Algorytm 31 Grupowanie konduktancyjne elementów

```
1: for każdego i-tego węzła z listy do
2:     for każdego elementu dołączonego do węzła i do
3:         pobierz drugi węzeł j elementu
4:         if węzły i oraz j nie były odwiedzane then
5:             wyzeruj g_ij, g_i, g_j
6:             for każdego elementu pomiędzy węzłami i oraz j do
7:                 g_ij = g_ij + maksymalna konduktancja w układzie
8:                 usuń element z grupy
9:             end for
10:        end if
11:        wyznacz zastępcze konduktancje widziane w stronę układu g_i, g_j
12:        wyznacz γ_∞
13:        if γ_∞ > α then
14:            grupuj węzły i oraz j
15:        end if
16:    end for
17: end for
```

Parametr α jest minimalną wartością γ, od której węzły są grupowane. Działanie algorytmu wymaga przejrzenia wszystkich węzłów i wyznaczeniu wartości współczynnika γ dla każdego węzła incydentnego z danym węzłem. Analogiczny podział wykonywany jest dla kondensatorów C i zwany jest podziałem C.

8.9.4 Podział hierarchiczny

Podział hierarchiczny [P.S93] jest dość często wykonywany w praktyce. Dla układów z tranzystorami MOS jest to najczęściej podział logiczny, który nie zawsze jest optymalny. Dlatego też stosuje się podział hierarchiczny, którego idea polega na podziale układu na bloki o jednokierunkowych sprzężeniach pomiędzy blokami.

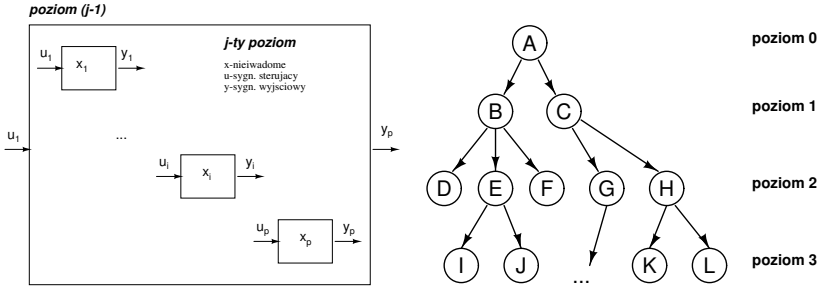
Poczynając od obwodu głównego, każdy obwód dzielony jest na bloki na kolejnych niższych poziomach (*ang. clusters*). W obrębie bloków znów dokonuje się podziału na niższych poziomach (*ang. levelizing*) do momentu, aż nie da się dalej podzielić bloku, gdyż zawiera tylko elementy bilateralne.

Na rys. 8.6 przedstawiono przykładowy sposób hierarchicznego podziału hipotetycznego układu (rys. 8.6a) i jego reprezentację w postaci drzewa (rys. 8.6b).

Poszczególne poziomy z rys. 8.6b odpowiadają przykładowo za:

- poziom 0 odpowiada za podział logiczny np. na poziomie przerzutników,
- poziomy 1 i 2 reprezentują podział na bramki logiczne w obrębie przerzutników,
- poziom 3 wykonany został dla tranzystorów wchodzących w skład bramek.

Taka drzewiasta struktura reprezentowana jest w postaci macierzy blokowych poszczególnych poziomów. W obrębie bloków stosuje się metodę Newtona, a pomiędzy



(a) systemu (b) drzewa

Rys. 8.6: Reprezentacja hierarchiczna

nimi iteracje blokowe Gaussa-Seidela. W trakcie analizy może się okazać, że istnieje potrzeba przegrupowania bloków ze względu na krótki krok analizy lub słabą zbieżność. Stosuje się wtedy przegrupowanie w czasie analizy. Omawiane tu podejście zastosowano w symulatorze PYRAMID [P.S88].

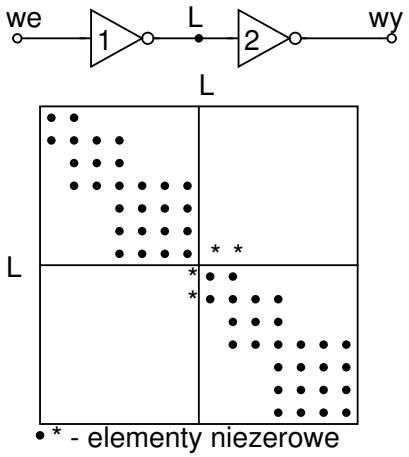
8.9.5 Podział logiczny

Podział logiczny [RS94] (*ang. gate level network separation technique*) układu opiera się na wykorzystaniu estymacji Jakobianu dla poziomów wysokiego i niskiego bez liczenia konkretnej wartości Jakobianu. Zauważono, że dla stanu wysokiej impedancji oznaczonego '*' lub dla poziomu wysokiego elementy spoza głównej przekątnej macierzy układu mają małe wartości w kolejnych iteracjach Newtona. Umożliwia to podział układu zgodnie ze strukturą logiczną jak to przedstawiono na rys. 8.7. Podział układu dokonywany jest jeżeli napięcie w punkcie L spełnia zależność $v_L^t > Th_{high}$, gdzie v_L jest napięciem wejściowym bramki w punkcie czasowym t, a Th_{high} jest najmniejszą wartością napięcia wejściowego bramki stanu wysokiego. Jeżeli zostanie spełniona ta zależność, to wartości spoza przekątnej oznaczone przez '*' mają relatywnie małe wartości i układ można rozdzielić na dwa podukłady, co odpowiada podziałowi macierzy układu z rys. 8.7 w punkcie L.

Do prawidłowego działania algorytmu potrzebna jest znajomość wartości Th_{high}, którą można uzyskać na podstawie badania Jakobianu dla danego modelu bramki logicznej lub ustalić doświadczalnie.

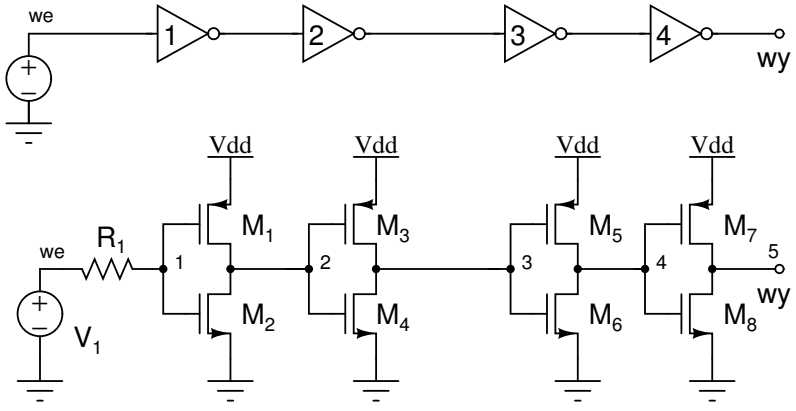
8.9.6 Podział dla układów z tranzystorami MOS

Układy z tranzystorami MOS są szczególnie dobrze akceptowane przez metody relaksacyjne ze względu na ich jednokierunkowość i słabe sprzężenia pomiędzy bramką sterującą a pozostałymi końcówkami. Jeśli układ złożony jest tylko z tranzystorów MOS, to nie wykonuje się podziału blokowego ze względu na istnienie silnych sprzężeń międzywęzłowych, gdyż takie nie występują. Dokonuje się podziału węzłowego. Bramka tranzystora traktowana jest jako sterowanie modelu. Ze względu na słabe sprzężenie bramki z pozostałymi węzłami, model tranzystora nie jest obliczany, gdy liczone jest napięcie w węźle dołączonym do bramki (obliczanie modelu nie wnosi tu żadnej informacji). Można powiedzieć, że podział układu z tranzystorami MOS przebiega na *granicy bramka-tlenek*. Rozdział równań układu odpowiada podziałowi węzłowemu.



Rys. 8.7: Struktura macierzy Jakobianu dla układu dwóch inwerterów

W układach cyfrowych podział węzłowy odpowiada często podziałowi logicznemu (*ang. gate level network separation technique*) i może być wykonywany dynamicznie w trakcie analizy. Wykorzystuje się tu Jakobian do badania sprzężeń pomiędzy bramkami. Taki algorytm zastosowano w programie SPLIT2 [M.N88]. Np. podział układu 4 inwerterów z rys. 8.8 odpowiada jego strukturze logicznej. Każda para tranzystorów tworzy bramkę cyfrową, która jest zarazem pojedynczym blokiem.



Rys. 8.8: Schemat logiczny i elektryczny ciągu 4 inwerterów

Układ ten jest jednym z najprostszych układów. Nie występują w nim pętle sprzężenia zwrotnego. Przepływ sygnału jest jednokierunkowy z wejścia do wyjścia. Jedyne sprzężenia zwrotne, które *psują* zbieżność analizy relaksacyjnej, to nieuziemione pojemności C_{GD} i C_{GS}.

W przypadku istnienia pętli sprzężeń zwrotnych należy stosować odpowiednią numerację węzłów odpowiadającą przepływowi sygnału.

W celu przyśpieszenia obliczeń często stosuje się uproszczone modele tranzystorów.

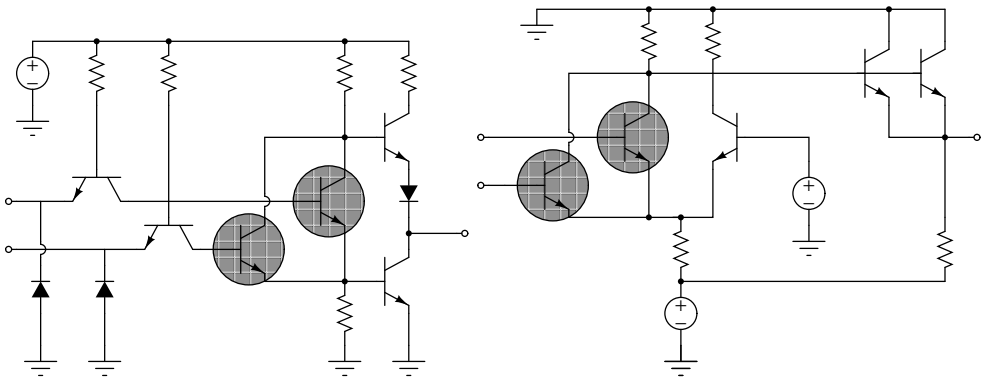
Ogólnie modele tranzystorów można podzielić na trzy grupy:

- modele dokładne,
- modele uproszczone,
- modele uproszczone z dokładniejszym modelowaniem niektórych zjawisk.

Jako modele dokładne stosuje się najczęściej modele SPICE'a [A+81, J.P96] poziomu 1 lub 3 z pojemnościami modelującymi efekt gromadzenia ładunku w tranzystorze. W modelach uproszczonych stosuje się elementy RC i przełączniki zapewniające proste odwzorowanie podstawowych zjawisk. Dają one nieco mniej dokładne wyniki, lecz przy nieco mniejszej dokładności, umożliwiają szybszą analizę. Stosowane są także modele uproszczone opisujące niektóre kluczowe zjawiska ważne z punktu widzenia symulacji [Ana97]. Dla modeli uproszczonych stosowane są czasami metody podziału układu na bloki (rozdział 8.9.3).

8.9.7 Podział dla układów z tranzystorami bipolarnymi

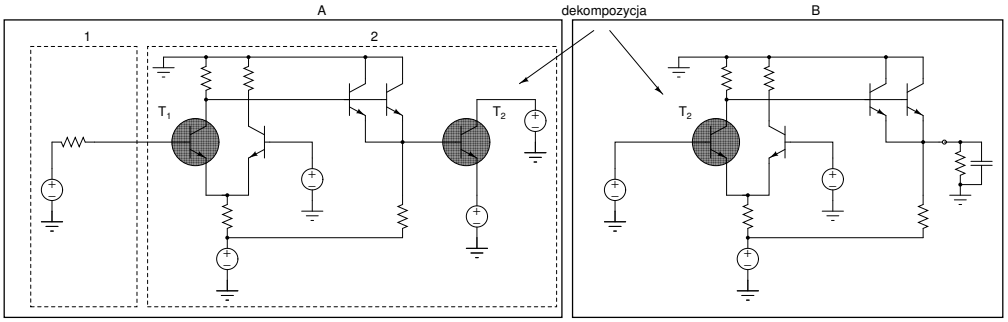
W [M.N91] opisano algorytm WR wykorzystany do analizy układów cyfrowych zbudowanych z tranzystorów bipolarnych. Algorytm ten zastosowano w symulatorze równoległym PNAP-1.1[N.H89]. Wykorzystuje on technikę okien (*ang. windowing* - rozdział 8.4.1), dynamiczny podział układu i uproszczone modele tranzystorów. Algorytm dynamicznego podziału opiera się na spostrzeżeniu, że węzły tranzystora bipolarnego są słabo sprzężone, gdy tranzystor jest zatkany. W praktyce napięcia sterujące muszą być mniejsze niż napięcie przełączania $U_{be} < U_{beth}$ i $U_{bc} < U_{bcth}$, które zostały doświadczalnie wyznaczone i wynoszą 0.5V. Jeżeli tranzystor wchodzi w tak zdefiniowany obszar zatkania, to program wywołuje funkcję, która dzieli układ. Podział polega na odseparowaniu bazy od kolektora/emitera przez umieszczenia jej w osobnym podobwodzie. Podział ten wykorzystywany jest tylko dla tranzystorów *wejściowych* bramek, które zostały zaznaczone kółkami na rys. 8.9 dla dwóch rodzajów bramek.



(a) TTL NOR (b) ECL NOR

Rys. 8.9: Przykład dekompozycji bramek zbudowanych z tranzystorów bipolarnych

Przykładowy układ po podziale przedstawiono na rys. 8.10. Jest on złożony z dwóch podobwodów A i B. Jeżeli w podobwodzie A tranzystor T_1 nie jest w stanie odcięcia to podobwody 1 i 2 są łączone razem tworząc podobwód A.



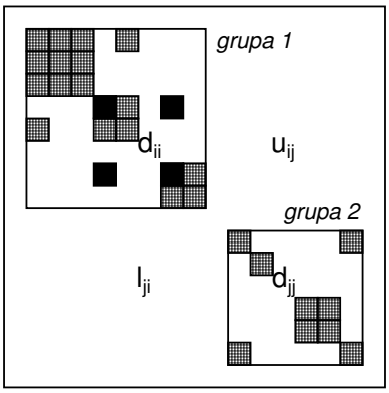
Rys. 8.10: Układ z tranzystorami bipolarnymi po podziale na podobwody

8.9.8 Podział dynamiczny

Podział dynamiczny stosowany jest najczęściej w metodach WR. Jest on oparty na badaniu Jakobianu [M.N93]. Jeżeli równania ZMPW układu zapiszemy w postaci macierzowej (rysunek 8.11), to sprzężenia pomiędzy dwoma węzłami i, j można zbadać wyznaczając współczynniki sprzężeń $c_{i,j}, c_i, c_j$ z (8.53).

$$c_{i,j} =\mid \frac{u_{i,j}l_{i,j}}{d_{i,i}d_{j,j}} \mid, c_i =\mid \frac{u_{i,j}}{d_{i,i}} \mid, c_j =\mid \frac{l_{j,i}}{d_{j,j}} \mid \tag{8.53}$$

gdzie: $u_{i,j}$ - element znad przekątnej macierzy, $l_{i,j}$ - element spod przekątnej, $d_{j,j}$ - element diagonalny.



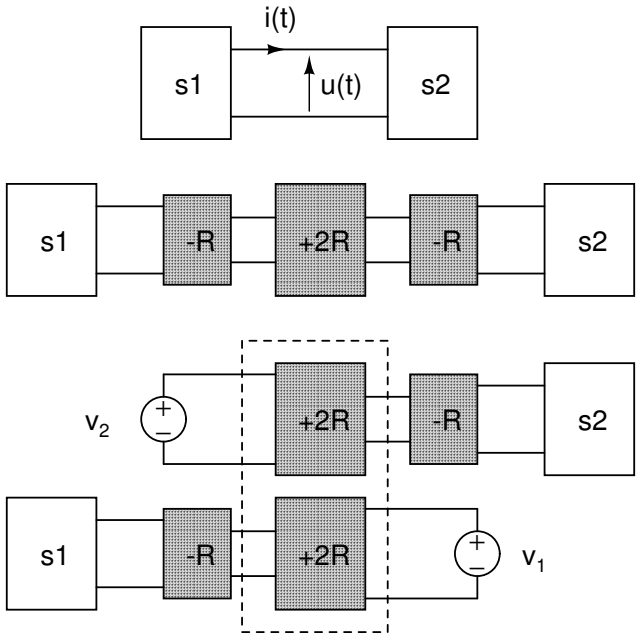
Rys. 8.11: Struktura macierzy układu a współczynniki sprzężeń równań

Można zadać współczynniki sprzężeń α i α_1, których wartości decydują o grupowaniu węzłów. Jeżeli nierówności $c_{i,j} < \alpha$ i $c_i < \alpha_1$ są spełnione, to węzły i oraz j mogą

być traktowane jako słabo sprzężone i ich równania można rozwiązywać metodami iteracyjnymi. W przeciwnym przypadku należy stosować techniki bezpośrednie.

8.9.9 Modyfikacje topologii

Zbieżność metod relaksacyjnych można zapewnić przez odpowiednie (niekoniecznie fizyczne) modyfikacje topologii układu. W [RW91] opisano metodę modyfikacji topologii na granicy silnie sprzężonych układów (*ang. overlap partitioning*). Pomiędzy dwoma podukładami (systemami) zostają umieszczone trzy rezystory o wartościach: -R,+2R,-R szeregowo z konduktancjami wejściowymi/wyjściowymi podobwodów (rys. 8.12).



Rys. 8.12: Idea modyfikacji układu silnie sprzężonych systemów - *ang. overlap partitioning*

Modyfikacja taka zapewnia uzyskanie zbieżności technik relaksacyjnych nawet wtedy, gdy bez modyfikacji nie osiągano zbieżności. W pracy pokazano, że przy takiej modyfikacji układu, każdy węzeł sieci musi mieć dołączoną uziemioną pojemność.

Zdefiniowano trzy typy układów liniowych podlegających podziałowi:

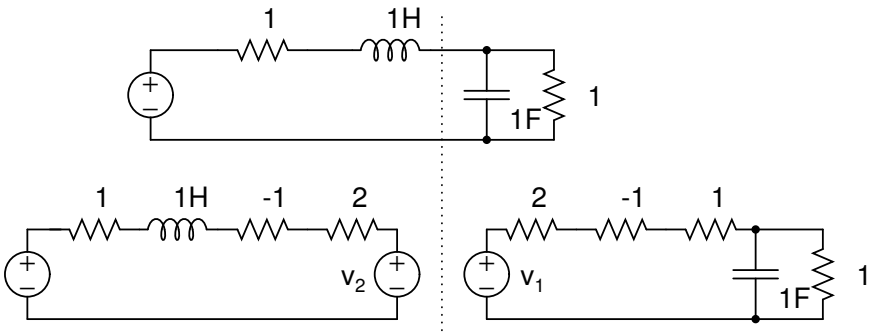
typ A - w którym żadne składowe wektora $i(t)$ i $v(t)$ na granicy podobwodów nie są zmiennymi stanu,

typ B - w którym niektóre składowe wektora $i(t)$ są zmiennymi stanu,

typ C - w którym niektóre składowe wektora $v(t)$ są zmiennymi stanu.

Podobwody $s1$, $s2$ nie mogą być jednocześnie typu B lub C. W zależności od typu podobwodów, podział wykonywany jest w różny sposób.

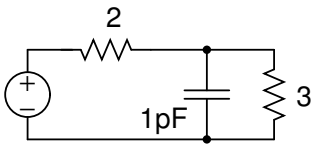
Na rys. 8.13 pokazano przykładowy podział dla podobwodów typu B-C wykonany wzdłuż linii kropkowanej. Dodano elementy dodatkowe: -R,+2R,-R oraz źródła napięciowe v_1 i v_2 reprezentujące podobwody $s1, s1$.



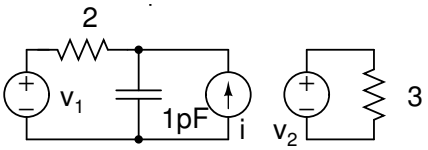
Rys. 8.13: Podział typu B-C

Dobór wartości rezystancji R został szczegółowo przeanalizowany w artykule[RW91]. Do jej wyznaczenia wykorzystuje się macierz rezystancji wejściowych podobwodu (przy wyzerowanych wszystkich źródłach niezależnych). Wartość R przyjmowana jest jako wartość rezystancji wejściowej dla i-tego węzła, dla którego wykonywany jest podział $R = diag(r_{ii})$, gdzie r_{ii} jest elementem z głównej przekątnej macierzy rezystancji dla i węzła. Gwarantuje to istotną redukcję promienia spektralnego macierzy układu i szybką zbieżność metody.

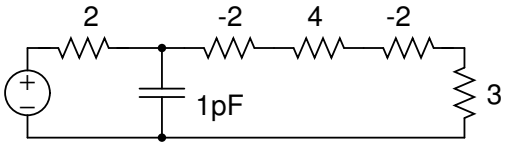
Na rys. 8.14a pokazano układ, który podzielono na dwa sposoby. Pierwszy typu i-i przedstawiony na rys. 8.14b nie gwarantuje zbieżności, natomiast drugi z rys. 8.14c typu u-u gwarantuje szybką zbieżność.



(a) Schemat układu



(b) Podział i-i nie gwarantujący zbieżności



(c) Podział u-u gwarantujący zbieżność

Rys. 8.14: Porównanie sposobów podziału układu

8.10 Metody doboru kroku analizy

Dobór kroku analizy czasowej jest jednym z trudniejszych zagadnień analizy czasowej. Jest to spowodowane wzajemnie sprzecznymi wymaganiami na krok analizy. Krok analizy powinien być tak dobrany, aby zapewnić:

1. szybkość - przez dobór możliwie najdłuższego kroku.
2. dokładność - przez dobór takiego kroku (z reguły krótkiego), który zapewnia utrzymanie lokalnego błędu obcięcia na zadanym poziomie,
3. zbieżność algorytmów relaksacyjnych (rozdział 5.6).

Należy mieć na uwadze fakt, że spełnienie tych (sprzecznych ze sobą) kryteriów nie jest zawsze możliwe i wybierany jest wariant pośredni będący kompromisem między dokładnością analizy a jej szybkością. Należy zwrócić także uwagę na fakt, że sam dobór nawet optymalnego kroku analizy może nie skracać czasu analizy jeżeli występują sprzężenia w układzie powodujące konieczność wykonania kroku wstecz (gdyż sygnały sterujące wzięte do rozwiązania węzłów zależnych obarczone były zbyt dużym błędem). Sytuacja taka występuje w analizie WR, gdzie w oknie czasowym analizowany jest zbiór węzłów wzajemnie sprzężonych. Może się tu zdarzyć konieczność odrzucenia rozwiązań z poprzednich chwil czasowych dla węzłów zależnych, jeżeli były one analizowane z mniejszym krokiem niż węzeł sterujący, którego napięcie nie zbiegło się. Dlatego też o szybkości analizy nie decyduje tylko dobór kroku analizy węzłów, ale także kolejność ich rozwiązywania. Należy stwierdzić, że zastosowanie tylko jednej z tych technik nie prowadzi do optymalnego doboru kroku i dlatego stosuje się równocześnie kilka kryteriów.

8.10.1 Dobór kroku na podstawie liczby iteracji NR

W przypadku metod relaksacyjnych użycie długiego kroku analizy może prowadzić do słabej zbieżności metod iteracyjnego rozwiązywania równań, a w konsekwencji do konieczności wykonania kroku wstecz. Dlatego też przy wyborze kroku należy brać pod uwagę nieliniowość układu w celu minimalizacji całkowitej liczby iteracji NR. Takie podejście stosuje wiele programów, jak np.: SPICE2 [L.W75], ADVICE [L.W80], MOTIS3 [D.T86, B.R75]. Krok dobierany jest na podstawie liczby wykonanych iteracji NR (*ang. iterative count time step control*) w następujący sposób:

1. Jeżeli liczba iteracji NR jest większa niż p_{max}, to krok jest redukowany z pewnym współczynnikiem.
2. Jeżeli liczba iteracji NR jest mniejsza od p_{min}, to krok jest wydłużany z pewnym współczynnikiem.
3. Jeżeli liczba iteracji NR zawiera się w przedziale $[p_{max}, p_{min}]$, to krok pozostaje niezmieniony.

Główną ideą tej metody jest utrzymanie stałej liczby iteracji NR w każdym punkcie czasowym analizy. Omawiane podejście efektywnie redukuje całkowite nakłady obliczeniowe, lecz mogą występować trudności z utrzymaniem zadanej dokładności[L.W75], gdyż metoda ta nie kontroluje poziomu dokładności. Na przykład dla układu liniowego zbieżność osiągana jest w jednej iteracji NR i krok będzie sukcesywnie wydłużany.

Dlatego też metodę tą należy stosować w połączeniu z metodą kontroli lokalnego błędu obcięcia (LBO). W przypadku przekroczenia maksymalnej zadanej liczby iteracji należy wykonać krok wstecz i skrócić krok.

8.10.2 Dobór kroku na podstawie badania przyrostów wartości sygnałów

Metoda doboru kroku na podstawie badania przyrostów wartości sygnału w kolejnych chwilach czasowych związana jest z usypianiem węzłów [C.L86] omówionym w rozdziale 8.12.3. Dobór kroku następuje na podstawie badania przyrostów sygnału pomiędzy obecnym - m-tym punktem czasowym, a punktem $m-l$-tym z przeszłości. Dla $l = 1$ otrzymujemy kryterium doboru kroku analizy pomiędzy dwoma punktami czasowymi. Przyrost sygnału w kolejnym punktach czasowych m i $m - l$ powinien zawierać się pomiędzy zadanymi przez użytkownika wartościami minimalnego dv_{min} i maksymalnego dopuszczalnego przyrostu dv_{max} sygnału. Wartości te należy wyznaczyć doświadczalnie. Dobór kroku odbywa się na podstawie (8.54).

$$\begin{array}{lll} \mid v(t_{m+l}) - v(t_m) \mid < dv_{min} & h_{m+1} = 2h_m & \text{wydłuż krok} \\ \mid v(t_{m+l}) - v(t_m) \mid > dv_{max} & h_{m+1} = \dfrac{h_m}{2} & \text{skróć krok} \\ dv_{min} < \mid v(t_{m+l}) - v(t_m) \mid < dv_{max} & h_{m+1} = h_m & \text{nie zmieniaj kroku} \end{array} \tag{8.54}$$

gdzie: l - jest kolejnym l-tym punktem czasowym względem chwili t_m.

