Paweł Niezgodzki

JĘZYK SYMULACYJNY LAHYSS DO MODELOWANIA SYSTEMÓW HYBRYDOWYCH NA MASZYNIE CYFROWEJ

WARSZAWA 1979

Praca wpłynęła do Redakcji dnia 14 marca 1980 r.

Zarejestrowana pod nr 9/1980

Praca została wykonana w ramach problemu węzłowego 05.12
"Wytrzymałość i optymalizacja konstrukcji maszynowych
i budowlanych"

IPPT PAN

Na prawach rękopisu

Instytut Podstawowych Problemów Techniki PAN
Nakład 150 egz. Ark.wyd. 5,6. Ark.druk.8,25
Oddano do drukarni w marcu 1980 r.
Nr zamówienia 210/0/80

Warszawska Drukarnia Naukowa, Warszawa,
ul.Śniadeckich 8

Paweł Niezgodzki
Zakład Układów Mechanicznych

JĘZYK SYMULACYJNY LAHYSS DO MODELOWANIA SYSTEMÓW HYBRYDOWYCH NA MASZYNIE CYFROWEJ

Wprowadzenie

Język symulacyjny **LAHYSS** /Language for Analog and HYbrid System Simulation/ jest językiem specjalizowanym służącym do modelowania układów dynamicznych oraz algorytmów ich badania na maszynie cyfrowej PDP 11/70. Sposób przygotowania programów w języku **LAHYSS** jest analogiczny z metodami stosowanymi przy programowaniu systemów hybrydowych.

Język **LAHYSS** jest językiem łatwym do nauczenia i stosowania. Wymaga od użytkownika jedynie znajomości podstawowych zasad programowania maszyn analogowych i cyfrowych. Duża automatyzacja programowania oraz zorientowana na problem struktura i treść programów sprawiają, że z języka mogą korzystać osoby o niewielkim doświadczeniu w programowaniu maszyn cyfrowych.

Język może być stosowany w wielu dziedzinach nauki i techniki do:

- rozwiązywania równań różniczkowych zwyczajnych i cząstkowych,
- badania przebiegów w układach dynamicznych,
- modelowania złożonych algorytmów badania układów dynamicznych,
- optymalizacji parametrycznej i identyfikacji,
- modelowania układów w których występują równocześnie wielkości o charakterze analogowym i dyskretnym /logiczne/.

Szczególnie przydatny jest on do rozwiązywania złożonych zadań analizy i syntezy w mechanice a zwłaszcza dynamice maszyn. Zadania te charakteryzują się skomplikowanymi algorytmami ba-

dania dynamicznych układów mechanicznych, opisanych zarówno równaniami zwyczajnymi jak i cząstkowymi.

Język LAHYSS jest wynikiem prac, związanych z zastosowaniem techniki analogowej i hybrydowej, a także metod symulacji cyfrowej do obliczania i optymalizacji drgań i stateczności dynamicznej konstrukcji maszynowych. Prace te prowadzone były w Zakładzie Układów Mechanicznych Instytutu Podstawowych Problemów Techniki PAN w ramach problemu węzłowego 05.12 pt. "Wytrzymałość i optymalizacja konstrukcji maszynowych i budowlanych".

Translator języka LAHYSS opracowany został dla maszyny cyfrowej PDP 11/70. Należy jednak podkreślić, że może być on, z niewielkimi zmianami, przeniesiony na inne maszyny, dysponujące translatorem języka FORTRAN /np. komputery serii RIAD/.

Maszyna cyfrowa PDP 11/70 jest maszyną czwartej generacji. Pracuje ona w systemie wielodostępnym, co bardzo ułatwia praktyczny dostęp do komputera. Maszyna posiada bogate oprogramowanie systemowe i aplikacyjne, w skład którego wchodzą m.in.: szeroki zestaw rozkazów dla zarządzania zbiorami danych, dobrze opracowany program - editor, pracujący w systemie interakcyjnym oraz biblioteka podprogramów dla obliczeń naukowo-technicznych. Środki te znakomicie upraszczają pisanie i uruchamianie programów, skracając do minimum okres czasu pomiędzy napisaniem programu, a uzyskaniem wyników. Dodatkowe zalety maszyny cyfrowej PDP 11/70 to: możliwość przechowywania i operowania programami na zbiorach dyskowych, co pozwala na uniknięcie kłopotów związanych z stosowaniem kart dziurkowanych oraz bogato oprogramowany graph-plotter współpracujący z EMC pozwalający na uzyskiwanie wyników w postaci graficznej.

Praca podzielona została na cztery rozdziały. /W nawiasach podane są pozycje literatury dotyczące poszczególnych zagadnień/.

Rozdział 1 zawiera wiadomości dotyczące modelowania hybrydowego oraz budowy i programowania systemów hybrydowych /[2], [7], [8], [12], [14], [18], [27], [29], [31]/. Omówiono również języki symulacyjne pod kątem ich zastosowania do rozwiązywania zagadnień modelowania hybrydowego/[3], [4], [6], [10], [13], [15], [16], [17], [20], [21], [22], [25], [34], również

pozycje wymienione poprzednio/ .

Rozdział 2 zawiera opis języka LAHYSS. Omówiono w nim strukturę programów użytkowych oraz zdefiniowano podstawowe elementy języka i zasady prawidłowej pisowni instrukcji tego języka. Język LAHYSS jest ściśle związany z językiem algorytmicznym FORTRAN /[1] , [30]/ , którego znajomość jest potrzebna do pełnego wykorzystania możliwości LAHYSS-u.

W rozdziale 3 przedstawiono metodykę programowania w języku LAHYSS. Programowanie w tym języku oparte jest na metodach modelowania hybrydowego, opisanych w rozdziale 1 i wymaga znajomości zasad programowania hybrydowych maszyn analogowych /[9] , [19] , [23] , [32] , [33] , [35]/ oraz programowania w języku FORTRAN. Rozdział zawiera również informacje dotyczące praktycznej strony pisania i uruchamiania programów na maszynie cyfrowej /[5, 11]/.

W rozdziale 4 przedstawiono przykłady kompletnych programów użytkowych wraz z wynikami symulacji komputerowej. Przykłady dotyczą problemów często występujących w badaniach układów dynamicznych /[28] , [8]/. Pozwalają one na prześledzenie całego procesu modelowania hybrydowego z wykorzystaniem języka LAHYSS począwszy od sformułowania zadania do uzyskania wyników.

1. Wstęp do programowania hybrydowego

W ostatnich latach obserwujemy wzrost zainteresowania zastosowaniem hybrydowej techniki obliczeniowej do rozwiązywania złożonych zadań analizy i syntezy układów dynamicznych. Pojęcie hybrydowej techniki obliczeniowej obejmuje zarówno jej środki techniczne - systemy hybrydowe, jak i specyficzne metody programowania.

W rozdziale przedstawimy podstawowe wiadomości o modelowaniu hybrydowym oraz budowę i zasady programowania systemów hybrydowych. Pozwoli to na lepsze zrozumienie podstawowej idei języka LAHYSS, polegającej na zastąpieniu rzeczywistego systemu hybrydowego przez środki symulacji cyfrowej. Sposób podejścia do rozwiązywanego problemu oraz metodyka tworzenia programów użytkowych w tym języku są analogiczne do metod stosowanych przy

programowaniu systemów hybrydowych.

1.1. Modelowanie hybrydowe

Modelowanie hybrydowe można podzielić na następujące zasadnicze etapy:

1/ Sformułowanie problemu
2/ Opracowanie ogólnego algorytmu hybrydowego dla całości problemu
3/ Opracowanie schematu analogowego
4/ Opracowanie algorytmu dla części cyfrowej
5/ Opracowanie i uruchomienie programu hybrydowego
6/ Obliczenia symulacyjne

Sformułowanie problemu może mieć różnorodny charakter. Najczęściej jest to opis matematyczny w którym wyraźnie wyodrębnione są dwie części: dynamiczna /analogowa/ i numeryczna /cyfrowa/. W części dynamicznej opisany jest badany układ dynamiczny. Opis może być podany w postaci układu równań różniczkowych zwyczajnych, bądź w postaci schematu blokowego układu. W części numerycznej sformułowany jest matematycznie problem dotyczący układu dynamicznego, nazywany dalej algorytmem badania układu dynamicznego. Dla ilustracji podamy przykład sformułowania zadania doboru optymalnych parametrów układu mechanicznego według całkowego kryterium jakości.

A. Część analogowa

Załóżmy, że układ mechaniczny opisany jest układem równań różniczkowych zwyczajnych w postaci:

$$m_i\ddot{x}_i + \varphi_i(\dot{x}_1,\dots,\dot{x}_n,l_1,\dots,l_s) + f_i(x_1,\dots,x_n,k_1,\dots k_s) = P_i\cos\gamma t \quad , \quad i=1,\dots,n$$

gdzie:

n - liczba stopni swobody
m_i - masa
x_i - współrzędna uogólniona
f_i - siła sprężysta
φ_i - siła tłumienia wiskotycznego
k_j - współczynniki sztywności sprężystej

l_j - współczynniki tłumienia

P_i - amplituda harmonicznej siły wymuszającej

w przedziale czasowym $0 \leq t \leq T$. Układ spełnia warunki początkowe:

$$x(0) = xp_i$$
$$\dot{x}(0) = \dot{x}p_i$$

$$i = 1,\ldots,n .$$

B. Część numeryczna

Należy dobrać wartości współczynników sztywności sprężystej k_j $j = 1,\ldots,n$ tak, ażeby całkowy wskaźnik jakości, określony wzorem:

$$J(k_1,\ldots,k_s) = \int_0^T F(t,x_1,\ldots,x_n,\dot{x}_1,\ldots,\dot{x}_n)\,dt$$

osiągał minimum. Obszar dopuszczalnych wartości zmiennych decyzyjnych $k_1,\ldots,k_s$ określony jest nierównościami:

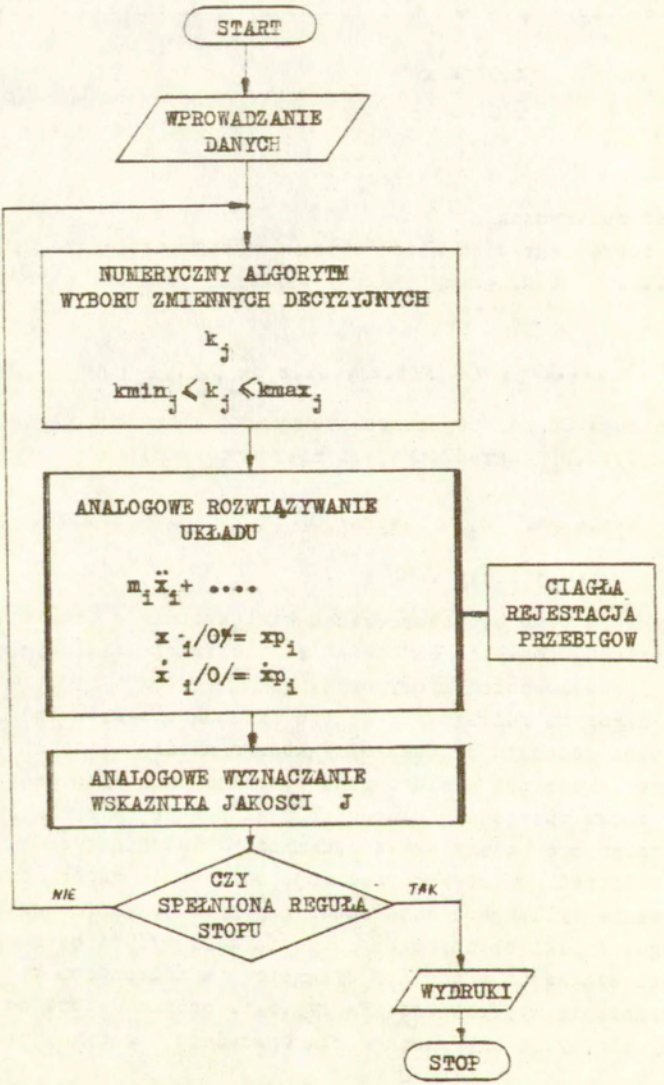
$$k_{j\min} \leq k_j \leq k_{j\max}$$

$$j = 1,\ldots,n .$$

Do przykładu tego będziemy wracać wielokrotnie w dalszej części rozdziału, pozwala on bowiem na ilustrację wielu pojęć związanych z modelowaniem hybrydowym.

Ogólny algorytm rozwiązania całego zadania przedstawiany jest w postaci schematu blokowego, w którym wyodrębnione są 4 bloki: danych, operacji analogowych, operacji cyfrowych oraz wyników. W bloku operacji analogowych realizowane są równania i relacje opisujące badany układ dynamiczny. Natomiast blok operacji /obliczeń/ cyfrowych realizuje obliczenia numeryczne oraz sterowanie wynikające z zadanego algorytmu badania układu dynamicznego. Wyniki obliczeń /symulacji/ mogą być rejestrowane w postaci ciągłej i w postaci dyskretnej w zależności od użytego urządzenia wyjściowego. Na rys.1.1. przedstawiono schemat blokowy algorytmu hybrydowego dla opisanego wcześniej przykładu.

W miejscu tym należy podkreślić, że w wielu przypadkach zależności opisane w części numerycznej problemu realizowane



Rys. 1.1. Schemat blokowy przykładowego algorytmu hybrydowego.

są w praktyce przez blok operacji analogowych. Obliczanie całkowego wskaźnika jakości użytego w omawianym przykładzie jest znacznie prościej zrealizować analogowo niż cyfrowo.

Schemat analogowy jest formą zapisu bloku operacji analogowych, przystosowaną do realizacji na maszynie analogowej, wchodzącej w skład systemu hybrydowego. Zasady budowy schematów analogowych wyjaśnione będą w rozdziale 3.

Algorytm dla części cyfrowej algorytmu hybrydowego zawiera szczegółowy opis obliczeń numerycznych, potrzebnych do rozwiązania całego problemu oraz instrukcje sterujące blokiem operacji analogowych. Sposób przygotowania tego algorytmu jest identyczny z metodami stosowanymi przy programowaniu uniwersalnych EMC i musi już uwzględniać właściwości języka programowania w jakim będzie zapisany.

Schemat analogowy oraz algorytm cyfrowy są podstawą do opracowania programu hybrydowego. Program taki składa się z dwóch podprogramów: analogowego i cyfrowego. Przy sporządzeniu programu, który w dalszej części pracy nazywać będziemy również programem użytkowym, należy pamiętać o zapewnieniu współpracy między obydwoma podprogramami. W tym celu należy wyodrębnić wielkości, które będą przekazywane pomiędzy obu podprogramami.

Opracowanie podprogramu analogowego przebiega zgodnie z zasadami programowania maszyn analogowych i jest ściśle związane z typem maszyny wchodzącej w skład systemu hybrydowego. Podprogram cyfrowy programu hybrydowego przygotowywany jest w języku algorytmicznym rozszerzonym o instrukcje hybrydowe, umożliwiające wymianę danych między podprogramami oraz sterowanie bloku operacji analogowych przez podprogram cyfrowy. Przy uruchamianiu programu hybrydowego dokonuje się sprawdzenia poszczególnych jego podprogramów oraz wykonuje się obliczenia testujące działanie całego programu. Szczególnie istotne jest sprawdzenie prawidłowego połączenia części analogowej, gdyż stanowi ona na ogół główne źródło błędnego działania programu hybrydowego. W dalszej części pracy oba podprogramy programu hybrydowego będziemy nazywali po prostu programami: analogowym i cyfrowym.

1.2. Systemy hybrydowe.

1.2.1. Struktura systemów hybrydowych.

W skład współczesnych systemów hybrydowych wchodzą następujące urządzenia:

1/ hybrydowa maszyna analogowa /MA/,
2/ uniwersalna maszyna cyfrowa /MC/,
3/ urządzenie sprzęgające,
4/ urządzenia wejścia - wyjścia.

Schemat blokowy systemu hybrydowego przedstawiono na rys. 1.2. Ważną częścią składową systemu, nie pokazaną na schemacie jest jego oprogramowanie /software/.

Hybrydowa maszyna analogowa służy do realizacji bloku operacji analogowych, a więc do rozwiązywania równań różniczkowych i relacji opisujących badany układ dynamiczny. W tym celu maszyna wyposażona jest w elementy liczące umożliwiające wykonywanie następujących operacji matematycznych typu ciągłego:

1/ całkowanie /integratory/,
2/ operacje arytmetyczne /sumatory, inwertory, mnożarki, elementy dzielące, potencjometry/,
3/ obliczanie funkcji jednej zmiennej /przetwornik funkcji/.

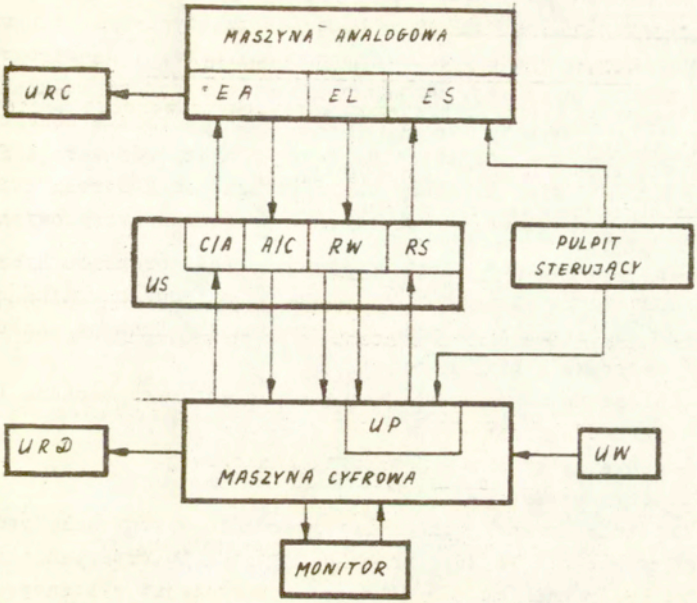
Oprócz tego hybrydowa maszyna analogowa posiada elementy umożliwiające wykonywanie operacji nieciągłych oraz elementy realizujące podstawowe funkcje logiczne. Należą do nich:

1/ komparatory z wyjściem logicznym,
2/ przekaźniki sterowane sygnałami logicznymi,
3/ pamięci analogowe,
4/ bramki typu NOR i NAND,
5/ przerzutniki typu D i JK,
6/ generatory impulsów.

Wyposażenie to sprawia, że wiele problemów może być zamodelowanych całkowicie na hybrydowej maszynie analogowej.

Maszyna analogowa przystosowana jest do sterowania ręcznego oraz automatycznego, poprzez sygnały sterujące z maszyny cyfrowej.

Maszyna cyfrowa pracująca w systemie hybrydowym to na ogół uniwersalny minikomputer. Minikomputer taki, oprócz oprogramo-



Oznaczenia:

EA - elementy analogowe
EL - elementy logiczne
ES - elementy sterujące
US - urządzenie sprzęgające
C/A - przetwornik
A/C - przetwornik
RW - rejestr wskaźnikowy
RS - rejestr sterujący
UP - układ przerwań
UW - urządzenie wejściowe
URC - urządzenia rejestracji ciągłej
URD - urządzenia rejestarcji dyskretnej

Rys. 1.2. Schemat blokowy systemu hybrydowego.

wania typowego dla tej klasy EMC, posiada pakiety programów systemowych umożliwiających pracę w czasie rzeczywistym, zapewniających współpracę i sterowanie maszyny analogowej. Na oprogramowanie maszyny cyfrowej, użytej w systemie hybrydowym składają się również: translator języka algorytmicznego rozszerzonego o instrukcje hybrydowe /najczęściej jest to hybrydowa wersja FORTRAN-u - HYTRAN/ oraz programy diagnostyczne do kontroli działania systemu hybrydowego i testowania programów hybrydowych.

Program dla maszyny cyfrowej stanowi część programu hybrydowego i służy do realizacji bloku operacji cyfrowych wynikających z ogólnego algorytmu hybrydowego. Przypomnijmy funkcje części cyfrowej programu hybrydowego:

1/ obliczenia numeryczne związane z algorytmem badania układu dynamicznego,

2/ sterowanie hybrydowej maszyny analogowej,

3/ obsługa przerwań.

Ta ostatnia funkcja polega na przerwaniu wykonywania programu hybrydowego i reakcji na sytuację jaka to przerwanie spowodowała. Przerwania mogą być dwojakiego rodzaju: systemowe i programowe. Przerwania systemowe wynikają z błędnego działania urządzeń systemu i błędów programowania. Reakcja na te przerwania jest automatyczna i polega na ogół na zawieszeniu wykonywania programu. Przerwania programowe wynikają z sytuacji przewidzianych przez programistę. Są nimi na ogół przypadki szczególne modelowanych algorytmów hybrydowych, występujące po stronie analogowej programu hybrydowego. Reakcja na te przerwania polega na wykonaniu obliczeń numerycznych przewidzianych przez programistę dla danego przerwania i kontynuacji /czasami zatrzymaniu/ obliczeń. Należy podkreślić, że niektóre sytuacje powodujące przerwania systemowe, mogą być obsługiwane programowo, nie powodując zawieszenia wykonywania programu. Do sytuacji takich należą na przykład przekroczenia dozwolonego zakresu napięcia przez sygnały występujące w maszynie analogowej.

Urządzenie sprzęgające zapewnia współpracę maszyny cyfrowej i hybrydowej maszyny analogowej. Obecność takiego urządzenia jest konieczna ze względu na całkowicie odmienną postać wielkości

występujących w obu maszynach. W maszynie cyfrowej dane występują w postaci dyskretnej, natomiast maszyna analogowa "obrabia" dane w postaci ciągłej. Spełnia ono następujące funkcje:

1/ przetwarzanie sygnałów analogowych na postać cyfrową /przetworniki A/C/,
2/ przetwarzanie danych numerycznych na sygnały analogowe /przetworniki C/A/,
3/ dopasowanie sygnałów logicznych,
4/ generowanie sygnałów przerwań.

Urządzenia wyjściowe służą do rejestracji wyników obliczeń symulacyjnych. Wyniki mogą być rejestrowane w dwojaki sposób:

1/ ciągły przez urządzenia rejestracji ciągłej połączone bezpośrednio z MA /rejestratory, oscyloskopy, woltomierze itp/,
2/ dyskretny przez urządzenia wyjściowe połączone z maszyną cyfrową /monitor, drukarka, pamięć zewnętrzna itp/.

Urządzenia te pozwalają na: bieżącą obserwację przebiegów /oscyloskop, monitor MC/, długotrwałe przechowywanie wyników w postaci cyfrowej /w pamięci zewnętrznej MC/ oraz bogatą dokumentację przeprowadzonych badań w formie graficznej /rejestrator/ oraz wydruków na drukarce wierszowej.

Urządzenia wejściowe służą do wprowadzania danych dla programu hybrydowego. Program hybrydowy wymaga danych parametrów dla programu analogowego i danych liczbowych programu cyfrowego. Wprowadzenie danych dla programu analogowego polega na: nastawieniu przetworników funkcyjnych, potencjometrów oraz warunków początkowych. Czynności te można wykonać dwojako: ręcznie, środkami hybrydowej maszyny analogowej lub programowo, za pośrednictwem programu cyfrowego. W tym drugim przypadku dane wczytywane są poprzez urządzenia wejściowe MC.

Maszyna cyfrowa posiada dwa zasadnicze urządzenia wejściowe: monitor oraz czytnik kart lub taśmy dziurkowanej. Monitor służy do wyprowadzania danych w trybie konwersacyjnym, natomiast czytnik wymaga wcześniejszego przygotowania danych dla programu hybrydowego. Ze względu na szybkość transmisji danych monitor używany jest do wprowadzania najistotniejszych, naj-

częściej zmienianych parametrów i danych liczbowych. Jego zasadniczą funkcją jest zapewnienie komunikacji z MC oraz sterowanie pracą całego systemu hybrydowego za pośrednictwem systemu operacyjnego MC.

Pojęcie sterowania w odniesieniu do systemów hybrydowych jest wieloznaczne. Obejmuje ono sterowanie wykonywaniem całego programu hybrydowego oraz sterowania programem analogowym. Sterowanie całego programu hybrydowego obejmuje takie czynności jak: translacja i uruchamianie programu oraz inicjacja obliczeń. Realizowane jest poprzez system operacyjny z monitora MC. Sterowanie programu analogowego polega, ogólnie mówiąc, na sterowaniu stanami pracy MA i wprowadzaniu parametrów /danych/ do poszczególnych bloków. Obejmuje ono również sterowanie urządzeniami peryferyjnymi współpracującymi z programem w sposób ciągły. Sterowanie to może być realizowane trojako: przez system operacyjny MC, poprzez program cyfrowy, bądź też ręcznie z pulpitu sterującego MA. W pracy pokażemy, że wiele algorytmów hybrydowych można zrealizować bez podprogramu cyfrowego. W takich przypadkach sterowanie wykonywaniem programu hybrydowego sprowadza się do sterowania programem analogowym.

1.2.2. Cechy charakterystyczne systemów hybrydowych.

Systemy hybrydowe są obecnie podstawowym urządzeniem modelowania hybrydowego, ale, jak pokażemy w dalszej części pracy, nie jedynym. Stanowią one próbę połączenia zalet maszyn analogowych i cyfrowych oraz eliminację wad obu tych maszyn stosowanych oddzielnie. Podstawowe cechy współczesnych systemów hybrydowych to:

1/ przetwarzanie danych w postaci ciągłej /analogowej/ i dyskretnej /cyfrowej/, a także danych numerycznych,
2/ połączenie szybkości obliczeń MA z dokładnością MC,
3/ możliwość modelowania rozbudowanych, pod względem decyzyjnym, algorytmów,
4/ duża elastyczność programowania,
5/ szerokie możliwości rejestracji i przechowywania wyników symulacji,
6/ możliwość ingerencji użytkownika w trakcie wykonywania obliczeń.

Należy podkreślić, że większość zalet systemów hybrydowych wynika z możliwości maszyny cyfrowej. Podstawową i praktycznie jedyną /z punktu widzenia systemu hybrydowego/ zaletą maszyny analogowej jest duża szybkość rozwiązywania równań różniczkowych, niezależna od liczby tych równań.

Wady systemów hybrydowych wynikają z faktu, że stanowią one połączenie urządzeń, których zasady działania oraz technologia wykonania są zupełnie odmienne. O ile w obecnych czasach obserwujemy ogromny postęp w budowie i możliwościach maszyn cyfrowych, o tyle w dziedzinie budowy maszyn analogowych postęp ten jest niewielki. Wynika to z faktu, że maszyny cyfrowe mają bardzo uniwersalne zastosowanie w dziedzinach gospodarki, nauki i techniki. Maszyny analogowe są urządzeniami specjalizowanymi o ograniczonych dziedzinach zastosowań, co znacznie limituje popyt, a więc i nakłady na rozwój tych maszyn.

Podstawowe wady istniejących systemów hybrydowych to:

1/ duża zawodność sprzętu, szczególnie części sprzęgającej MA i MC
2/ ograniczona pojemność i dokładność obliczeń MA,
3/ duże straty czasu na przygotowanie i uruchomienie programu hybrydowego, co wynika z konieczności rozwiązywania problemów, nie związanych z istotą modelowania zadania /skalowanie MA, realizacja części sprzęgającej itp./,
4/ trudności w przechowywaniu, manipulacji i wymianie gotowych programów hybrydowych.

Dodatkowe ograniczenia w szerokim rozpowszechnieniu systemów hybrydowych /ale nie metod modelowania hybrydowego/ w Polsce to:

1/ wysoki koszt pełnego systemu hybrydowego i związane z zakupem wydatki dewizowe,
2/ wymagane od użytkowników systemów wysokie kwalifikacje w zakresie programowania i obsługi maszyn analogowych i minikomputerów.

1.3. Języki symulacyjne a modelowanie hybrydowe.

Współczesne maszyny cyfrowe pozwalają już na rozwiązywanie /symulację/ większości problemów związanych z modelowaniem hybrydowym. Do celów symulacji cyfrowej można użyć każdy język algorytmiczny, np. FORTRAN, jednak nakład czasu i pracy potrzebny dla napisania i uruchomienia programu powoduje, że stosowanie tych języków staje się nieekonomiczne. Fakt ten stał się bezpośrednią przyczyną podjęcia na początku lat 60-tych prac nad językami zorientowanymi problemowo. Prace te doprowadziły do powstania licznej grupy specjalizowanych języków symulacyjnych przeznaczonych do modelowania układów dynamicznych.

Języki symulacyjne posiadają następujące zalety w porównaniu z językami uniwersalnymi:

1/ ścisły związek programu z modelowanym problemem,
2/ skrócenie czasu i łatwość programowania,
3/ efektywniejsze metody wykrywania błędów,
4/ zwięzłość i jasność wyrażeń opisujących modelowany układ,
5/ automatyzacja wielu czynności programowania i obliczeń,
6/ możliwość prowadzenia obliczeń w trybie konwersacyjnym,
7/ rozszerzone możliwości rejestracji wyników symulacji.

Cechy te sprawiają, że z języków symulacyjnych mogą korzystać osoby o małym doświadczeniu w programowaniu EMC.

Do chwili obecnej powstało na świecie i w Polsce wiele specjalizowanych języków symulacyjnych do modelowania układów dynamicznych. Do najbardziej znanych należą:

- CSMP /Continous System Modeling Program, [4] /, opracowany przez firmę IBM i posiadający translatory na większość komputerów tej firmy,
- CSSL /Continous System Simulation Language, [34] /, opracowany przez grupę naukowców związanych z The SCi Simulation Software Committe jako język "wzorcowy",
- Mimic /[25]/ , posiadający translatory na niektóre maszyny firm IBM i CDC.

Z polskich opracowań należy wymienić języki: AM-F-1 /[12]/ oraz CEMMA /[21,22]/.

Języki te posiadają rozbudowane możliwości modelowania układów dynamicznych. Nie pozwalają jednak /lub czynią to w bardzo skomplikowany sposób/ na symulację wielu algorytmów typowych dla modelowania hybrydowego.

Przedstawione w rozdziale fakty, w szczególności:

- duża przydatność metod modelowania hybrydowego do rozwiązywania szerokiej klasy zadań związanych z układami dynamicznymi, a jednocześnie
- trudności i ograniczenia w wykorzystaniu rzeczywistego systemu hybrydowego do rozwiązywania zadań modelowania hybrydowego,
- coraz większe możliwości uniwersalnych EMC, w szczególności: rosnące szybkości i malejące koszty obliczeń oraz wielodostępny, interakcyjny system pracy,
- pewne braki istniejących języków symulacyjnych występujące przy próbach symulacji złożonych algorytmów hybrydowych, skłoniły do podjęcia próby stworzenia języka symulacyjnego, który umożliwiałby całkowicie cyfrową symulację zadań rozwiązywanych dotychczas za pomocą systemów hybrydowych, tj. sprzężonych ze sobą maszyn: analogowej i cyfrowej.

2. Opis języka symulacyjnego LAHYSS

Język symulacyjny LAHYSS został opracowany zgodnie z wymaganiami sformułowanymi przez The SCi Simulation Software Committee /[29]/ i zapewnia:

1/ prostą, jasną postać zdań opisujących modelowany układ,
2/ możliwość prowadzenia obliczeń hybrydowych,
3/ rozszerzoną możliwość współpracy z urządzeniami wejścia-wyjścia EMC,
4/ możliwość prowadzenia obliczeń w trybie konwersacyjnym /interakcyjnym/,
5/ możliwość budowy programu zorientowanego na blokowy lub równaniowy opis metod badanego układu dynamiczne-

go,

6/ możliwość współpracy z językiem algorytmicznym FORTRAN,

7/ wbudowane mechanizmy całkowania i ustalania kolejności obliczeń,

8/ kompletną diagnostykę programu użytkowego,

9/ dużą elastyczność programowania.

Przy opracowaniu języka szczególną uwagę zwrócono na zagadnienie: możliwości symulacji algorytmów typowych dla modelowania hybrydowego oraz współpracy z urządzeniami peryferyjnymi EMC PDP, pozwalającej na bogatą dokumentację prowadzonych obliczeń. Język LAHYSS przewiduje również prowadzenie obliczeń w trybie konwersacyjnym /interakcyjnym/, który pozwala na bezpośrednią współpracę użytkownika z programem. Praca w trybie konwersacyjnym jest niezwykle przydatna do prowadzenia obliczeń w przypadkach, gdy zachodzi konieczność częstego podejmowania decyzji dotyczących dalszego przebiegu obliczeń w zależności od uzyskanych już wyników.

Przed przystąpieniem do omówienia zasad programowania w języku LAHYSS wyjaśnimy jeszcze rolę języka algorytmicznego FORTRAN, który jest językiem związanym dla języka LAHYSS. Język FORTRAN pełni trzy podstawowe funkcje.

Po pierwsze, translacja programów napisanych w języku LAHYSS na kod wewnętrzny EMC polega, w początkowej fazie, na tłumaczeniu tych programów na ciąg instrukcji języka FORTRAN. Funkcja ta jest dla użytkownika "niewidoczna", przebiega bowiem automatycznie.

Po drugie, język FORTRAN służy do zapisu części numerycznej /cyfrowej/ modelowanego problemu hybrydowego.

Po trzecie, język FORTRAN pełni funkcję "dydaktyczną". Przy omawianiu zasad programowania w języku LAHYSS przyjmujemy założenie, że użytkownik tego języka zna zasady programowania w FORTRAN-ie. Założenie to zwalnia nas z konieczności omawiania podstawowych zasad programowania, które są wspólne dla języków LAHYSS i FORTRAN. Dlatego też, przy omawianiu języka LAHYSS, w wielu miejscach, zamiast ścisłych definicji powoływać się będziemy na analogie z językiem FORTRAN.

2.1. Ogólne zasady programowania w języku LAHYSS

Na wstępie przedstawimy przykład programu użytkowego napisanego w języku LAHYSS. Na rys. 2.1. przedstawiono tabulogram programu symulacji liniowego układu mechanicznego o jednym stopniu swobody opisanego równaniem różniczkowym:

$$m\ddot{x} + l\dot{x} + kx = P\sin(\omega t), \text{ z warunkami początkowymi:}$$

$$x(0) = C ,$$

$$\dot{x}(0) = C .$$

Program wyznacza przebiegi czasowe przemieszczenia x i prędkości $\dot{x}$ w przedziale czasowym $0 \leq t \leq 50s$, dla różnych wartości częstości wymuszenia ω = 0.5, 1.0, 1,5 rd/s, przy zadanych parametrach układu:

masa - m = 1.0 kg,

współczynnik tłumienia wiskotycznego - l = 0.1 Ns/m,

stała sprężyny - k = 1 N/m,

amplituda harmonicznej siły wymuszającej - P = 1 N.

Przykład pozostawimy narazie bez komentarza, powracając do niego w dalszej części rozdziału. Należy tylko wyjaśnić, że liczby po prawej stronie każdego wiersza programu zostały wprowadzone sztucznie w celu numeracji poszczególnych wierszy programu i w rzeczywistym programie nie występują. Na numery te będziemy się powoływać przy wyjaśnianiu zasad programowania w języku LAHYSS.

2.1.1. Struktura programu w języku LAHYSS

Program użytkowy w języku LAHYSS ma strukturę blokową (rys.2.2.). Składa się on z nagłówka oraz trzech segmentów: SYSTEM, NUMERICAL i CONTROL, w których opisane są oba podprogramy analogowy i cyfrowy modelowanego programu hybrydowego oraz sposób sterowania obliczeniami.