Badanie bezwzględnych przyrostów może w praktyce prowadzić do błędów. Sytuacja taka została przedstawiona na rys. 8.20d. Wartość sygnału początkowo rośnie, po czym łagodnie opada. Pomiędzy zmianami występuje zakres powolnych zmian sygnału. Jeżeli l jest małe (np. 1), to możliwe jest wybranie zbyt długiego kroku. Wybierając $l > 1$ można częściowo zabezpieczyć się przed doborem zbyt długiego kroku dla sygnału wolnozmiennego. Jeśli jednak sygnał "oscyluje" wokół jednej wartości, to podejście to jest niewystarczające. Dlatego też metoda ta nie może być stosowana samodzielnie.

8.10.3 Dobór kroku na podstawie lokalnego błędu obcięcia

Dobór kroku na podstawie badania lokalnego błędu obcięcia[J.O95, RS94] daje jedne z najlepszych rezultatów. Z tego powodu metoda ta jest szeroko stosowana. Do kontroli lokalnego błędu obcięcia (LBO) (*ang. Local Truncation Error* - LTE) używa się zadanych przez użytkownika błędów bezwzględnego δ i względnego ϵ w każdym kroku całkowania. Wartość lokalnego błędu obcięcia zadaną przez użytkownika można wyznaczyć z (8.55) na podstawie znajomości δ i ϵ.

$$LBO_{user} = \delta + \epsilon \max(\mid v_{m+1} \mid, \mid v_m \mid) \tag{8.55}$$

Dla schematów różnicowych zamkniętych (*ang. Backward Differential Formulas* - BDF) P-tego rzędu, wartość lokalnego błędu obcięcia może być obliczona na podstawie różnicy pomiędzy wartością obliczoną v_{m+1}, a wartością przewidywaną $v^P(t_{m+1})$ ze wzoru (8.56), przy czym v^P musi być obliczone predyktorem rzędu P.

$$LBO_P = \left[\frac{h_m}{t_{m+1} - t_{m-P}}\right] \left(v_{m+1} - v^P(t_{m+1})\right) \tag{8.56}$$

Rozwiązanie jest prawidłowe jeżeli spełnia warunek dokładności (8.57).

$$| LBO_P | < LBO_{user} \tag{8.57}$$

Aby wykorzystać równanie (8.57) do wyznaczenia nowego kroku analizy należy wprowadzić współczynnik r będący ilorazem wartości dopuszczalnego i aktualnego lokalnego błędu obcięcia (8.58).

$$r = \frac{| LBO_{user} |}{| LBO_P |} \tag{8.58}$$

Wartości LBO zależą od h i współczynnik $r = \left[\frac{h_{max}}{h_m}\right]^{P+1}$ także zależy od h. Krok h_{max} jest optymalnym krokiem czasowym, dla którego wartość lokalnego błędu obcięcia wynosi LBO_{user}. Iloraz $\frac{h_{max}}{h_m}$ oznaczmy jako r_{LBO} i potraktujmy jako mnożnik kroku analizy gwarantujący utrzymanie zadanej dokładności analizy (8.59).

$$h_{m+1} = r_{LBO} h_m \tag{8.59}$$

Jeżeli $r_{LBO} > 1$ oznacza to, że wartość lokalnego błędu obcięcia jest mniejsza niż założona i otrzymane rozwiązanie jest prawidłowe. Współczynnik r_{LBO} jest funkcją r (8.60), a więc zależy pośrednio od zadanych przez użytkownika błędów δ, ϵ i wyznaczonego na ich podstawie LBO_{user} (8.55) użytego do obliczenia r.

$$r_{LBO} = \frac{h_{max}}{h_m} = r^{(\frac{1}{P+1})} \tag{8.60}$$

W praktyce predykowany krok h_{m+1} (8.59) podlega raptownym zmianom, co powoduje problemy ze stabilnością [R.K77]. Stopniowe zmiany kroku można uzyskać stosując ograniczanie przyrostów kroku (8.61).

$$\begin{array}{rll} r_{LBO} < 1 & \text{to skróć krok} & h_{m+1} = h_m \max(s_l, r_{LBO}) \\ 1 < r_{LBO} < \alpha & \text{krok stały} & h_{m+1} = h_m \\ r_{LBO} > \alpha & \text{to wydłuż krok} & h_{m+1} = h_m \min(s_u, \beta r_{LBO}) \end{array} \tag{8.61}$$

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 (8.59). Ponieważ wartość LBO jest estymowana, to czasami może się zdarzyć przypadek doboru za długiego kroku i konieczność wykonania kroku wstecz (ze względu na przekroczenie dopuszczalnych zadanych błędów analizy). Wyborowi zbyt długiego kroku zapobiega mnożnik β, który nieco skraca wydłużany krok i w ten sposób (w wielu przypadkach) zapobiega występowaniu omawianego zjawiska. W praktyce stosuje się następujące wartości parametrów: $s_l = 0.25, s_u = 2.0, \alpha = 1.2, \beta = 0.9$.

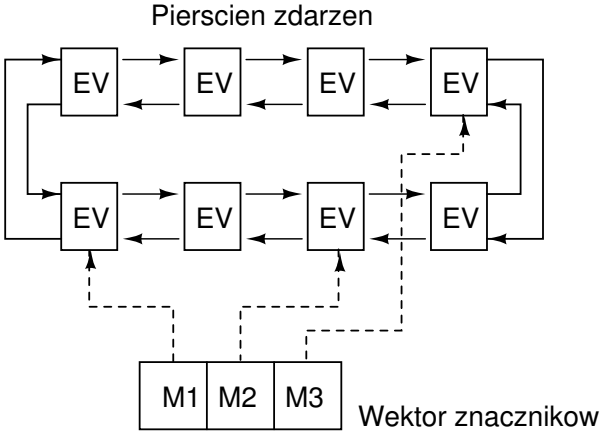
8.11 Zarządzanie zdarzeniami

Zarządzanie zdarzeniami jest jedną z najbardziej krytycznych części algorytmu kierowanego zdarzeniami ze względu na dużą czasochłonność obsługi list zdarzeń. W trakcie analizy istnieje konieczność nie tylko prostego wstawiania czy usuwania, lecz także wstawiania zdarzenia pomiędzy dwa zdarzenia, sortowania w ustalonej kolejności, usunięcia zdarzenia niepotrzebnego. Operacje wstawiania i usuwania muszą być bardzo efektywne.

Organizacja list zdarzeń powinna gwarantować efektywne sortowanie zdarzeń. Czas obsługi listy zdarzeń jest stosunkowo krótki w analizie układowej, gdzie sama analiza jest bardzo czasochłonna, lecz może być dominujący w analizie na poziomie przełączników (*ang. switch-level circuit simulation*) lub na wyższych poziomach, gdzie czas analizy jest stosunkowo krótki. Ogólnie przyjmuje się, że dobrze zaprojektowana obsługa zdarzeń powinna zajmować nie więcej niż 10% do 15% całkowitego czasu analizy.

8.11.1 Organizacja listy zdarzeń

W klasycznych symulatorach układów cyfrowych stosowana jest dwukierunkowa lista pierścieniowa. Listę taką, zastosowaną w symulatorze SAMSON[K.A85], przedstawia rys. 8.15.

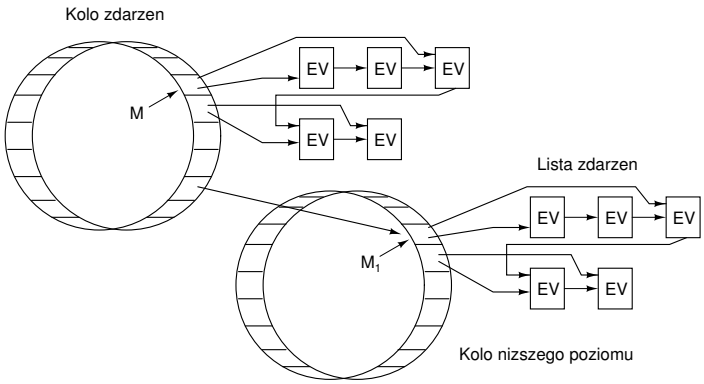


Rys. 8.15: Lista indeksowana stosowana w symulatorach kierowanych zdarzeniami

Każdy element listy zawiera informację o zdarzeniach następnym i poprzednim, o czasie analizy i węzłach, które powinny zostać przeanalizowane. Dostęp realizowany jest poprzez wskaźnik do jednego z elementów listy. Wstawienie zdarzenia polega na alokacji odpowiedniej struktury i umieszczeniu jej w odpowiednim miejscu pierścienia. Wektor znaczników M zawiera znaczniki wskazujące końce odpowiednich odcinków czasu. Cała dziedzina czasu podzielona jest na odcinki o długości Δt. Każdemu znacznikowi przyporządkowane są chwile czasowe $t_0 = t_1 - \Delta t, t_1 = t_2 - \Delta t, t_2 = t_3 - \Delta t, \ldots$. Odcinki pomiędzy znacznikami zawierają uporządkowane zdarzenia o czasie $t \in [t_m, t_{m+1}]$. Szukanie chwili pomiędzy znacznikami jest powolne, ze względu na konieczność przejrzenia listy. Lista jest jednak krótka. Dostęp do odpowiednich odcinków (przez tablicę znaczników) jest szybki. Aby zminimalizować średni czas dostępu należy dążyć do tego, aby liczba zdarzeń na każdym odcinku (pomiędzy znacznikami) była jednakowa. Wymaga to zastosowania zmiennego Δt (dobieranego metodami adaptacyjnymi) dostosowanego do gęstości i rozmieszczenia zdarzeń w czasie.

Podobne podejście, to zastosowanie tzw. koła czasowego (*ang. time-wheel*). Dziedzina czasu, tak jak poprzednio, dzielona jest na odcinki Δt, do których znaczniki końca i początku znajdują się w tabeli znaczników (w kole czasowym znaczników) - rys. 8.16.

Pomiędzy znacznikami zdarzenia mogą być zorganizowane w formie listy dwukierunkowej lub zdalnej (posortowanej lub nie) (*ang. sorted or unsorted remote list*). Zaletą



Rys. 8.16: Organizacja koła zdarzeń

tego podejścia jest: i) z góry określona zajętość pamięci zależna od przyjętego Δt; ii) dobrze określone położenia znacznika czasowego M, gdyż znane są odstępy czasowe. Jeżeli założymy, że tablica ma długość m i wszystkie zdarzenia zostały już przeanalizowane, to można ją powtórnie użyć przesuwając wskaźnik M na początek tablicy. W ten sposób realizowane jest koło czasowe.

W przypadku konieczności wstawienia zdarzenia o czasie $t > m\Delta$ (poza zakresem reprezentowanym w kole czasowym) istnieje konieczność realokacji pamięci (wydłużenia tablicy) i zwiększenia liczby punktów lub użycia następnego koła czasowego. Odcinki czasu pomiędzy znacznikami mogą być reprezentowane przez koła czasowe, w których znaczniki są rozmieszczone w mniejszych odstępach czasowych, np. co $\Delta t_1 < \Delta t/2$. Umożliwia to podzielenie niektórych odcinków na mniejsze (z góry ustalone) odcinki o szybkim dostępie do zdarzeń. Użycie kół czasowych ze stałymi odstępami czasowymi Δt znaczników umożliwia bezpośredni dostęp do określonego odcinka czasowego, ze względu na możliwość wyliczenia indeksów znacznika - zamiast przeglądania listy.

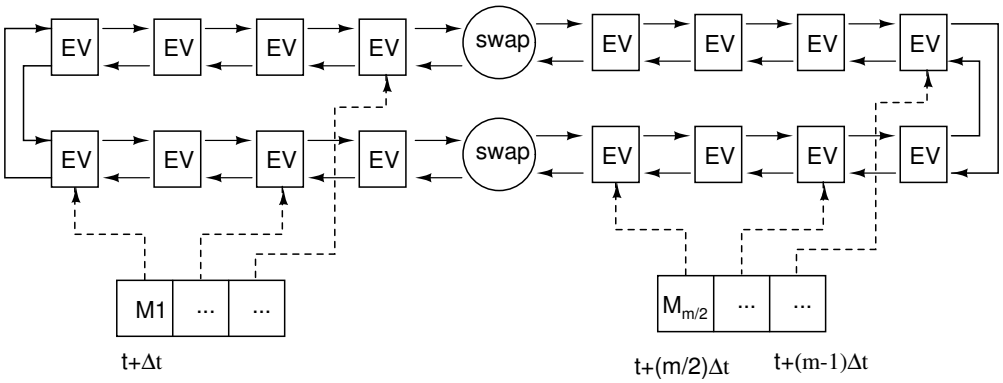
W programie iSPLICE3[R.A89a] zastosowano technikę łączonych list zdarzeń z różnymi odstępami czasowymi Δt dla każdej z list - rys. 8.17. Generowane zdarzenia mogą być umieszczane zarówno w liście pierwszej, drugiej jak i w listach zdalnych. Po osiągnięciu końca pierwszej - druga lista staje się aktywna, a pierwsza reprezentuje następne $m/2$ punktów.

8.11.2 Zarządzanie analizą węzłów

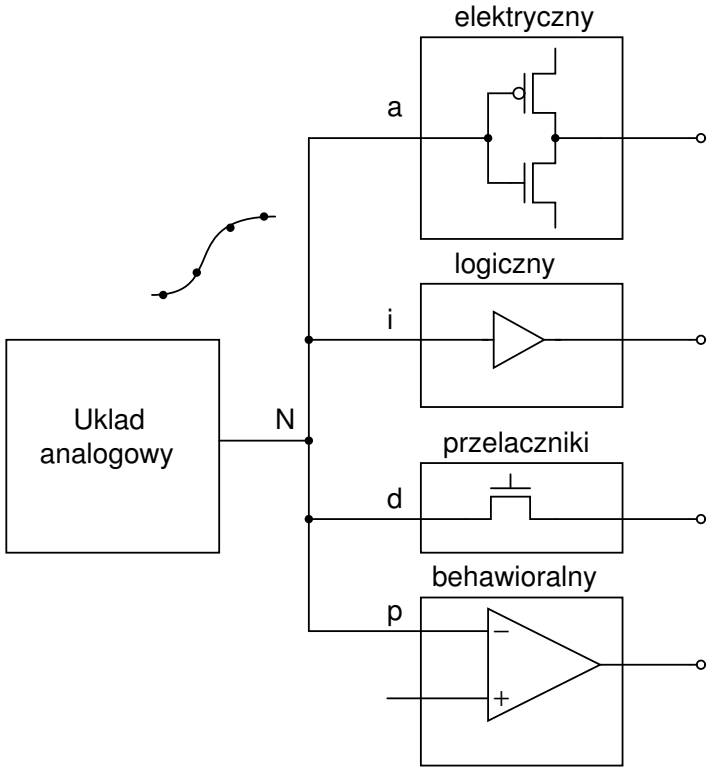
W ostatnich latach pojawiły się symulatory, które umożliwiają symulację układów na różnych poziomach i w różnych dziedzinach (*ang. multilevel and mixed-domain simulation*). W takich symulatorach prawidłowa gospodarka kierowaniem węzłów do analizy (*ang. scheduling*), a w szczególności węzłów zależnych, nastręcza wiele problemów.

Rodzaje używanych technik i występujące problemy zostaną omówione na przykładzie symulatora iMACSIM [R.A96], umożliwiającego symulację bloków z różnych poziomów. Załóżmy, że układ analogowy, przedstawiony w postaci bloku na rys. 8.18, steruje węzłem N. Do węzła N dołączono bloki z różnych poziomów symulacji.

Każdy dołączony blok wymaga innych kryteriów kierowania bloku do analizy, aby zapobiec sytuacji niepotrzebnej analizy bloków. W zależności od dołączonego bloku wyróżniono kilka sposobów kierowania do analizy (*ang. scheduling*) węzłów zależnych



Rys. 8.17: Łączone listy zdarzeń zaimplementowane w programie iSPLICE3



Rys. 8.18: Mixed-mode scheduling

(*ang. fanouts*):

- *ang. continues scheduling* odpowiada za kierowanie do analizy bloków poziomu elektrycznego, gdy następuje zmiana wartości sygnału (węzeł a).
- *ang. delta scheduling* wykorzystuje się, gdy występuje zmiana sygnału o z góry

ustaloną wartość ϵ (węzeł i).

- *ang. treshold scheduling* wykorzystuje się, gdy napięcie przekracza predefiniowany poziom napięcia (poziom przełączania) w którąkolwiek ze stron (węzeł d).

- *ang. signal direction-based scheduling* jest to *ang. treshold scheduling* z rozróżnianiem kierunków zmian sygnału (opadający - narastający) (węzeł i).

Najwięcej problemów nastręcza *ang. treshold scheduling*. Jeżeli zmiany sygnału w węźle N są *a priori* znane, to można z góry przewidzieć chwilę przełączenia $t_{treschold}$ i skierować blok i węzły zależne do analizy. Trudniejsza sytuacja występuje, gdy zmiana sygnału węzła N nie jest znana i należy wykryć chwilę przełączenia. Istnieją dwie metody wykrywania chwili przełączenia:

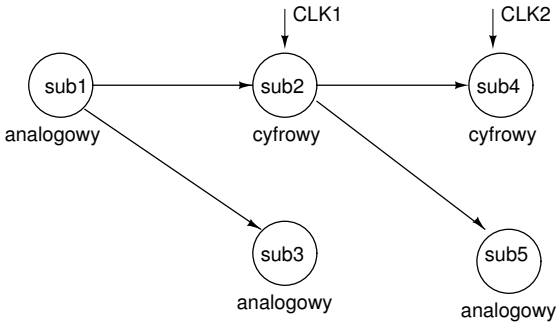
1. Predykowany jest krok czasowy t_{new} dla węzła N (bez uwzględnienia poziomu przełączania) [M.B80]. Następnie sprawdza się czy przekroczono poziom przełączania po wykonaniu kroku czasowego. Jeżeli przekroczono poziom przełączenia, to należy cofnąć się i wyznaczyć nowy krótszy krok tak, aby trafić w poziom przełączenia. W tym przypadku stosowana jest predykcja liniowa (np. Eulera). Wadą metody jest konieczność wykonania wielu małych kroków czasowych, gdy odległość pomiędzy zdarzeniem $\Delta t = t_{new} - t_{treschold}$ jest mała. W przypadku tym należy zaostrzyć kryterium kontroli lokalnego błędu obcięcia.

2. Akceptowany jest wyznaczony krok czasowy t_{new} i wartość napięcia v_{new}. Jeżeli w czasie wykonywania kroku wystąpiło przełączenia, to należy znaleźć chwilę przełączenia na podstawie interpolacji napięcia w węźle N. Zaletą rozwiązania jest eliminacja efektu "dochodzenia" poprzedniego rozwiązania do chwili przełączenia. Wadą jest potrzeba składowania wyników w kolejnych punktach i konieczność pomijania rozwiązań, które otrzymano po przełączeniu.

Układy analogowo-cyfrowe

W układach mieszanych tj. z czasem dyskretnym (cyfrowych) i ciągłym (analogowych), należy uwzględnić dwa podstawowe czynniki:

1. Zapewnienie otrzymania prawidłowych rozwiązań w poszczególnych punktach czasowych i na granicy pomiędzy układami analogowymi a cyfrowymi. Dla układu z czasem ciągłym (analogowego), węzły lub podukłady zależne kierowane są do analizy wtedy, gdy wystąpiła zmiana sygnałów sterujących, co z reguły oznacza zmianę napięcia w węźle. Układ analogowy może skierować do analizy zarówno inny układ analogowy jak i cyfrowy. Dla układu cyfrowego najczęściej wprowadzany jest sygnał zegarowy determinujący chwile czasowe, w których układ może być analizowany. Na rys. 8.19 przedstawiono przykładowy układ, w którego skład wchodzą bloki analogowe i cyfrowe. Blok sub1 może skierować do analizy układ analogowy sub3 (każdorazowo, gdy zmienią się sygnały wyjściowe bloku sub1) i blok cyfrowy sub2 (tylko w chwilach zdeterminowanych zegarem CLK1). Jeśli sygnały wyjściowe układu cyfrowego (np. sub2) zmienią się, to do analizy zostaje skierowany każdy sterowany układ analogowy (tutaj sub5). Zastosowanie zegara CLK gwarantuje analizę układu cyfrowego tylko w określonych chwilach czasowych i umożliwia łatwe wykorzystanie usypiania (*ang. latency*).



Rys. 8.19: Kombinacja układów analogowych i cyfrowych

Okresy zegarów CLK mogą być nieznane. Użytkownik powinien mieć możliwość zadania (przed analizą) okresów analizy podobwodów cyfrowych. Omawiany problem może występować także w układach analogowych, których elementy opisane modelami behawioralnymi reagują na poziomy przełączania (*ang. treshold voltage*).

2. Kłopoty ze zbieżnością na skutek istnienia nieciągłości. W przypadku idealnych przełączników zmiana ich stanów może powodować szybkie zmiany sygnałów w węzłach napięciowych, które mogą być węzłami wejściowymi układów analogowych. Metody całkowania numerycznego są bardzo wrażliwe na błędy związane z nieciągłościami. Kontrola lokalnego błędu obcięcia używana w połączeniu z metodami całkowania numerycznego musi uwzględniać fakt, że takie raptowne zmiany napięć w węzłach nie mogą być używane do predykcji napięcia w kolejnych chwilach czasowych. Spowodowane jest to zbyt szybkimi zmianami napięć w tych węzłach, które nie mogą być podstawą do predykcji. Proponowana metodologia polega na: i) ustaleniu dokładnej chwili przełączenia, gdy przełącznik przewodzi; ii) wstawieniu jako rozwiązania pseudo rozwiązania dla dc. W tym przypadku jeden z węzłów przełącznika może być określony jako sterujący, a drugi jako sterowany. W chwili przełączenia węzeł sterujący jest wykorzystywany do wyznaczenia wartości sygnału w węźle sterowanym.

8.12 Metody przyśpieszania analizy

W rozdziale 1.3 wymieniono podstawowe techniki redukcji nakładów obliczeniowych. W tym rozdziale omówione zostaną wybrane techniki należące do grupy technik usuwających nadmiary obliczeniowe, a mianowicie:

1. technika omijania [J.O95][M.N91],
2. technika omijania (zamrażania) bloków [RA89b][T.S88],
3. technika usypiania węzłów/bloków w czasie [J.O95][RS94],
4. techniki modelowania:
   modele odcinkowo-liniowe [R.K94][J.D99],
   modele uproszczone [J.D99][B.R75][D.O77],

5. technika dopuszczania obliczeń niedokładnych.

8.12.1 Omijanie

Omijanie(*ang. bypassing*) [J.O95][RS94] jest standardową techniką redukcji nakładów obliczeniowych stosowaną w większości programów symulacji układów analogowych. Jest rozszerzeniem testu stopu. Polega ona na sprawdzeniu warunku przerwania iteracji NR gałęzi nieliniowych przed obliczeniem ich modeli iteracyjnych w bieżącej iteracji. Jeżeli wartości sterowań ustaliły się na tyle, że nie ma potrzeby obliczania dla niej modelu iteracyjnego, to wystarczy zapamiętać ostatnio wyliczane wartości i używać ich aż do końca analizy w danym punkcie czasowym. Każda gałąź spełniająca kryterium omijania spełnia także kryterium testu stopu NR. Dochodzenie do zbieżności dla różnych gałęzi nieliniowych (różnych elementów) nie jest równomierne, więc omijanie obliczeń parametrów gałęzi nieliniowych, dla których nie uzyskano zbieżności, prowadzi do redukcji czasu obliczeń w przypadku, gdy część gałęzi wciąż wymaga wykonywania iteracji NR. Ze względu na skomplikowanie modeli elementów nieliniowych (np. tranzystorów MOS, bipolarnych) najwięcej czasu analizy zajmuje obliczenie modeli nieliniowych. Stosując omijanie nakłady te można istotnie ograniczyć.

8.12.2 Zamrażanie bloków

Technika zamrażania bloków [J.O95][RS94], stosowana w analizie blokowej, jest rozszerzeniem techniki omijania na bloki elementów. Jeżeli wszystkie sygnały w bloku niższego poziomu ustaliły się na tyle, że możliwe jest ominięcie dalszego procesu iteracyjnego NR na poziomie tego bloku, to zostają zapamiętane wartości rozwiązań i pochodnych z ostatniej iteracji NR i blok uzyskuje status omijanego (zamrożonego). Dalsze iteracje NR w bloku wyższego poziomu wykonywane są dla ustalonych wartości rozwiązań i pochodnych bloku niższego poziomu.

W analizie blokowej, przed wejściem w iteracje dolnego poziomu dla bloków, należy dla każdego z bloków sprawdzić czy nie jest on już omijany. Jeżeli nie jest, to należy sprawdzić czy:

1. zmienne blokowe od ostatniej $p-1$ iteracji zmieniły się dostatecznie mało, tzn. czy każda i-ta zmienna w danym bloku spełnia warunek $\mid x_i^p - x_i^{p-1} \mid \leq \delta + \epsilon \max(\mid x_i^p \mid, \mid x_i^{p-1} \mid)$,
2. wszystkie elementy nieliniowe danego bloku zostały ominięte.

Jeżeli powyższe warunki zostały spełnione, to blok może otrzymać status omijanego. Jeśli blok ma już status bloku omijanego, to status ten należy podtrzymać, gdy wszystkie i-te zmienne sterujące (na zaciskach bloku) zmieniły się mało w porównaniu z wartościami, przy których blok ten otrzymał status omijanego, tzn. wszystkie spełniają zależność $\mid x_i^p - x_i^{p_b} \mid \leq \delta + \epsilon \max(\mid x_i^p \mid, \mid x_i^{p_b} \mid)$, gdzie p_b jest indeksem iteracji NR pierwszego ominięcia bloku.