Nagłówek (wiersz 001) programu z rys.2.1 jest pierwszym wierszem programu i służy do identyfikacji programu i wyników symulacji.

```
PROGRAM(MASA-TLUMIK-SPREZYNA)                                    001
SYSTEM                                                           002
C       DEKLARACJE SYGNALÓW I PARAMETRÓW                         003

        ANALOG      X,DX,D2X,PWYM                                004
        PARAMETER   M,L,K,P OMEGA                                005

DYNAMIC                                                          006

C       OPIS BADANEGO UKLADU                                     007

        X=INT(DX,.0)                                             008
        DX=INT(D2X,.0)                                           009
        PWYM=GS1N(P,OMEGA)                                       010
        D2X=AREX((PWYM-L*DX-K*X)/M)                              011

PERIPHERAL                                                       012

C       DEKLARACJA ELEMENTU REJESTRUJACEGO                       013

        REX7(1,X,DX)                                             014

CONTROL                                                          015

C       METODA I KROK CALKOWANIA                                 016

        RUNGE(4,0.01)                                            017

C       KROK REJESTRACJI                                         018

        ACCESS(1,0.05)                                           019

C       ZADANIE PARAMETROW STALYCH                               020

        LET(M,1.0)                                               021
        LET(L,0.1)                                               022
        LET(K,1.0)                                               023
        LET(P,1.0)                                               024

C       OBLICZENIA DLA ROZNYCH OMEGA                             025

        LET(OMEGA,0.5)                                           026
        RUN(50.0)                                                027
        LET(OMEGA,1.0)                                           028
        RUN(50.0)                                                029
        LET(OMEGA,1.5)                                           030
        RUN(50.0)                                                031

        FINISH                                                   032
END                                                              033
```

Rys. 2.1. Przykład programu napisanego w języku LAHYSS.

Segment SYSTEM (wiersze 002 - 014) odpowiada programowi analogowemu i podzielony jest na dwie sekcje: DYNAMIC i PERIPHERAL. W sekcji DYNAMIC (wiersze 006 - 011) , odpowiadającej blokowi operacji analogowych opisany jest model badanego układu dynamicznego. W sekcji PERIPHERAL (wiersze 012 - 014) zadeklarowane są wszystkie urządzenia peryferyjne współpracujące z modelami układu w sposób ciągły. Zwracamy uwagę, że przed zapisem obu sekcji segmentu należy zadeklarować wszystkie użyte w segmencie sygnały i parametry (wiersze 004 - 005) .

Segment NUMERICAL zawiera podprogram cyfrowy programu hybrydowego, zapisany w języku algorytmicznym FORTAN. Wspomnieliśmy już, że wiele programów hybrydowych może być zrealizowane bez programu cyfrowego. Takim właśnie programem jest program przedstawiony na rys.2.1., stąd brak w nim segmentu NUMERICAL. Segment składa się z dwóch procedur: EXEC i HOLD, może również zawierać procedury własne użytkownika, wykorzystywane w procedurach EXEC i HOLD. W procedurze EXEC zapisany jest algorytm badania układu dynamicznego, na który składają się instrukcje języka FORTRAN oraz rozkazy sterujące modelem zapisanym w segmencie SYSTEM. W procedurze HOLD opisany jest sposób obsługi przerwań programowych powstających w trakcie wykonywania programu.

W segmencie CONTROL opisany jest sposób sterowania obliczeniami hybrydowymi. Sterowanie może się odbywać bezpośrednio z segmentu CONTROL (wiersze 015 - 032) , wówczas podejmowane są działania wynikające z kolejnych rozkazów sterujących zapisanych w segmencie. Przy tym sposobie sterowania nie jest możliwe modelowanie złożonych algorytmów hybrydowych, ponieważ rozkazy wykonywane są kolejno i nie jest możliwy powrót do rozkazów już wykonanych. Sterowanie z segmentu CONTROL stosowane jest do sterowania programów, które nie zawierają programu cyfrowego oraz przy uruchamianiu i testowaniu pełnych programów hybrydowych. Dodajmy jeszcze, że rozkazy sterujące,przy podanym wyżej sposobie sterowania, mogą być również wyprowadzane ręcznie z terminala maszyny PDP. W tym celu należy w segmencie CONTROL umieścić rozkaz MANUAL, przekazujący sterowanie z segmentu do terminala.

Drugim sposobem sterowania jest sterowanie programem analo-

```
PROGRAM ( name )
SYSTEM
        Deklaracje sygnałów i parametrów
DYNAMIC
        Opis badanego układu dynamicznego
PERIPHERAL
        Opis urządzeń peryferyjnych
CONTROL
        Sterowanie obliczeń
NUMERICAL
        SUBROUTINE EXEC
        Opis algorytmu badania układu
        END
        SUBROUTINE HOLD
        Opis obsług. przerwań
        END
        Procedury własne użytkownika
END
```

Rys. 2.2. Struktura programu w języku LAHYSS.

gowym przez program cyfrowy procedury EXEC segmentu NUMERICAL. Ten sposób sterowania stosowany jest przy modelowaniu złożonych algorytmów hybrydowych. Przekazanie sterowania programowi cyfrowemu następuje rozkazem DIGITAL umieszczonym w segmencie CONTROL.

Niezależnie od sposobu sterowania z terminala, z segmentu CONTROL, z programu cyfrowego , można używać wszystkich rozkazów sterujących, które przedstawione będą w p.2.4.

2.1.2. Zdania języka LAHYSS.

Program użytkowy napisany w języku LAHYSS składa się ze zdań (statement) pisanych w oddzielnych wierszach programu. W języku LAHYSS istnieją następujące typy zdań / w nawiasach podano numery wierszy programu z rys.2.1. odpowiadających poszczególnym typom zdań / :

A. Dyrektywy / 001, 002, 006, 012, 015, 033 / .

Dyrektywy służą do sterowania translacją programu napisanego w języku LAHYSS na język FORTRAN, jednocześnie oddzielają od siebie poszczególne segmenty i sekcje programu. Dyrektywa składa się ze słowa sterującego i ewentualnie parametrów.

B. Deklaracje / 004, 005 / .

Deklaracje służą do opisu zmiennych użytych w segmencie SYSTEM. Deklaracja składa się ze słowa określającego typ zmiennych i listy nazw zmiennych.

C. Instrukcje strukturalne / 008 - 011, 014 / .

Zdania tego typu służą do zapisu elementów liczących, z których zbudowany jest model badanego układu dynamicznego, a także do deklaracji urządzeń peryferyjnych współpracujących z modelem. Postać instrukcji strukturalnych jest ściśle zależna od operacji jaką dany element wykonuje i będą szczegółowo omówione w p. 2.3.

D. Instrukcje rozkazy sterujące / 017, 019, 021 - 024, 026 - 032 /.

Rozkazy służą do sterowania układu opisanego w segmencie SYSTEM, a także do sterowania obliczeniami symulacyjnymi w segmencie ogólnym / np. wybór metody całkowania numerycznego / .

Instrukcja sterująca składa się z nazwy określającej rodzaj rozkazu i odpowiednich parametrów.

E. Instrukcje języka związanego FORTRAN

Instrukcji FORTRAN-u można używać tylko w segmencie NUMERICAL. W zasadzie dostępne są wszystkie konstrukcje tego języka, włącznie z deklaracjami zmiennych. Pewne ograniczenia wynikające z organizacji programu realizującego obliczenia symulacyjne będą omówione w pkt 2.5.

F. Komentarze / 003, 007, 013, 016, 018, 020, 025 / .

Zdania komentarza są pomijane w trakcie tłumaczenia programu. Służą one jedynie celom poglądowym, do dodatkowego, słownego opisu struktury i działania programu użytkowego. Zasady zapisu komentarzy w języku LAHYSS są identyczne do zasad obowiązujących w FORTRAN-ie.

2.1.3. Zasady zapisu programu w języku LAHYSS

Programowanie w języku LAHYSS charakteryzuje się dużą elastycznością. Dotyczy ona zarówno sposobów realizacji modelowanego algorytmu hybrydowego a także kolejność zapisu poszczególnych zdań programu. Pierwsza z wymienionych cech omówiona będzie w rozdziale 3, w tym punkcie skupimy się nad formalną stroną zapisu programu użytkowego.

Podana na rys.2.2. struktura programu w języku LAHYSS dotyczy pełnego programu hybrydowego, nie jest jednak sztywna. Przy zapisie programów użytkowych należy przestrzegać następujące zasady ogólne:

1/ Program rozpoczyna się nagłówkiem - dyrektywą PROGRAM, zaś kończy dyrektywą END,

2/ Kolejność zapisu poszczególnych segmentów programu jest w zasadzie obojętna, niemniej zalecamy następującą kolejność segmentów:

SYSTEM
CONTROL
NUMERICAL

3/ Poszczególne segmenty programu rozpoczynają się dyrektywą - SYSTEM, CONTROL, NUMERICAL, zaś kończą dyrektywą

rozpoczynającą następny segment lub dyrektywą END.

4/ W programie może występować najwyżej jeden segment danego typu.

5/ Dla prawidłowego działania programu konieczne są segmenty: SYSTEM i CONTROL. Segment NUMERICAL może być pominięty.

6/ Zdania komentarza mogą być umieszczane w dowolnych miejscach.

7/ Poszczególne zdania programu należy zapisywać w odpowiednim formacie. Formaty te zostaną opisane przy omawianiu poszczególnych segmentów programu (pp. 2.2., 2.3., 2.4., 2.5) .

Przy zapisie segmentu SYSTEM obowiązują następujące zasady ogólne:

1/ Deklaracje nazw sygnałów są konieczne i muszą poprzedzać opis badanego modelu.

2/ Kolejność zapisu poszczególnych sekcji jest obojętna, niemniej zalecamy kolejność: DYNAMIC, PERIPHERAL.

3/ Kolejność zapisu poszczególnych instrukcji strukturalnych w sekcjach segmentu jest obojętne, bowiem translator automatycznie sortuje poszczególne instrukcje w celu zapewnienia właściwej kolejności obliczeń.

4/ W segmencie nie muszą występować jednocześnie obie sekcje. Np. można zrezygnować z sekcji PERIPHERAL, a wyniki wyprowadzać przez segment NUMERICAL.

Przy zapisie segmentu NUMERICAL należy pamiętać, że:

1/ Kolejność zapisu poszczególnych procedur jest obojętna.

2/ Kolejność zapisu i wykonywania instrukcji wewnątrz procedur jest ściśle określona i wynika z przyjętego algorytmu badania układu dynamicznego.

Podane wyżej zasady programowania w języku LAHYSS mają charakter ogólny. Jak widać, zasady te nie stanowią istotnych

ograniczeń dla programisty, a nawet ułatwiają programowanie dzięki:

1/ dużej dowolności w kolejności zapisu poszczególnych zdań i fragmentów programu,

2/ możliwości pominięcia fragmentów niepotrzebnych z punktu widzenia modelowanego algorytmu.

2.2. Podstawowe elementy języka LAHESS

2.2.1. Zbiór znaków

Zbiór znaków (alfabet) używanych do kodowania programów w języku LAHYSS obejmuje:

1. Litery: A, B, C, ... , X, Y, Z.
2. Cyfry: 0, 1, ... , 9.
3. Znaki:
   - operatory: * , + , - , / , =
   - separatory: (,) , , (przecinek) , . (kropka dziesiętna) , '(apostrof),
   - znaki specjalne, dostępne na EMC PDP 11/70.

2.2.2. Stałe i zmienne

Wielkości występujące w programie użytkowym w LAHYSS-ie mogą być reprezentowane dwojako:

1/ stałe - wielkości reprezentowane przez wartość,

2/ zmienne - wielkości reprezentowane przez nazwę.

Zasady zapisu stałych są identyczne do zasad zapisu stosowanych w FORTRAN-ie. W zależności od kontekstu, stałe użyte w programie dotyczą wielkości ciągłych występujących w segmencie SYSTEM lub wielkości numerycznych występujących w segmencie NUMERICAL. Przypominamy, że stałe pod względem postaci zapisu dzielą się na: całkowite INTEGER , rzeczywiste REAL, DOUBLE PRECISION , logiczne LOGICAL , tekstowe literały .

Typy zmiennych

Zmienne języka LAHYSS dzielą się na następujące typy:

A. Zmienne ciągłe

Zmienne ciągłe interpretowane są jako sygnały napięciowe i nastawy /parametry/ elementów z których zbudowany jest model badanego układu dynamicznego. Dzielą się one na:

- sygnały typu ANALOG - odpowiadające sygnałom na wyjściach analogowych elementów liczących, zmieniających się w trakcie symulacji w sposób "ciągły",
- sygnały typu LOGIC - odpowiadające sygnałom na wejściach elementów logicznych i sterujących; zmienne tego typu mogą przybierać dwie wartości: "1" i "0" logiczne w dowolnej chwili czasu,
- parametry zmienne typu PARAMETER - odpowiadające nastawom poszczególnych elementów; ich wartość nie zmienia się w trakcie pojedyńczego przebiegu symulacji; zmian parametrów dokonuje się w segmencie CONTROL lub NUMERICAL.

Ze względu na fakt, że cały proces symulacji odbywa się na maszynie cyfrowej, zmienne ciągłe w rzeczywistości reprezentowane są przez zmienne numeryczne odpowiednich typów wg tabeli:

Zmienna ciągła	Reprezentacja maszynowa
ANALOG	DOUBLE PRECISION
LOGIC	LOGICAL
PARAMETER	DOUBLE PRECISION

Zmienne ciągłe mogą być wyłącznie zmiennymi prostymi w sensie FORTRAN-u.

B. Zmienne numeryczne

Zmienne numeryczne to zmienne występujące w podprogramie cyfrowym programu hybrydowego / segmencie NUMERICAL/ . Ich interpretacja jest identyczna jak zmiennych w języku FORTRAN.

Zmienne numeryczne mogą być następujących typów:

- INTEGER
- REAL
- DOUBLE PRECISION
- LOGICAL
- COMPLEX.

Zmienne numeryczne mogą być zarówno zmiennymi prostymi jak i indeksowanymi.

Zasady ogólne dotyczące zmiennych

1. W języku LAHYSS należy zadeklarować wszystkie zmienne ciągłe. Deklaracji dokonuje się w segmencie SYSTEM (p. 2.3.1)

2. Zmienne numeryczne deklarowane są zgodnie z zasadami FORTRAN-u w segmencie NUMERICAL (p. 2.4.).

3. Obszarem działania zmiennych numerycznych są procedury segmentu NUMERICAL, w których zostały zadeklarowane. Wymiana wartości tych zmiennych między procedurami następuje przez parametry procedur lub przez obszary wspólne dyrektywa COMMON języka FORTRAN .

4. Obszarem działania zmiennych typu ciągłego jest cały program tzn. mogą być one używane (ich wartości są przekazywane automatycznie) zarówno w segmencie SYSTEM jak i NUMERICAL. W segmencie NUMERICAL zmienne ciągłe traktowane są jak zmienne numeryczne, zgodnie z przedstawioną już tabelą.

5. Nazwy zmiennych numerycznych i ciągłych muszą spełniać zasady języka FORTRAN:

- nazwa może zawierać conajwyżej 8 znaków alfanumerycznych, tzn. litery A - Z i cyfry 0 - 9,
- nazwa musi rozpoczynać się od litery, np.

L3, A, X, D2X,

natomiast nazwy:

1X2 - rozpoczyna się cyfrą,
X2/1 - zawiera operator,
ZMIENNA - zawiera za dużo znaków,

są błędne.

6. W języku LAHYSS istnieją następujące nazwy zastrzeżone dla zmiennych posiadających stałe znaczenie:

DT - zmienna numeryczna (typ DOUBLE PRECISION) : krok całkowania w użytej metodzie całkowania numerycznego,

T - zmienna ciągła (typ ANALOG) : czas rzeczywisty w badanym układzie (zmienna niezależna) , automatycznie uaktualniana w trakcie symulacji,

TRUE, FALSE - w zależności od kontekstu zmienne numeryczne (typ LOGICAL) lub ciągłe (typu LOGIC) ,

PRE - zmienna ciągła typ LOGIC : przyjmuje wartość TRUE w przypadku, gdy układ dynamiczny jest w stanie PREPARE (warunki początkowe) ,

SOL - zmienna ciągła (typ LOGIC) : przyjmuje wartość TRUE w przypadku, gdy układ dynamiczny jest w stanie SOLVE (rozwiązywanie) ,

HLD - zmienna ciągła (typu LOGIC) : przyjmuje wartość TRUE w przypadku przerwania obliczeń analogowych,

TERM - zmienna ciągła (typ LOGIC) : przyjmuje wartość TRUE w przypadku zakończenia pojedynczego przebiegu symulacji.

2.2.3. Dyrektywy języka

Jak już powiedzieliśmy dyrektywy służą do sterowania translacją programów napisanych w języku LAHYSS. Ogólna postać dyrektywy:

słowo sterujące [(parametry)]

Zwracamy uwagę,że dyrektywy zapisuje się począwszy od pierwszej kolumny wiersza zdania , w odróżnieniu od pozostałych zdań języka LAHYSS, które zapisujemy tak, jak w FORTRAN-ie (tzn. co najmniej od 7 kolumny) .

W języku LAHYSS istnieją następujące dyrektywy sterujące:

PROGRAM (nazwa programu) - oznacza początek programu użytkowego, ciąg znaków zawarty w nawiasie zapewnia identyfikację programu.

SYSTEM - oznacza początek segmentu SYSTEM,
CONTROL - oznacza początek segmentu CONTROL,
NUMERICAL - oznacza początek segmentu NUMERICAL,
DYNAMIC - oznacza początek sekcji DYNAMIC w segmencie SYSTEM,
PERIPHERAL - oznacza początek sekcji PERIPHERAL segmantu SYSTEM.
END - oznacza koniec programu napisanego w LAHYSS-ie.

2.2.4. Elementy liczące

Elementy liczące to odpowiedniki bloków hybrydowej maszyny analogowej, za pomocą których modelowany jest badany układ dynamiczny. Opisu elementów liczących dokonuje się w sekcji DYNAMIC segmentu SYSTEM (p.2.3.2) przy pomocy instrukcji strukturalnych języka LAHYSS. W języku LAHYSS można stosować następujące elementy liczące:

A. Analogowe elementy liczące - to elementy, w których sygnał wyjściowy jest sygnałem typu ANALOG, niezależnie od typów sygnałów podawanych na wejścia elementów. Dzielą się one z kolei na 4 grupy:

Elementy dynamiczne

INT - integrator,
INTR - integrator sterowany,
AMEM - pamięć analogowa,
DERIV - element różniczkujący,
DELAY - element opóźniający,
XMAX, XMIN - elementy pomiarowe,

Elementy operacji algebraicznych

INW - inwentor,
SUM - sumator,
POT - potencjometr,
MULT - mnożarka,
DIV - dzielarka,
POW - podnoszenie do potęgi.

Operacje wykonywane przez te elementy mogą być również zapisane w postaci wyrażeń arytmetycznych zapisanych zgodnie z zasadami FORTRAN-u z pomocą elementu:

AREX - wyrażenie arytmetyczne.