8.12.3 Usypianie

Jedną z najważniejszych technik przyśpieszania obliczeń jest technika usypiania węzłów (ang.latency). Jeśli w pewnej chwili czasowej stwierdzimy, że sygnał w danym węźle mało się zmienia, to węzeł można uznać za uśpiony (zamrożony w czasie) i w kolejnych punktach czasowych nie obliczać wartości sygnału w tym węźle. W analizie blokowej

odpowiada to nieobliczaniu wartości sygnału dla bloku uśpionego niższego poziomu (nie stosuje się obliczeń w pętli NR na poziomie bloku uśpionego). Węzeł można uznać za uśpiony, jeśli spełnia kryterium uśpienia, czyli dostatecznie małej zmiany sygnału od ostatniej chwili $| x_{i,t_m} - x_{i,t_{m-1}} | \leq \delta + \epsilon \max(| x_{i,t_m} |, | x_{i,t_{m-1}} |)$. Dla bloków indeks węzłów i przebiega przez wszystkie węzły bloku. Analogiczne kryterium należy zastosować do wszystkich zmiennych sieciowych w bloku, tzn. dla prądów pojemności, napięć indukcyjności oraz zmiennych ładunkowych i strumieniowych. Dodatkowo należy zabezpieczyć się przed wzięciem za stan ustalony odpowiedzi układu o dużych stałych czasowych. W blokach należy dokonać oszacowania lokalnych stałych czasowych związanych z reaktancjami i sprawdzać warunek na przyrosty sygnału na reaktancjach [J.O95]. Dodatkowo należy badać kryteria przyrostów wartości bezwzględnych sygnału od chwili zamrożenia.

W analizie relaksacyjnej uśpienie węzła powoduje, że nie jest on w ogóle obliczany - wartość sygnału w kolejnych chwilach czasowych przyjmowana jest jako wartość z chwili uśpienia. Gdy węzeł jest uśpiony niewielka zmiana sygnału sterującego (nie powodująca *przebudzenia* węzła) nie propaguje się na dalsze węzły. Nawet dla małych zmian sygnału może to być przyczyną dużych błędów obliczeniowych w węzłach zależnych (ang. fanouts). Dlatego też bardzo ważne jest ustalenie prawidłowych kryteriów usypiania/przebudzania węzłów. Ze względu na specyfikę zmian sygnału można podać kilka kryteriów przebudzania węzłów. Realizowane są one przez wprowadzenie współczynników opisujących dopuszczalne zmiany sygnału w węźle. Graficzną interpretację omawianych kryteriów przedstawia rys. 8.20.

Kryterium 1

Najprostszym kryterium (8.62) jest badanie zmian sygnału pomiędzy kolejnymi punktami czasu (rys. 8.20a). Użytkownik musi zadać wartość względnego przyrostu, poniżej którego zmiana sygnału spowoduje uśpienie węzła.

$$| v(t_{m+1}) - v(t_m) | < \delta_v \tag{8.62}$$

Kryterium to nie może być stosowane samodzielnie, gdyż można pokazać prosty przykład, gdzie stan ustalony nie jest osiągany pomimo spełnienia tego kryterium.

Kryterium 1a

Kryterium 1a ma zastosowanie do sygnałów wolnozmiennych, gdy kryterium 1 nie daje prawidłowej informacji. Powolna zmiana sygnału powoduje, że kryterium 1 jest spełnione pomiędzy dwoma sąsiednimi punktami czasowymi pomimo, że sygnał nie osiągnął stanu ustalonego. Kryterium to służy do badania sygnałów wolnozmiennych (rys. 8.20b). W odróżnieniu do kryterium 1 tutaj badany jest przyrost sygnału pomiędzy $r > 1$ punktami czasu (8.63), co lepiej oddaje charakter zmian sygnału. Jeżeli sygnał osiągnie stan ustalony, to kryteria 1 i 1a zostaną spełnione, przy czym kryterium 1a jest silniejsze.

$$| v(t_{m+r}) - v(t_m) | < \delta_v \quad r > 1 \tag{8.63}$$

Kryterium 2

Kryterium to bada pochodne czasowe sygnału w kolejnych punktach czasu - rys. 8.20c dla $r = 1$.

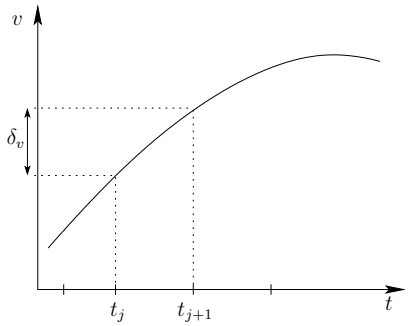
$$\frac{\mid v(t_{m+1}) - v(t_m) \mid}{h_m} < \delta_{\dot{v}} \tag{8.64}$$

Jest ono podobne do kryterium 1, jednak zamiast przyrostów sygnałów bada zmiany pochodnych. Jest ono także zawodne. Jeśli kolejne punkty analizy zostaną wybrane jak na rys. 8.20d, to pomimo nieosiągnięcia stanu ustalonego, kryterium to jest spełnione.

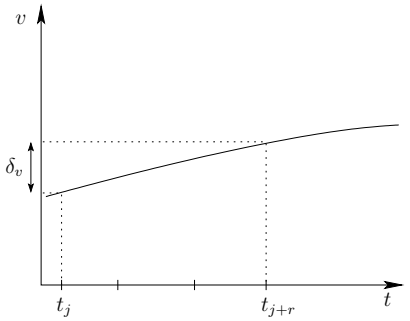
Kryterium 2a

Badaniu podlega zmiana pochodnej czasowej sygnału w r ostatnich punktach czasowych - rys. 8.20c.

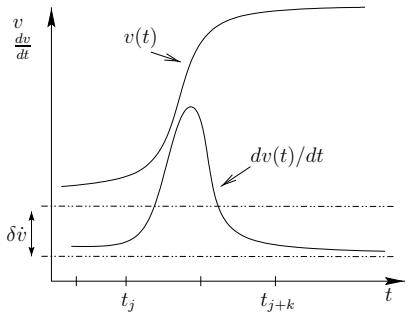
$$\frac{1}{r}\sum_{s=1}^{r} \frac{\mid v(t_{m+2-s}) - v(t_{m+1-s}) \mid}{h_{m+1-s}} < \delta_{\dot{v}} \quad r > 1 \tag{8.65}$$



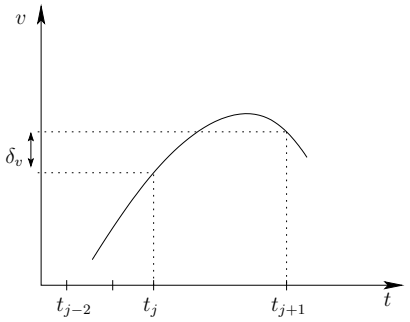
(a) Kryterium 1 - badanie bezwzględnych przyrostów sygnału w kolejnych punktach czasowych



(b) Kryterium 1a - badanie przyrostów sygnału pomiędzy kolejnymi punktami czasowymi



(c) Kryterium 2 - badanie pochodnych czasowych pomiędzy punktami czasowymi



(d) Ilustracja problemu wykrywania stanu ustalonego sygnału zmiennego

Rys. 8.20: Graficzna interpretacja kryteriów usypiania węzłów

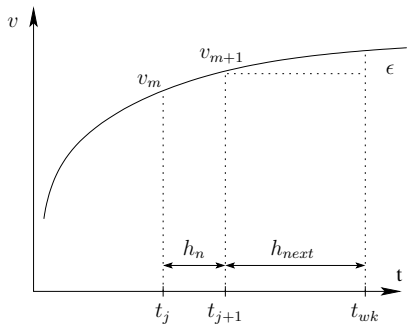
Węzeł nie może pozostawać uśpiony nieskończenie długo, aby nie powstało zafałszowanie wyników analizy. Powolna zmiana sygnału powoduje ciągłe uśpienie węzła i otrzymane wyniki symulacji będą błędne. Dlatego wprowadza się warunek przebudzenia (8.66).

$$h_{next}\frac{\mid v_{t_m+1} - v_{t_m} \mid}{h_{m+1}} > \delta_v \tag{8.66}$$

gdzie: v_{t_m+1} i v_{t_m} są napięciami w kolejnych chwilach czasowych. Liczby $\delta_{\dot{v}}$ i δ_v należy wyznaczyć doświadczalnie lub metodami adaptacyjnymi.

W chwili usypiania węzła wyznacza się maksymalny czas uśpienia (krok h_{next}), po którym węzeł musi znowu stać się aktywny. Mechanizm przebudzenia węzła ilustruje rys. 8.21. Czas ponownego przebudzenia węzła wynikający z wyznaczonego wcześniej h_{next} dany jest równaniem (8.67).

$$t_{wake-up} = t_{m+1} + h_{next} \tag{8.67}$$



Rys. 8.21: Ilustracja mechanizmu wyznaczania czasu przebudzenia węzła

8.12.4 Uproszczone modele elementów

Modele odcinkowo-liniowe Modele odcinkowo-liniowe [R.K94][J.D99][Eij83][Bok81] są często stosowane w przypadku, gdy wymagane jest odwzorowanie charakterystyk modeli elementów, a zależy nam na szybkości analizy. Charakterystyki wyjściowe elementów dzielone są na przedziały, w których stosuje się przybliżenie liniowe wymagające obliczenia uproszczonych funkcji gałęziowych. Na granicy przedziałów popełniane są zazwyczaj największe błędy przybliżeń. Podejście to jest w wielu przypadkach wystarczające. Jest ono stosowane np. w analizie układów cyfrowych, gdy wymagane jest dobre oszacowanie opóźnień w układzie, a modelowanie stanów logicznych i opóźnień wnoszonych przez bramki jest niewystarczające.

Modele stabelaryzowane W przypadku bardziej skomplikowanych modeli elementów można stosować stabelaryzowane modele elementów (*ang. lookup table models*) [B.R75][D.O77]. Wartości funkcji gałęziowych wyznaczane są na etapie tworzenia modelu lub przez pomiar charakterystyk na wyjściach elementów i zapisywane są w wielowymiarowych tablicach. Upraszcza to proces modelowania, gdyż pomijany jest proces

identyfikacji i ekstrakcji parametrów modelu elementu. W przypadku analizy, wartości funkcji gałęziowych są odczytywane z odpowiednich miejsc tabel i nie ma potrzeby wykonywania skomplikowanych obliczeń.

8.13 Metody relaksacyjne w wybranych symulatorach

8.13.1 ELDO-XL

W symulatorze ELDO-XL [Ana97] użyto kombinacji technik relaksacyjnych i bezpośrednich określanych jako opartych na technice Newtona-Raphsona (rozdział 4.2.2). Program wykorzystuje technikę OSR (rozdział 8.3). Wprowadzone modyfikacje przyśpieszają obliczenia dla układów *cyfrowych*. Spostrzeżono, że dla układów złożonych z elementów o jednokierunkowym przepływie sygnału, do osiągnięcia zbieżności potrzeba około dwóch relaksacji. Druga relaksacja wykonywana jest w celu sprawdzenia czy osiągnięto zbieżność. Zakładając, że zbieżność osiągnięto po jednej relaksacji, zrezygnowano ze sprawdzania testu stopu i wykonywania dalszych relaksacji. Założenie to jest słuszne dla węzłów słabo sprzężonych. Dlatego też firma zaleca wykonanie dwóch symulacji próbnych, dwoma metodami: tradycyjną i relaksacyjną. Jeżeli otrzymane wyniki będą *takie same*, to można stosować technikę relaksacyjną.

W programie nie zaimplementowano dynamicznego ani statycznego podziału blokowego układu. Użytkownik we własnym zakresie musi zdecydować, które podukłady będą traktowane jako *analogowe* (analizowane metodami bezpośrednimi), a które jako *cyfrowe* (analizowane metodami relaksacyjnymi). Podanie odpowiedniej opcji w linii deklaracji podobwodu określa jego rodzaj. Zadeklarowanie podobwodu jako *cyfrowego* powoduje przyjęcie uproszczonych modeli tranzystorów MOS (modele odcinkowo-liniowe RC).

Takie podejście prowadzi do analizy prostych układów RC ze źródłami sterowanymi i blokami *analogowymi* analizowanymi metodami bezpośrednimi, co istotnie przyśpiesza analizę kosztem niewielkiego pogorszenia dokładności. Należy mieć to na uwadze przy interpretacji wyników. Przyjęcie uproszczonych modeli tranzystorów prowadzi do dodatkowych błędów będących wynikiem pominięcia niektórych zjawisk w modelach elementów.

8.13.2 PYRAMID

Symulator układów elektronicznych PYRAMID [P.S88][P.S93] przeznaczony jest do analizy dużych układów złożonych z tranzystorów FET i bipolarnych, których rozmiar nie przekracza osiemnastu tysięcy elementów. W programie wykorzystano następujące techniki:

1. hierarchiczną analizę przebiegów czasowych (HWR - *ang. Hierarchical Waveform Relaxation*) wspomaganą dodatkowo techniką okien czasowych (*ang. windowing* - roz. 8.4.1);

2. dynamiczne łączenie i przegrupowywanie tzw. super bloków[5] opierające się na badaniu kryteriów zbieżności metody HWR; W rezultacie otrzymano skrócenie czasu analizy dla układów z elementami dwukierunkowymi w porównaniu z podziałem hierarchicznym (logicznym, bramkowym) układu. Wstępnie wykonywany jest podział hierarchiczny układu, a następnie na podstawie badania Jakobianu dokonuje

[5]Przez *super blok* twórcy programu rozumieją podobwód (blok) zawierający elementy dwukierunkowe takie jak: rezystory, pływające kondensatory, tranzystory bipolarne.

się dalszego podziału, tak aby uzyskać jak największą liczbę super bloków. Dla bloków stosuje się metody macierzowe, a pomiędzy nimi metody relaksacyjne rozwiązywania równań. Jeśli liczba super bloków jest mała, tzn. zawierają one wiele węzłów, analiza WR staje się nieefektywna.

3. analiza poszczególnych części układu (poziomów) odbywa się z wykorzystaniem indywidualnych kryteriów oceny błędów, co umożliwia analizę krytycznych części układu z większą dokładnością.

Proces symulacji przebiega w kilku etapach. Po sprawdzeniu składni i wczytaniu układu tworzone jest drzewo podukładów. W następnym kroku wykonywany jest podział hierarchiczny (rozdział 8.9.5), tzn. podukłady, zawierające elementy, dzielone są na mniejsze podukłady (*ang. clusters*) silnie sprzężonych elementów, co prowadzi do stworzenia nowego podobwodu na niższym poziomie (*ang. levelizing*).

Po fazie wstępnej wykonywana jest analiza dc. Wszystkie podobwody niższego poziomu zawierające silnie sprzężone podukłady (*ang. clusters*) rozwiązywane są metodą NR (rozdział 5.2) z macierzami pełnymi, natomiast podukłady wyższego poziomu rozwiązywane są techniką iteracyjną Gaussa-Seidela(rozdział 4.2.2). Na każdym poziomie równania lokalne rozwiązuje się aż do uzyskania zbieżności, a następnie przechodzi się na wyższy poziom. Program dołącza do każdego węzła uziemiony kondensator w celu przyśpieszenia zbieżności. Dynamiczny podział jest możliwy w przypadku wolnej zbieżności. Otrzymane rozwiązania wykorzystywane są jako warunki początkowe analizy czasowej.

W analizie czasowej zakres analizy $[0, T]$ dzielony jest na okna czasowe, co umożliwia uzyskanie szybszej zbieżności [J.W87][J.W83]. Niższe poziomy podukładów analizowane są tylko wtedy, gdy sygnały wejściowe z wyższego poziomu zmienią się o ustaloną wartość. W takim przypadku inicjalizowane jest nowe okno czasowe, w którym analizowane są węzły podukładów z niższych poziomów. W przeciwnym razie podukłady z niższego poziomu nie są analizowane - pozostają uśpione (*ang. latency*). Tak jak w innych symulatorach, do każdego węzła dołączany jest uziemiony kondensator przyśpieszający zbieżność. W przypadku słabej zbieżności lub zbyt krótkiego kroku analizy, następuje przegrupowanie podukładów z niższego poziomu.

8.13.3 SPLICE

Program SPLICE[A.R79] powstał w latach 70-tych. Jego pierwsza wersja umożliwiała wykonanie następujących symulacji:

- logicznej,
- symulacji *timing* wykorzystującej metodę SOR z wewnętrzną pętlą Newtona-Raphsona,
- symulacji układowej wykorzystującej techniki bezpośrednie (macierze pełne i iteracje Newtona).

Analiza hybrydowa wykorzystywała wszystkie powyższe techniki jednocześnie. Pomiędzy symulatorami *logicznym* i *analogowym* wprowadzono interfejs, którego zadaniem była zamiana poziomów logicznych (H,1,*,0) na poziomy napięć i odwrotnie.

W programie zastosowano odcinkowo-liniowe modele elementów. Wartości opisujące charakterystyki modeli zostały umieszczone w tabelach (*ang. look-up tables*). Skróciło to czas analizy, gdyż wyeliminowano konieczność obliczania Jakobianu. W analizie układowej stabelaryzowane wartości muszą być interpolowane w celu likwidacji nieciągłości

funkcji i zapobiec problemom ze zbieżnością. W analizie *timing* interpolacja nie była stosowana, gdyż wykonywany był tylko jeden krok SOR-Newton'a w każdym punkcie czasowym i dlatego też nie było problemów ze zbieżnością.

SPLICE3

W nowej wersji programu SPLICE3[RA89b] zastosowano nowe techniki symulacji *timing*:

- zaawansowaną wersję ITA wykorzystującą usypianie węzłów;
- algorytm kierowany zdarzeniami oparty na ITA, który wykorzystuje:
  - *ang. multirate behavior of circuit* [RA89b],
  - kierowane zdarzeniami przybliżanie chwil czasowych analizy węzłów zarządzający cofaniem analizy w czasie (*ang. step rejection*) znanym *ang. selective backup*.

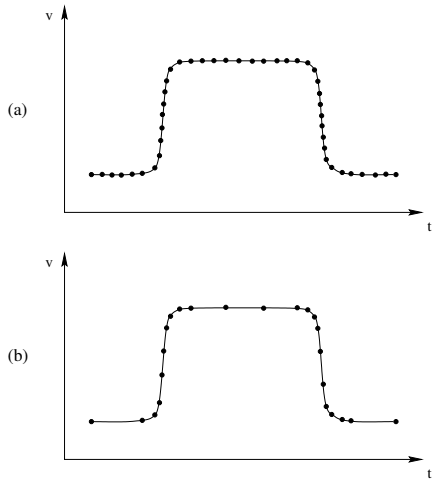
W celu przyśpieszenia zbieżności, węzły silnie sprzężone umieszczane są w podobwodach analizowanych technikami bezpośrednimi. Jeżeli dwa węzły są sprzężone bilateralnie, to umieszcza się je w jednym obwodzie. W przypadku silnego sprzężenia unilateralnego (np. węzeł A jest silnie sprzężone z B, ale węzeł B nie jest silnie sprzężony z A), węzły zostają umieszczone w różnych obwodach. Ważna jest wtedy kolejność analizy węzłów zgodnie z kierunkiem przepływu sygnału (węzeł A przed B).

Program umożliwia wprowadzenie podziału blokowego: statycznego[J.W85b] i dynamicznego wykonywanego w trakcie analizy[A.R84].

W programie wykorzystano informacje o kształtach przebiegów *ang. multirate behavior*. Każdy sygnał ma inny kształt i powinien być analizowany w różnych punktach czasowych w przedziale przyjętego okna czasowego. Aby wyznaczyć prawidłowe punkty czasowe analizy wykorzystano post-procesor, który na podstawie dokładnego kształtu sygnału (otrzymanego przez analizę sygnału ze stałym krokiem metodami bezpośrednimi w przedziale przyjętego okna czasowego), pozostawia tylko punkty prawidłowo opisujące przebieg (z zadaną dokładnością). Na rys. 8.22 przedstawiono przebieg przykładowego sygnału i ten sam sygnał po wyjściu z post-procesora.

Post-procesor wykorzystuje dwa algorytmy klasyfikacji punktów analizy: algorytm przyrostowy i algorytm całkowania drugiego rzędu.

Algorytm przyrostowy sprawdza zmiany sygnału względem ustalonego punktu t_k. Wszystkie punkty $t_m > t_k$ spełniające zależność $\mid v(t_m) - v(t_k) \mid < \epsilon$ są eliminowane. Następnie jako punkt odniesienia przyjmowany jest punkt, który nie spełnił powyższego warunku i procedura powtarza się aż do osiągnięcia końca przedziału. Zadana wartość ϵ wpływa na liczbę generowanych punktów czasowych. Uzyskiwane przyśpieszenie zależne jest od liczby usuniętych punktów czasowych i może zostać oszacowane z zależności: $S = \frac{N * T_{total}}{\sum_{i=1}^{N} T_{removed}}$, gdzie: N - liczba węzłów, T_{total} - początkowa liczba punktów analizy, $T_{removed}$ - liczba usuniętych punktów analizy. Jeżeli $S = 1$ to sygnały w układzie szybko się zmieniają. $S >> 1$ oznacza, że sygnały w niektórych węzłach zmieniają się powoli i zastosowanie usypiania da wymierne korzyści.

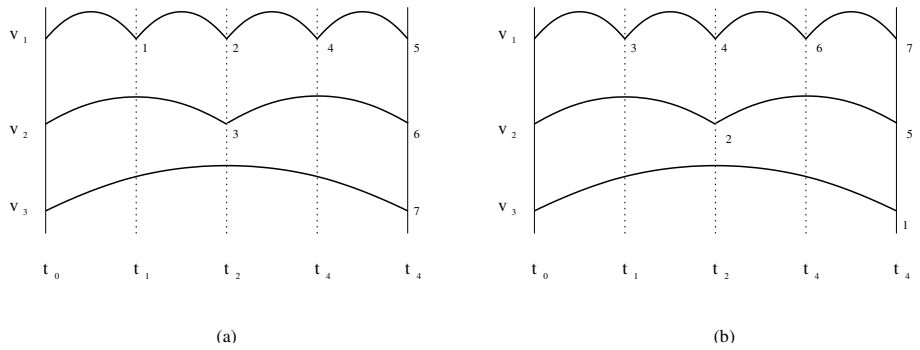


Rys. 8.22: Punkty analizy przed (a) i po działaniu post-procesora (b)

Algorytm całkowania drugiego rzędu przewiduje wartości sygnału w każdym punkcie t_m i począwszy od punktu t_k sprawdza czy przewidywane wartości mieszczą się w granicach dokładności ϵ zadanej przez użytkownika. Poziom błędu ϵ wpływa na liczbę generowanych punktów czasowych. Przyśpieszenie analizy można oszacować tak samo jak dla poprzedniego algorytmu, jednak inna jest interpretacja wyników: $S = 1$ oznacza, że punkty czasowe najlepiej opisują dany sygnał; $S >> 1$ oznacza, że punktów analizy było za dużo i sygnał był *podatny* na zastosowany algorytm.

W programie zastosowano usypianie elementów (*ang. dormant models*) i węzłów. W trakcie analizy występują błędy związane z rozdziałem węzłów w czasie (*ang. time-domain decoupling*) - brakiem synchronizacji czasowej sygnałów i pochodnych w węzłach. Błędy te propagują się na węzły zależne i mogą wpływać na dokładność rozwiązania całego układu. W programie zastosowano techniki ekstrapolacji lub interpolacji wartości sygnałów w celu synchronizacji czasów analizy węzłów - usunięcie rozdziału czasowego węzłów. Technika ta nie zawsze jest skuteczna, gdyż nigdy nie ma pewności czy rozwiązanie będzie prawidłowe. Węzły wzajemnie wpływają na siebie zaburzając rozwiązanie. Po jakimś czasie może okazać się, że rozwiązanie w danym węźle nie było prawidłowe i należy wykonać krok wstecz (także dla węzłów zależnych), tracąc w ten sposób poniesiony nakład obliczeniowy. Aby zabezpieczyć się przed cofaniem analizy w węzłach zależnych należy wykonać analizę węzłów w odpowiedniej kolejności. Możliwe są tu dwa podejścia:

- całkowanie najpierw z najkrótszym krokiem (*ang. Smallest-Step-First Multirate Method*) - rys. 8.23(a). Analiza węzłów przebiega w kolejności rosnącego kroku. Wadą metody jest konieczność cofania analizy dla wszystkich węzłów zależnych, jeżeli węzeł analizowany z najdłuższym krokiem nie uzyskał zbieżności.
- całkowanie najpierw z najdłuższym krokiem (*ang. Largest-Step-First*), w którym najpierw analizuje się węzły o najdłuższym kroku całkowania - rys. 8.23(b). W przypadku braku zbieżności skraca się krok tylko dla tego jednego węzła i ponownie wykonuje się obliczenia. Zabezpiecza to przed cofaniem dla wszystkich węzłów zależnych.



Rys. 8.23: Kolejność obliczania węzłów dla metody SSF(a) i LSF(b)

Jak już wcześniej wspomniano niespełnienie kryteriów dokładności analizy w danym węźle powoduje konieczność cofnięcia i doboru nowego kroku całkowania. Sygnał takiego węzła wpływa na węzły sąsiednie i dla nich także należałoby wykonać krok wstecz i wybrać nowy krok analizy. Jest to jednak działanie dosyć kosztowne. Dlatego w programie SPLICE zastosowano technikę *ang. selective backup strategy*. Rozwiązania pamiętane są tylko dla tych węzłów, które prawdopodobnie będą wymagać wykonania kroku w tył w przypadku niespełnienia kryteriów dokładności. Np. jeśli w przypadku ciągu inwerterów należy wykonać krok wstecz dla węzła v_i, to krok w tył należy wykonać także dla węzłów z sąsiedztwa: $v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}$.

Pierwszym krokiem strategii *ang. selective backup* jest wykonanie analizy z nowym krokiem w zakresie $[t_{start}, t_{old}]$ (t_{old} jest punktem, z którego należy wykonać krok wstecz). Następnie należy znaleźć punkt $t_{diff} \in [t_{start}, t_{end}]$, dla którego nowa wartość v_i^{new} różni się od wartości v_i^{old}. Po określeniu t_{diff} węzły zależne analizuje się z nowym krokiem w zakresie $[t_{diff}, t_{end}]$, zamiast $[t_{start}, t_{end}]$. Algorytm ten stosuje się dla kolejnych węzłów zależnych w celu określenia punktów czasowych, od których rozwiązania różnią się i należy je ponownie przeanalizować dobierając nowy krok analizy.

8.13.4 MOTIS-3

Symulator MOTIS[D.T86][B.R75] przeznaczony jest do symulacji układów z tranzystorami MOS. W programie zastosowano algorytm *timing* oraz *fast-timing*. Program jest do trzech rzędów wielkości szybszy od symulatora SPICE. W celu przyśpieszenia działania wykorzystywane są stabelaryzowane modele odcinkowo-liniowe (*ang. table look-up model*) otrzymane na podstawie danych pomiarowych lub symulacji. Węzły sprzężone dwukierunkowo grupowane są w podobwody. Elementy nieliniowe w podobwodach zastępowane są stabelaryzowanymi makromodelami (źródła prądowe). Po dołączeniu kondensatorów podobwody zastępowane są makromodelami złożonymi tylko z pojemności i źródeł prądowych.

W programie zastosowano kwantyzację poziomów sygnałów z korekcją. Przedział napięć $[v_{min}, v_{max}]$ dzielony jest na S poziomów. Tak więc napięcie $v_{s+1} = v_s + \Delta v$. Krok czasowy analizy h wyznaczany jest na podstawie Δv. Szybkość i dokładność analizy zależy od Δv. Jeżeli użyto zbyt wielu poziomów kwantyzacji, to analiza może okazać się wolniejsza od analizy tradycyjnej (np. wykonanej programem SPICE).

Zakładając, że do każdego wyjścia bloku lub bramki dołączony jest kondensator C, napięcie w punkcie czasowym t_m wynosi $v_{t_m} = v(t_m)$ i zadany przez użytkownika po-

ziom kwantyzacji napięcia wynosi $\Delta v = v_{s+1} - v_s$, proces analizy przebiega następująco:

1. Obliczany jest prąd płynący z wyjścia I, na podstawie tablicy wartości z makromodelu.
2. Obliczany jest krok czasowy pomiędzy dwoma sąsiadującymi poziomami napięć v_{s+1} i v_s z modelu liniowego $h_s = C\frac{\Delta v}{I}$.
3. Obliczane jest napięcie $v_{s+1}^{'}$ w punkcie t_{m+1} algorytmem Gaussa-Seidela.
4. Wykonywana jest korekcja kroku czasowego $h_s^{'} = h_s(v_{s+1}^{'} - v_s)/\Delta v$, gdzie $h_s^{'}$ jest odległością czasową pomiędzy poziomami napięć v_s i v_{s+1}. Skorygowana wartość kroku używana jest do wyznaczenia kolejnego punktu czasowego $t_{m+1} = t_m + h_s^{'}$.

8.13.5 WCaZM

WCAZM[D.J92] jest wersją symulatora CAZM[D+90] przeznaczoną do analizy dużych układów CMOS. Program wykorzystuje dwie techniki relaksacyjne:

- *ang. Waveform Ralaxation Newton (WRN)* - rozdział 8.4.2,
- *ang. Newton Waveform Ralaxation (NWR)* - rozdział 8.4.3[D.J89].

W programie zastosowano technikę okienną (*ang. windowing*) i blokowy algorytm Gaussa-Seidela. Algorytm NWR dobrze sprawdza się w przypadku sprzężeń pomiędzy węzłami. Węzły silnie sprzężone grupowane są w bloki analizowane metodami bezpośrednimi. Otrzymane wyniki pokazują, że algorytm NWR jest co najmniej 20% szybszy od WRN. W niektórych przypadkach metoda WRN jest wolniejsza niż analiza bezpośrednia zaimplementowana w symulatorze CAZM (analiza bezpośrednia jest bardzo dobrze zaimplementowana).

8.13.6 SPLIT

Program SPLIT2.1[M.N88] wykorzystuje technikę *ang. Waveform Ralaxation* w połączeniu z metodą bezpośrednią wykorzystywaną do analizy bloków węzłów silnie sprzężonych (*ang. mixed mode simulator*). Blok węzłów silnie sprzężonych traktowany jest jako *ang. "big node"*[6]. W programie zastosowano:

- Techniki dynamicznego podziału układu:
  - *ang. dynamic network separation* badającą Jakobian (rozdział 8.9.8). Separowane są także bloki, których sygnały wejściowe wykazują małe zmiany. Jeżeli sygnały wejściowe bloku, w dwóch sąsiednich chwilach czasowych, spełniają nierówność
    $$\frac{\mid v(t_m) - v(t_{m-1}) \mid}{dt} < \epsilon_{separate}$$
    , to dany blok uznawany jest za uśpiony i jest separowany. Parametru $\epsilon_{separate}$, zadany przez użytkownika, wyznaczany jest doświadczalnie,
  - technikę *ang. gate level network separation technique* (rozdział 8.9.5) wykorzystującą fakt, że dla stanu wysokiej impedancji bramki elementy pozadiagonalne są w każdej iteracji Newtona odpowiednio małe.

[6]Termin "big node" używany jest przez twórców oprogramowania

- Hierarchiczne usypianie węzłów (*ang. hierarchical time domain latency* lub *ang. selective trace*) (rozdział 8.9.4) wykorzystujące badanie pochodnych czasowych napięć (rys. 8.20c). Algorytm korzysta z zależności

$$\max_i(\frac{\mid v_i(t_m) - v_i(t_{m-1}) \mid}{dt}) < \epsilon_{latent}$$

, gdzie: $v_i(t)$ jest węzłem napięciowym pomiędzy separowanymi blokami, ϵ_{latent} jest parametrem decydującym o usypianiu bloku. Wszystkie węzły wejściowe v_i usypianego bloku muszą spełniać powyższe kryterium.

Testy szybkości wykazują około czterokrotne przyśpieszenie w stosunku do analizy z wykorzystaniem tylko dekompozycji hierarchicznej (bez usypiania) dla liczby węzłów ok. 400. Dla mniejszych układów przyśpieszenie jest nieco mniejsze.

8.13.7 PSPLAX

Program PSPLAX[E.Z92] opracowany został z myślą o obliczeniach równoległych wykorzystujących kilka mikroprocesorów (tutaj 8). Program używa algorytmu Gauss-Seidel WRN. Zrównoleglanie obliczeń zostało wykorzystane w trzech obszarach:

1. Zrównoleglenie obliczeń podobwodów SLP (*ang. subcircuit level parallelism*),
2. Zrównoleglanie obliczeń punktów czasowych PTP (*ang. parallel time point method*) - rozdział węzłów w czasie. W obrębie okna czasowego można wykonać jednocześnie analizę tylu punktów (w k-tej relaksacji), ile jest wolnych w danym momencie procesorów (zasobów) uwzględniając fakt, że część zasobów może być potrzebna do zrównoleglania obliczeń na poziomie podobwodów.
3. Zrównoleglanie obliczeń elementów PME (*ang. parallel model evaluation*). W obrębie podobwodu należy przeliczyć modele elementów i ułożyć macierz układu. Obliczenia modeli i układanie równań można zrównoleglić przyspieszając w ten sposób czasochłonny proces układania równań. Zrównoleglanie to stosowane jest w przypadku analizy dużych podobwodów.

Szybkość pracy programu silnie zależy od liczby procesorów. Wykorzystanie ośmiu procesorów skraca czas analizy od 3 do 6 razy. Dla dwóch procesorów przyśpieszenie nie przekraczało 2 razy a dla 4 procesorów 3.7 raza.

8.13.8 MOTA

MOTA [C.L86] jest symulatorem układów CMOS wykorzystującym jednokrokową technikę Gaussa-Seidela lub Gaussa-Jacobiego OSR (*ang. one-sweep relaxation*). Program umożliwia zdefiniowanie podobwodów silnie sprzężonych węzłów, dla których wykorzystywane są techniki bezpośrednie oparte na macierzach pełnych. W celu przyśpieszenia obliczeń zastosowano technikę omijania [J.O95] i stabelaryzowano wartości wyjść modeli elementów (*ang. look-up table models*). Punkty pomiędzy wartościami zawartymi w tabeli są interpolowane. Zastosowano zmienny krok analizy z bardzo prostym algorytmem doboru kroku. Założono, że jeżeli przyrost wartości napięcia pomiędzy dwoma punktami czasowymi jest mniejszy niż Δv_{min}, to następny krok czasowy analizy jest podwajany. Jeśli przyrost jest większy niż Δv_{max}, to następny krok ulega skróceniu.

Zastosowano także prosty algorytm omijania. Jeżeli zmiana napięcia wejściowego jest mniejsza niż Δv_c, to nie oblicza się powtórnie wartości prądów lub konduktancji elementu, lecz przyjmuje się stare wartości z poprzedniego punktu czasowego.

Szybkość programy jest o około 30% mniejsza niż programu MOTIS-C [B.R75, D.O77], jednak otrzymano lepszą dokładność. Powodem różnicy są użyte w programie MOTIS-C jednowymiarowe modele tablicowe elementów i brak interpolacji wartości wyjść modeli pomiędzy punktami z tabeli.

8.13.9 KMIX

KMIX [YHJ89] jest systemem symulacji układów analogowo-cyfrowych (*ang. mixed-mode simulator*). Łączy on w sobie kilka symulatorów wykorzystujących wspólną bazę danych układu. Należą do nich:

- symulator logiczny działający na poziomie bramek (*ang. gate level simulator*) - KLOG [Y.H88],
- symulator działający na poziomie przełączników (*ang. switch level simulator*) - KSWITCH [H.J88],
- symulator *timing* - KTIME [Y+89].
- symulator układowy (*ang. circuit simulator*) - SPICE2 [A+81].

Układ elektroniczny dzielony jest przez użytkownika na podukłady. Użytkownik może wybrać do symulacji każdego z nich jeden z wymienionych symulatorów. Umożliwia to łatwe zastosowanie programu w projektowaniu hierarchicznym. Interfejs pomiędzy sygnałami analogowymi a cyfrowymi zrealizowany został przez zastosowanie funkcji przełączającej (*ang. threshold function*).

8.13.10 ELOSIM

W programie ELOSIM [Y.H89] zastosowano oryginalne podejście do symulacji poprzez inne modelowanie elementów. Zamiast poszukiwać rozwiązania dla zmiennych sieciowych w danym punkcie czasu (tak jak w konwencjonalnych symulatorach), wyznaczane są odcinki czasu determinujące zmianę wartości zmiennych sieciowych o określoną wartość, zwaną poziomem logicznym. Odstęp pomiędzy poziomami definiowany jest przez użytkownika.

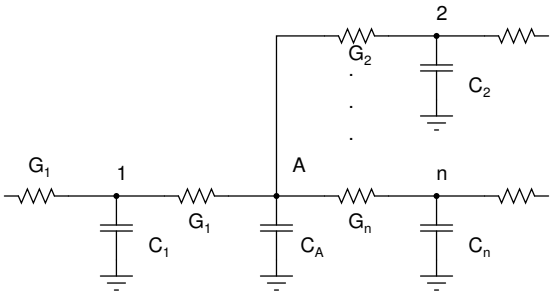
W programie zastosowano algorytmy analizy ELOGIC-1,2,3 oparte na metodzie potencjałów węzłowych. Zapewniają one większą prędkość lub precyzję pomiędzy poziomami symulacji układowej i logicznej (na poziomie przełączników). Algorytmy te są dokładniejsze od algorytmów symulacji logicznej przy porównywalnej prędkości.

Algorytm ELOGIC

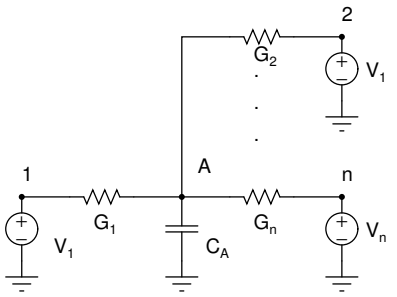
Zadaniem algorytmu ELOGIC jest wyznaczenie odcinków czasu Δt, w których sygnał zmienia się o Δv. Sygnały mogą przyjmować tylko określone wartości z ustalonego wcześniej zbioru, np. co $0.2V$. Są one traktowane jako uogólnione poziomy logiczne. Wartości sygnałów określone są tylko w punktach odpowiadających zdefiniowanym poziomom logicznym. Wektorem wejściowym dla układu jest wektor zmian stanów w dyskretnych punktach czasu. Wektor ten (*ang. waveform fragment*) jest wektorem wejściowym dla modeli elementów ze skończoną liczbą stanów wewnętrznych (*ang. finite-state-machine*

model). Przebiegi sygnałów obliczane są za pomocą symulatorów relaksacyjnych (np. iSPLICE). Jeżeli wartości nie odpowiadają zdefiniowanym poziomom logicznym to są przybliżane do najbliższego poziomu. Propagacja sygnału w układzie generuje szereg zdarzeń. Jest ich tym więcej, im mniejszy jest odstęp pomiędzy poziomami Δv.

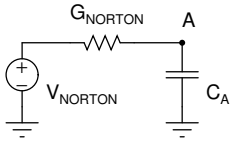
ELOGIC-1 Algorytm ELOGIC-1 wykorzystuje SR Eulera otwarty do dyskretyzacji pojemności uziemionych. Pojemności nieuziemione nie są akceptowane. Na rys. 8.24a przedstawiono przykładowy schemat układu.



(a) układ oryginalny



(b) układ uproszczony



(c) układ ze źródłem zastępczym

Rys. 8.24: Przykładowy schemat układu

W danym punkcie czasu analizowany jest węzeł A. Po zastąpieniu węzłów oddziałujących - stałymi źródłami napięcia, otrzymamy schemat uproszczony przedstawiony na rys. 8.24b. Dla węzła A można wyznaczyć parametry źródła Nortonowskiego (8.68).

$$\begin{aligned} G_{NORTON} &= \sum_{i=1}^{N} G_i \\ I_{NORTON} &= \sum_{i=1}^{N} v_i * G_i \end{aligned} \tag{8.68}$$

Następnie można wyznaczyć parametry zastępczego źródła Thevenina (8.69). Otrzymany uproszczony układ zastępczy, pokazany na rys. 8.24c, służy do obliczenia napięcia w węźle A.

$$V_{THEV} = \frac{I_{NORTON}}{G_{NORTON}} \tag{8.69}$$

Zmiana sygnału w węźle A, pomiędzy kolejnymi stanami, wystąpi po czasie h_A (8.70).

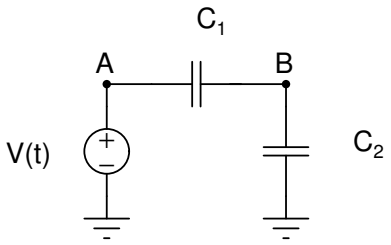
$$h_A = \frac{v_{A,t_m+1} - v_{A,t_m}}{\dot{v}_{A,t_m}} = \frac{(v_{A,t_m+1} - v_{A,t_m})C_A}{\dot{i}_{A,t_m}} = \frac{(v_{A,t_m+1} - v_{A,t_m})C_A}{I_{NORTON} - v_{A,t_m}G_{NORTON}} \tag{8.70}$$

Punkt przełączenia (zmiany stanu) ustalono jako $v_{THRESH} = \frac{v_{t_m+1} - v_{t_m}}{2}$. Jeśli $v_{THEV} < V_{THRESH}$, to przyjmuje się, że napięcie w węźle pozostaje niezmienione. W przeciwnym przypadku przyjmowany jest następny stan v^{i+1}. Zdarzenie generowane jest dla wyznaczonego następnego czas analizy $t_{m+1} = t_m + h$.

Słabą stroną algorytmu jest możliwość występowania samooscylacji w węzłach, w których dokładna wartość sygnału oscyluje wokół V_{THRESH}. Dokładniejsze omówienie tego tematu można znaleźć w literaturze [Y.H89].

ELOGIC-2 Algorytm ten, w odróżnieniu od ELIGIC-1, do dyskretyzacji pojemności dołączonych do masy używa SR trapezowego.

ELOGIC-3 Jak wcześniej wspomniano podstawowy algorytm ELOGIC uniemożliwia analizę układów z pojemnościami pływającymi. Wprowadzone modyfikacje umożliwiły analizę takich układów. Algorytm zmodyfikowany nazwano ELOGIC-3.



Rys. 8.25: Układ z pojemnością pływającą.

Jeśli pomiędzy dwa węzły A i B włączona jest pojemność pływająca (rys. 8.25) i napięcie w węźle A zmienia się, to zmianę napięcia w węźle B wyznacza się z zależności (8.71).

$$\Delta v_B = \Delta v_A = \frac{C_1}{C_1 + C_2} \tag{8.71}$$

8.13.11 Inne symulatory

W pracy nie omówiono wszystkich symulatorów stworzonych w ciągu wielu lat, jakie upłynęły od opracowania relaksacyjnych metod analizy. Omówiono tylko niektóre z nich. W tym rozdziale zostaną omówione w skrócie inne programy symulacji. Podana zostanie literatura, w której można znaleźć bardziej szczegółowe informacje.

iMACSIM Program [R.A96] przeznaczony jest do symulacji dużych systemów z czasem ciągłym i dyskretnym, w tym także do symulacji układów analogowo-cyfrowych i analogowych należących do równych środowisk (*ang. mixed-domain*). Umożliwia on symulację blokową na różnych poziomach: funkcjonalnym, logicznym, elementów. Każdy blok należący do innego poziomu może być analizowany innymi metodami, z uwzględnieniem specyficznych kryteriów (kierowania bloków do analizy, błędów itp.). Na poziomie bloków elementów MNA stosowana jest analiza tradycyjna, natomiast pomiędzy blokami analiza relaksacyjna ITA.

Rsim W programie [R.K94] zastosowano analizę na poziomie przełączników. Modele elementów zostały uproszczone - zastąpiono je modelami odcinkowo-liniowymi złożonymi z przełączników oraz pojemności i rezystancji o odpowiednich wartościach. Pozwoliło to na praktyczną symulację układów złożonych z paruset tysięcy tranzystorów w niezbyt długim czasie. Błędy obliczeń opóźnień zawierają się w granicach od 10% do 20%.

SRP Symulator SRP[YC89] jest programem hybrydowym używającym dwóch relaksacyjnych metod rozwiązywania równań: Gaussa-Seidela - dla komputerów o architekturze von-Neumann'a i Gaussa-Jacobiego - dla komputerów zbudowanych na architekturze sieci neuronowej lub wykorzystujących przetwarzanie równoległe. W programie wykorzystano metodę WR i technikę wirtualnych pojemności dołączanych do węzłów układu, dla których Jakobian ma słabo dominującą przekątną. Przed dołączeniem pojemności wirtualnych, elementy pływające o dużej konduktancji zastępowane są uziemionymi źródłami prądowymi dołączonymi do węzłów podłączenia elementów pływających. Otrzymane wyniki dają redukcję nakładów w stosunku do analizy tradycyjnej już przy kilkuset węzłach.

Sisal W symulatorze[B.K89] zastosowano technikę WRN (Waveform Relaxation Newton)[J.W85a] z pojedynczą iteracją Newtona w każdej relaksacji. Zastosowano także technikę okien czasowych przyśpieszającą proces analizy. Uzyskano nieco lepszą wydajność w stosunku do metody WR (rzędu 10%), dla układów z tranzystorami MOS, przy czym wydajność zwiększa się ze wzrostem liczby elementów.

CSWAN W programie CSWAN[P.O90] zastosowano metodę WR, zrównoleglanie obliczeń oraz podział układu na bloki. Umożliwiło to symulację układów o silnie sprzężonych węzłach. Zastosowano podział układu oparty na: 1) podziale statycznym wykonywanym przez użytkownika, 2) podziale na podstawie wyników analizy DC, 3) na podstawie badania dominacji przekątnej macierzy układu (*ang. Norton partitioning*). Zrównoleglanie obliczeń wykonywane jest na kilku poziomach: 1) podobwodu, 2) podgrupy, 3) okna czasowego, 4) obliczania modelu elementu.

SUPER-SPICE Program SUPER-SPICE[T.S88] jest nakładką na symulator SPICE umożliwiającą wykonanie analizy relaksacyjnej na poziomie blokowym. Wykorzystano algorytm WR z oknami czasowymi i metodę GS. Układ dzielony jest na bloki analizowane przez pojedyncze programy SPICE. W miejscu podziału stosowane jest powielanie części bloków tak, aby umożliwić prawidłową symulację. Procesem symulacji i podziału na bloki kieruje program sterujący. Wydajność jest porównywalna do wydajności programu SPICE2. Przy analizie układów zawierających więcej niż 1000 elementów jest ona większa (dla 2000 elementów - dwa razy).

PNAP W programie PNAP[H$^+$88] wykorzystano algorytm WR i metodę GS. Zastosowano także zrównoleglanie obliczeń sterowane przez specjalnie opracowany algorytm wykorzystujący trójpoziomowy algorytm zarządzania wykorzystaniem wolnych procesorów. Program został zaimplementowany na komputerze Balance 8000 złożonym z 12 procesorów wykorzystujących pamięć dzieloną. Dekompozycja układu następuje w pamięci operacyjnej. Program sterujący przydziela każdemu analizowanemu blokowi osobny proces oraz procesor. Wyniki analizy przebiegów w oknie czasowym dla dwóch kolejnych iteracji wykonywane są przez osobne procesy i zapisywane w pamięci dzielonej. Po przeanalizowaniu każdego podobwodu w oknie czasowym, dane są uaktualniane. Wydajność programu dla jednego procesora jest podobna do wydajności symulatora SPICE2. Największy wzrost wydajności (około dziesięciokrotny) uzyskano w przypadku analizy układów z elementami unilateralnymi (bez sprzężeń zwrotnych) i w przypadku analizy układów z silnie sprzężonymi węzłami. W analizie układów z elementami bilateralnymi o średniej sile sprzężeń wydajność była zdecydowanie mniejsza (dwukrotna).

Inne programy symulacji W literaturze spotykanych jest wiele innych programów symulacji wykorzystujących techniki relaksacyjne. Należą do nich np.: MSPICE[J.T84], DIANA[H$^+$80], TOGGLE[H.Y85], RELAX[J.W83, J.W85b].

Rozdział 9

Małosygnałowa analiza częstotliwościowa

Rozważmy sieć opisaną kanonicznym równaniem algebraiczno-różniczkowym postaci (9.1) przy wymuszeniach sinusoidalnych o pulsacji ω.

$$f(x, \dot{x}, t) = s(t) \tag{9.1}$$

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)[J.O94]. 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 siecią przyrostową. Powoduje to 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.

9.1 Równania sieci

Rozważmy równania sieci nieliniowej postaci (9.1) zapisane w postaci zawierającej po prawej stronie wymuszenia $s(t)$. Odpowiedź stałoprądowa sieci jest rozwiązaniem równania (9.2)

$$f(x_0, 0, 0) = s_0 \tag{9.2}$$

gdzie pochodne w równaniu (9.1) zastąpiono zerami, a wymuszenia - stałymi s_0. Praca z małymi sygnałami przyrostowymi odpowiada przybliżeniu równania (9.1) w otoczeniu odpowiedzi stałoprądowej. Rozwijając obie strony równania (9.1), 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

$$f(x_0, 0, 0) + \frac{\delta f}{\delta x}(x - x_0) + \frac{\delta f}{\delta \dot{x}}(\dot{x} - 0) = s_0 + [s(t) - s_0] \tag{9.3}$$

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 (9.3) składniki rzędu zerowego są takie same jak w równaniu kanonicznym (9.2) 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 (9.4).

$$\frac{\delta f}{\delta x}\hat{x} + \frac{\delta f}{\delta \dot{x}}\frac{d\hat{x}}{dt} = \hat{s}(t) \tag{9.4}$$

Otrzymane równanie liniowe stanowi opis kanoniczny sieci identyczny strukturalnie z siecią oryginalną (9.1). 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 (9.4) w stanie ustalonym przy sinusoidalnych wymuszeniach przyrostowych.

Po transformacji sieci immitancyjnej w dziedzinę $j\omega$ powstaje układ immitancyjny małoprzyrostowy postaci (9.5).

$$\left(\frac{\delta f}{\delta x} + j\omega\frac{\delta f}{\delta \dot{x}}\right)\bar{x} = \bar{s} \tag{9.5}$$

$\bar{x}$ 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. W programach symulacji wymuszenie opisywane jest wzorem (9.6).

$$s = MAGN \cdot e^{(jPHASE)} \tag{9.6}$$

gdzie $MAGN$ jest amplitudą wymuszenia, a $PHASE$ jest fazą wymuszenia.

9.2 Algorytm małosygnałowej analizy częstotliwościowej

Algorytm 32 opisuje analizę małosygnałową sieci przyrostowej.

Algorytm 32 Algorytm analizy małosygnałowej sieci.

Require: x_0 ▷ punkt pracy układu do linearyzacji równań
Require: $f_{min}, f_{max}, \Delta_f$ ▷ zakresy analizy i krok analizy
Require: Ustaw wartość s ▷ Wartości wymuszeń małosygnałowych
1: Linearyzuj równania sieci (9.3)
2: **for** $f = f_{min} \dots f_{max}$ **do** ▷ Pętla czasu
3: ułóż równania sieci przyrostowej (9.5)
4: wyznacz rozwiązanie $\bar{x}$ np. metodą rozkładu LU - 4.1.2.
5: zapisz wyniki analizy $\bar{x}$
6: $f = f + \Delta_f$
7: **end for**
8: KONIEC ▷ warunki początkowe dla stanu ustalonego wyznaczone

Do prawidłowego działania algorytmu wymagana jest linearyzacja równań w punkcie pracy. Wymaga to wykonania analizy OP celem wyznaczenia punktu pracy lub wczytania punktu pracy z pliku zewnętrznego - linia 0. W linii 2 rozpoczyna się pętla analizy w zakresie od f_{min} do f_{max} z krokiem Δ_f. Należy ułożyć równania zlinearyzowane (linia 3) oraz rozwiązać równania lub układ równań jedną z metod omówioną w rozdziale 4, np. metodą rozkładu LU.

Rozdział 10

Wybrane metody optymalizacji

10.1 Optymalizacja deterministyczna

Zdefiniujmy wektor parametrów projektowych

$$x = [x_1, \ldots, x_N],$$

gdzie: N oznacza liczbę parametrów projektowych, a $x_1, \ldots, x_i, \ldots, x_N$ są poszczególnymi parametrami projektowymi [1].