Pozostałe elementy algebraiczne to:

ALGEB- pętla algebraiczna,
FTAB - funkcja zadana w postaci tablicy.

Elementy funkcji matematycznych

SQRT - pierwiastek kwadratowy,
EXP - funkcja eksponencjalna,
LN - logarytm naturalny,
LOG - logarytm dziesiętny,
ABS - moduł,
SIGN - znak,
FIX - obcięcie,
SIN - sinus,
COS - cosinus,
TAN - tangens,
ARSIN -
ARCOS - } funkcje odwrotne funkcji SIN, COS, TAN,
ARTAN -
SINH - sinus hiperboliczny,
COSH - cosinus hiperboliczny,

Elementy specjalne i hybrydowe

DIP - dioda,
DIN - dioda odwrócona,
LIM - ograniczenie,
DEHDSP - strefa nieczułości,
HIST - histereza,
SWI - klucz elektroniczny,
DSWI - podwójny klucz elektroniczny,
PRZ - przekaźnik trójpołożeniowy.

Generatory funkcji czasu

GSTEP - generator skoku jednostkowego
GSIN - generator sinusoidy

GCOS - generator cosinusoidy,
GRECT - generator fali prostokątnej,
GTRIAL - generator fali piłokształtnej,
GRAND - generator liczb losowych rozkład równomierny,
GAUSS - generator liczb losowych rozkład normalny.

B. Elementy logiczne - to elementy, w których sygnał wyjściowy jest typu LOGIC, niezależnie od typów wejść:

Elementy pamięci logicznej:

RS - przerzutnik typu R-S,
JK - przerzutnik typu J-K,
DE - przerzutnik typu D,
DERL - logiczny element różniczkujący.

Elementy operacji logicznych:

NOT - zaprzeczenie /negacja/,
OR - suma logiczna /alternatywa/,
AND - iloczyn logiczny /koniunkcja/ ,
NOR - negacja alternatywy,
NAND - negacja koniunkcji.

Komparator -

COMP - precyzyjny komparator elektroniczny

Generatory -

GEN - generator impulsów,
TPER - generator przedziałów czasowych.

C. Elementy sterujące - to elementy pozwalające na przerwanie obliczeń, których sygnałem wyjściowym jest zmienna zarezerwowana.

HOLD - element realizujący przerwanie obliczeń analogowych i wywołanie procedury obsługi przerwań,
TERMIN - element powodujący zakończenie pojedynczego przebiegu symulacyjnego przed upływem zadanego czasu.

Postać instrukcji strukturalnych, działanie oraz sygnały

wejściowe elementów liczących omówione będą dokładnie w p.2.3.

2.2.5. Urządzenia peryferyjne

Urządzenia peryferyjne służą do rejestracji wyników symulacji w sposób "ciągły". Ciągłość jest tu pojęciem symbolicznym, ponieważ w języku LAHYSS wszystkie przebiegi są rejestrowane na fizycznych urządzeniach wyjściowych EMC w sposób dyskretny. Pojęcie to służy do podkreślenia różnicy pomiędzy rejestracją wyników w trakcie przebiegów symulacyjnych w systemie hybrydowym dokonywaną na oscyloskopie i rejestratorach , a rejestracją wyników pomiędzy przebiegami i wydrukami dokonywanymi przez podprogram cyfrowy na urządzeniach EMC drukarka, monitor . Urządzenia peryferyjne współpracujące z badanym układem dynamicznym opisywane są w sekcji PERIPHERAL segmentu SYSTEM (p.2.3.3) za pomocą instrukcji strukturalnych.

W języku LAHYSS istnieją następujące urządzenia rejestrujące:

Rejestracja sygnałów analogowych

OUTPUT - rejestracja sygnałów analogowych w pamięci EMC bez wyprowadzania wyników (odpowiednik magnetofonu - ZAPIS)

REXT - rejestracja sygnałów analogowych i wyprowadzenie wyników w postaci wykresów czasowych (rejestrator XT) ,

REXY - rejestracja sygnałów analogowych i wyprowadzenie wyników w postaci wykresów na płaszczyźnie XY (rejestrator XY) ,

TABLE - rejestracja sygnałów analogowych i wyprowadzenie wyników w postaci tabeli,

TYPE - bezpośrednia rejestracja sygnałów analogowych z natychmiastowym wyprowadzeniem wyników na terminal użytkownika.

Rejestracja sygnałów logicznych

OUTLOG - rejestracja sygnałów logicznych i wyprowadzenie wyników w postaci tabeli,

SIGLOG - sygnalizacja zmiany stanu sygnałów logicznych z natychmiastowym wyprowadzeniem wyników na terminal użytkownika.

W języku LAHYSS przewidziana jest również możliwość wprowadzania sygnałów analogowych do układu dynamicznego /odpowiednik magnetofonu - ODCZYT/ w sposób ciągły poprzez element:

INPUT.

Rejestracja wyników i wydruki przez program cyfrowy dokonywane są zgodnie z zasadami FORTRAN-u.

2.2.6. Rozkazy sterujące

Rozkazy sterujące języka LAHYSS służą do:

Sterowanie stanami pracy maszyny analogowej:

PREPARE	- wprowadzenie warunków początkowych,
SOLVE	- całkowanie,
COMPUTE	- wykonanie kolejno rozkazów PREPARE, SOLVE,
RUN	- jak rozkaz COMPUTE, z równoczesną inicjacją urządzeń peryferyjnych i wyprowadzeniem wyników symulacji,
ITER	- obliczenia iteracyjne.

Sterowanie pojedyńczych elementów liczących:

INIT	- nadanie wartości sygnałowi typu ANALOG,
LET	- nadanie wartości sygnałowi typu PARAMETER,
SET	- zerowanie sygnału typu LOGIC,
RESET	- ustawianie sygnału typu LOGIC.

Sterowanie urządzeń peryferyjnych

OPENR	- otwarcie dostępu do urządzenia,
ACCESS	- otwarcie dostępu do urządzenia i ustalenie kroku rejestracji,
CLOSER	- zamknięcie dostępu do urządzeń,
EDIT	- wyprowadzenie wyników rejestracji na drukarkę,
PLOT	- wyprowadzenie wyników rejestracji na graph-pletter,
DELETE	- kasowanie zarejestrowanych wyników,

BACK - powrót na początek zarejestrowanych wyników /przewinięcie/ ,

COPY - kopiowanie wyników rejestracji z jednego urządzenia na drugie,

EXCH - zamiana wyników rejstracji między urządzeniami,

APPEND - dołączanie wyników rejestracji jednego urządzenia do drugiego,

SCALE - skalowanie urządzeń do rejestracji wyników w postaci wykresów REXT i REXY .

Ponadto do słownego opisu wydruków i wykresów:

MEAD - wyprowadzenie nagłówka,

TEXT - wyprowadzenie opisu słownego.

Rozkazy specjalne:

NEWRUN - inicjacja wszystkich urządzeń peryferyjnych,

ENDRUN - wyprowadzenie wyników symulacji na wszystkich urządzeniach peryferyjnych,

CONTI - kontynuacja obliczeń analogowych w przypadku przerwania przez element HOLD,

ABORT - kasowanie wszystkich wyników symulacji,

FINISH - zakończenie obliczeń symulacyjnych.

Rozkazy sterujące sposobem wykonywania obliczeń:

RUNGE - wybór metody i kroku całkowania metody numerycznej,

DIGITAL - przekazanie sterowania obliczeń procedurze EXEC segmentu NUMERICAL,

MANUAL - sterowanie "ręczne" bezpośrednio z terminala EMC.

Rozkazy sterujące mogą być używane zarówno w segmencie CONTROL jak i NUMERICAL. Mogą być one również wprowadzane przez użytkownika z terminala EMC. W przypadku sterowania z segmentu CONTROL lub "ręcznego" format rozkazów jest następujący:

rozkaz [(parametry)]

gdzie:

rozkaz - nazwa rozkazu sterującego

parametry - wymagane parametry dla danego rozkazu.

W przypadku sterowania z segmentu NUMERICAL postać rozkazu sterującego jest następująca:

CALL rozkaz [(parametry)] ,

zatem rozkaz sterujący równoznaczny jest z wywołaniem odpowiedniej procedury programu wykonującego obliczenia symulacyjne.

Dokładne omówienie poszczególnych rozkazów sterujących znajdzie czytelnik w p.2.4., natomiast wskazówki dotyczące stosowania rozkazów w segmencie NUMERICAL w p.2.5.

2.3. Zasady zapisu segmentu SYSTEM.

Segment SYSTEM podzielony jest na trzy sekcje /rys.2.2/: deklaracji sygnałów i parametrów użytych do opisu modelu badanego układu dynamicznego /p.2.3.1/; DYNAMIC /p.2.3.2/, opisującą strukturę modelu oraz PERIPHERAL /p.2.3.3/, w której zadeklarowane są urządzenia peryferyjne, służące do ciągłej rejestracji wyników symulacji.

Segment rozpoczyna się słowem sterującym SYSTEM, zaś kończy słowem rozpoczynającym kolejny segment programu użytkowego /NUMERICAL lub CONTROL/.

2.3.1. Deklaracje sygnałów i parametrów.

Bezpośrednio po słowie sterującym SYSTEM, rozpoczynającym segment należy zadeklarować nazwy wszystkich sygnałów i parametrów /zmiennych/ występujących w układzie dynamicznym.

A. Postać deklaracji sygnałów jest następująca:

typ wykaz nazw sygnałów

gdzie:

typ - typ deklarowanych sygnałów /ANALOG lub LOGIC/.
wykaz nazw sygnałów - lista nazw oddzielonych przecinkami.

ANALOG X1, X2, DX1, DX2

LOGIC L3, BAD, GOOD

B. Postać deklaracji parametrów jest następująca:

PARAMETER wykaz nazw parametrów

gdzie:

wykaz nazw parametrów - lista nazw oddzielonych przecinkami

Przykład:

PARAMETER TLUMIK, OMEGA

Przy wyborze i deklarowaniu nazw sygnałów i parametrów należy przestrzegać zasady omówione w p.2.2.2. Przypominamy, że należy zadeklarować wszystkie nazwy użyte do zapisu modelu badanego układu.

2.3.2. Sekcja DYNAMIC.

W sekcji DYNAMIC zapisany jest model badanego układu dynamicznego, odpowiadający ściśle blokowi operacji analogowych opisanych w p.1.1. Sposób budowy modelu układu przystosowany do zapisu w języku LAHYSS przedstawiony będzie w rozdziale 3. W tym punkcie omówimy jedynie formalną stronę zapisu sekcji DYNAMIC, skupiając uwagę na opisie elementów liczących, służących do zapisu modelu.

Sekcja DYNAMIC rozpoczyna się słowem sterującym DYNAMIC, zaś kończy słowem PERIPHERAL, rozpoczynającym kolejną sekcję segmentu SYSTEM /rys.2.2/. Kolejność zapisu poszczególnych elementów nie jest istotna, ponieważ translator automatycznie ustala kolejność wykonywania poszczególnych operacji.

2.3.2.1. Analogowe elementy liczące.

Pod tym pojęciem rozumieć będziemy elementy liczące, w których sygnał wyjściowy jest sygnałem analogowym /ANALOG/, niezależnie od typów sygnałów podawanych na wejściu elementu.

Ogólna postać deklaracji analogowego elementu liczącego jest następująca:

sa = ael (wykaz sygnałów i parametrów)

gdzie:

sa - nazwa sygnału wyjściowego, uprzednio zadeklarowana dyrektywą ANALOG,

ael - nazwa analogowego elementu liczącego,

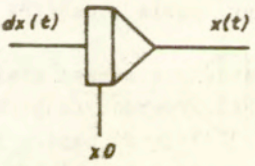
wykaz sygnałów i parametrów - lista sygnałów i parametrów podawanych na wejścia elementu, oddzielonych przecinkami; typy poszczególnych sygnałów muszą odpowiadać typom wejść danego elementu.

A. Elementy dynamiczne.

Do elementów tych zaliczamy elementy realizujące operacje których wynik zależy zarówno od stanu wejść, jak i stanu poprzedniego elementu. Do elementów tych należą: elementy całkujące /INT, INTR/, pamięci analogowej /AMEM/, element różniczkujący /DERIV/, element opóźniający /DELAY/ oraz elementy pomiarowe /XMAX, XMIN/.

A1. Integrator - INT.

Na rys.2.3 pokazane jest oznaczenie /symbol/ elementu używane przy sporządzaniu schematów analogowych.



Rys.2.3. Integrator - INT.

a. Postać deklaracji:

x = INT (dx, xØ)

gdzie:

x - sygnał typu ANALOG,

dx, xØ - zmienna typu ANALOG lub PARAMETER, np.:

X = INT (DX, XP) .

b. Działanie:

Element realizuje operację całkowania:

$$x(t) = \int_{tp}^{tk} dx(t) \cdot dt + x\emptyset$$

gdzie:

tp, tk - przedział całkowania zadawany rozkazami sterującymi,

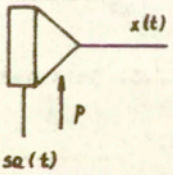
dt - krok całkowania.

Całkowanie realizowane jest numerycznie metodami /do wyboru : prostokątów, Runge-Kutta 2 lub Runge-Kutta 4. Wyboru metody oraz określenie kroku całkowania dokonuje się rozkazem RUNGE.

Całkowanie odbywa się w stanie SOLVE modelu układu, zaś warunki początkowe wprowadzane są w stanie PREPARE.

A2. Pamięć analogowa - AMEM.

Symbol elementu pokazano na rys.2.4.



Rys.2.4. Pamięć analogowa - AMEM

a. Postać deklaracji:

x = AMEM (sa, P)

gdzie:

x, sa - sygnały typu ANALOG,

p - sygnał ty pu LOGIC, np.:

AX = AMEM (XW, L2) .

b. Działanie:

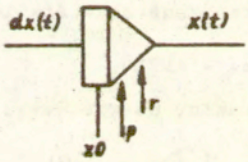
Element realizuje operację opisaną wzorem:

$$x(t) = \begin{cases} sa(t), & \text{jeśli sl = .TRUE.} \\ x_p, & \text{jeśli sl = .FALSE.} \end{cases}$$

Element śledzi sygnał na wejściu sa w przypadku gdy sygnał logiczny sl ma wartość TRUE i zapamiętuje ostatnią wartość sygnału sa przy zmianie sygnału sl na FALSE. Zapamiętana wartość xp przechowywana jest do chwili, gdy sygnał logiczny ponownie osiąga wartość TRUE.

A3. Integrator sterowany - INTR.

Symbol elementu pokazano na rys.2.5.



Rys.2.5. Integrator sterowany - INTR

a. Postać deklaracji:

X = INTR (dx, xØ, r, p)

gdzie:

x - sygnał typu ANALOG,

dx, xØ - zmienne typu ANALOG lub PARAMETER,

r, p - sygnały typu LOGIC, np.:

X = INT (V, XP, SOLVE, PREPARE) .

b. Działanie:

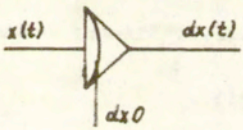
Element realizuje 3 różne operacje w zależności od stanu wejść logicznych r i p :

$$x(t) = \begin{cases} \int_{t_p}^{t_k} dx(t) \cdot dt + x\emptyset, & \text{jeśli } r = \text{TRUE i } p = \text{FALSE} \\ x\emptyset, & \text{jeśli } p = \text{TRUE} \\ x_p, & \text{jeśli } r = \text{FALSE i } p = \text{FALSE} \end{cases}$$

Element działa zatem jak integrator /INT, p.A1./ w przypadku, gdy r = TRUE i p = FALSE, natomiast w przypadku, gdy r = FALSE pracuje jak pamięć analogowa /AMEM, p.A2/.

A4. Element różniczkujący - DERIV.

Symbol elementu pokazano na rys.2.6.



Rys.2.6. Element różniczkujący.

a. Postać deklaracji:

dx = DERIV (x, dxØ)

gdzie:

x, dx - sygnał typu ANALOG

dxØ - zmienne typu ANALOG lub PARAMETER, np.:

DX = DERIV (X,DXP)

b. Działanie:

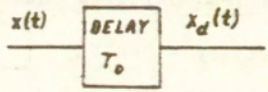
Element realizuje operację różniczkowania:

$$dx(t) = \frac{d\,x(t)}{dt}$$

Ze względu na fakt, że operacja realizowana jest numerycznie, należy również podać początkową wartość pochodnej $dx(0) = dx\emptyset$, która jest podstawiana analogicznie jak warunek początkowy w elemencie całkującym INT, tzn. w stanie PREPARE. Właściwe różniczkowanie odbywa się w stanie SOLVE.

A5. Element opóźniający - DELAY.

Symbol elementu pokazano na rys.2.7.



Rys.2.7. Element opóźniający

a. Postać deklaracji:

xd = DELAY (x, T_o)

gdzie:

xd, x - sygnał typu ANALOG,

T_o - zmienna typu PARAMETER, np.:

XOP = DELAY (XWE, TOPZ).

b. Działanie:

Element realizuje czyste opóźnienie, opisane wzorem:

$$xd(t) = \begin{cases} x(t - T_o) , & \text{dla} \quad t \geqslant T_o \\ 0 , & \text{dla} \quad t < T_o \end{cases}$$

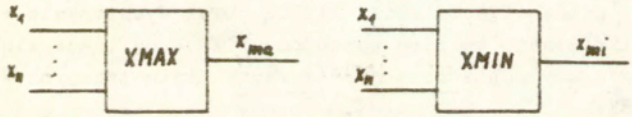
Sygnał na wyjściu elementu xd jest opóźniony o T_o sekund w stosunku do sygnału wejściowego x. Opóźnienie realizowane jest za pomocą automatycznie generowanej tablicy typu DOUBLE PRECISION, przy czym ilość wykorzystanych elementów tej tablicy dla jednego elementu opóźniającego wynosi:

$$n = E\left(\frac{T_o}{h} + .5\right) \quad , \text{ gdzie}$$

h - krok całkowania numerycznego.

A6. Elementy pomiarowe - XMAX i XMIN.

Symbole elementów pokazano na rys.2.8.



Rys.2.8. Elementy pomiarowe

a. Postać deklaracji:

xma = XMAX (x1,...,xn)

lub

xmi = XMIN (x1,...,xn)

gdzie:

xmi, xma , x1,...,xn - sygnały typu ANALOG, przy czym liczba sygnałów wyjściowych $n \leq 6$, np.:

XMAKS = XMAX (X1, XW, 3.0) .

b. Działanie:

Elementy znajdują odpowiednio: maksimum lub minimum sygnałów wejściowych w czasie, wg wzorów:

xma = DMAX1 (xma, x1,...,xn)

i

xmi = DMIN1 (xmi, x1,...,xn) ,

gdzie:

DMAX1, DMIN1 - funkcje standardowe języka FORTRAN.

Zerowanie elementów następuje automatycznie w stanie PREPARE, lub programowo rozkazem INIT.

B. Elementy operacji algebraicznych.

Do elementów tych zaliczamy elementy realizujące podstawowe operacje matematyczne, odpowiadające blokom maszyny analogowej /INW, SUM, POT, MULT, DIV, POW/ . Do grupy tej zaliczamy również elementy wykonujące operacje bardziej złożone: rozwiązywanie pętli algebraicznej ALGEB oraz wyznaczanie wartości funkcji opisanej tablicą wartości / FTAB / , a także element umożliwiający zapis złożonych wyrażeń arytmetycznych w FORTRAN-ie / AREX / .

B1. Inwertor - INW.

Symbol elementu pokazano na rys.2.9.

Rys.2.9. Inwertor

a. Postać deklaracji:

xn = INW (X)

gdzie:

xn, x - sygnały typu ANALOG, np.:

PWYM = INW (PW)

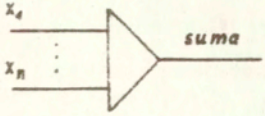
b. Działanie:

Element realizuje zmianę znaku sygnału wejściowego x :

$xx = -x$.

B2. Sumator - SUM.

Symbol sumatora podano na rys.2.10.



Rys.2.10. Sumator.

a. Postać deklaracji

$$suma = SUM(x_1,\ldots,x_n)$$

gdzie:

suma, $x_1,\ldots,x_n$ - sygnały typu ANALOG, przy czym liczba sygnałów wejściowych $n \leqslant 6$, np.:

DELTA = SUM (PX, PWYM)

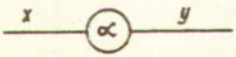
b. Działanie:

Element realizuje operację:

$$suma = \sum_{i=1}^{n} X_i .$$

B3. Potencjometr - POT.

Symbol potencjometru pokazano na rys.2.11.



Rys.2.11. Potencjometr.

a. Postać deklaracji:

y = POT (α, x)

gdzie:

x, y - sygnały typu ANALOG,

α - współczynnik typu PARAMETER, np.:

SILA = POT (MASA, D2X)

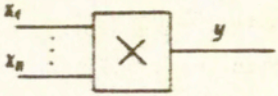
b. Działanie:

Element realizuje operację:

$$y = \alpha \cdot x \ .$$

B4. Mnożarka - MULT.

Symbol mnożarki pokazano na rys.2.12.



Rys.2.12. Mnożarka.

a. Postać deklaracji:

$$y = \text{MULT}\,(x_1, \ldots, x_n)$$

gdzie:

$y, x_1, \ldots, x_n$ - sygnały typu ANALOG, np.

ILOCZ = MULT (X1, X2, X3)

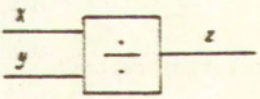
b. Działanie:

Element realizuje operację:

$$y = \prod_{i=1}^{n} x_i \ .$$

B5. Dzielarka DIV.

Symbol elementu pokazano na rys.2.13



Rys.2.13. Dzielarka.

a. Postać deklaracji:

z = DIV (x, y)

gdzie:

x,y,z - sygnały typu ANALOG, np.:

D2X = DIV (PWYM, MASA)

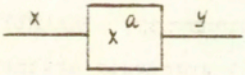
b. Działanie:

Dzielarka wykonuje operację:

$z = x/y$.

B6. Podnoszenie do potęgi - POW.

Symbol elementu pokazano na rys.2.14.



Rys.2.14. Element POW.

a. Postać deklaracji:

y = POW (x, a)

gdzie:

x,y - sygnały typu ANALOG

a - parametr, np.:

Y = POW (X, 2.)

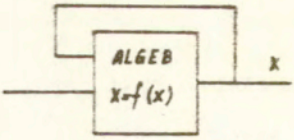
b. Działanie:

Element wykonuje operację:

$y = x^a$.

B7. Rozwiązywanie pętli algebraicznej - ALGEB

Symbol elementu pokazano na rys.2.15.



Rys.2.15. Pętla algebraiczna

a. Postać deklaracji:

x = ALGEB (xØ, f(x) , eps)

gdzie:

x - sygnał typu ANALOG

xØ, eps - zmienne typu PARAMETER

f(x) - wyrażenie arytmetyczne, zapisane w FORTRAN-ie opisujące prawą stronę rozwiązywanego równania, np., instrukcja

X = ALGEB (1.0, A * EXP(-T) + B * SIN(X),.0001)

deklaruje element rozwiązujący równanie:

$x = A\ e^{-t} + B \sin x.$

b. Działanie:

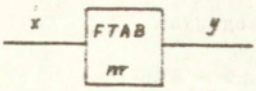
Element rozwiązuje pętlę algebraiczną w postaci:

x = f(x) ,

Takiej zależności nie można modelować w inny sposób, ze względu na działanie algorytmu sortującego. Algorytm ten traktuje pojawienie się tego samego sygnału po obu stronach znaku równości jako błąd /niezamierzona pętla algebraiczna/.
Równanie x = f(x) rozwiązywane jest metodą Newtona-Raphsona. Parametr xØ określa przybliżenie początkowe rozwiązania /wprowadzane w stanie PREPARE/, natomiast eps - błąd graniczny, określający warunek zakończenia obliczeń iteracyjnych.

BB. Funkcja zadana tablicą - FTAB.

Symbol elementu pokazano na rys.2.16



Rys.2.16. Element FTAB.

a. Postać deklaracji:

y = FTAB (nr, n, x)

dane

gdzie:

x, y - sygnały typu ANALOG

nr - numer identyfikacyjny elementu FTAB

n - liczba punktów, w których podane są wartości funkcji, $n \leq 50$,

dane - wartości kolejnych argumentów i wartości funkcji dla tych argumentów, podawane parami w kolejnych wierszach programu; argumenty muszą być uporządkowane w kolejności rosnącej.

Przykład:
Element opisany instrukcją:

```
y = FTAB ( 1, 5, x )
- 1.0, 1.0
- 0.5, 0.25
  0.0, 0.0
  0.5, 0.25
  1.0, 1.0
```

modeluje funkcję:

$y = x^2$.

b. Działanie:

Element oblicza funkcję jednej zmiennej zadaną tablicą

wartości liczbowych. Wartości funkcji obliczane są metodą interpolacji liniowej wprowadzonych danych.
Element FTAB odpowiada uniwersalnemu przetwornikowi diodowemu stosowanemu w maszynach analogowych.

B9. Wyrażenie arytmetyczne - AREX.

Element AREX stosowany jest przy metodzie zapisu równaniowego modelu badanego układu dynamicznego /p.3.1/ i nie ma swojego odpowiednika w blokach maszyny analogowej. Postać deklaracji elementu jest następująca:

y = AREX (wyrażenie)

gdzie:

y - sygnał typu ANALOG,

wyrażenie - wyrażenie arytmetyczne zapisane w FORTRAN-ie, np.:

SILA = AREX (X * K + L * DX + SILWYM)

Element wykonuje obliczenie wyrażenia arytmetycznego zapisanego w nawiasach zgodnie z zasadami FORTRAN-u.

C. Elementy funkcji matematycznych.

Elementy te służą do obliczania podstawowych funkcji matematycznych, omówionych wystarczająco w p.2.2.4. Wszystkie te elementy / SQRT, EXP, LOG, ABS, SIGN, FIX, SIN, COS, TAN, ARSIN, ARCOS, ARTAN, SINH, COSH / mają jednakową postać deklaracji:

y = efm(x) ,

gdzie:

x, y - sygnały typu ANALOG,

efm - nazwa funkcji matematycznej, np.:

X = EXP (T)

SN = SIN (ALFA)

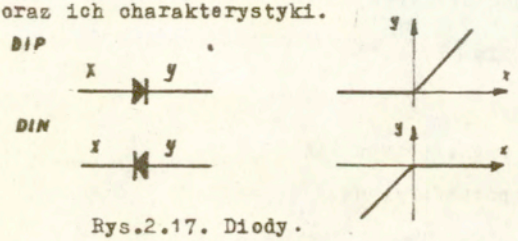
Elementy funkcji matematycznych odpowiadają blokom przetworników funkcyjnych maszyny analogowej i wykonują działania na sygnałach analogowych. Dlatego też nie należy ich mylić z funkcjami standardowymi języka FORTAN, używanymi w segmencie NUMERICAL i operującymi na zmiennych numerycznych.

D. Elementy specjalne i hybrydowe.

Do danych elementów zaliczamy elementy modelujące specjalne funkcje nieliniowe często występujące w układach dynamicznych /DIP, DIN, LIM, DEADSP, HIST/ oraz elementy przełączające /SWI, DSWI, PRZ/.

D1. Diody - DIP, DIN.

Na rys.2.17 pokazano symbole elementów używane w schematach analogowych oraz ich charakterystyki.



Rys.2.17. Diody.

a. Postać deklaracji:

x = DIP (y)

gdzie:

x, y - sygnały typu ANALOG, np.:

XPLUS = DIP (X)

XMINUS = DIN (X) .

b. Działanie:

Elementy modelują funkcje opisane zależnością:

- DIP

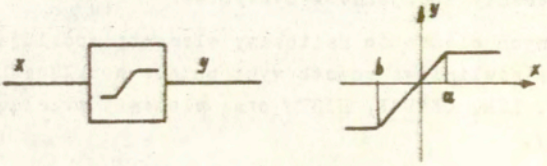
$$y = \begin{cases} x & \text{, dla } x \geq 0 \\ 0 & \text{, dla } x < 0 \end{cases}$$

- DIN

$$y = \begin{cases} 0 & \text{, dla} \quad x > 0 \\ x & \text{, dla} \quad x \leqslant 0 \end{cases} .$$

D2. Ograniczenie - LIM.

Symbol elementu i realizowaną zależność przedstawiono na rys.2.18.



Rys.2.18. Ograniczenie.

a. Postać deklaracji:

y = LIM (x, a, b)

gdzie:

y, x - sygnały typu ANALOG

a, b - paramétry, np.

OGR = LIM (XWE, 1.0 - 2.0) .

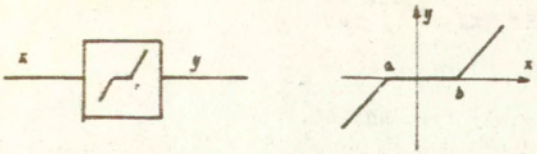
b. Działanie:

Element realizuje zależność:

$$y = \begin{cases} a & \text{, dla} \quad x > a \\ x & \text{, dla} \quad b \leqslant x \leqslant a \\ b & \text{, dla} \quad x > b \end{cases} .$$

D3. Strefa nieczułości - DEADSP

Symbol elementu i modelowaną funkcję przedstawiono na rys.2.19.



Rys.2.19. Strefa nieczułości

a. Postać deklaracji:

y = DEADSP (x, a, b)

gdzie:

x, y - sygnały typu ANALOG

a, b - parametry, np.:

NCZUL = DEADSP (XWE,0.5,- 0.1) .

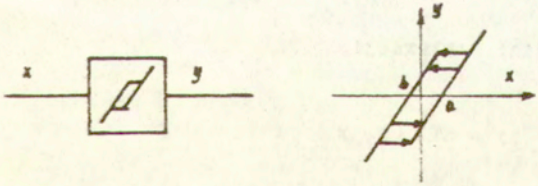
b. Działanie:

Element realizuje zależność:

$$y = \begin{cases} x - a & , \; x < a \\ 0 & , \; a \leqslant x \leqslant b \\ x - b & , \; x > b \end{cases} .$$

D4. Pętla histerezy - HIST.

Symbol elementu i modelowaną funkcję pokazano na rys.2.20.



Rys.2.10. Pętla histerezy.

a. Postać deklaracji:

y = HIST (x, a, b, yp)

gdzie:

x, y - sygnały typu ANALOG,

a, b, yp - parametry, np.:

LUZ = HIST (XWE, 0.1, - 0.1, 3.0)

b. Działanie:

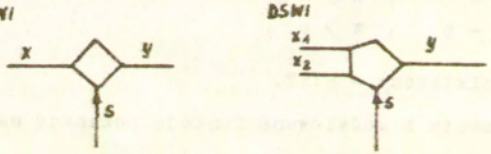
Element modeluje funkcję /nazywaną również luzem maszynowym/:

$$y = \begin{cases} x - a & \text{, dla } x^i - x^{i-1} > 0 \text{ i } x^{i-1} \leqslant y - a \\ x - b & \text{, dla } x^i - x^{i-1} < 0 \text{ i } x^{i-1} \geqslant y - b \\ y^{i-1} & \text{, w pozostałych przypadkach.} \end{cases}$$

Parametr yp potrzebny jest do ustalenia początkowego stanu elementu i jest wprowadzany w stanie PREPARE.

D5. Klucze elektroniczne - SWI i DSWI

Symbole elementów pokazano na rys.2.21



Rys.2.21. Klucze elektroniczne

a. Postać deklaracji:

- SWI

y = SWI (s, x)

- DSWI

y = DSW (s, x1, x2)

gdzie:

x, x1, x2, y - sygnały typu ANALOG

s - sygnał typu LOGIC, np.

Y1 = SWI (L1, X)

Y2 = DSWI (L3, X, NX)

b. Działanie:

Elementy realizują operacje:

- SWI:

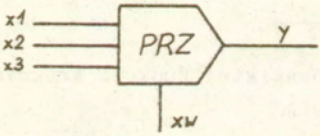
$$y = \begin{cases} x & \text{, dla} \quad s = \text{TRUE} \\ 0 & \text{, dla} \quad s = \text{FALSE} \end{cases}$$

- DSWI:

$$y = \begin{cases} X1 & \text{, dla} \quad s = \text{TRUE} \\ X2 & \text{, dla} \quad s = \text{FALSE} \end{cases}$$

D6. Przekaźnik trójpołożeniowy - PRZ.

Symbol elementu pokazano na rys.2.22.



Rys.2.22. Przekaźnik trójpołożeniowy.

a. Postać deklaracji:

y = PRZ (xw, x1, x2, x3)

gdzie:

xw, x1, x2, x3, y - sygnały typu ANALOG, np.

Y = PRZ (DELTA, X2, .0, X3)

b. Działanie:

Element realizuje zależność:

$$y = \begin{cases} X1 & \text{,dla} \quad xw < 0. \\ X2 & \text{,dla} \quad xw = 0. \\ X3 & \text{,dla} \quad xw > 0. \end{cases}$$

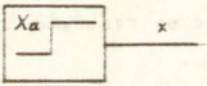
E. Generatory wymuszeń.

Do generatorów wymuszeń zaliczamy elementy, w których sygnał wyjściowy zależy jedynie od czasu T, nie zależy natomiast od stanu pozostałych elementów modelu /generatory nie posiadają wejść analogowych/.

W języku LAHYSS stosować można zarówno generatory wymuszeń zdeterminowanych /GSTEP, GSIN, GCOS, GRECT, GTRIAL/, jak i stochastycznych /GRANDOM, GAUSS/.

E1. Generator funkcji skokowej - GSTEP.

Symbol elementu pokazano na rys.2.23.



Rys. 2.23. Generator funkcji skokowej.

a. Postać deklaracji:

x = GSTEP (xa)

gdzie:

x - sygnał typu ANALOG

xa - parametr, np.:

PWYM = GSTEP (2.0)

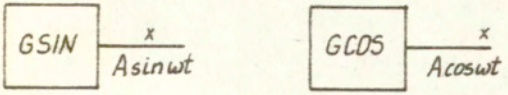
b. Działanie:

Element generuje funkcję:

$$x = \begin{cases} 0 & \text{, dla} \quad T = 0. \\ xa & \text{, dla} \quad T > 0. \end{cases}$$

E2. Generatory wymuszeń harmonicznych - GSIN, GCOS.

Symbole elementów pokazano na rys.2.24.



Rys.2.24. Generatory wymuszeń harmonicznych.

a. Postać deklaracji:

x = GSIN (ω , A)

gdzie:

x - sygnał typu ANALOG ,

ω , A - parametry, np.:

PW = GSIN (OMEGA, PAMPL)

XW = GCOS (2.0, XMAKS)

b. Działanie:

Elementy generują funkcję:

- GSIN:

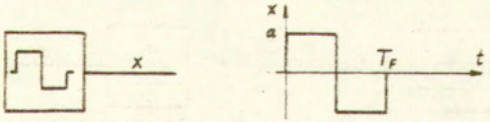
x = A·sin ωt

- GCOS:

x = A·cos ωt.

E3. Generator fali prostokątnej - GRECT.

Symbol elementu i generowaną funkcję pokazano na rys.2.25



Rys.2.25. Generator fali prostokątnej.

a. Postać deklaracji:

x = GRBCT (T_p, a)

gdzie:

x - sygnał typu ANALOG

a, T_p - parametry, np.:

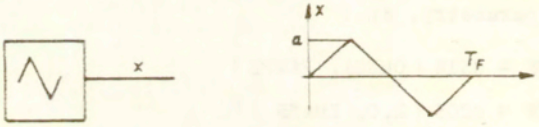
WYM = GRBCT (10.0, 1.0)

b. Działanie:

Element generuje falę prostokątną o okresie T_F i amplitudzie a.

B4. Generator fali trójkątnej.

Symbol elementu i generowaną funkcję pokazano na rys.2.26.



Rys.2.26. Generator fali trójkątnej.

Deklaracja i działanie generatora jest podobne jak elementu poprzedniego. Różnica polega jedynie na kształcie generowanej fali, co ilustruje rys.2.26.

B.5. Generatory wymuszeń losowych - GRAND, GAUSS.