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[2]

$$s = s_1, \ldots, s_M$$

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 należy zbudować 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 $FC(x)$

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

[1] Parametry projektowe można ustawić w programie symulacji (zobacz [Pla13] - dyrektywa *.VARP*)

[2] Specyfikacje projektowe ustawiane są w programie symulacji (zobacz [Pla13] - dyrektywa *.SPEC*)

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 nieograniczonym zakresem zmienności:

- 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 leży poza 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 współczynnikiem tłumienia λ.

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

Z równania wyznaczany jest wektor d_U

$$d_U = -(J_k^T J_k + \lambda I)^{-1} J_k^T FC(z^k) \tag{10.1}$$

gdzie: J jest Jakobianem[3] - 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 oznaczymy 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:

$$d_{NS} = -J^T FC(z^k) \tag{10.2}$$

[3]Do estymacji Jakobianu stosuje się SR 1 rzędu

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

$$d_U = -(J_k^T J_k + \lambda \; diag(J^T J))^{-1} J_k^T FC(z^k) \tag{10.3}$$

Proces optymalizacji można zapisać w postaci algorytmu 33 lub algorytmu alg:optim-lv.

Algorytm 33 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 34 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 (10.3)
7: $(S.3)$ $x^{k+1} = P_X(x^k + x^{k+1})$, $k = k + 1$, idź do 2
8: **end if**
9: STOP

W symulatorze *Dero* zastosowano algorytm 35.

10.1.1 Wybrane elementy algorytmu

Obliczenie pochodnej funkcji celu względem parametrów projektowych

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

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

Algorytm 35 Algorytm optymalizacji zastosowany w programie *Dero*.

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

Pochodne funkcji celu względem wewnętrznych zmiennych projektowych dane są zależnością:

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

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[4].

Estymacja Jakobianu

W celu wykonania estymacji Jakobianu J należy wykonać 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) można zastosować 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 ograniczany. Opcje STIN oraz STMAX umożliwiają ograniczenie wartości początkowej oraz maksymalnej r.

Dobór kroku

Oznaczmy kroku wykorzystuje funkcje celu: FC_{min} - najmniejsza wartość funkcji celu, FC_{sq} - suma kwadratów składowych funkcji celu.

$$FC_{np} = FC_{best} + \sum_{i=0}^{M}(FC_i * \frac{dFC_i}{dy})^2 \tag{10.4}$$

$$FC_{sq} = \sum_{i}^{M} FC_i^2 \tag{10.5}$$

$$\tau_{ns} = FC_{min} * 0.9 + FC_{np} * 0.1 - FC_{sq} \tag{10.6}$$

Dla $\tau_{ns} >= 0$ należy wydłużyć krok.

[4]Zmianę wartości można uzyskać przez zadanie odpowiednich współczynników skalowania w programie symulacji (zobacz [Pla13] - dyrektywa $.VAR$)

$$sp = \sum_{i=0}^{M} |F(i)^2 - DU2_{NDIR}[i])| \tag{10.7}$$

$$ss = \sum_{i=0}^{M} (F(i) - DU2_{NDIR}(i))^2 \tag{10.8}$$

$$pj = \frac{\tau_{ns}}{\max(sp + \sqrt{sp^2 + \tau_{ns} * ss}, 1e-20)} + 1.0 \tag{10.9}$$

$$sp = \min(\min(4.0, \tau_{inc}^{(old)}), pj) \tag{10.10}$$

$$dd = \min(STMX^2, sp * dd) \tag{10.11}$$

$$FC_{min} = FC_{sq} \tag{10.12}$$

$$\tau_{inc} = \frac{pj}{sp} \tag{10.13}$$

Dla $\tau_{ns} < 0$ skróć krok

$$dd = max(STIN^2, dd * 0.25) \tag{10.14}$$

$$\tau_{inc} = 1 \tag{10.15}$$

gdzie: $DU2_{NDIR}(i) = \sum_i^M LAMBDA^2 * \Delta Z_{best}(i) + \sum_i^M (J^T * \Delta FC_{best})$ jest tablicą ortogonalnych kierunków optymalizacji funkcji celu.

10.2 Wyznaczanie obszarów sprawności układów liniowych

W niniejszej rozdziale zostaną omówione teoretyczne podstawy techniki jednowymiarowych przeszukiwań ortogonalnych ODOS (*ang. One Dimensional Ortogonal Search*) [J.O84] wykorzystującą analizę biliniowej funkcji układowej. Wynikiem działania jest wyznaczony obszar sprawności przedstawiony na rys. 10.4. W symulatorze *Dero*zastosowano modyfikację techniki ODOS i wprowadzono trzeci parametr zmienny, co umożliwiło wyznaczanie przestrzeni sprawności w trzech wymiarach - rys. 10.5.

Zastosowano następujące oznaczenia:

p - parametr rzeczywisty,

$f_p = f(p)$ - funkcja zależna od parametru p,

x - parametr zespolony zależny od parametru p,

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

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

10.2.1 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:

$$Yx = B \tag{10.16}$$

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

Jeżeli założymy, że $detY \neq 0$, to wówczas istnieje jednoznaczne rozwiązanie postaci (10.17).

$$x = Y^{-1}B \tag{10.17}$$

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 (10.18).

$$x_i = x_i(p) \tag{10.18}$$

Np. jeżeli $y_c = j\omega C$ to $x_i = y_c = x_i(C)$ i $p = C$. Jeżeli we wzorze (10.17) B jest stałe (wymuszenia stałe) to wzór (10.19) opisuje zespoloną funkcją układową w przestrzeni rzeczywistych parametrów sieci P_p.

$$X(x(p)) = X(p) = Y^{-1}B \tag{10.19}$$

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 10.2.1. *(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 biliniowa względem x_i, tzn. jest ilorazem funkcji liniowych*

$$X(p) = \frac{n(p)}{d(p)} = \frac{n_0 + n_l p}{d_0 + d_l p} \tag{10.20}$$

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).

$$x(p_i) = x(p_{i0}) + \frac{x'_i(p_i - p_{i0})}{1 + q_i(p_i - p_{i0})} \tag{10.21}$$

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 12). 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:

$$p_i = p_{i0} + \frac{p'_i(p_i - p_{i0})}{1 + qq_i(p_i - p_{i0})} \tag{10.22}$$

Współczynniki we wzorze (10.22) obliczane są z modelu elementu na podstawie zależności (10.23).

$$\begin{aligned}
p_{i0} &= f_i(p_{i0}) \\
p'_i &= \tfrac{\delta f_i(p_{i0})}{\delta p_i} \\
qq_i &= \tfrac{\delta p'_i}{\delta p_i} \approx \tfrac{\Delta p'_i}{\Delta p}
\end{aligned} \tag{10.23}$$

Dzięki zależności (10.22) 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.

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

Algorytm obliczania 7C

Sposób obliczania 7C wrażliwościowych w przestrzeni X ($7CX$) przedstawia algorytm 36. Obliczenie 7C wymaga wykonania 3 analiz: jednej układu pierwotnego i dwóch układów zastępczych z wartościami współczynników p_i, p_j.

Algorytm 36 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

10.2.2 Analiza biliniowej funkcji układowej

W praktyce najczęściej bada się rzeczywiste funkcje układowe modułu i fazy zespolonej biliniowej funkcji układowej postaci (10.21), 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 (10.24).

$$\|x(p_i)\|^2 = \|x_0 + [x(p_i) - x_0]\|^2 \tag{10.24}$$

Wyznaczając z (10.21) różnicę $x(p_i) - x_0$ i podstawiając do (10.24) otrzymujemy wzór (10.25) na kwadrat modułu zespolonej biliniowej funkcji układowej (ZBFU) wyrażony przy pomocy trzech współczynników (3C).

$$\Delta p_i = p_i - p_{i0} \tag{10.25}$$

$$\|x(p_i)\|^2 = \|x_0\|^2 + \frac{a_1 \Delta p_i + a_2 \Delta p_i}{a_3 \Delta p_i + a_4 \Delta p_i + 1} \tag{10.26}$$

gdzie współczynniki a wyrażone są przy pomocy tzw. trzech współczynników (3C :

x_0,x'_i, q_i).

$$\begin{array}{lcl} a_1 & = & \|x'_i\|^2 + 2Re(x_0^* x'_i q_i^*) \\ a_2 & = & 2Re(x_0^* x'_i) \\ a_3 & = & \|q_i\|^2 \\ a_4 & = & 2Re(q_i) \end{array} \tag{10.27}$$

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

Definicja 10.2.1. *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. 10.2

$$S_L^2(\omega) \leq ||x(p, j\omega)||^2 \leq S_U^2(\omega) \tag{10.28}$$

$$S_L^2 \leq x(p, 0) \leq S_U \tag{10.29}$$

2. ograniczenia amplitudowe i fazowe nałożone na charakterystyki amplitudowe i fazowe w dyskretnych punktach częstotliwości (AC - analiza częstotliwościowa) - rys. 10.1

$$x(p, j\omega) \in \angle(r_L, r_U) \tag{10.30}$$

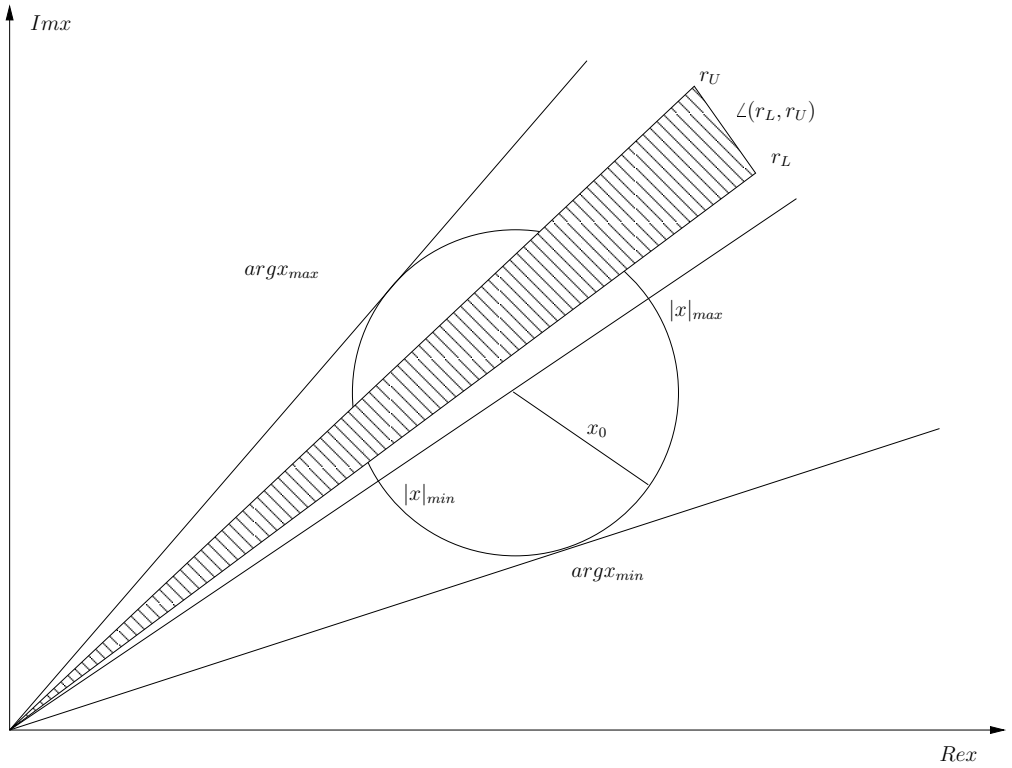
gdzie : S_L, S_U są ograniczeniami dolnym i górnym nałożonym na charakterystykę amplitudową, $x(p, j\omega)$ jest zespoloną biliniową funkcją układową sieci, a $\angle(r_L, r_U)$ reprezentuje stożek na płaszczyźnie zespolonej wyznaczony przez dwie proste r_L,r_U określone przez dolne i górne ograniczenie 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. 10.1).

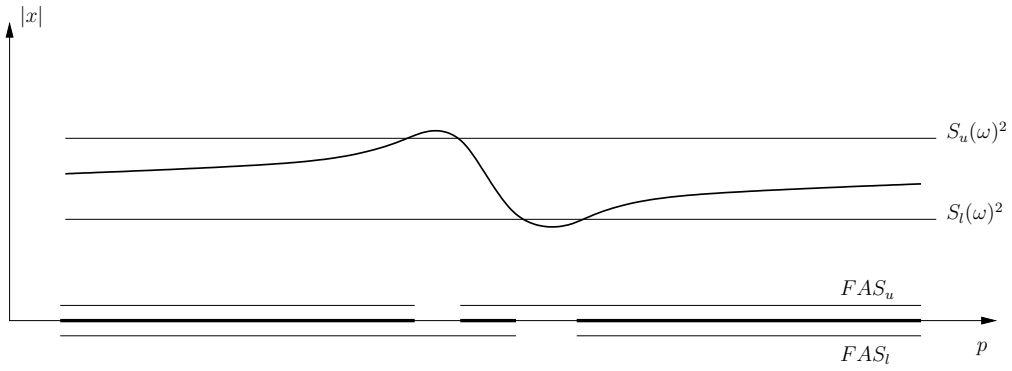
W projektowaniu komputerowym wzory (10.28,10.29,10.30) wykorzystywane są do wyznaczania rodzin odcinków sprawności, przy pomocy których aproksymuje się obszary sprawności. Jeżeli założymy, że wszystkie składowe wektora rzeczywistych parametrów sieci przyjmują wartości nominalne p_0 oprócz jednego parametru 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.

$$\mathrm{LM_i(p_0) = \{p \in R^m p_k = p_{k0} \text{ dla } k = 1, ..m \text{ i } k \neq i\}} \tag{10.31}$$

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



Rys. 10.1: Interpretacja graficzna biliniowej funkcji układowej i ograniczeń fazowych na płaszczyźnie zespolonej dla $\xi > 0$.



Rys. 10.2: Przebieg kwadratu modułu biliniowej funkcji układowej z ograniczeniami amplitudowymi i elementarnymi rodzinami odcinków sprawności dla ograniczenia dolnego i górnego.

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 odcin-

ków 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 (10.28) otrzymujemy dwie nierówności dla ograniczenia dolnego (10.32) i górnego (10.33) (rys. 10.2)

$$\begin{aligned} \mathrm{S}_{\mathrm{L}}^2(\omega_k) - ||\mathrm{x}(\mathrm{p},\mathrm{j}\omega_k)||^2 &\leq 0 \qquad (10.32)\\ ||\mathrm{x}(\mathrm{p},\mathrm{j}\omega_k)||^2 - \mathrm{S}_{\mathrm{U}}^2(\omega_k) &\leq 0 \qquad (10.33) \end{aligned}$$

Nierówności te można rozwiązać jako nierówność kwadratową (10.34).

$$c_2(p_i - p_{i0})^2 + c_1(p_i - p_{i0}) + c_0 \leq 0 \qquad (10.34)$$

Współczynniki dla nierówności (10.32) dane są wzorami:

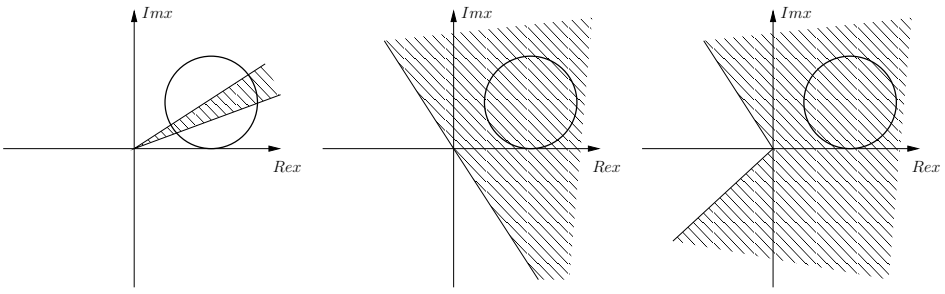
$$\begin{aligned} c_2 &= (S_L^2(\omega_k) - ||x_0||^2)a_3 - a_1\\ c_1 &= (S_L^2(\omega_k) - ||x_0||^2)a_4 - a_2\\ c_0 &= S_L^2(\omega_k) - ||x_0||^2 \end{aligned} \qquad (10.35)$$

Współczynniki dla nierówności (10.34) dane są wzoram:

$$\begin{aligned} c_2 &= (||x_0||^2 - S_U^2(\omega_k))a_3 + a_1\\ c_1 &= (||x_0||^2 - S_U^2(\omega_k))a_4 - a_2\\ c_0 &= ||x_0||^2 - S_U^2(\omega_k) \end{aligned} \qquad (10.36)$$

Współczynniki $a_1 \ldots a_4$ są to współczynniki występujące we wzorze (10.27).

Dla ograniczeń fazowych rodziny elementarne znajdujemy w sposób podobny jak dla ograniczeń amplitudowych. Dla ustalonej ω_k należy rozwiązać dwie nierówności wynikające z zależności (10.30). 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. 10.1. Wygląd obszaru w zależności od wartości ξ przedstawiony został na rys. 10.3.



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

Rys. 10.3: Graficzna ilustracja ograniczeń fazowych na płaszczyźnie zespolonej.

.

Biorąc pod uwagę zależność (10.30) dla przypadku $\xi > 0$ otrzymujemy układ nierówności:

$$\begin{aligned} -Im(r_L)Re(x) + Re(r_L)Im(x) &\geq 0 \quad (10.37) \\ Im(r_U)Re(x) - Re(r_U)Im(x) &\geq 0 \quad (10.38) \end{aligned}$$

Nierówności tę można rozwiązać tak jak nierówność kwadratową (10.34) dla nierówności (10.38) ze współczynnikami

$$\begin{aligned} c_2 &= Im[r_L q_i (x_0 q_i + x_i')^*] \\ c_1 &= Im[r_L (2(Re q_i)x_0 + x_i')^*] \\ c_0 &= Im[r_L x_0^*] \end{aligned} \quad (10.39)$$

Dla nierówności (10.38) w (10.39) 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 (10.29).

$$\begin{aligned} S_L - x(p, 0) &\leq 0 \quad (10.40) \\ x(p, 0) - S_U &\leq 0 \quad (10.41) \end{aligned}$$

Wyznaczając $x(p, o)$ ze wzoru 10.21 otrzymujemy dwie biliniowe nierówności typu

$$\frac{e_0 + e_1(p_i - p_{i0})}{1 + q_i(p_i - p_{i0})} \leq 0 \quad (10.42)$$

ze współczynnikami,

$$\begin{aligned} e_1 &= (S_L - x_0)q_i - x_i' \\ e_0 &= S_L - x_0 \end{aligned} \quad (10.43)$$

dla nierówności 10.41.

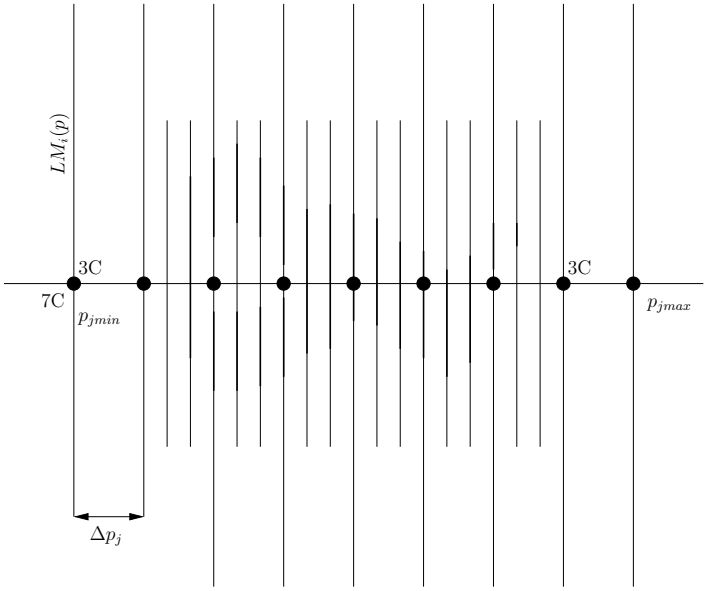
Dla nierówności 10.41 należy zamienić S_L z S_U i zmienić znaki. Mianownik nierówności 10.42 jest różny od zera, więc nierówność tą można rozwiązać tak jak nierówność kwadratową 10.34. W praktyce projektowej jednocześnie aktywnych jest wiele ograniczeń, co implikuje istnienie jednocześnie wielu rodzin 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).*

$$\mathrm{SA}_P(N) = \bigcup_{k=1}^{l} \bigcup_{i \in P} \mathrm{FAS}_i(p_{p^{(k)}}) \quad (10.44)$$

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. 10.4.



Rys. 10.4: Aproksymacja odcinkowo-liniowa obszaru sprawności.

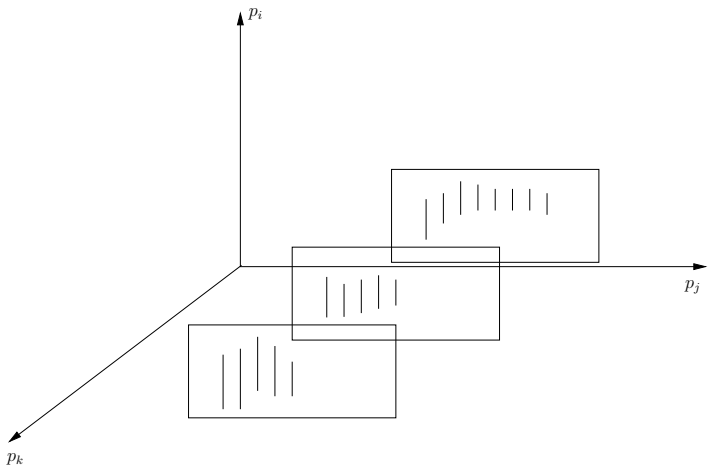
Ustalamy zbiór punktów z zakresu zmienności parametru p_j. Dla ustalonej pulsacji i ustalonej wartości parametru p_j wykonujemy analizę ODOS z punktu nominalnego p_0 w kierunku p_i. Następnie przesuwamy się wzdłuż osi p_j do następnego punktu i ponownie wykonujemy analizę ODOS w kierunku p_i. W ten sposób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

$$\Delta\hat{p_j} = \sqrt{[(p_{j_{max}} - p_{j_{min}})\Delta p_j} \tag{10.45}$$

Wyznaczanie aproksymacji odcinkowo-liniowej przestrzeni sprawności

Wprowadzając trzeci parametr p_k i wykonując algorytm 36 w kolejnych punktach p_k można wyznaczyć aproksymacje odcinkowo-liniową trójwymiarowego obszary sprawności 3D, tak jak zostało to przedstawione na rys. 10.5. Wyznaczanie obszarów sprawności w 3D zostało zaimplementowane z programie *Dero*.



Rys. 10.5: Trójwymiarowa aproksymacja odcinkowo-liniowa obszaru sprawności po wprowadzeniu parametru zmiennego p_k

Rozdział 11

Różniczkowanie symboliczne

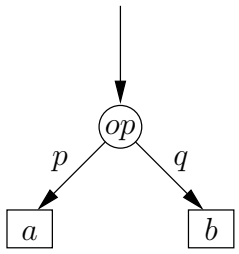
Algorytm różniczkowania symbolicznego wykorzystywany jest w modelach MDL do obliczania pochodnych funkcji względem zmiennych (wyjść względem sterowań oraz czasu). Są one wymagane przez algorytmy obliczeniowe symulatora.

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.

11.1 Drzewo parsowania

Drzewo parsowania przedstawione na rys. 11.1 to struktura, która umożliwia reprezentację działań elementarnych (dodawania, odejmowania ...). Drzewo parsowania posiada



Rys. 11.1: 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.

11.2 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.