Symbole generatorów pokazano na rys.2.27.

Rys.2.27. Generatory wymuszeń losowych.

a. Postać deklaracji:

- Grand:

x = GRAND (N)

- GAUSS:

x = GAUSS (N, E, σ)

gdzie:

x - sygnał typu ANALOG,

N - dowolna liczba całkowita dodatnia,

E - wartość oczekiwana,

σ - odchyłka standardowa, np.:

STOCH = GRAND (271) ,

STOGAS = GAUSS (152, 3.52, 0.3)

b. Działanie:

Elementy generują funkcje losowe:

- GRAND:

funkcja o rozkładzie równomiernym w przedziale $\langle 0,1 \rangle$.

- GAUSS:

funkcja o rozkładzie normalnym.

Działanie elementów oparte jest na procedurach funkcyjnych dostępnych w systemie PDP 11/70.

2.3.2.2. Elementy logiczne.

Pod pojęciem tego typu będziemy rozumieć elementy realizujące funkcje logiczne /przełączające/, w których sygnał wyjściowy jest sygnałem logicznym /LOGIC/ niezależnie od typów sygnałów podawanych na wejścia elementów.

Ogólna postać deklaracji elementu logicznego jest następująca:

sl = el (wykaz sygnałów i parametrów)

gdzie:

sl - nazwa sygnału wyjściowego, uprzednio zadeklarowana dyrektywą LOGIC,

el - nazwa elementu logicznego,

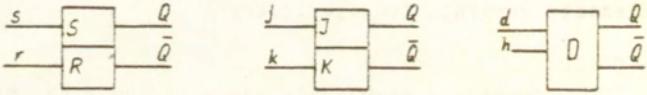
wykaz sygnałów i parametrów - lista sygnałów i parametrów podawanych na wejścia elementów, oddzielonych przecinkami.

A. Elementy pamięci logicznej.

Do elementów tych zaliczamy elementy realizujące operacje logiczne, których wynik zależy zarówno od stanu wejść, jak i stanu poprzedniego elementu. Do elementów tych należą: przerzutniki /RS, JK, DH/ oraz logiczny element różniczkujący /DERL/.

A1. Przerzutniki - RS, JK, DH.

Symbole przerzutników pokazano na rys.2.28.



Rys.2.28. Przerzutniki.

a. Postać deklaracji:

- RS:

q = RS (r, s)

- JK:

q = JK (j, k)

- DH:

q = DH (h, d)

gdzie:

q, r, s, j, k, h, d - sygnały typu LOGIC, np.:

Q1 = RS (L2, L3)

Q2 = JK (Q1, K3)

Q3 = DH (CLOCK, Q2) .

b. Działanie:

Działanie poszczególnych przerzutników pokażemy za pomocą tablic przejść, w których użyto następujących oznaczeń:

Q - stan poprzedni przerzutnika,

Q^{i+1} - stan następny przerzutnika,

0 - "zero" logiczne FALSE,

1 - "jedynka" logiczna TRUE

RS

Q \ sr	00	01	11	10
0	0	0	—	1
1	1	0	—	1

Q^{i+1}

D

Q \ d	0	1
0	0	1
1	0	1

Q^{i+1}

JK

Q \ jk	00	01	11	10
0	0	0	1	1
1	1	0	0	1

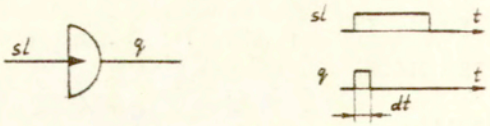
Q^{i+1}

Przerzutniki RS i JK są przerzutnikami asynchronicznymi, tzn. zmiana ich stanu może nastąpić w dowolnej chwili. Przerzutnik DH jest elementem synchronicznym. Zmiana jego stanu /zgodnie z podaną tablicą/ może nastąpić tylko przy zmianie sygnału taktującego h z "0" na "1".

Do ustawiania i zerowania przerzutników służą dodatkowo rozkazy SET i RESET.

A2. Logiczny element różniczkujący - DERL.

Symbol elementu i modelowaną funkcję pokazano na rys.2.29.



Rys.2.29. Logiczny element różniczkujący.

a. Postać deklaracji:

q = DERL (sl)

gdzie:

q, sl - sygnały typu LOGIC, np.:

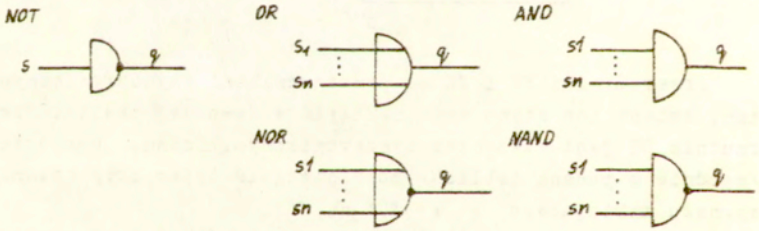
DK = DERL (K)

b. Działanie:

Na wyjściu elementu pojawia się sygnał q = TRUE o czasie trwania równym dt w przypadku zmiany sygnału wejściowego sl z FALSE na TRUE, tzn. na zboczu narastającym.

B. Elementy operacji logicznych.

Elementy /rys.2.30/ realizują podstawowe funkcje logiczne:



Rys.2.30. Elementy operacji logicznych.

- NOT / negacja /:

q = ¬ s (.NOT. s)

- OR /alternatywa/:

$q = s1 \vee \ldots \vee sn$ (s1.OR. ... .OR. sn)

- AND /koniunkcja/:

$q = s1 \wedge \ldots \wedge sn$ (s1.AND. ... .AND.sn)

- NOR /negacja alternatywy/:

$q = \neg (s1 \vee \ldots \vee sn)$

- NAND /negacja koniunkcji/:

$q = \neg (s1 \wedge \ldots \wedge sn)$.

C. Komparator - COMP.

Symbol elementu pokazano na rys.2.31.

Rys.2.31. Komparator.

a. Postać deklaracji:

k = COMP (x1,...,xn)

gdzie:

k - sygnał typu LOGIC

x1,...,xn - sygnały typu ANALOG, $n \leqslant 4$, np.:

GOOD = COMP (X, - 0.1)

b. Działanie:

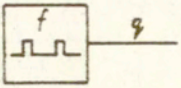
Element realizuje funkcję:

$$k = \begin{cases} \text{TRUE} & \text{, dla} \quad \sum_i x_i \geqslant 0 \\ \text{FALSE} & \text{, dla} \quad \sum_i x_i < 0 \end{cases}.$$

D. Generatory sygnałów logicznych.

D1. Generator impulsów - GEN.

Symbol elementu i generowaną funkcję pokazano na rys.2.32.



Rys.2.32. Generator impulsów.

a. Postać deklaracji:

q = GEN (f)

gdzie:

q - sygnał typu LOGIC,

f - parametr, np.:

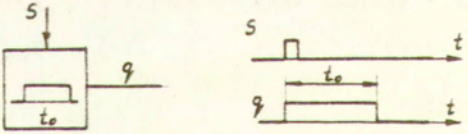
CLOCK = GEN (10.0)

b. Działanie:

Element generuje impulsy logiczne o częstotliwości f i czasie trwania dt . Generator uruchamiany jest automatycznie w stanie SOLVE.

D2. Generator przedziałów czasowych - TPER.

Symbol elementu i generowaną funkcję pokazano na rys.2.33.



Rys.2.33. Generator przedziałów czasowych.

a. Postać deklaracji:

q = TPER (s, to)

gdzie:

q, s - sygnał typu LOGIC,

to - parametr, np.:

PERIOD = TPER (GO, 2.5)

b. Działanie:

Element generuje sygnał logiczny q = TRUE w czasie trwania to sekund, z chwilą pojawienia się sygnału wyzwalającego s = TRUE.

2.3.2.3. Elementy sterujące.

Do elementów sterujących należą dwa elementy: HOLD i TERMIN. Pozwalają one na przerywanie obliczeń symulacyjnych i przekazywaniu sterowania do innych segmentów programu.

A. Przerywanie obliczeń - HOLD.

Element HOLD odpowiada rejestrowi przerwań występującemu w rzeczywistym systemie hybrydowym.

a. Postać deklaracji:

HOLD (s1,...,sn)

gdzie:

s1,...,sn - sygnały typu ANALOG, $n \leq 4$, np.

HOLD (K3, CLOCK, ST)

b. Działanie:

Pojawienie się na którymkolwiek z wejść s1,...,sn sygnału TRUE powoduje:

- przerwanie przebiegu symulacyjnego,
- nadanie zmiennym standardowym HLD wartości

HLD = TRUE ,

- wywołanie procedury HOLD segmentu NUMERICAL,
- dalsze obliczenia podejmowane są zgodnie z rozkazami

zawartymi w procedurze HOLD / p.2.5 / .

B. Zakończenie obliczeń - TERMIN.

a. Postać deklaracji:

TERMIN (s1,...,sn)

gdzie:

s1,...,sn - sygnały typu LOGIC, n $\leq$ 4 , np.:

TERMIN (BAD, EXIT)

b. Działanie:

Pojawienie się na którymkolwiek z wejść s1,...,sn sygnału TRUE powoduje zakończenie obliczeń symulacyjnych dla aktualnie wykonywanego przebiegu oraz przekazanie sterowania do segmentu, z którego nastąpiło uruchomienie obliczeń / CONTROL lub procedura EXEC segmentu NUMERICAL/.

2.3.3. Sekcja PERIPHERAL.

W sekcji PERIPHERAL deklarowane są urządzenia peryferyjne współpracujące z modelem, opisanym w sekcji DYNAMIC, w sposób ciągły. W punkcie tym omówimy formalną stronę zapisu instrukcji deklarujących urządzenia peryferyjne oraz standardowe działanie tych urządzeń /bardziej złożone metody korzystania z urządzeń peryferyjnych przedstawione będą w p.3.3/.

Sekcja rozpoczyna się słowem sterującym PERIPHERAL, zaś kończy słowem rozpoczynającym kolejny segment programu /CONTROL - rys.2.2/.

Ogólna postać deklaracji jest następująca:

device (lun , wykaz sygnałów)

gdzie:

device - nazwa typ urządzenia / p.2.2.5 /,

lun - numer programowy / logiczny / urządzenia,

wykaz sygnałów - lista rejestrowanych sygnałów, oddzielonych przecinkami.

Każdemu urządzeniu peryferyjnemu użytemu w programie musi być przyporządkowany numer programowy /liczba całkowita bez znaku/. Wyjątkiem jest urządzenie TYPE, któremu na stałe przyporządkowany jest numer 5. Numer programowy służy do identyfikacji urządzenia w rozkazach sterujących urządzeniami peryferyjnymi /p.2.4.2/ oraz określa zbiór w pamięci dyskowej, na którym rejestrowane są przebiegi. Przy nadawaniu numerów logicznych poszczególnym urządzeniom należy przestrzegać następujące zasady:

- numer musi zawierać się w przedziałach 1 - 4 i 7 - 10 /numery 5 i 6 są na stałe przyporządkowane odpowiednio monitorowi terminalowi użytkownika i drukarce wierszowej/,
- każdemu urządzeniu musi być przyporządkowany oddzielny numer,
- numery programowe użyte w sekcji PERIPHERAL nie mogą być użyte w instrukcjach wejścia/wyjścia FORTRAN-u w segmencie NUMERICAL.

Wykaz sygnałów musi zawierać sygnały jednakowego typu /ANALOG lub LOGIC/ w zależności od typu urządzenia. Dla urządzeń do rejestracji sygnałów analogowych wykaz ten będziemy oznaczać symbolicznie:

a1,...,an , gdzie n - dopuszczalna liczba sygnałów dla danego elementu. Dla urządzeń do rejestracji sygnałów logicznych wykaz oznaczać będziemy przez ciąg symboli:

l1,...,ln .

Wszystkie urządzenia peryferyjne wymagają podania odstępu próbkowania rejestracji wyników /rozkaz ACCESS - p.2.4.2/.

2.3.3.1. Rejestracja sygnałów analogowych

Do rejestracji sygnałów analogowych można stosować:

- urządzenia do rejestracji wyników w postaci numerycznej /TYPE, TABLE/,
- urządzenia do rejestracji graficznej /REXT, REXY/,
- urządzenia do rejestracji w pamięci dyskowej /OUTPUT/,

Urządzenie OUTPUT współpracuje na ogół z urządzeniem INPUT pozwalającym na odczytywanie przebiegów zapisanych w pamięci dyskowej.

A. Rejestarcja wyników w postaci tabeli - TABLE

a. Postać deklaracji:

TABLE (lun, a1,...,an)

n ⩽ 6, np.:

TABLE (3, X1, X2)

b. Działanie:

Urządzenie realizuje rejestrację zadeklarowanych sygnałów a1,...,an, a następnie wyprowadzanie wyników na drukarkę wierszową. Wyniki wyprowadzane są po zakończeniu poszczególnych przebiegów symulacyjnych w postaci tabeli zawierającej:

- nagłówek,
- opis tabeli,
- w kolejnych wierszach: czas symulacyjny T i odpowiadające mu wartości sygnałów a1,...,an w formacie E15.6.

B. Rejestracja bieżąca - TYPE

a. Postać deklaracji:

TYPE (a1,...,an)

n ⩽ 4, np.:

TYPE (D2X, X3, PWYM).

b. Działanie:

Urządzeniu na stałe przyporządkowany jest numer 5. Element powoduje wyprowadzanie czasu symulacyjnego T oraz zadeklarowanych sygnałów a1,...,an bezpośrednio na monitor użytkownika w formacie E15.6. Umożliwia to bieżącą obserwację i ocenę przebiegów w modelowanym układzie dynamicznym.

C. Rejestracja wykresów czasowych - REXT

a. Postać deklaracji:

REXT (lun, a1,...,an)

n ⩽ 6, np.:

REXT (7, X1, X2, X3)

b. Działanie:

Urządzenie rejestruje sygnały a1,...,an w pamięci dyskowej, a następnie wyprowadza wyniki w postaci wykresów czasowych /na płaszczyźnie XT/. Wyniki wyprowadzane są po zakończeniu poszczególnych przebiegów symulacyjnych na drukarkę wierszową, a także graph-plotter. Wykresy skalowane są automatycznie /co pozwala na ich maksymalne rozciągnięcie na płaszczyźnie wykresu/ lub przez rozkaz SCALE /co umożliwia porównanie przebiegów o podobnym charakterze, ale różnych amplitudach/.

D. Rejestracja na płaszczyźnie fazowej - REXY

a. Postać deklaracji:

REXY (lun, ax, ay)

np.:

REXY (1, X, DX)

b. Działanie:

Działanie urządzenia jest analogiczne jak urządzenie REXT, z tą różnicą, że wykres sporządzany jest na płaszczyźnie (ax, ay).

E. Rejestracja w pamięci dyskowej - OUTPUT

a. Postać deklaracji:

OUTPUT (lun, a1,...,an)

n ⩽ 8, np.:

OUTPUT (3, X, DX, D2X)

b. Działanie:

Urządzenie działa analogicznie jak magnetofon w stanie ZAPIS, tzn. rejestruje zadeklarowane sygnały a1,...,an w pamięci dyskowej bez wyprowadzania jakichkolwiek wyników na urządzenia maszyny cyfrowej.

F. Odczyt przebiegów z pamięci dyskowej - INPUT.

a. Postać deklaracji:

INPUT (lun, a1, ...,an)

n ≤ 8, np.:

INPUT (3, X, DX, D2X)

b. Działanie:

Urządzenie działa analogicznie jak magnetofon w stanie ODCZYT, a więc odczytuje sygnały zapisane uprzednio w pamięci dyskowej. Urządzenie współpracuje z urządzeniem OUTPUT. W celu zapewnienia współpracy między tymi urządzeniami należy przestrzegać następujące reguły:

- liczba zadeklarowanych sygnałów w obu współpracujących elementach musi być taka sama,
- krok /odstęp/ próbkowania przy zapisie i odczycie /tzn. dla obu urządzeń/ musi być taki sam.

Urządzenia stosowane są przy modelowaniu złożonych algorytmów iteracyjnych oraz do wymiany przebiegów pomiędzy różnymi programami użytkowymi.

2.3.3.2. Rejestracja sygnałów logicznych.

Do rejestracji sygnałów logicznych służą dwa urządzenia: OUTLOG i SIGLOG.

A. Rejestracja sygnałów logicznych - OUTLOG.

a. Postać deklaracji:

OUTLOG (lun, l1,...,ln)

n ⩽ 8, np.:

OUTLOG (lun, GOOD, WAR, ST)

b. Działanie:

Urządzenie rejestruje zadeklarowane sygnały, a następnie powoduje wyprowadzenie wyników w postaci tabeli, zawierającej wartości czasu symulacyjnego T i odpowiadające mu wartości zarejestrowanych sygnałów logicznych w formacie E15.6, 8L8. Wyniki wyprowadzane są po zakończeniu przebiegu symulacyjnego na drukarkę wierszową.

B. Rejestracja zmiany stanu sygnału logicznego - SIGLOG.

a. Postać deklaracji:

SIGLOG (l1,...ln)

n ⩽ 8, np.:

SIGLOG (K2, K3, Q1)

b. Działanie:

Urządzenie powoduje wyprowadzenie czasu symulacyjnego T i odpowiadających mu wartości sygnałów l1,...,ln bezpośrednio na monitor w formacie E15,6, 8L8. Wyniki wyprowadzane są tylko w przy padku zmiany stanu /tzn. z TRUE na FALSE lub z FALSE na TRUE/, któregokolwiek z zadeklarowanych sygnałów. Urządzenie umożliwia bieżącą obserwację i ocenę stanu logicznego badanego układu.

2.4. Rozkazy sterujące.

Rozkazy sterujące służą do sterowania modelem układu dynamicznego zapisanym w segmencie SYSTEM. Przy pomocy tych rozkazów zapisywany jest algorytm badania układu. Rozkazy sterujące dzielą się na 5 grup:

1. Rozkazy sterujące stanami modelu układu dynamicznego - powodują przejście całego modelu opisane w segmencie

SYSTEM do stanu PREPARE /warunki początkowe/ lub SOLVE /rozwiązywanie/ i wykonanie obliczeń symulacyjnych.

2. Rozkazy sterujące pojedynczymi elementami - umożliwiają sterowanie stanami pracy poszczególnych elementów sekcji DYNAMIC, bez zmiany stanu pozostałych elementów modelu.
3. Rozkazy sterujące urządzeniami peryferyjnymi - zapewniają sterowanie poszczególnymi urządzeniami zadeklarowanymi w sekcji PERIPHERAL.
4. Rozkazy specjalne - służą do sterowania czynnościami związanymi z rozpoczęciem, przerwaniem i zakończeniem przebiegów symulacyjnych.
5. Rozkazy sterujące sposobem wykonywania obliczeń - określają sposób sterowania obliczeniami symulacyjnymi oraz metody całkowania.

Rozkazy sterujące mogą być wywoływane w segmencie CONTROL, ręcznie z terminala EMC lub w segmencie NUMERICAL. W dwóch pierwszych przypadkach postać ogólna wywołania rozkazu jest następująca:

rozkaz [(parametry)]

Natomiast w przy padku wywołania rozkazu w segmencie NUMERICAL

CALL rozkaz [(parametry)] .

Nawiasy kwadratowe /[,]/ oznaczają możliwość pominięcia zawartych w nich treści.

2.4.1. Rozkazy sterujące stanami modelu.

Rozkazy powodują uruchomienie obliczeń symulacyjnych /całkowania/ modelu układu dynamicznego. Zaliczamy do nich rozkazy: PREPARE, SOLVE, COMPUTE, RUN oraz ITER. Rozkazy PREPARE, SOLVE, COMPUTE i ITER nie mają żadnego wpływu na stan pracy urządzeń peryferyjnych - ich wykonanie powoduje jedynie rejestrację przebiegów. Natomiast rozkaz RUN powoduje inicjację urządzeń peryferyjnych/ustawienie zbiorów przyporządkowanych urządze-

niom na początek/ oraz wyprowadzenie wyników po zakończeniu obliczeń.

A. Wprowadzanie warunków początkowych - PREPARE.

a. Postać wywołania:

[CALL]PREPARE (to)

gdzie:

to - czas początkowy, np.

PREPARE (.0)

CALL PREPARE (2.0)

b. Działanie:

Rozkaz powoduje ustalenie czasu symulacyjnego T = to i przejście całego układu w stan PREPARE. Po wykonaniu rozkazu czas T nie ulega zmianie.

B. Rozwiązywanie - SOLVE

a. Postać wywołania:

[CALL] SOLVE (tk)

gdzie:

tk - czas rozwiązywania równań opisujących układ, np.:

SOLVE (10.0)

CALL SOLVE (TSYM).

b. Działanie:

Rozkaz powoduje rozwiązywanie układu opisanego w segmencie SYSTEM układu w przedziale czasowym:

$tp \leqslant T \leqslant tp + tk$, gdzie

tp - czas symulacyjny T wchwili wywołania rozkazu. W czasie rozwiązywania rejestrowane są sygnały na wszystkich włączonych urządzeniach peryferyjnych.

Zakończenie rozwiązywania następuje w 2 przypadkach:
1. po upływie tk sekund,
2. poprzez element TERMIN zadeklarowany w sekcji DYNAMIC.

C. Obliczenie pojedynczego przebiegu - COMPUTE.

a. Postać wywołania:

[CALL] COMPUTE (tk)

gdzie:

tk - czas rozwiązywania, np.:

```
COMPUTE (2.0)
CALL COMPUTE (TK)
```

b. Działanie:

Rozkaz powoduje wykonanie kolejno rozkazów:

```
PREPARE (.0)
SOLVE (tk)
```

D. Wykonanie pojedynczego przebiegu - RUN.

a. Postać wywołania:

[CALL] RUN (tk)

gdzie:

tk - czas rozwiązywania, np.

```
RUN (10.0)
CALL RUN  tk
```

b. Działania:

Rozkaz powoduje wykonanie kolejno rozkazów:

```
NEWRUN /p.2.4.4/
PREPARE(.0)
SOLVE (tk)
```

ENDRUN /p.2.4.4./

E. Obliczenia iteracyjne - ITER.

a. Postać wywołania:

[CALL] ITER (tk, noit, warhon)

gdzie:

tk - czas rozwiązywania,

noit - maksymalna liczba iteracji,

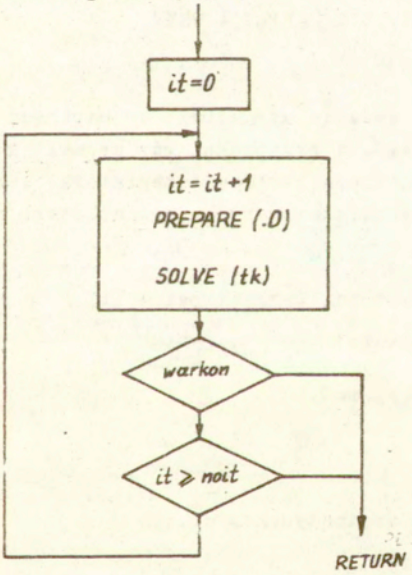
warhon - warunek zakończenia iteracji, np.:

ITER (10.0, 20, K2)

CALL ITER (TK, NOIT, XMAX .GT. 3.0)

b. Działanie:

Rozkaz powoduje wykonanie obliczeń iteracyjnych wg schematu blokowego pokazanego na rys.2.3.4.



Rys.2.34. Schemat algorytmu dla rozkazu ITER.

Zakończenie obliczeń następuje w dwóch przypadkach:

1. Wartość wyrażenia logicznego warkon przyjmuje wartość TRUE,
2. Wykonano maksymalną noit liczbę iteracji.

2.4.2. Rozkazy sterujące pojedynczymi elementami.

Rozkazy służą do nadawania wartości sygnałom analogowym /INIT/ , parametrom /LET/ oraz sygnałom logicznym /SET, RESET/.

A. Nadawanie wartości sygnałowi analogowemu-INIT.

a. Postać wywołania:

[CALL] INIT (X, wa)

gdzie:

x - sygnał typu ANALOG,
wa - wyrażenia arytmetyczne, np.:

INIT (X1, 2,5)

CALL INIT (LX, SIN (ALFA) + WSP)

b. Działanie:

Rozkaz powoduje nadanie sygnałowi x wartości równej obliczonej wartości wa . W przypadku, gdy sygnał x opisuje wyjście elementu dynamicznego /np. integratora/ , rozkaz jest równoznaczny z wprowadzeniem warunku początkowego na ten element.

B. Nadawanie wartości parametrowi - LET.

a. Postać wywołania:

[CALL] LET (p, wa)

gdzie:

p - parametr,
wa - wyrażenie arytmetyczne, np.:

LET (NI, 0.1)

CALL LET (NI, 2. * OMEGA1 + OMEGA2) .

b. Działanie:

Rozkaz powoduje nadanie parametrowi p wartość równą wa .

C. Ustawianie sygnałów logicznych - SET, RESET.

a. Postać wywołania:

[CALL] SET (l)

[CALL] RESET (l)

gdzie:

l - sygnał typu LOGIC,

SET (Q1)

CALL SET (Q3)

b. Działanie:

Rozkaz SET powoduje nadanie sygnałowi l wartość TRUE, zaś rozkaz RESET nadaje syganłowi l wartość FALSE.

2.4.3. Rozkazy sterujące urządzeniami peryferyjnymi.

Rozkazy służą do sterowania urządzeniami zadeklarowanymi w sekcji PERIPHERAL segmentu SYSTEM. Zapewniają one:

- włączanie lub wyłączanie urządzeń /ACCESS, OPENR, CLOSER/,
- wyprowadzanie wyników rejestracji /EDIT, PLOT/ ,
- kasowanie zarejestrowanych przebiegów /DELETE, BACK/ ,
- przesyłanie zarejestrowanych przebiegów pomiędzy urządzeniami /COPY, EXCH, APPEND/,
- kasowanie zarejestrowanych przebiegów /DELETE, BACK/ ,
- skalowanie wykresów /SCALE/ ,
- wyprowadzanie opisu słownego /HEAD, TEXT/ .

A. Otwarcie dostępu włączenia urządzenia - ACCESS, OPENR.

a. Postać wywołania:

[CALL] ACCESS (lun, dtout)

[CALL] OPENR (lun)

gdzie

lun - numer programowy urządzenia,
dtout - odstęp krok rejestracji przebiegów, np.:

```
ACCESS (1, 0.01)
CALL OPENR(2)
```

b. Działanie:

Rozkazy powodują włączenie urządzenia peryferyjnego o numerze programowym lun , zadeklarowanego w sekcji PERIPHERAL. Rozkaz ACCESS dodatkowo określa krok próbkowania rejestrowanych sygnałów dtout.

B. Zamknięcie dostępu wyłączenie urządzenia - CLOSER.

a. Postać wywołania:

[CALL] CLOSER (lun)

gdzie:

lun- numer programowy urządzenia, np.

```
CLOSER (2)
CALL CLOSER (NL)
```

b. Działanie:

Rozkaz powoduje wyłączenie urządzenia o numerze lun . Dotychczas zarejestrowane w urządzeniu przebiegi są nadal przechowywane.

C. Wyprowadzanie wyników na drukarkę wierszową - EDIT.

a. Postać wywołania:

[CALL] EDIT (lun)

gdzie:

lun - numer programowy urządzenia, np.:

EDIT (3)

CALL EDIT (ACTIVE)

b. Działanie:

Rozkaz powoduje wyprowadzenie wyników na drukarkę wierszową tylko dla włączonego urządzenia o numerze lun . Rozkaz wykonywany jest dla urządzeń: TABLE, REXT, REXY oraz OUTLOG, dla pozostałych urządzeń nie jest wykonywany /p.2.3.3./.

D. Wyprowadzanie wyników na graph-plotter - PLOT.

a. Postać wywołania:

[CALL] PLOT (lun)

gdzie:

lun - numer programowy urządzenia, np.:

PLOT (7)

CALL PLOT (2)

b. Działanie:

Rozkaz powoduje wyprowadzenie zarejestrowanych na urządzeniu o numerze lun przebiegów na graph-plotter. Rozkaz wykonany jest tylko dla urządzenia REXT i REXY.

E. Kasowanie zarejestrowanych przebiegów - DELETE.

a. Postać wywołana:

[CALL] DELETE (lun)

gdzie:

lun - numer programowy urządzenia, np.:

DELETE (2)

CALL DELETE (LUN)

b. Działanie:

Rozkaz powoduje wyzerowanie zbioru w pamięci dyskowej

przydzielonego urządzeniu o numerze lun.

F. Przejście na początek zbioru - BACK.

a. Postać wywołania:

[CALL]BACK (lun)

gdzie:

lun - numer programowy urządzenia, np.:

CALL BACK (9)

BACK (1).

b. Działanie:

Rozkaz powoduje "przewinięcie" zbioru przydzielonego urządzeniu o numerze lun na początek. Umożliwia to powtórne odczytanie zawartości zbioru przez urządzenie INPUT.

G. Kopiowanie zarejestrowanych przebiegów - COPY.

a. Postać wywołania:

[CALL] COPY (lun1, lun2)

gdzie:

lun1, lun2 - numery programowe urządzeń, np.:

CALL COPY (1, 2)

COPY (2, 3)

b. Działanie:

Rozkaz powoduje przesłanie przebiegów zarejestrowanych w urządzeniu lun1 do urządzenia lun2, bez zmiany zawartości zbioru przyporządkowanemu urządzeniu lun1. Przesyłanie może następować między urządzeniami, w których zadeklarowana jest ta sama liczba rejestrowanych sygnałów.

H. Zamiana zarejestrowanych przebiegów między urządzeniami - EXCH.

a. Postać wywołania:

[CALL] EXCH (lun1, lun2)

gdzie:

lun1, lun2 - numery programowe urządzeń, np.:

CALL EXCH (1, 2)

EXCH (2, 3).

b. Działanie:

Rozkaz powoduje zamianę zawartości zbiorów przydzielonych urządzeniom lun1 i lun2. Oznacza to, że przebiegi zarejestrowane na urządzeniu lun1 znajdują się po wykonaniu rozkazu w urządzeniu lun2 i odwrotnie.

Dwa ostatnie rozkazy stosowane są do sterowania urządzeniami INPUT i OUTPUT przy modelowaniu algorytmów iteracyjnych.

E. Dołączanie zapisów - APPEND.

a. Postać wywołania:

[CALL] APPEND (lun1, lun2)

gdzie:

lun1, lun2 - numery programowe urządzeń, np.:

CALL APPEND (N1, N2)

APPEND (1, 3)

b. Działanie:

Rozkaz powoduje dołączenie przebiegów zarejestrowanych na urządzeniu lun1 do zbioru przydzielonego urządzeniu lun2. Rozkaz pozwala na wykonywanie kilku wykresów na jednym rysunku.

I. Skalowanie wykresów - SCALE.

a. Postać wywołania:

[CALL] SCALE (lun, xmax, xmin, ymax, ymin)

gdzie:

lun - numer programowy urządzenia,
xmax, xmin, ymax, ymin - wyrażenia arytmetyczne typu REAL,
np.:

```
CALL SCALE (1, 20.0,- 20.0, YMAX,.0)
SCALE (2, TK, .0, 10.0,- 10.0)
```

b. Działanie:

Rozkaz powoduje wyskalowanie poszczególnych osi przed wykonaniem wykresu /xmax i xmin - osi czasu T dla urządzenia REXT i osi x dla urządzenia REXY; ymax i ymin - osi Y dla obu urządzeń/ . W przypadku przekroczenia przez sygnał wartości granicznych jest on obcinany do tej wartości. Rozkaz jest wykonywany tylko dla urządzeń REXT i REXY.

K. Wyprowadzanie opisu słownego - HEAD, TEXT.

a. Postać wywołania:

```
[CALL] HEAD (lun, nagłówek)
[CALL] TEXT (lun, podpis)
```

gdzie:

lun - numer programowy urządzenia,
nagłówek, podpis - ciąg znaków (literał) o długości $\leqslant$ 40,
np.:

```
CALL HEAD (1,'WYKRES CZASOWY')
TEXT (1,'RYS.2.1. PRZEMIESZCZENIE X1')
```

b. Działanie:

Rozkazy powodują wyprowadzanie tekstów: nagłówek i podpis na wykresach sporządzanych przez graph plotter dla urządzeń REXT i REXY. Przy wyprowadzaniu wyników na drukarkę wierszową dla urządzeń TABLE, REXT i REXY wyprowadzany jest tylko nagłówek. W przypadku nie użycia tych rozkazów wyprowadzone są napisy standardowe:

- nagłówek:
  PROGRAM (name) REXT lun
- podpis:
  40 spacji /40 H␣/.

2.4.4. Rozkazy specjalne.

Rozkazy służą do:

1/ grupowego sterowania włączonymi urządzeniami peryferyjnymi /NEWRUN, ENDRUN/ ,

2/ wyboru dalszego toku przerwanych obliczeń /CONTI, ABORT, FINISH/.

A. Grupowe sterowanie urządzeniami peryferyjnymi - NEWRUN, ENDRUN.

a. Postać wywołania:

```
CALL NEWRUN
CALL  ENDRUN
```

rozkazy nie posiadają parametrów.

b. Działanie:

Rozkaz NEWRUN powoduje przewinięcie na początek zbiorów przyporządkowanych wszystkim włączonym urządzeniom peryferyjnym. Dotychczasowa zawartość tych zbiorów ulega wymazaniu.

Rozkaz ENDRUN powoduje wyprowadzenie wyników na drukarkę wierszową i graphplotter dla wszystkich włączonych urządzeń typu REXT, REXY, TABLE. Zawartość pamięci przyporządkowanej tym zbiorom nie ulega zmianie.

B. Reakcja na przerwanie obliczeń - CONTI, ABORT, FINISH.

a. Postać wywołania:

```
CALL CONTI
CALL ABORT
CALL  FINISH
```

rozkazy nie posiadają parametrów.

b. Działanie

Rozkazy CONTI i ABORT mogą być wywoływane w procedurze HOLD segmentu NUMERICAL. Procedura ta jest wywoływana po wzbudzeniu elementu HOLD w segmencie SYSTEM /p.2.3.2.3/ .

Rozkaz CONTI powoduje kontynuację przerwanych obliczeń symulacyjnych natomiast rozkaz ABORT zakończenie aktualnego przebiegu symulacyjnego.

Rozkaz FINISH służy do zakończenia wykonywania całego programu symulacyjnego i przekazuje sterowanie systemowi operacyjnemu maszy ny PDP 11/70.

2.4.5. Rozkazy sterujące sposobem wykonywania obliczeń.

Rozkazy MANUAL i DIGITAL służą do przekazywania sterowania obliczeń odpowiednio, do: terminala /monitora/ EMC lub procedury EXEC segmentu NUMERICAL.

Rozkaz RUNGE służy do określenia użytej metody i kroku całkowania numerycznego. Postać wywołania tego rozkazu jest następująca:

[CALL] RUNGE (nn, dt)

gdzie

nn - liczba całkowita /1, 2, 4/
nn = 1, metoda prostokątów,
nn = 2, metoda trapezów,
nn = 4, metoda RUNGE-Kutta 4,

dt - wyrażenie arytmetyczne typu REAL, np.:

CALL RUNGE (4, TK/100.0)

RUNGE (2, 0.01)

2.5. Zasady zapisu segmentów CONTROL i NUMERICAL.

2.5.1. Segment CONTROL.

Segment CONTROL rozpoczyna się słowem sterującym CONTROL, zaś kończy słowem rozpoczynającym kolejny segment programu

/NUMERICAL lub END - rys.2.2/. Ten ostatni przypadek równoznaczny jest z brakiem segmentu NUMERICAL.

W segmencie tym można umieszczać tylko rozkazy sterujące /p.2.4/, które wykonywane są kolejno, bez możliwości powrotu /skoku/ do raz wykonanego rozkazu. Napotkanie rozkazu MANUAL powoduje przekazanie sterowania na terminal, co umożliwia ręczne wprowadzanie rozkazów sterujących. Rozkaz DIGITAL powoduje wywołanie procedury EXEC segmentu NUMERICAL. Po zakończeniu wykonywania tej procedury sterowanie ponownie przekazywane jest segmentowi CONTROL.

Segment CONTROL w języku LAHYSS jest odpowiednikiem czynności wykonywanych ręcznie za pomocą pulpitu sterującego rzeczywistego systemu hybrydowego.