11.2.1 Podstawowe przekształcenia drzewa

Na rys. 11.2 i 11.3 przedstawiono podstawowe przekształcenia drzewa, wynikające z podstawowych reguł obliczania pochodnych dla podstawowych działań matematycznych. "Prim" w nazwie zmiennej oznacza pochodną. Na przykład pochodne wyrażeń:

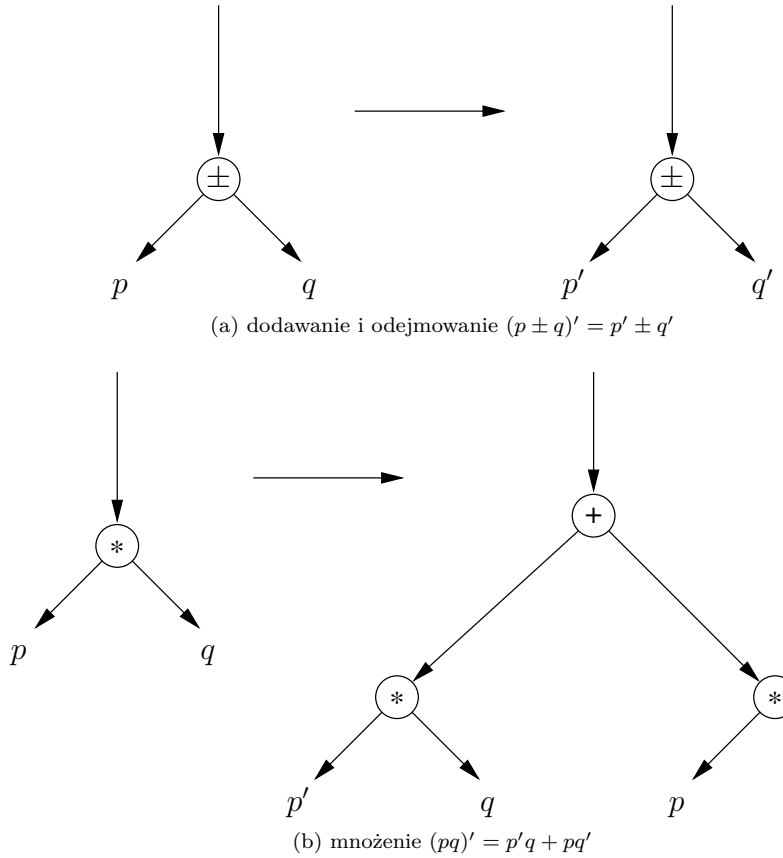
$$(pq)' = p'q + pq'$$

$$(p \pm q)' = p' \pm q'$$

Opisane powyżej przekształcenia realizuje algorytm 37. 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 37 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 $(pq)' = p'q + pq'$

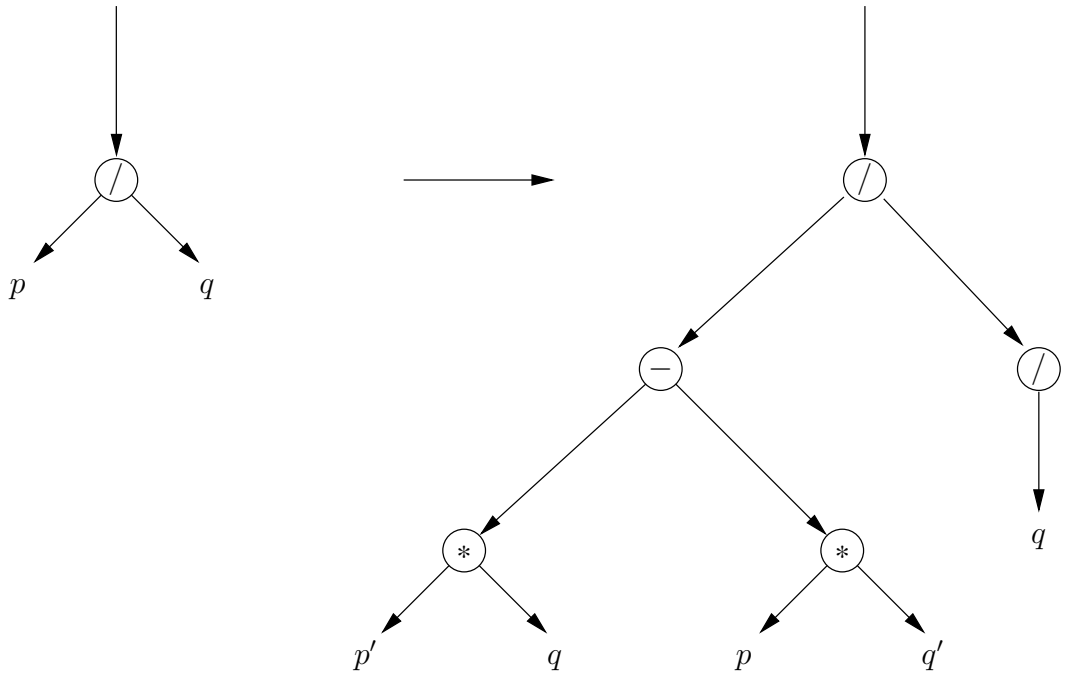
Rys. 11.2: Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych

11.2.2 Redukcja drzewa

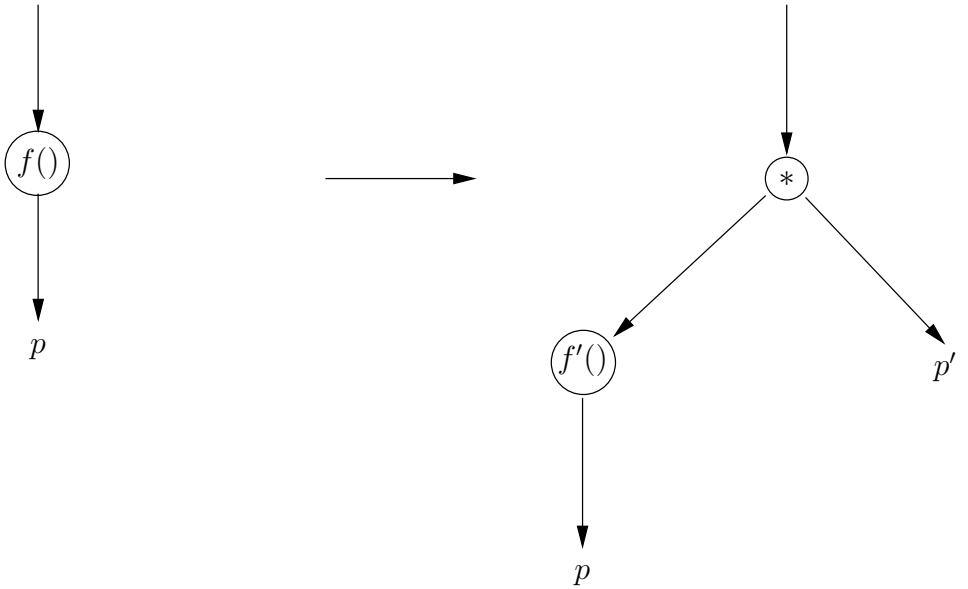
Przekształcone drzewo parsowania opisuje zróżniczkowane wyrażenie względem wszystkich zmiennych (liści drzewa). W celu wygenerowania pochodnej względem jednej zmiennej należy zredukować drzewo parsowania. Redukcja drzewa polega na usunięciu tych gałęzi drzewa, które na skutek wykonanych działań zerują się. Weźmy pod uwagę pochodną $(p + q)' = p'q + pq'$ przedstawioną w postaci drzewa na rys. 11.4. Obliczając pochodną względem zmiennej p, zmienna q traktowana jest jako stała i jej pochodna $q' = 0$. Czynnik pq' zeruje się. Działanie pq' opisane 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 38.

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 w wierz-



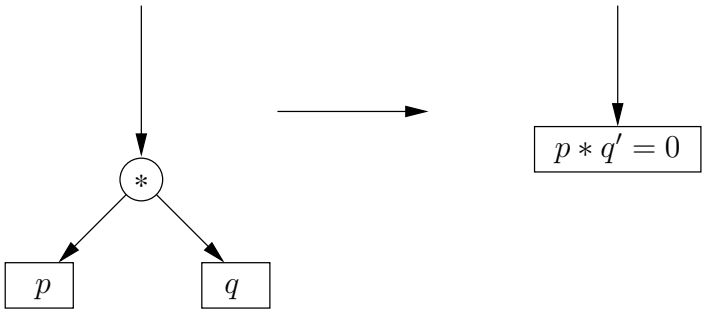
(a) dzielenie $(p/q)' = \frac{p'q - pq'}{q^2}$



(b) funkcja $f(p)' = f(p)' * p'$

Rys. 11.3: Przekształcenia węzłów drzewa dla podstawowych operacji arytmetycznych

chołkach. Na podstawie danych umieszczonych w drzewie można wygenerować postać symboliczną pochodnej.

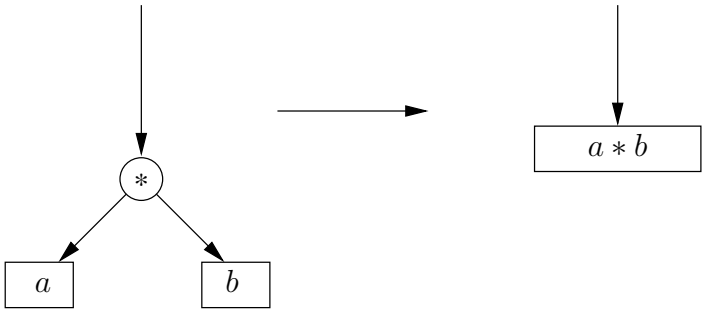


(a) operator mnożenia

Rys. 11.4: Redukcja drzewa

11.2.3 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. 11.5.



(a) operator mnożenie

Rys. 11.5: Generacja postaci symbolicznej dla operacji mnożenia

Zamianę można zapisać w postaci algorytmu 39. 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 (11.1). Drzewo parsowania dla wyrażenia przedstawiono na rys. 11.6.

$$a(b + dc) \tag{11.1}$$

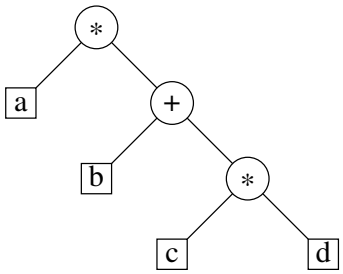
Przekształcone drzewo parsowania przedstawiono na rys. 11.7. Drzewo zostało zredukowane względem zmiennej a i zawiera pochodną wyrażenia po tej zmiennej.

Algorytm 38 Redukcja gałęzi drzewa

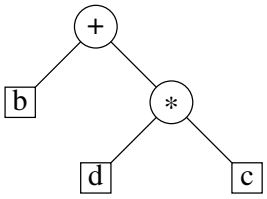
```
1: i - indeks warstwy, j - indeks w warstwie
2: n_i - liczba wierzchołków w warstwie
3: d - liczba warstw
4: for i = d ... 1 do
5:     for j = 1 ... n_i do
6:         if w_j ==' *' then
7:             if w_j → p == 0 w_j → q == 0 then
8:                 eliminuj węzeł w_j
9:             end if
10:        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 39 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. 11.6: Przykład drzewa parsowania dla wyrażenia (11.1).



Rys. 11.7: Drzewo pochodnej wyrażenia (11.1) obliczonej względem a.

Rozdział 12

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,

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

- wykorzystanie języków opisu sprzętu.

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 tab. 12.1 zamieszczono zmienne uogólnione dla wybranych środowisk.

Tab. 12.1: 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)	$\int_q v dq$, $\int_\psi v d\psi$	$\int_V P dV$, $\int_{P_p} \phi dP_p$	$\int_x F dx$, $\int_p V dx$	$\int_\phi \tau d\phi$, $\int_{P_t} \omega dP_t$
P moc (power)	$v(t)i(t)$	$P(t)\phi(t)$	$F(t)V(t)$	$\tau(t)\omega(t)$

Podejście to umożliwia tworzenie obwodów równoważnych 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 tab. 12.2.

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

Tab. 12.2: 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

dokładności z jakimi obliczane są przepływy i wymuszenia indywidualnie dla każdego środowiska.

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 (tab. 12.3): typu siła-prąd FI (*ang. force-current*) i siła-napięcie FV (*ang. force-voltage*).

Tab. 12.3: 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.

12.1 Równania sieci w dziedzinie elektrycznej

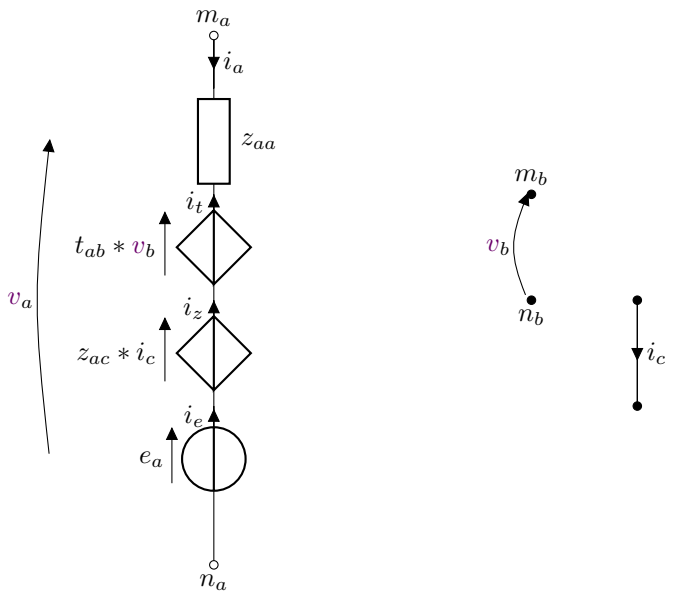
W dziedzinie elektrycznej x przyjmuje postać (12.1)

$$x = [v, i, q, \psi]^T \tag{12.1}$$

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.

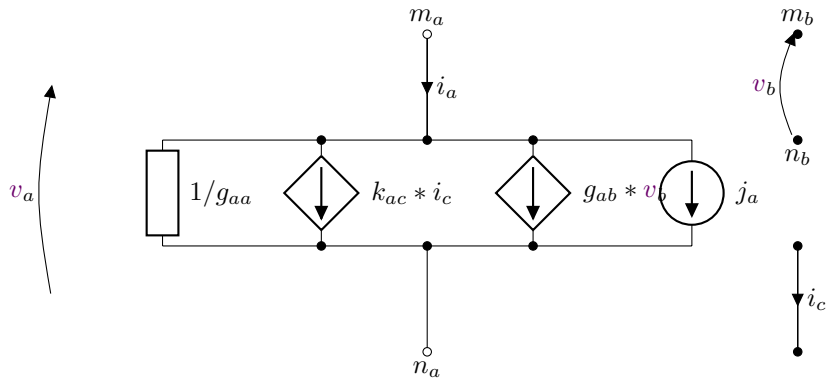


Rys. 12.1: Gałąź typu napięciowego

12.1.1 Gałąź typu prądowego

Gałąź typu prądowego przedstawiono na rys. 12.2. Gałąź opisana jest równaniem

$$i_a = G_{aa} v_a + G_{ab} v_b + K_{ac} i_c + J_a$$



Rys. 12.2: Gałąź typu prądowego

Rys. 12.3: 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. 12.4: 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	$-J_a$

12.1.2 Gałąź typu napięciowego

Gałąź typu napięciowego z prądem jako niewiadomą pokazana na rys. 12.1.

$$v_a = Z_{aa} i_a + T_{ab} v_b + Z_{ac} i_c + E_a$$

Rys. 12.5: 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}	$-E_a$

12.1.3 Gałąź opisana równaniem ładunkowym

Gałąź opisana jest równaniem ładunkowym

$$\begin{aligned} i_a &= \frac{dq}{dt} \\ q_a &= C_{aa}v_a + C_{ab}v_b + q_{a0} \end{aligned} \tag{12.2}$$

gdzie: C_{xx} są to pojemności dynamiczne.

Rys. 12.6: 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			$+1$			$-\gamma$	$-\gamma p_{qa}$

12.1.4 Gałąź opisana równaniem strumieniowym

$$\begin{aligned} v_a &= \frac{d\psi}{dt} \\ \psi_a &= L_{aa}i_a + L_{ac}i_c + \psi_{a0} \end{aligned} \tag{12.3}$$

gdzie: L_{xx} są to indukcyjności.

Rys. 12.7: 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				$+1$		$-\gamma$	$-\gamma p_{\psi a}$

12.1.5 Szablony ZMPW dla elementów liniowych

W rozdziale przedstawiono predefiniowane szablony elementów wykorzystywane do układania równań w klasycznych analizach AC, OP, DC, TRAN 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.

$$R\ i_a = v_a$$

$$\begin{bmatrix} & & 1 \\ & & -1 \\ -1 & 1 & R \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.4}$$

Konduktancja liniowa

Szablon dla wszystkich rodzajów analiz. Wyjście konduktancyjne stosowane jest do obsługi wyjścia R, o ile wartość rezystancji $R > 0$.

$$G\ (v_{ma} - v_{na}) = i$$

$$\begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{12.5}$$

Pojemność liniowa

Pojemność liniowa opisana jest równaniem prądowym postaci:

$$i = C\ \frac{dv}{dt} = C\ \dot{v} \tag{12.6}$$

Postać równania zależy od rodzaju analizy.

Analiza czasowa (TRAN) W analizie czasowej stanu nieustalonego pochodna $\dot{x}$ obliczana jest za pomocą schematu różnicowego postaci (6.2).

$$i_a = C(\gamma v_a + d_{v_a})$$

Równanie jest następnie linearyzowane (algorytm Newtona-Rapsona - rozdział 5.2). Po dyskretyzacji i linearyzacji równanie oraz po pogrupowaniu czynników otrzymujemy równanie gałęzi prądowej ZMPW, które może być układane przy pomocy szablonu opisanego w rozdziale 12.1.

$$\gamma C v_a = -C d_{v_a} \tag{12.7}$$

Zapisując równanie macierzowo otrzymamy szablon do automatycznego układania równania.

$$\begin{bmatrix} \gamma C & -\gamma C \\ -\gamma C & \gamma C \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} -C d_{v_{ma}} \\ C d_{v_{na}} \end{bmatrix} \tag{12.8}$$

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 pojemności liniowe traktowana jest jak sieć nieliniowa. Stosując SR Eulera równania gałęzi (12.6) 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 (12.7). 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 (12.9) podobną do (12.8), dla $\gamma = j\omega$ i $d_{v_a} = 0$.

$$\begin{bmatrix} j\ \omega C & -j\ \omega C \\ -j\ \omega C & j\ \omega C \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{12.9}$$

Indukcyjność liniowa

$$v_a = L\ \frac{di_a}{dt}$$

Pochodną $\frac{di}{dt}$ obliczamy ze schematu różnicowego (dyskretyzacja).

$$v_a = L\ (\gamma i_a + d_i)$$

[2]W literaturze spotyka się różne określenia, np.: węzły pływające, węzły odosobnione, 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.

Równanie należy zlinearyzować (algorytm Newtona-Raphsona - rozdział 5.2), a następnie rozwiązać. Po pogrupowaniu otrzymujemy równanie gałęzi napięciowej ZMPW (roz. 12.1).

$$-v_{ma} + v_{na} + \gamma L i_a = -L d_{i_a}$$

Zapisując równanie macierzowo otrzymamy szablon do automatycznego układania równania.

$$\begin{bmatrix} & & 1 \\ & & -1 \\ -1 & 1 & \gamma L \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -L d_{i_a} \end{bmatrix} \tag{12.10}$$

Liniowe źródło prądowe

$$i = aux\ j_a$$

Analiza OP,DC,TRAN Wprowadzono mnożnik wydajności źródła $aux \in < 0, 1 >$ wykorzystywany w analizie kontynuacji. W klasycznej analizie OP, DC, TRAN mnożnik $aux = 1$. Szablon równania prądowego ZMPW przedstawiono poniżej.

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \end{bmatrix} = \begin{bmatrix} -aux\ j_a \\ aux\ j_a \end{bmatrix} \tag{12.11}$$

W programie zastosowano także szablon równania prądowego ZMPW (12.12), z prądem gałęzi jako niewiadomą (Jx).

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ i_a \end{bmatrix} = \begin{bmatrix} \\ \\ -aux\ j_a \end{bmatrix} \tag{12.12}$$

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.

$$i = \sum_b G_{ab}\ (v_{mb} - v_{nb}) + \sum_c K_{ac} i_c$$

$$\begin{bmatrix} 0 & 0 & G_{ab} & -G_{ab} & -K_{ac} \\ 0 & 0 & -G_{ab} & G_{ab} & K_{ac} \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.13}$$

Liniowe napięciowe źródło sterowane

Szablon dla wszystkich rodzajów analiz.

$$v = \sum_b T_{ab}\ (v_{mb} - v_{nb}) + \sum_c Z_{ac} i_c$$

$$\begin{bmatrix} & & & & & 1 \\ & & & & & -1 \\ & & & & & \\ & & & & & \\ -1 & 1 & T_{ab} & -T_{ab} & Z_{ac} & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.14}$$

Idealny wzmacniacz operacyjny (liniowy)

Szablon dla wszystkich rodzajów analiz.

$$\begin{bmatrix} & & & & 1 \\ & & & & -1 \\ & & & & \\ & & & & \\ -1 & 1 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_a \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.15}$$

12.1.6 Szablony ZMPW dla elementów nieliniowych

Nieliniowe prądowe źródło sterowane

Wartości sterowań źródeł sterowanych x podlegają ograniczaniu, co gwarantuje lepszą zbieżność zastosowanego tłumionego algorytmu NR (rozdział 5.2.2).

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

$$\underbrace{\left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n}\right)}_{\delta f_x/\delta x} {x_n}^{(p)} = -\underbrace{f(x^{(p-1)})}_{f_x} + \left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n} + \gamma \frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n}\right) x_n^{(p-1)}$$

$$\begin{bmatrix} 0 & 0 & \frac{\delta f_x}{\delta v_b} & -\frac{\delta f_x}{\delta v_b} & -\frac{\delta f_i}{\delta i} \\ 0 & 0 & -\frac{\delta f_x}{\delta v_b} & \frac{\delta f_x}{\delta v_b} & \frac{\delta f_i}{\delta i} \\ & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \end{bmatrix} = \begin{bmatrix} -f_x + \frac{\delta f_x}{\delta v_b}\ v_b + \frac{\delta f_x}{\delta i_c}\ i_c) \\ +f_x - \frac{\delta f_x}{\delta v_b}\ v_b - \frac{\delta f_x}{\delta i_c}\ i_c) \\ \\ \end{bmatrix} \tag{12.16}$$

Nieliniowe napięciowe źródło sterowane

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

$$-v_{ma}+v_{na}+\underbrace{\left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n}+\gamma\frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n}\right)}_{\delta f_x/dx} x_n{}^{(p)} = -\underbrace{f(x^{(p-1)})}_{f_x}+\left(\frac{\delta f(x_n^{(p-1)})}{\delta x_n}+\gamma\frac{\delta f(x_n^{(p-1)})}{\delta \dot{x}_n}\right) x_n^{(p-1)}$$

$$\begin{bmatrix} & & & & & 1 \\ & & & & & -1 \\ & & & & & \\ & & & & & \\ -1 & 1 & \frac{\delta f_x}{\delta v_b} & -\frac{\delta f_x}{\delta v_b} & \frac{\delta f_x}{\delta i_c} & \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ i_a \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \\ -f_x + \frac{\delta f_x}{\delta v_b}\, v_b + \frac{\delta f_x}{\delta i_c}\, i_c \end{bmatrix} \tag{12.17}$$

Wyjście ładunkowe (nieliniowe)

Gałąź opisana równaniem ładunkowym jest gałęzią nieliniową o równaniu postaci (12.18)

$$i = \frac{dq}{dt} \tag{12.18}$$

gdzie q jest funkcją nieliniową zależna od wartości sterowań $q(v)$. Pochodną $\frac{dq}{dt}$ obliczymy ze schematu różnicowego (6.2). Po podstawieniu otrzymamy pierwsze równanie.

$$i = \gamma q(v) + d_q \tag{12.19}$$

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

$$i = \gamma q + d_q \tag{12.20}$$

$$q = q(v^{(p-1)}) + \frac{\delta q(v)}{\delta v}\,(v^{(p)} - v^{(p-1)}) \tag{12.21}$$

Po pogrupowaniu czynników stronami otrzymujemy układ równań

$$\gamma q = -d_q \tag{12.22}$$

$$\frac{\delta q(v_b^{(p-1)})}{\delta v}\, v_b^{(p)} - q = -q(v_b^{(p-1)}) + \frac{\delta q(v_b^{(p-1)})}{\delta v}\, v_b^{(p-1)} \tag{12.23}$$

gdzie $\frac{\delta q(v^{(p-1)})}{\delta v}$ jest pojemnością dynamiczną C.

$$\begin{bmatrix} & & & & & \gamma \\ & & & & & -\gamma \\ & & & & & \cdot \\ & & & & & \cdot \\ & & & & & \cdot \\ \cdot & \cdot & \frac{\delta q(v_b^{(p-1)})}{\delta v} & -\frac{\delta q(v_b^{(p-1)})}{\delta v} & \cdot & -1 \end{bmatrix} \begin{bmatrix} v_{ma} \\ v_{na} \\ v_{mb} \\ v_{nb} \\ i_c \\ q \end{bmatrix} = \begin{bmatrix} -d_q \\ d_q \\ \cdot \\ \cdot \\ \cdot \\ -q(v^{(p-1)}) + \frac{\delta q(v^{(p-1)})}{\delta v} \cdot v_b \end{bmatrix} \tag{12.24}$$

Gałąź opisana równaniem strumieniowym

Gałąź opisana równaniem strumieniowym jest gałęzią nieliniową o równaniu Postaci (12.25)

$$v = \frac{d\psi}{dt} \tag{12.25}$$

gdzie ψ jest funkcją nieliniową zależna od wartości sterowań $\psi(i)$. Pochodną $\frac{d\psi}{dt}$ obliczymy ze schematu różnicowego (6.2). Po podstawieniu otrzymamy pierwsze równanie.

$$v = \gamma\psi(i) + d_\psi \tag{12.26}$$

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

$$v = \gamma\psi + d_\psi \tag{12.27}$$

$$\psi = \psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ (i^{(p)} - i^{(p-1)}) \tag{12.28}$$

Grupując stronami otrzymamy

$$-v_{ma} + v_{na} + \gamma\psi = -d_\psi \tag{12.29}$$

$$\frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p)} - \psi = -\psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p-1)} \tag{12.30}$$

gdzie $\frac{\delta\psi(i^{(p-1)})}{\delta i}$ jest indukcyjnością dynamiczną L.

$$\begin{bmatrix} & & 1 & & \cdot \\ & & -1 & & \cdot \\ -1 & 1 & & & \gamma \\ & & & & \cdot \\ \cdot & \cdot & \cdot & \frac{\delta\psi(i^{(p-1)})}{\delta i} & -1 \end{bmatrix} \begin{bmatrix} v_{ma}^{(p)} \\ v_{na}^{(p)} \\ i_a^{(p)} \\ i^{(p)} \\ \psi \end{bmatrix} = \begin{bmatrix} \cdot \\ \cdot \\ -d_\psi \\ \cdot \\ -\psi(i^{(p-1)}) + \frac{\delta\psi(i^{(p-1)})}{\delta i}\ i^{(p-1)} \end{bmatrix} \tag{12.31}$$

12.2 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. 12.1. W tab. 12.4 przedstawiono szczegółowe analogie pomiędzy zmiennymi występującymi w środowisku elektrycznym i ich transformacje na zmienne w środowisku mechanicznym 1 i mechanicznym 2.

Tab. 12.4: 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 $[kg]$
Transformator $[N1 : N2]$	Dźwignia L1:L2	Dźwignia L1:L2
	K Sprężystość	K Sprężystość

W tab. 12.5 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. 12.5: 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	

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.

Dodatek A

Metoda Powell'a

Opis metody Powell'a można znaleźć np. w http://en.wikipedia.org/wiki/Powell's_method. Załóżmy, że wektor X_0 jest wektorem, dla którego funkcja $z = f(X) = f(x_1, x_2, \ldots, x_n)$ przyjmuje minimum. Załóżmy, że pochodne cząstkowe funkcji $f(X)$ nie są dostępne. Intuicyjnie w celu znalezienia minimum funkcji $f(X)$ można wyznaczyć nową aproksymację X_1 przesuwając się w kierunku minimum funkcji $f(X)$ wzdłuż każdego z n wektorów bazowych $E_k = (0, \ldots, 0, 1_k, 0, \ldots, 0)$. Przesuwanie powoduje, że funkcja f staje się funkcją jednej zmiennej. Proces generuje sekwencję punktów

$$X_0 = P_0, P_1, P_2, \ldots, P_n = X_1$$

Proces ten może zostać powtórzony iteracyjnie, co spowoduje wygenerowanie zbioru wektorów punktów $\{X_k\}_{k=0}^{\infty}$ Metoda ta zwana *ang. taxi-cab method* jest w ogólności mało efektywna w przypadku funkcji wielu zmiennych. Sekwencja $P_0, P_1, P_2, \ldots, P_n$ jest używana w pierwszym kroku metody Powell'a.

Metoda Taxi Cab Ideą metody aproksymacji lokalnego minimum $f(X)$, gdzie f jest funkcją i $X = (x_1, x_2, \ldots, x_n)$, jest szukania minimum funkcji wzdłuż wektorów bazowych $\{E_k\}_{k=1}^{\infty}$ startując z punktu P_0.

A.1 Oryginalna metoda Powell'a

Istotą metody Powell'a jest dodanie dodatkowych kroków w procesie poszukiwania wektora poprawy. Wektor $P_n - P_0$ reprezentuje w pewnym sensie uśredniony kierunek przesunięcia poprzez kolejne kroki $P_0, P_1, P_2, \ldots, P_n$ w iteracji. w kolejnej iteracji. Tak jak poprzednio funkcja f jest funkcją jednej zmiennej przy przesuwaniu się w określonym kierunku, i jej minimum w kierunku przesuwania się można łatwo znaleźć (np. metodą złotego podziału). Jeżeli $P_n - P_0$ reprezentuje prawidłowy kierunek zmian, to w kolejnej iteracji może on wymienić jeden z wektorów kierunków poprawy. Iteracje są powtarzane używając nowy zbiór wektorów poprawy P co generuje zbiór punktów $\{X_k\}_{k=0}^{\infty}$. Modyfikuje to sposób dochodzenia do minimum funkcji f.

Załóżmy, że X_0 będzie warunkiem początkowym do minimalizacji funkcji

$$z = f(X) = f(x_1, x_2, \ldots, x_n)$$

.

Niech $E_k = (0, \ldots, 0, 1_k, 0, \ldots, 0)$ dla $k = 1, 2, 3, \ldots, n$ będzie zbiorem wektorów bazowych. Zainicjalizuj wektor $U_k = E_k$ dla $k = 1, 2, 3, \ldots, n$ i użyj wektorów transponowanych tak, aby utworzyły kolumny macierz U:

$$U = [U_1^t, U_2^t, \ldots, U_n^t]$$

Algorytm 40 Orgyginalny algorytm Powell'a

ustaw licznik i = 0
(S.1) $P_0 = X_i$
(S.2) dla $k = 1, 2, 3, \ldots, n$ znajdź wartość $\gamma = \gamma_k$, która minimalizuje $f(P_{k-1} + \gamma U_k)$ i ustaw $P_k = P_{k-1} + \gamma_k U_k$
(S.3) ustaw $U_j = U_{j+1}$ dla $j = 1, 2, 3, \ldots, n-1$ oraz $U_n = P_n - P_0$
(S.4) zwiększ licznik kierunków $i = i + 1$
(S.5) znajdź wartość $\gamma = \gamma_{min}$, która minimalizuje $f(P_0 + \gamma U_n)$ i ustaw $X_i = P_0 + \gamma_{min} U_n$
(S.6) powtarzaj kroki S.1 do S.6 aż do osiągnięcia zbieżności

A.2 Zmodyfikowana metoda Powell'a

W kroku S.3 oryginalnego algorytmu 40 pierwszy wektor U_1 był odrzucany i średni wektor $P_n - P_0$ był dodawany do listy wektorów kierunków. Lepiej jest usunąć wektor U_r spośród wektorów, dla których wystąpił największy spadek f. Wydaje się, że wektor U_r jest dużym komponentem wektora uśrednionego $U_n = P_n - P_0$. Jednak ze wzrostem liczby iteracji wektory mają tendencję do stawania się liniowo zależnymi. W takim zależności liniowej wektorów jeden lub więcej kierunków poprawy zostanie utracony i zbiór punktów $\{X_k\}_{k=0}^{\infty}$ nie doprowadzi do punktu, gdzie funkcja osiąga lokalne minimum. W punkcie (S.3) założona, że średni kierunek poprawy reprezentuje dobry kierunek poszukiwań. Nie musi być to jednak regułą.

Algorytm 41 Rozszerzony algorytm Powell'a

ustaw licznik i = 0
(S.1) $P_0 = X_i$
(S.2) Dla $k = 1, 2, 3, \ldots, n$ znajdź wartość $\gamma = \gamma_k$, która minimalizuje $f(P_{k-1} + \gamma U_k)$ i ustaw $P_k = P_{k-1} + \gamma_k U_k$
(S.3) Niech $\Delta f_k = f(P_k) - f(P_{k-1})$ dla $k = 1, 2, 3, \ldots, n$. Znajdź taki indeks r, że $\Delta f = |\Delta f_r| = max(|\Delta f_k|)$ jest największym spadkiem f oraz wektor U_r dla najszybszego spadku spośród wszystkich wektorów kierunków w kroku S.2
(S.4) zwiększ licznik kierunków $i = i + 1$
(S.5) Niech $f_k = f(P_k)$ dla $k = 1, 2, 3, \ldots, n$.
Niech $f_E = f(2P_n - P_0)$ będzie wartością funkcji w rozszerzonym kierunku $2(P_n - P_0)$ z P_0.
Jeżeli $f_E \geq f_0$ ustaw $X_i = P_n$ i przejdź do kroku S.1 lub
Jeżeli $2(f_0 - 2f_n + f_E)(f_0 - f_n - \Delta f)^2 \geq \Delta f(f_0 - f_E)^2$ to ustaw $X_i = P_n$ i przejdź do kroku S.1
W przeciwnym przypadku idź do S.6
(S.6) ustaw $U_n = P_n - P_0$, gdzie r zostało obliczone w kroku S.3
(S.7) znajdź wartość $\gamma = \gamma_{min}$, która minimalizuje $f(P_0 + \gamma U_r)$ i ustaw $X_i = P_0 + \gamma_{min} U_r$
(S.8) powtarzaj kroki S.1 do S.7 aż do osiągnięcia zbieżności

Spis tabel

Spis rysunków

Lista algorytmów

Oznaczenia

Θ Kat. 194

κ wskaźnik uwarunkowania macierzy. 31

ϵ błąd reprezentacji. 21, 32

ω Pulsacja $\omega = \frac{1}{2\pi f}$. 174, 178–180

ω Zmienna przyspieszenia katowowego. 194, 195

ψ Strumień. 194, 196, 198, 204

ψ zmienna strumieniowa. 105

τ Zmienna momentu obrotowego. 194, 195

a Wymuszenie uogólnione. 195

a Przepływ uogólniony. 195

B wektor prawych stron. 174, 176, 177

B Tarcie. 205, 206

C Pojemnosc. 205, 206

d Zmienna przesuniecia. 195

e Wymuszenie. 194

F składowa funckji celu zadania optymalizacji. 169–172

f funkcja układowa optymalizacji. 167, 168

f Przeplyw. 194

f Sila. 194, 205

f Zmienna sily. 195

FC funkcja celu zadania optymalizacji. 167–173

fr flow rate. 195

hfr heat flow rate. 195

i Prad. 194–206

i Zmienna pradowa. 195

i zmienna pradowa. 104, 105

J Jakobian. 169–172

K Sprezystosc. 205, 206

L Indukcyjnosc. 205, 206

L Smarnosc. 205

m indeks chwili czasowej. 105–114

m Masa. 195, 205, 206

P Moc. 194

P Cisnienie. 194

p Ped. 194

p parametr rzeczywisty. 173–178, 180–183

p Polozenie. 194

p Zmienna cisnienia. 195

q Ladunek. 194, 196, 198, 203, 204

q zmienna ladunkowa. 105, 110

R Rezystancja. 205

S bit znaku - liczba jest ujemna, gdy bit znaku jest równy 1, w przeciwnej sytuacji ma on wartość 0. 26

s specyfikacje projektowe optymalizacji. 167, 168

t Temperatura. 195

V Przyspieszenie. 194, 195, 205

V Objetosc. 194

v Napiecie. 194, 196–206

v zmienna napieciowa. 106–113, 120, 132

v Zmienna napieciowa. 104, 105, 194, 195

w waga danej specyfikacji projektowej optymalizacji. 168

WU wskaźnik uwarunkowania. 27

X parametr zespolony zależny od p. 174–177

x parametry projektowe optymalizacji. 167–170, 172

x Zmienna, Przesuniecie. 194, 196

x wektor zmiennych. 79–81, 105

Y macierz admitancyjna układu. 174, 176, 177

y odpowiedź funkcji układowych y=f(x) optymalizacji. 167, 168, 172

z wewnętrzne zmienne optymalizacji. 168–172

z zmienna uogólnioną. 80, 81, 110

Słownik

fanin w analizie relaksacyjnej oznacza węzeł oddziałujący na węzeł zależny (*ang. fanout*).. 113

fanout w analizie relaksacyjnej oznacza węzeł, w którym zmiana sygnału spowodowana jest zmianą sygnału w węźle oddziałującym (*ang. fanin*).. 113

LBO Błąd *ang. local truncation error* będący różnicą między wartością dokładną, a przewidywaną w danym kroku czasowym.. 89

predykcja przewidywanie wartości startowej za dla schematu różnicowego, ang. prediction. 88, 144

Indeks

Literatura

[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+02] A.Ushida, Y.Yamagami u. a.: An efficient algorithm for finding multiple dc operating points based on SPICE oriented Newton homotopy method. In: *IEEE Trans. Computer-Aided Design* 21 (2002), Mar, Nr. 3, S. 337–348

[ABG08] A. Barari, A.R. G. M. Omidvar O. M. Omidvar ; Ganji, D.D.: Application of homotopy perturbation method and variational iteration method to nonlinear oscillator differential equations. In: *Acta Applicandae Mathematicae* 104 (2008), Nr. 2, S. 161–171

[AD+99] A. Dyes, E. C. u. a.: Simple implementations of homotopy algorithms for finding dc solutions of nonlinear circuits. In: *Proc. Int. Symp. Circuits Systems* (1999), May, S. 290–293

[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

[AESD12] A.M.A. El-Sayed, I.L.El-Kalla A.Elsaid ; D.Hammad: A homotopy perturbation technique for solving partial differential equations of fractional order in finite domains. In: *Applied Mathematics and Computation* 218 (2012), Nr. 17, S. 8329–8340

[Agu90] Aguilar, J.Cardenas: *Circuit analysis by hierarchical decomposition*. Warszawa : IPE PW, 1990. – Analiza układow elektronicznych metoda dekompozycji hierachicznej

[Ami12] Aminikhah, H.: The combined Laplace transform and new homotopy perturbation methods for stiff systems of ODEs. In: *Applied Mathematical Modelling* 36 (2012), Nr. 8, S. 3638–3644

[Ana] Analogy Corporation (Hrsg.): *SABER User's Manual*. USA: Analogy Corporation

[Ana97] Anacad Corporation (Hrsg.): *ELDO-XL User's Manual*. USA: Anacad Corporation, Mar 1997

[A.R79] A.R.Newton: Techniques for the simulation of large-scale integrated circuits. In: *IEEE Transactions On Circuits And Systems* CAS-26 (1979), Sep, S. 741–749

[A.R84] A.R.Newton, A.Sangiovanni-Vincentelli: Relaxation-based circuit simulation. In: *IEEE Transactions Computer-Aided Design* CAD-3 (1984), Oct, S. 308–330

[A.R85] A.R.Newton, J.White R.Saleh A.Sangiovanni-Vincentelli: Accelerating relaxation algorithms using waveform Newton, step refinement and parallel techniques. In: *Proc. Int. Conf. Computer-Aided Design* (1985), Sep, S. 84–87

[AR+03] A. Reibiger, W. M. u. a.: Mathematical foundations of the TC-method for computing multiple DC-operating points. In: *International Journal of Applied Electromagnetics and Mechanics* 17 (2003), S. 169–191

[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

[BA09] Biazar, J. ; Aminikhah, H.: Study of convergence of homotopy perturbation method for systems of partial differential equations. In: *Computers & Mathematics with Applications* 58 (2009), Nr. 11-12, S. 2221–2230

[BE11] Biazar, J. ; Eslami, M.: A new homotopy perturbation method for solving systems of partial differential equations. In: *Computers & Mathematics with Applications* 62 (2011), Nr. 1, S. 225–234

[BG08] Biazar, J. ; Ghazvini, H.: Homotopy perturbation method for solving hyperbolic partial differential equations. In: *Computers & Mathematics with Applications* 56 (2008), Nr. 2, S. 453–458

[BG09] Biazar, J. ; Ghazvini, H.: Convergence of the homotopy perturbation method for partial differential equations. In: *Nonlinear Analysis. Real World Applications* 10 (2009), Nr. 5, S. 2633–2640

[B.H85a] B.Henion, D.Coquelle P.Senn: A new algorithm for third generation circuit simulator. In: *IEEE Autom. Conf.* (1985)

[B.H85b] B.Henion, P.Senn: ELDO: A new third generation circuit simulator using one-step relaxation method. In: *Proceedings of ISCAS 85* (1985), S. 1065–1068

[B.K89] B.Klaassen, P.G.Ploeger K.L.Paap: Improved Waveform-Relaxation-Newton Method. In: *ISCAS'89 PROCEEDINGS* (1989), S. 856–859

[Bok81] Bokhoven, W.M.G. v.: *Piecewise-linear Modeling and Analysiss.* Deventer : Kluwer Academic Publishers, 1981

[B.R75] B.R.Chawla, H.K.Gummel: MOTIS - A MOS Timing Simulator. In: *IEEE Trans. on Circuits and Systems* CAS-22 (1975), Dec, S. 901–910

[C.A86] C.A.Zukowski: *The bounding Approach to VLSI Circuit Simulation.* Boston : Kluwer Academic Publishers, 1986

[Cal72] Calahan, D.A.: Computer Aided Network Design. (1972)

[CD86] Chua, L. ; Deng, A.-C.: Canonical piecewise-linear analysis: Generalized breakpoint hopping algorithm. In: *Int. J. Circuit Theory Appl.* 14 (1986), Nr. 1, S. 35–52

[CHB75] C.W. Ho, A.E. R. ; Brennan, P.A.: The modified nodal ap- proach to network analysis. In: *IEEE Trans. Circuits Syst.* CAS-22 (1975), Jun, Nr. 6, S. 504–509

[CL82] Chua L.O., Ying R.: Finding all Solutions of Piecewise Linear Circuits. 10 (1982), S. 201–229

[C.L86] C.L.Lee, S.J.Jou C.W.Jen W.: MOTA: a MOSFET timing simulator. In: *IEE Proceedings* 133 (1986), Oct, Nr. 5, S. 193–199

[Col66] Collatz, L.: *The numerical treatment of differential equations.* Springer-Velag, 1966

[CP91] Cox P.F., Burch R.G. et a.: Direct circuit simulation algorithms for parallel processing. In: *IEEE Trans. on Computer-Aided Design* CAD-10 (1991), June, Nr. 6, S. 714–725

[C.T95] C.T.Kelley: *Iterative methods for linear and nonlinear equations.* Philadelfia : SIAM, 1995

[CU76] Chua, L.O. ; Ushida, A.: A switching parameter algorithm for finding multiple solutions of nonlinear resistive circuits. In: *International Journal Circuit Theory and Applications* 4 (1976), S. 215–239

[CZ02] Crutchley, D.A. ; Zwolinski, M.: Using evolutionary and hybrid algorithms for DC operating point analysis of nonlinear circuits. In: *Proc. 2002 Congress on Evolutionary Computation (CEC'02)* (2002), May 12th-17th

[D.82] D., Agnew: Techniques for Robust DC Algorithms for Circuit Simulation. In: *Proc. Int. Symp. Circ. Syst.* (1982), S. 1198–1201

[D+87] D.Dumlugol, P.Odent u. a.: Switch-electrical segmented waveform relaxation for digital MOS VLSI and its acceleration on parallel computers. In: *IEEE Trans. on Computer-Aided Design* CAD-6 (1987), Nov, S. 992–1005

[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

[D.B95] D.Bukat, J.Ogrodzki: *OPTIMA 2.0 uniwersalny analizator układów elektronicznych.* Warsaw : WNT, 1995. – ISBN 83–204–1816–X

[DBP09] D. Bahuguna, A. U. ; Pandey, D.N.: A comparative study of numerical methods for solving an integro-differential equation. In: *Computers & Mathematics with Applications* 57 (2009), Nr. 9, S. 1485–1493

[DGJ07] D.D. Ganji, H. T. ; Jooybari, M.B.: Variational iteration method and homotopy perturbation method for nonlinear evolution equations. In: *Computers & Mathematics with Applications* 54 (2007), Nr. 7-8, S. 1018–1027

[D.J89] D.J.Erdman, D.J.Rose: A Newton waveform relaxation algorithm for circuit simulation. In: *in Proc. IEEE Int. Conf. Computer Aided Design* (1989), Nov, S. 404–407

[D.J92] D.J.Erdman, D.J.Rose: Newton waveform relaxation techniques for tightly coupled systems. In: *IEEE Trans. On Computer-Aided Design* 11 (1992), May, Nr. 5, S. 598–606

[D.O77] D.O.Pederson, S.P.Fan M.Y.Hsueh A.: MOTIS-C: A new circuit simulator for MOS LSI circuit. In: *Proc. IEEE Int. Symp. on Circ. and Syst.* (1977), S. 700–703

[D.T86] D.Tsao, C.Chen: A fast-timing simulator for digital MOS circuits. In: *IEEE Trans. on Computer-Aided Design* CAD-5 (1986), Oct, S. 537–540

[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

[Eij83] Eijndhoven, J.T.J. van: The modelling of MOS-transistors for a PL circuit simulator. In: *Proc. ISCAS* (1983)

[E.L82] E.Lelarasmee, Alberto L.Sangiovanni-Vincentelli Albert E. Albert E.Ruehli: The waveform relaxation method for time-domain analysis of large scale integrated circuits. In: *IEEE Trans. Computed-Aided Design* 1 (1982), Jul, S. 131–145. – Opis symulatora RELAX

[EO11] Erdogan, U. ; Ozis, T.: A smart nonstandard finite difference scheme for second order nonlinear boundary value problems. In: *Journal of Computational Physics* 230 (2011), Nr. 17, S. 6464–6474

[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

[E.Z92] E.Z.Xia, R.A.Saleh: Parallel Waveform-Newton algorithms for circuit simulation. In: *IEEE Transactions On Computer-Aided Design* 11 (1992), Apr, Nr. 4, S. 432–442

[F+08] Fernandez, V. u. a.: Finding all the operating points in piecewise-linear electrical networks: an iterative decomposed approach. In: *15th IEEE International Conference on Electronics Circuits and Systems* (2008), S. 304–307

[FH.72] FH., Branin: Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. In: *IBM Journal on Research and Dev.* 16 (1972), S. 504–22

[Fog00] Fogel, Evolutionary Computation: Towards a New Philosophy of Machine I. D.B.: In: *2nd Ed.* (2000), IEEE Press, NY. S.

[FZ09] Fortuna Z., Wąsowski J. Macukow B. B. Macukow B.: *Metody numeryczne*. Warszawa : WNT, 2009

[Gaw93] Gawryś, Michał: *UNIX: Narzędzia programistyczne yacc i lex*. Warszawa : Wydawnictwo PLJ, 1993

[GD74] G. Dahlquist, A. B.: *Numerical methods*. Prientice-Hall, 1974

[G.D98] G.D.Gristede, C.A.Zukowski A.E.Ruehli: Convergence properties of waveform relaxation circuit simulation methods. In: *IEEE Transactions On Circuits And Systems* 45 (1998), Jul, S. 726–738

[GDM82] G. De Micheli, A.Sangiovanni-Vincentelli: Characterization of Integration Algorithms for Timing analysis of MOS VLSI circuits. In: *Int. J. Circ. Theory Appl.* 10 (1982), S. 299–309

[GEF67] George E. Forsythe, Cleve B. M.: *Computer Solution of Linear Algebraic Systems*. Prientice-Hall, 1967

[GG05] Goldgeisser, L.B. ; Green, M.M.: A method for automatically finding multiple operating points in nonlinear circuits. In: *IEEE Transactions on Circuits and Systems* 52 (2005), Apr, S. 776–784

[GGP08] G. Gajani, A. B. ; Premoli, A.: Numerical determination of possible multiple DC solutions of nonlinear circuits. In: *IEEE Transactions on Circuits and Systems* 55 (2008), S. 1074–1083

[GM83] G.De Micheli, A.Sangiovanni-Vincentelli A.R.Newton: Symmetric Displacement Algorithms for the Timing Analysis of MOS VLSI. In: *IEEE Trans. Circuit Systems* 7 (1983), S. 167–179

[G.M85] G.Marong, A.Sangiowanni-Vincentelli: Waveform relaxation and dynamic partitioning for transient simulation of large scale bipolar circuits. In: *Proc. Int. Conf. Circuit Computer-Aided Design* (1985), Nov, S. 32–34

[GM93] Gray, P.R. ; Meyer, R.G.: Analysis and Design of Analog Integrated Circuits. In: *New York: Wiley* (1993)

[Gol89] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. (1989)

[Gol99] Goldgeisser, L.: Enhanced dc operating point simulation using probability-one continuation methods. In: *Ph.D. dissertation* (1999)

[Gre95] Green, M.M.: An efficient continuation method for use in globally convergent dc circuit simulation. In: *Proceedings of URSI International Symposium on Signals, Systems and Electronics (ISSSE '95)* (1995), october, S. 497–500

[Gre98] Green, M.M.: The augmentation principle of nonlinear circuits and its application to continuation methods. In: *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* 45 (1998), Sep, Nr. 9, S. 1002–1006

[H+80] H.DeMan u. a.: DIANA: Mixed Mode Simulator with a Hardware Description Language for Hierarchical Design of VLSI. In: *IEEE ICC'80 Conf. Proc.* (1980), Oct, S. 356–360

[H+88] H.Yoshida, S.Kumagai u. a.: A parallel implementation of large-scale circuit simulation. In: *ISCAS'88 PROCEEDINGS* (1988), S. 321–324

[HA89] Hideki Asai, Takeshi K.: Under relaxation techniques for stable convergence in nonlinear circuit simulation. In: *ISCAS Proceedings* (1989), S. 2023–2026

[Hag81] Hageman, Young: *Applied iterative methods.* New York, 1981

[He99] He, J.-H.: Homotopy perturbation technique. In: *Computer Methods in Applied Mechanics and Engineering* 178 (1999), Nr. 3-4, S. 257–262

[He00] He, J.-H.: A coupling method of a homotopy technique and a perturbation technique for non-linear problems. In: *International Journal of Non-Linear Mechanics* 35 (2000), Nr. 1, S. 37–43

[He03] He, J.-H.: Homotopy perturbation method: a new nonlinear analytical technique. In: *Applied Mathematics and Computation* 135 (2003), Nr. 1, S. 73–79

[He04] He, J.H.: Comparison of homotopy perturbation method and homotopy analysis method. In: *Applied Mathematics and Computation* 156 (2004), Nr. 2, S. 527–539

[He09] He, J.-H.: An elementary introduction to the homotopy perturbation method. In: *Computers & Mathematics with Applications* 57 (2009), Nr. 3, S. 410–412

[H.J88] H.Jun, S.B.Park K.Jun: An Accurate and Efficient Multiple Delay Simulator for MOS Logic Circuits Using Polynomial Approximation. In: *Proc. of the IEEE Int. Symp. of Circuits and Systems* (1988), S. 2117–2120

[HL88] Huang, Q. ; Liu, R.: A simple method for all solutions of a piecewise linear network. In: *Proc. Int. Symp. Circuits Systems* (1988), Jun, S. 1233–1236

[Hou65] Householder, A.: *The theory of matrices in numerical analysis.* New York : Blaisded Publishing Company, 1965

[HQ89] Huang Q., Liu R.: A simple Algorithm for Finding all Solutions of Piecewise Linear Networks. In: *IEEE Transactions Circuits And Systems* 36 (1989), S. 600–609

[Hun80] Hundsdorfer: *The numerical solution of nonlinear systems, stiff initial value problems: an analysis of one step methods.* Amsterdam, 1980

[HVl+11] H. Vazquez-leal, L. Hernandez-Martinez u. a.: Homotopy method with a formal stop criterion applied to circuit simulation. In: *IEICE Electronics Express* 8 (2011), Nr. 21, S. 1808–1815

[HVL+12a] H. Vazquez-Leal, A. Y. R. Castaneda-Sheissa u. a.: Biparameter homotopy-based direct current simulation of multistable circuits. In: *British Journal of Mathematics & Computer Science* 2 (2012), Nr. 3, S. 137–150

[HVL+12b] H. Vazquez-Leal, U. Filobello-Nino u. a.: Modified hpms inspired by homotopy continuation methods. In: *Mathematical Problems in Engineering* 2012 (2012)

[HVLCS05] H. Vazquez-Leal, A. Sarmiento-Reyes L. Hernández-Martínez ; Castaneda-Sheissa, R.: Numerical continuation scheme for tracing the double bounded homotopy for analysing nonlinear circuits. In: *Proceedings of the International Conference on Communications, Circuits and Systems* (2005), may, S. 1122–1126

[HVLMA03] H. Vazquez-Leal, A. Sarmiento-Reyes L. Hernández-Martinez ; Murphy-Arteaga, R.S.: Improving multi-parameter homotopy via symbolic analysis techniques for circuit simulation. In: *Proceedings of the European Conference on Circuit Theory and Design* 2 (2003), S. 402–405

[HVLSR05] H. Vazquez-Leal, L. Hernaandez-Martinez ; Sarmiento-Reyes, A.: Double-bounded homotopy for analysing nonlinear resistive circuits. In: *International Symposium on Circuits and Systems* 4 (2005), S. 3203–3206

[H.Y85] H.Y.Hsieh, P.Ledak A.Ruehli: Progress on toggle: A waveform relaxation VLSI-MOSFET CAD program. In: *in Proc. Int. Symp. Circuits and Systems* (1985), S. 213–216

[I.D85] I.Dziubiński, T.Swiątkowski: *Poradnik Matematyczny*. Warszawa : PWN, 1985

[IEEa] *IEEE 754: Standard for Binary Floating-Point Arithmetic*. IEEE Standard Association, . – http://grouper.ieee.org/groups/754/

[IEEb] *IEEE Standard for Floating-Point Arithmetic*. IEEEXplore digital library, . – http://ieeexplore.ieee.org/

[ISO] *ISO/IEC/IEEE 60559:2011 Information technology – Microprocessor Systems – Floating-Point arithmetic*. ISO standards catalogue, . – http://www.iso.org/

[JCL94] Jai-Cheol Lee, Yu-Hen H.: EDLICS: A New Relaxation-Based Electrical Circuit Simulation Technique. In: *IEEE* (1994), S. 1–5

[JCT97] John C. Tannehill, Richard H. P. Dale Arden Anderson A. Dale Arden Anderson: *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, 1997. – ISBN 1–56032–046–X

[J.D99] J.Dąbrowski, Jerzy: *Piecewise Linear Approach to Functional-level Macrosimulation of Analogue and Mixed AD Systems*. Gliwice, Politechnika Śląska, Diss., 1999

[J.H04] J.H.He: The homotopy perturbation method nonlinear oscillators with discontinuities. In: *Applied Mathematics and Computation* 151 (2004), Nr. 1, S. 287–292

[J.M] J.Molenaar: Multigrid methods for semiconductor device simulation.

[J.O84] J.Ogrodzki: One dimensional ortogonal search. In: *??* (1984), S. ??

[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

[J.P96] J.Porębski, P.Korochoda: *SPICE program analizy nieliniowej układów elektronicznych.* Warszawa, 1992,1996

[JS83] J. Stoer, R. B.: *Wstęp do analizy numerycznej.* Warszawa : PWN, 1983. – ISBN 83–01–01626–4

[JS93] J. Stoer, R. B.: *Introduction to Numerical Analysis.* Springer, 1993. – ISBN 0–387–97878–X

[JS02] J. Stoer, R. B.: *Introduction to Numerical Analysis.* Springer, 2002

[J.T84] J.T.Deutsch, A.R.Newton: A Multiprocessor Implementation of Accurate Electrical Circuit Simulation. In: *Proc. 19th IEEE/ACM Design Automation Conference* (1984), Jun

[J.W83] J.White, A.Sangiovanni-Vincentelli: RELAX2: A new waveform relaxation approach for the analysis of LSI MOS circuits. In: *IEEE Int. Symp. Circuits and Systems* (1983), May, S. 756–759

[J.W85a] J.White: *The multirate integration properties of waveform relaxation with application to circuit simulation and parallel computation.* Berkeley, CA, University of California, Diss., Now 1985

[J.W85b] J.White, A.Sangiovanni-Vincentelli: Partitioning algorithms and parallel implementation of waveform relaxation algorithms for circuit simulation. In: *in Proc. IEEE Int. Symp. Circuits Systems* (1985), June

[J.W87] J.White, A.Sangiovanni-Vincentelli: *Relaxation Techniques for the Simulation of MOS VLSI Circuits.* Boston MA : Kluwer Academic Publishers, 1987

[K.A85] K.A.Sakallah, S.W.Director: *SAMSON A mixed circuit-logic-level simulator.* Greenwich,Connecticut,USA : Jai Press Inc., 1985

[KF11] Khan, Y. ; Faraz, N.: Application of modified Laplace decomposition method for solving boundary layer equation. In: *Journal of King Saud University* 23 (2011), Nr. 1, S. 115–119

[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

[Kol00] Kolev, L.: An interval method for global nonlinear analysis. In: *IEEE Transactions on Circuits and Systems* 47 (2000), May, S. 675–683

[KYI99] K. Yamamura, T. S. ; Inoue, Y.: A fixed-point homotopy method for solving modified nodal equations. In: *IEEE Transactions on Circuits and Systems* 46 (1999), Nr. 6, S. 654–665

[LC01] Lee, J. ; Chiang, H.-D.: Convergent regions of the Newton homotopy method for nonlinear systems: theory and computational applications. In: *IEEE Transactions on Circuits and Systems I* 48 (2001), Nr. 1, S. 51–66

[LD98] Lynn D.Gabbay, Stephen E.: Automatic generation of dynamic macromodels using quasistatic simulations with modal analysis. In: *Tech. Dig. IEEE Solid-State Sensor and Actuator Workshop* (June 1998), S. 197–200

[LG64] L.Kantorovich ; G.Akilov: *Functional analysis in normed spaces.* Oxford : Permagon Press, 1964. – translated by D.Brown and A.Robertson

[Lin81] Lin, Leon O.Chua Pen-Min: *Komputerowa analiza układów elektronicznych.* Warszawa : WNT, 1981

[LK90] L.V. Kolev, V.M. M.: An interval method for finding all operating points of non-linear resistive circuits. In: *International Journal of Circuit Theory and Applications* 18 (1990), S. 257–267

[LN77] L.O.Chua ; N.N.Wang: On the application of degree theory to the analysis of resistive nonlinear networks. In: *Int. J. Circuit Theory Appl.* 5 (1977), S. 35–68

[LTS98] L. Trajkovic, E. F. ; Sanders, S.: HomSPICE: Simulator with homotopy algorithms for finding dc and steady-state solutions of nonlinear circuits. In: *Proc. Int. Symp. Circuits Systems* (1998), May, S. 227–231

[L.V87] L.V., Kolev: A Quasi-Newton Algorithm for Finding all Solutions of Piecewise Linear Networks. In: *IEEE Transactions Circuits And Systems* 15 (1987), S. 600–609

[LV89] L. Vandenberghe, J. V. B.L.D. Moor M. B.L.D. Moor: The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits. In: *IEEE Transactions on Circuits and Systems* 36 (1989), Nov, S. 1382–1391

[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: ADVICE for circuit simulation. In: *Proc. IEEE Int. Symp. on Circ. and Syst.* (1980), May

[LWM87] L. Watson, S. B. ; Morgan, A.: ALGORITHM 652 HOMPACK: A suite of codes for globally convergent homotopy algorithms. In: *ACM Trans. Math. Softw.* 13 (1987), S. 281–310

[LZ97] Litovski, V. ; Zwolinski, M.: *VLSI Circuit Simulation and Optimization.* London : Chapman and Hall, 1997 http://eprints.soton.ac.uk/255734/

[Lár09] Lárusson, Finnur: *The homotopy theory of . . .* 2009

[M.B80] M.B.Carver: Efficient integration over discontinuities in ordinary differential equation simulation. In: *Mathematics and Computers in Simulation XX* (1980), S. 190–196

[M.D82] M.Dryja, J.M.Jankowscy: *Przegląd metod i algorytmów numerycznych.* Warszawa : WNT, 1982

[Miz95] Mizutani, H.: An effective method to find all solutions of piecewise linear circuits by using partitioning. In: *Electron. Comm. Jpn.* 78 (1995), S. 20–31

[MJ94] M.M.Green ; Jr., A.N.Willson: (Almost) half of all operating points are unstable. In: *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* 41 (1994), Mar, Nr. 3, S. 286–293

[M.M95] M.M.Green, R.C. M.: Sufficient conditions for finding multiple operating points of dc circuits using continuation methods. In: *Proc. IEEE Int. Symp. Circuits Systems* (1995), May, S. 117–120

[MMY11] M. Madani, Y. K. M. Fathizadeh F. M. Fathizadeh ; Yildirim, A.: On the coupling of the homotopy perturbation method and Laplace transformation. In: *Mathematical and Computer Modelling* 53 (2011), Nr. 9-10, S. 1937–1945

[M.N88] M.Nishigaki, H.Asai N.Tanaka: Mixed Mode Circuit Simulator SPLIT2.1 Using Dynamic Network Separation and Selective Trace. In: *IEEE* (1988), S. 9–12

[M.N91] M.Nakano, I.Shirakawa Y.Mae: A Waveform Relaxation method Applicable to Bipolar Digital Circuits. In: *IEEE* (1991), Apr, S. 2276–2279

[M.N93] M.Nishigaki, H.Asai N.Tanaka: Mixed mode circuit simulation using dynamic partitioning. In: *IEICE Trans. Fund.* E76-A (1993), Mar, Nr. 3, S. 292–298

[MP89] Madhaw P.Desai, Ibrahim H.: On the convergence of block relaxation methods for circuit simulation. In: *IEEE Transactions On Circuits And Systems* 36 (1989), Jul, S. 948–958

[MS09] M.Tadeusiewicz ; S.Hałgas: Improved algorithm for computing all the DC operating points of diode-transistor circuits. In: *European Conference on Circuit Theory and Design* (2009), S. 489–492 CD–ROM)

[M.T92] M.Tadeusiewicz: A method for finding bounds on all the DC solutions of transistor circuits. In: *IEEE Transactions on Circuits and Systems* 39 (1992), S. 557–564

[M.T94] M.Tadeusiewicz, K.Głowienka: A contraction algorithm for finding all the DC solutions of piecewise-linear circuits. In: *Journal of Circuits, Systems and Computers* 4 (1994), S. 319–336

[MT97a] Melville, R.C. ; Trajković, L.: Artificial parameter homotopy methods for the dc operating point problem. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12 (1997), Nr. 6, S. 861–877

[M.T97b] M.Tadeusiewicz: DC analysis of circuits with idealized diodes considering reverse bias breakdown phenomenon. In: *IEEE Transactions on Circuits and Systems* 44 (1997), Apr, S. 312–326

[M.T06a] M.Tadeusiewicz, S.Hałgas: Finding all the DC solutions in circuits containing bipolar transistors. In: *Proceedings of the International Conference on Signals and Electronic Systems* (2006), S. 361–365

[M.T06b] M.Tadeusiewicz, S.Hałgas: A method for the analysis of transistor circuits having multiple DC solutions. In: *International Journal of Electronics and Communications* 60 (2006), Nr. 8, S. 582–589

[N.H89] N.Hatano, I.Shirakawa S.Kumagai: On a windowing technique in the waveform relaxation method. In: *IEICE Tech. Rept.* (1989), S. CAS88–114. – in Japanese

[Nis89] Nishi, T.: An efficient method to find all solutions of piecewise-linear resistive circuits. In: *IEEE International Symposium on Circuits and Systems* (1989), S. 2052–2055

[NJ80] Nielsen, R.O. ; Jr., A.N. W.: A fundamental result concerning the topology of transistor circuits with multiple equilibria. In: *Proc. IEEE* 68 (1980), Feb, Nr. 2, S. 196–208

[OJ70] Ortega J.M., Rheinboldt W.: *Iterative Solution of Nonlinear Equations in Severae Variables.* New York : Acad. Press, 1970

[Owe94] Owe, Axelson: *Iterative solution methods.* Cambridge, 1994

[P+07] Press, William H. u. a.: *Numerical Recipes - The Art of Scientific Computing.* Cambridge Uniwersity Press, 2007. – ISBN 978–0–521–88407–5

[Pas09] Pastore, S.: Fast and efficient search for all DC solutions of PWL circuits by means of oversized polyhedra. In: *IEEE Transactions on Circuits and Systems* (2009), S. 2270–2279

[Ped89] Pedro, Pierre P. d.: Mecanique Vibratoire, Systemes discrets lineaires. In: *Presses Polytechniques Romandes,Laussane* (1989)

[Pla01a] Plaskura, P.: *Kierowana zdarzeniami symulacja systemów analogowych o opisie behawioralnym.* Warszawa, PhD dissertation, 2001

[Pla01b] Plaskura, P.: *OPTIMA v4, User Manual.* http://epub.aiva.pl. Version: 2001

[Pla13] Plaskura, P.: *Symulator mikrosystemów Dero v4. Metody i algorytmy obliczeniowe, modelowanie behawioralne, przykłady.* AIVA, 2013 http://epub.aiva.pl/?isbn=978-83-937245-1-2. – ISBN 978–83–937245–1–2

[P.O90] P.Odent, H. De M. L.M.Claesen: *Acceleration of Relaxation-Based Circuit Simulation Using A Multiprocessor System*, Oct 1990. (10) . – 1063–1072 S.

[PP96] Pastore, S. ; Premoli, A.: Capturing all branches of any one-port characteristic in piecewise-linear resistive circuits. In: *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.* 43 (1996), Jan, Nr. 1, S. 26–33

[P.P99] P.Plaskura, J.Ogrodzki ; Institute of Electronic Systems, Warsaw University of Technology (Hrsg.): *OPTIMA v4 User Manual*. Warsaw: Institute of Electronic Systems, Warsaw University of Technology, 1999

[P.S88] P.Saviz, O.Wing: PYRAMID - A hierarchical waveform relaxation-based circuit simulation program. In: *IEEE Int. Symp. Circuits and Systems* (1988), May

[P.S93] P.Saviz, O.Wing: Circuit Simulation by Hierarchical Waveform Relaxation. In: *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems* 12 (1993), Jun, Nr. 6, S. 845–860

[R.A83] R.A.Saleh: *Iterated Timing Analysis and SPLICE 1.* Berkeley, CA, 1983

[R.A89a] R.A.Saleh: *iSPLICE3 User's guide*, 1989

[RA89b] Resve A.Saleh, A.R.Newton: The exploitation of latency and multirate behavior using nonlinear relaxation for circuit simulation. In: *IEEE Transactions On Computer-Aided Design* 8 (1989), Dec, S. 1286–1298

[R.A96] R.A.Saleh, J.Singh B.A.A.Antao: Multilevel and Mixed-Domain Simulation of Analog Circuits and Systems. In: *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems* 15 (1996), Jan, Nr. 1, S. 68–81

[Rec65] Rechenberg, I.: Cybernetic solution path of an experimental problem. In: *Library Translation* (1965), Aug, Nr. 1122

[RGM99] R. Geoghegan, J.C. L. ; Melville, R.C.: Threading homotopies and dc operating points of nonlinear circuits. In: *SIAM Journal on Optimization* 9 (1999), Nr. 1, S. 159–178

[RJ90] R.Saleh ; J.White: Acceleration relaxation-based circuit simulation using waveform-Newton and step refinement. In: *IEEE Transactions Computer-Aided Design* 9 (1990), Sep, S. 951–958

[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

[R.K94] R.Kao, M.Horowitz: Timing Analysis for Piecewise Linear Rsim. In: *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems* 13 (1994), Dec, Nr. 12, S. 1498–1512

[RM] R.Geoghegan, J.C. L. ; Melville, R.C.: Threading homotopies and dc operatin points of nonlinear circuits. In: *SIAM J.Optim.* 9, S. 159–178

[RM06] Roychowdhury, J. ; Melville, R.: Delivering global DC convergence for large mixed-signal circuits via homotopy/continuation methods. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 25 (2006), Nr. 1, S. 66–78. http://dx.doi.org/10.1109/TCAD.2005.852461. – DOI 10.1109/TCAD.2005.852461. – ISSN 0278–0070

[RMW93] R.C. Melville, S.-C. F. L. Trajkovic T. L. Trajkovic ; Watson, L.T.: Artificial parameter homotopy methods for the dc operating point problem. In: *IEEE Trans. Computer-Aided Design* 12 (1993), Jun, Nr. 6, S. 861–877

[R.S65] R.S., Varga: *Matrix iterative analysis.* Englewood Cliffs,N.Y : Prentice-Hall, 1965

[RS94] Resve Saleh, A.Richard N. Shyh-Jye Jou J. Shyh-Jye Jou: *Mixed-mode simulation and analog multilevel simulation.* Norwell,Massachusetts 02061,USA : Kluwer Academic Publishers, 1994. – ISBN 0–7923–9473–9

[RW91] Rui Wang, Omar W.: Waveform relaxation on tightly coupled systems. In: *ISCAS Proceedings* (1991), S. 2280–2283

[SA97] S.Pastore ; A.Premoli: Finding all DC solutions of nonlinear resistive circuits by exploring both polyhedral and rectangular circuits. In: *IEEE Proceedings, Circuits, Devices and Systems* 9 (1997), S. 17–21

[SBE07] S.H. Behiry, I.L. El-Kalla H. H. H. Hashish ; Elsaid, A.: A new algorithm for the decomposition solution of nonlinear differential equations. In: *Computers & Mathematics with Applications* 54 (2007), Nr. 4, S. 459–466

[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

[Sch65] Schwefel, H.-P.: Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. (1965). – Diploma Thesis

[SK09] Sweilam, N.H. ; Khader, M.M.: Exact solutions of some coupled nonlinear partial differential equations using the homotopy perturbation method. In: *Computers & Mathematics with Applications* 58 (2009), Nr. 11-12, S. 2134–2141

[SMA09] S. Momani, G.H. E. ; Alnasr, M.H.: The modified homotopy perturbation method for solving strongly nonlinear oscillators. In: *Computers & Mathematics with Applications* 58 (2009), Nr. 11-12, S. 2209–2220

[S.P93] S.Pastore, A.Premoli: Polyhedral elements: A new algorithm for capturing all the equilibrium points of piecewise-linear circuits. In: *IEEE Transactions on Circuits and Systems* 40 (1993), S. 124–132

[SP95] Storn, R. ; Price, K.: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. In: *Technical Report TR-95-012, ICSI* (1995)

[Sto96] Storn, R.: On the usage of differential evolution for function optimization. (1996)

[SW69] Sandberg, I. ; Willson, A.: Some theorems on properties of DC equations of nonlinear networks. In: *Bell System Technical Journal* 48 (1969), S. 1–34

[Tad90] Tadeusiewicz, M.: A method for finding bounds on the location of all the solutions of DC piecewise-linear circuits. In: *International Journal of Circuit Theory and Applications* 18 (1990), S. 165–174

[Tad97] Tadeusiewicz, M.: DC analysis of circuits with idealized diodes considering reverse bias breakdown phenomenon. In: *IEEE Trans. Circutis Sys. I, Fundam. Theory Appl.* 44 (1997), Apr, Nr. 4, S. 312–326

[T.C97] T.Chruściel: *Implementacja i badanie algorytmu kierowanego zdarzeniami w czasowej analizie układów elektronicznych.* Warszawa, Polska, ISE, Politechnika Warszawska, Diplomarbeit, 1997

[TD91] T.A., Johnson ; D.J., Zukowski: Waveform-relaxation-based circuit simulation on the Victor V256 parallel processor. In: *IBM J. Res. Develop.* 35 (1991), Sept/Nov., Nr. 5/6

[TH09] Tadeusiewicz, M. ; Hałgas, S.: Finding all the DC solutions of circuits containing diodes and bipolar transistors. In: *Elektronika* 12 (2009), S. 39–42

[TM91] Tadeusiewicz M., Wykrota W.: An Algorithm for Analysis of Nonlinear Resistive Circuits Having Multiplr D.C. Solution. In: *KKTIUE* (1991)

[T.S88] T.Shima, Y.Kamatani: A circuit simulator based on the waveform relaxation method using selective overlapped partition and classified latencies. In: *ISCAS Proceedings* (1988), S. 1651–1654

[UA84] Ushida A., Chua L.: Tracing Solution Curves of Non-linear Equations with Sharp Turning points. 12 (1984), S. 1–21

[UFNo12] U. Filobello-Nino, Y. K. H. Vazquez-Leal ; others.: HPM applied to solve nonlinear circuits: a study case. In: *Applied Mathematical Sciences* 6 (2012), Nr. 85-88, S. 4331–4344

[V.L05] V.Lakshmikatham, S.K. S.: *Computational Error and Complexity in Science And Engineering.* Elsevier B.V, 2005. – ISBN 044–45–1860–6

[VL12] Vazquez-Leal, H.: Rational homotopy perturbation method. In: *Journal of Applied Mathematics* 2012 (2012)

[VM97] V., Litovski ; M., Zwolinski: VLSI: Circuit Simulation and Optimization. (1997)

[Wika] Wiki: *Carl Friedrich Gauss*. http://pl.wikipedia.org/wiki/Carl_Friedrich_Gauss

[Wikb] Wiki: *Homotopia*. http://pl.wikipedia.org/wiki/Homotopia

[Wikc] *Macierze*. Wikipedia, . – http://pl.wikipedia.org/wiki/Kategoria:Macierze

[Wikd] *Matrix*. Wikipedia, . – http://en.wikipedia.org/wiki/Matrix_(mathematics)

[Wike] *Matrix decomposition*. Wikipedia, . – http://en.wikipedia.org/wiki/Matrix_decomposition

[Wikf] *Matrix norm*. Wikipedia, . – http://en.wikipedia.org/wiki/Matrix_norm

[Wikg] *Schur decomposition*. Wikipedia, . – http://en.wikipedia.org/wiki/Schur_decomposition

[Wikh] *Wektor*. Wikipedia, . – http://pl.wikipedia.org/wiki/Wektor

[Wiki] Wiki: *Zagadnienie uwarunkowania macierzy*. http://numericalmethods.eng.usf.edu/mws/gen/04sle/mws_gen_sle_spe_adequacy.pdf

[Wik16] Wiki: *Bifurkacja*. http://pl.wikipedia.org/wiki/Bifurkacja_(matematyka). Version: 2016

[Wil63] Wilkinson, J.H.: *Rounding Errors in Algebraic Processes*. Prientice-Hall, 1963

[Wil65] Wilkinson, J.H.: *The algebraic Eigenvalue Problem*. Oxford : Clarendon Press, 1965

[Wil75] Willson, A.N.: The no-gain property for networks containing three-terminal networks. In: *IEEE Trans. Circuits Syst.* CAS-22 (1975), Aug, Nr. 8, S. 678–687

[Wir04] Wirth, Niklaus: *Algorytmy + struktury danych = programy*. Warszawa : Wydawnictwo Naukowo-Techniczne, 2004

[WMM02] W. Ma, L. T. ; Mayaram, K.: HomSSPICE: A homotopy based circuit simulator for periodic steady-state analysis of oscillators. In: *Proc. Int. Symp. Circuits Systems* (2002), May, S. 26–29

[WS96] Wolf, D.M. ; Sanders, S.R.: Multiparameter homotopy methods for finding dc operating points of nonlinear circuits. In: *IEEE Transactions on Circuits and Systems* 43 (1996), Nr. 10, S. 824–838

[Y+89] Y.H.Jun u. a.: Timing Simulator by Waveform Relaxation Considering Feedback Effect. In: *Proc. of the IEEE Int. Symp. of Circuits and Systems* (1989), S. 608–611

[Yam84] Yamamoto, Y.: A variable dimension fixed point algorithm and the orientation of simplices. In: *Mathematical Programming* 30 (1984), Nr. 3, S. 301–312

[YC89] Yasutami Chigusa, Mamoru T. Mitsuo Asai A. Mitsuo Asai: SRP: A relaxation-based circuit simulator with error-collecting-learning method by adaptive virtual capacitors. In: *ISCAS Proceedings* (1989), S. 1149–1152

[Y.H88] Y.H.Jun, S.B.Park S.H.Lee: High Speed Logic Simulation for VLSI Using Bitwise Operation. In: *Proc. of the KIEE JTC-CSCC* (1988)

[Y.H89] Y.H.KIM, A.R.Newton S.H.Hwang: Electrical-Logic Simulation and Its Applications. In: *IEEE Transactions On Computer-Aided Design* 8 (1989), Jan, Nr. 1, S. 8–22

[YHJ89] Young-Hyun Jun, Song-Bai P.: KMIX: A Mixed-Mode Simulator for Analog-Digital Circuits Using Event Driven Waveform Relaxation Method. In: *ISCAS Proceedings* (1989), S. 877–880

[YI04] Yamamura, K. ; Igarashi, N.: An interval algorithm for finding all solutions of non-linear resistive circuits. In: *International Journal of Circuit Theory and Applications* 32 (2004), S. 47–55

[YIT01] Y. Inoue, K. Y. S. Kusanobu K. S. Kusanobu ; Takahashi, T.: Newton-fixed-point homotopy method for finding dc operating points of nonlinear circuits. In: *Proceedings of the International Technical Conference on Circuits/Systems, Computer and Communications (ITC-CSCC '01)* 1 (2001), July, S. 370–373

[YK02] Y.Inoue, S.Kusanobu ; K.Yamamura: A practical approach for the fixed-point homotopy method using a solution-tracing circuit. In: *IEICE Trans. Fund.* E85-A (2002), Jan, Nr. 1, S. 222–233

[YK+12] Y. Khan, Q. W. u. a.: A new fractional analytical approach via a modified riemannliouville derivative. In: *Applied Mathematics Letters* 25 (2012), Nr. 10, S. 1340–1346

[YLS08] Y. Lin, J.A. E. ; Stadtherr, M.A.: Enclosing all solutions of two-point boundary value problems for ODEs. In: *Computers and Chemical Engineering* 32 (2008), Nr. 8, S. 1714–1725

[YM96] Yamamura, K. ; Mishina, M.: An algorithm for finding all solutions of piecewise-linear resistive circuits. In: *Int. J. Circuit Theory Appl.* 24 (1996), S. 223–231

[YM03] Y.Inoue, K.Yamamura S.Kusanobu ; M.Ando: An effective initial solution algorithm for globally convergent homotopy methods. In: *Proc. Int. Symp. Circuits Systems* (2003), May, S. 196–199

[YM08] Yamamura, K. ; Machida, A.: An efficient algorithm for finding all DC solutions of piecewise-linear circuits. In: *International Journal of Circuit Theory and Applications* 36 (2008), S. 989–1000

[Z.F93] Z.Fortuna, J.Wąsowski B.Macukow: *Metody numeryczne.* Warszawa : WNT, 1993

[ZMY00] Zwolinski M., Crutchley D. ; Yang, Z.R.: Evolutionary computing for operating point analysis of nonlinear circuits. In: *Proceedings of ICSES 2000* (Oct 17th-20th, 2000)

[ZMZ00] Zwolinski M., Crutchley D. ; Z.R., Yang: Evolutionary computing for operating point analysis of nonlinear circuits. In: *Proceedings of ICSES 2000* (2000), Oct 17th-20th

W pracy przedstawiono wybrane, zaawansowane metody i algorytmy obliczeniowe wykorzystane w symulatororach układów elektronicznych. Omówiono sposoby automatycznego układania równań zmodyfikowaną metodą potencjałów węzłowych oraz podano odpowiednie szablony. Przedstawiono metodę obliczania pochodnych w postaci symbolicznej. Wiele z opisywanych algorytmów została zaimplementowana w praktyce w symulatorze *Dero*, który dostępny jest w sieci: `http://dero.aiva.pl`.

Autor od wielu lat zajmuje się systemami operacyjnymi Unix, w szczególności systemem Linux. Poza omawianą tematyką główne obszary zainteresowań autora obejmują: techniki symulacji układów elektronicznych oraz mikrosystemów, języki behawioralnego opisu sprzętu, systemy zarządzania tokiem kształcenia, platformy e-learningowe, techniki programowania obiektowego odporne na błędy. Jest autorem szeregu systemów, m.in.: symulatora *Dero*, platformy obsługi toku kształcenia i nauczania zdalnego *Quela*, systemu różniczkowania symbolicznego *Deriv*.

ISBN 978-83-937245-0-5