2.5.2. Segment NUMERICAL.

Segment NUMERICAL rozpoczyna się słowem sterującym NUMERICAL zaś kończy słowem sterującym END, kończącym cały program użytkowy /rys.2.2/. W segmencie zapisywane są poszczególne procedury podprogramu cyfrowego /EXEC, HOLD i procedury własne użytkownika/ .

A. Procedura EXEC.

Procedura służy do zapisu algorytmu badania modelowego układu dynamicznego w języku algorytmicznym FORTRAN. Procedura EXEC musi być zapisana w następującej postaci:

```
      SUBROUTINE EXEC
      INCLUDE 'ZMIENNE.FTN'

      Treść procedury zapisana w FORTRAN-ie

      RETURN
      END
```

Procedura nie posiada parametrów, bowiem wartości wszystkich zmiennych sygnałów i parametrów przenoszone są przez

obszary wspólne /COMMON/ opisane w zbiorze 'ZMIENNE.FTN' . Instrukcja INCLUDE, stanowiąca rozszerzenie języka FORTRAN /dla maszyn PDP/, powoduje dołączenie tego zbioru do procedury w trakcie kompilacji.

Procedura może zawierać wszystkie instrukcje FORTRAN-u, w tym również instrukcje deklaracji zmiennych. Należy jednak pamiętać, że obszarem ich działania będzie wyłącznie segment NUMERICAL.

Rozkazy sterujące zapisane w procedurze EXEC wywoływane są jako odpowiednie procedury, w kolejności wynikającej z zapisanego algorytmu.

Wyjście z procedury następuje zgodnie z zasadami FORTRAN-u tzn. po napotkaniu instrukcji RETURN. Sterowanie przekazywane jest wówczas do segmentu CONTROL lub, jeśli ostatnio zapisaną instrukcją było wywołanie rozkazu:

CALL FINISH

do systemu operacyjnego EMC.

Zakres pracy nie pozwala na szczegółowe omówienie programowania w języku FORTRAN. Ograniczymy się jedynie do podania kilku typowych przykładów, które znajdzie Czytelnik w rozdziałach 3 i 4.

B. Procedura HOLD.

Procedura HOLD wywoływana jest tylko w przypadku wzbudzenia co najmniej jednego z wejść elementu HOLD zadeklarowanego w segmencie SYSTEM. Zasady tworzenia i zapisu procedury wyjaśnimy na przykładzie.

Załóżmy, że deklaracja elementu HOLD ma postać:

HOLD (K1, L3)

Algorytm obsługi przerwań przewiduje, że jeśli:

1/ wzbudzone zostało wejście I /K1/ , to należy zmienić wartość parametru A i kontynuować obliczenia,

2/ wzbudzone zostało wejście II /L3/ , to należy wyprowadzić odpowiedni komunikat i kontynuować obliczenia,

3/ wzbudzone zostały oba wejścia, to należy przerwać

obliczenia.

Procedura realizująca taki algorytm obsługi przerwań przedstawiono na rys.2.35.

```
      SUBROUTINE HOLD
      INCLUDE 'ZMIENNE.FTN'
      IF(K1.AND.L3) GOTO 3
      IF(L3) GOTO 2
1     A=AMAX
      CALL CONTI
      RETURN
2     TYPE *,' L3=TRUE'
      CALL CONTI
      RETURN
3     CALL ABORT
      RETURN
      END
```

Rys.2.35. Przykład procedury HOLD.

C. Procedury własne użytkownika.

Procedury te pisane są zgodnie z wszystkimi zasadami FORTRAN-u i mogą być wywoływane tylko w segmencie NUMERICAL. Przenoszenie wartości zmiennych może następować trzema drogami:

1/ poprzez parametry aktualne,
2/ poprzez obszary COMMON zadeklarowane we współpracujących ze sobą procedurach,
3/ przez obszary COMMON zadeklarowane automatycznie w zbiorze ZMIENNE.FTN. W tym przypadku należy w procedurze umieścić omówioną już instrukcję INCLUDE.

3. Metodyka programowania w języku LAHYSS.

W rozdziale przedstawione zostaną sposoby realizacji w języku LAHYSS podstawowych problemów występujących w badaniach symulacyjnych układów dynamicznych. Do problemów tych należą:

- budowa modeli symulacyjnych badanych układów dynamicznych /p.3.1/,
- wprowadzanie i zadawanie wartości liczbowych parametrom /p.3.2/ ,
- wykorzystanie urządzeń elementów peryferyjnych /p.3.3/.

W rozdziale zostaną przedstawione zarówno uwagi ogólne dotyczące tych zagadnień, jak również przykłady programów napisanych w języku LAHYSS realizujące typowe zadania. Przykłady ilustrują również elastyczność programowania w języku LAHYSS, pokazują bowiem różne sposoby rozwiązania poszczególnych czynności, występujących w modelowaniu algorytmów hybrydowych.

3.1. Budowa modelu układu dynamicznego.

W języku LAHYSS możliwe są dwie metody budowy i zapisu modelu badanego układu dynamicznego:

1/ metoda równoważnego schematu analogowego /blokowa/ ,

2/ metoda zapisu równaniowego.

Pierwszy sposób polega na stworzeniu, a następnie zapisie schematu analogowego, jaki należałoby zbudować przy modelowaniu układu na rzeczywistej maszynie analogowej. W drugim przypadku model układu dynamicznego budowany jest wprost w oparciu o równania różniczkowe opisujące badany układ. W praktyce stosowana jest również metoda mieszana polegająca na budowie modelu, którego część zorientowana jest na zapis blokowy, a część na zapis równaniowy.

Przedstawiony podział dotyczy metod przygotowania i zapisu programów w języku LAHYSS i nie jest bezpośrednio związany z podziałem sposobów opisu układów dynamicznych, które mogą być opisane zarówno układem równań różniczkowych, jak i w postaci schematu blokowego. I tak, na przykład, układ dynamiczny opisany układem równań różniczkowych może być modelowany metodą równoważnego schematu analogowego.

A. Metoda równoważnego schematu analogowego.

Tok postępowania przy stosowaniu tej metody jest następujący:

1/ Przygotowanie opisu układu do budowy odpowiadającego mu schematu analogowego.

2/ Budowa równoważnego schematu analogowego w oparciu o dostępne elementy liczące.

3/ Oznaczenie /nazwanie/ wyjść /sygnałów/ wszystkich użytych w schemacie elementów oraz parametrów. Nazwy sygnałów i

i parametrów muszą odpowiadać warunkom przedstawionym w p.2.2.2.

4/ Zapis przygotowanego modelu w języku LAHYSS;

a/ deklaracja nazw sygnałów i parametrów,

b/ zapis poszczególnych elementów liczących występujących w schemacie analogowym.

Metodyka przygotowania i budowy schematów analogowych opisana jest szczegółowo w literaturze /[8] , [9] , [33]/ , dlatego też wyjaśnienie przedstawionego toku postępowania ograniczymy do pokazania kilku przykładów. W miejscu tym należy jedynie podkreślić, że przy modelowaniu cyfrowym schematu analogowego unika się konieczności skalowania poszczególnych elementów liczących. Przy korzystaniu z rzeczywistej maszyny analogowej czynności związane ze skalowaniem zajmują większą część czasu zużytego na opracowanie schematu analogowego.

Przykład 1.

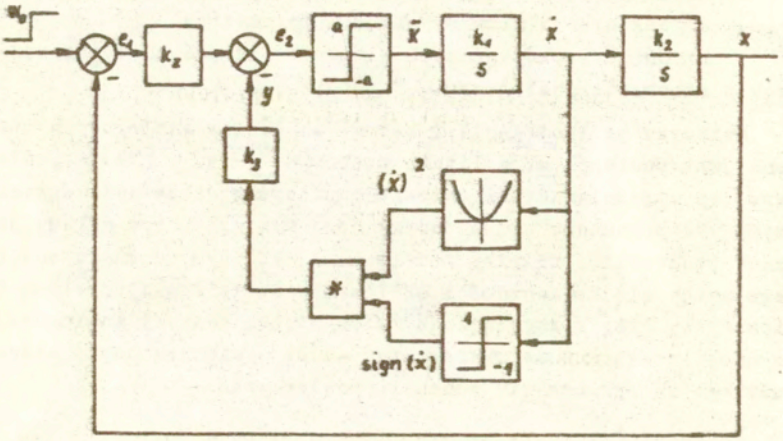
Badany jest czasooptymalny układ regulacji /[14]/ , którego schemat blokowy pokazano na rys.3.1. Przedmiotem badań jest zachowanie się układu pod wpływem wymuszenia skokowego. W układzie występują następujące parametry: W_0 /wartość wymuszenia/ , k_z, a, k_1, k_2, k_3 oraz sygnały: x, $\dot{x}$, $\ddot{x}$, e_1, e_2, y, $(\dot{x})^2$, sign $\dot{x}$ zależne od czasu.

Na podstawie schematu blokowego tworzymy równoważny mu schemat analogowy /rys.3.2/ i oznaczamy występujące w nim wielkości nazwami, które stanowią prostą transkrypcję symboli użytych do opisu schematu blokowego.

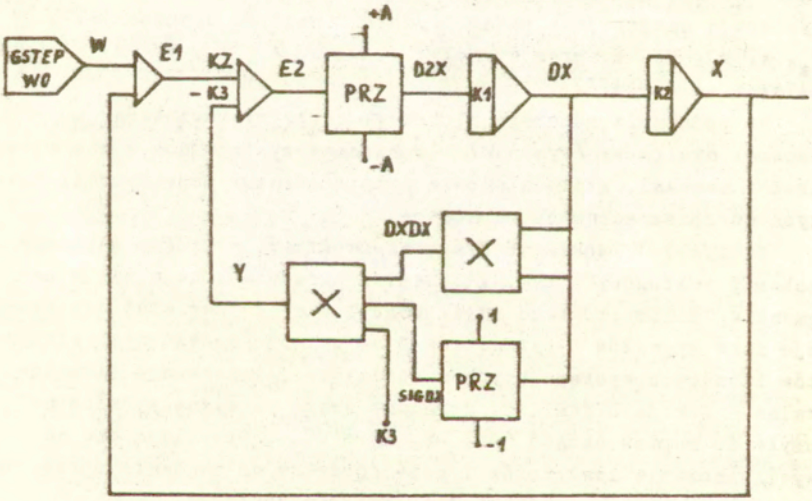
Na rys.3.3 pokazano fragment programu, w którym zapisany schemat analogowy z rys.3.2. Zapisu układu dokonuje się w segmencie SYSTEM programu użytkowego i musi on zawierać deklaracje nazw sygnałów i parametrów oraz opis poszczególnych elementów liczących występujących w schemacie /instrukcje strukturalne - sekcja DYNAMIC/ . Zwracamy uwagę na sygnały "złożone" użyte do zapisu układu /-X, -Y, KZ * E1/ . Pozwalają one na wyeliminowanie inwertorów i potencjometru co znacznie upraszcza schemat analogowy i odpowiednio program użytkowy.

Zapis układu można uprościć jeszcze bardziej, zastępując elementy Y, DXDX, SIGDX elementem AREX w postaci:

Y = AREX (K3 * DX * DX * SIGN (DX)).



Rys.3.1. Schemat blokowy układu czasooptymalnego.



Rys.3.2. Schemat analogowy dla układu czasooptymalnego.

```
SYSTEM

        ANALOG W,E1,E2,X,DX,D2X,DXDX,SIGDX
        PARAMETER  WO,K1,K2,K3,KZ,A

DYNAMIC

        X=INT(K2*DX,.0)
        DX=INT(K1*D2X,.0)
        D2X=PRZ(E2,-A,.0,A)
        E1=SUM(W,-X)
        E2=SUM(KZ*E1,-Y)
        Y=MNO(K3,DXDX,SIGDX)
        DXDX=MNO(DX,DX)
        SIGDX=PRZ(DX,-1.0,.0,1.0)
```

Rys.3.3. Model symulacyjny zapisany w języku LAHYSS opisujący układ z przykładu 1.

Przykład 2.

Rozważmy układ mechaniczny o jednym stopniu swobody z nieliniową charakterystyką sprężystości opisany równaniem:

$\ddot{x} + l\dot{x} + k\,(x + \mu x^3) = 0$, z warunkami początkowymi:

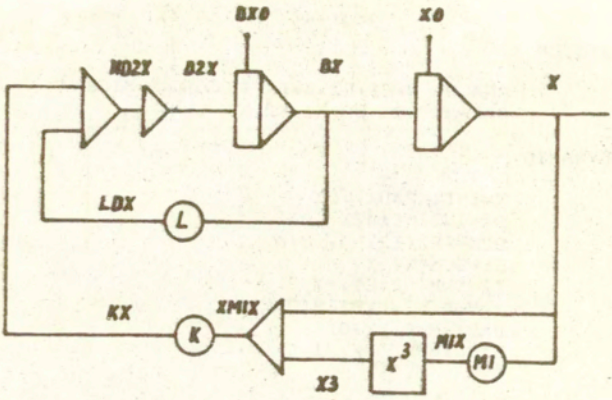
$x(0) = x_0$,

$\dot{x}(0) = \dot{x}_0$.

Dla układu tego można zbudować schemat analogowy przedstawiony na rys.3.4. Schemat ten przystosowany jest do realizacji na maszynie analogowej i zawiera wszystkie potrzebne do tego elementy /inwertory i potencjometry/ . Na rys.3.5 przedstawiono odpowiedni fragment programu w języku LAHYSS modelujący ten schemat.

Uwzględniając możliwości stosowania sygnałów złożonych można dla badanego układu dynamicznego zbudować znacznie uproszczony schemat analogowy /rys.3.6/ .

Dla takiego schematu analogowego odpowiadający mu fragment programu /rys.3.7/ jest znacznie krótszy w porównaniu z progra-



Rys.3.4. Pełny schemat analogowy dla układu nieliniowego /przykład 2/.

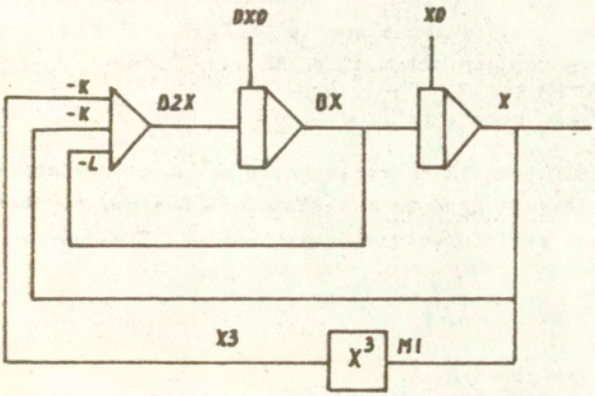
```
SYSTEM

        ANALOG X,DX,D2X,MIX,X3,XMIX,KX,LDX,NDX
        PARAMETER X0,DX0,L,K,MI

DYNAMIC

        X=INT(DX,X0)
        DX=INT(D2X,DX0)
        D2X=INW(ND2X)
        ND2X=SUM(KX,LDX)
        KX=POT(K,XMIX)
        XMIX=SUM(X,X3)
        X3=POW(MIX,3.0)
        MIX=POT(X,MI)
        LDX=POT(L,DX)
```

Rys.3.5. Model symulacyjny dla schematu z rys.3.4.



Rys.3.6. Uproszczony schemat analogowy dla układu nieliniowego /przykład 2/.

```
SYSTEM

        ANALOG X,DX,D2X,X3
        PARAMETER X0,DX0,L,K,MI

DYNAMIC

        X=INT(DX,X0)
        DX=INT(D2X,DX0)
        DZX=SUM(-K*X,-L*DX,-K*X3)
        X3=POW(MI*X,3.0)
```

Rys.3.7. Model symulacyjny dla schematu z rys.3.6.

mem z rys.3.5.

Omawiany układ można również zamodelować metodą zapisu równaniowego, z wykorzystaniem elementu AREX. W tym celu przekształcamy równanie opisujące układ od postaci:

$$\ddot{x} = -l\dot{x} - k(x + \mu x^3).$$

Element AREX pozwala na zapis wzoru na x w postaci wyrażenia arytmetycznego zgodnie z zasadami FORTRAN-u, co jeszcze bardziej skraca zapis modelu w języku LAHYSS /rys.3.8/ .

```
SYSTEM

        ANALOG X,DX,D2X
        PARAMETER X0,DX0,L,K,MI

DYNAMIC

        X=INT(DX,X0)
        DX=INT(D2X,DX0)
        D2X=AREX(-L*DX-K*(X+MI*X**3))
```

Rys.3.8. Model symulacyjny dla przykładu 2 /metoda zapisu równaniowego/.

B. Metoda zapisu równaniowego.

Metoda zapisu równaniowego jest szczególnie przydatna do zapisu modeli układów mechanicznych o wielu stopniach swobody, które opisywane są układami równań różniczkowych zwyczajnych w postaci /p.1.1/ :

$$m_i\ddot{x}_i + \varphi_i(\ldots,\dot{x}_i,\ldots l_i) + f_i(\ldots,x_i,\ldots k_i) = P_i \cos \gamma t.$$

z warunkami początkowymi:

$$x_i(0) = x_{oi}$$

$$\dot{x}_i(0) = \dot{x}_{oi}, \qquad i = 1,\ldots,n$$

Przygotowanie modelu symulacyjnego dla takiego układu przebiega następująco:

1/ Przekształcenie układu równań drugiego rzędu do równoważnego układu równań pierwszego rzędu w postaci:

$$\begin{cases} \frac{dx_i}{dt} = dx_i \;, & x_i(0) = x0_i \\ \frac{d\,dx_i}{dt} = d2x_i \;, & dx_i(0) = dx0_i \\ d2x_i = \frac{1}{m_i}\,(P_i \cos \gamma t - \varphi_i - f_i) \;, \end{cases}$$

$i = 1,\ldots,n.$

2/ Deklaracje zmiennych i parametrów:

ANALOG X1,..., XN, DX1,..., DXN, D2X1,...,D2XN

PARAMETER L1,...,LS, K1,...,KS, NI

3/ Bezpośredni zapis układu z punktu 1/ w języku LAHYSS wykorzystując następujące elementy liczące:

a/ INT - do modelowania równania różniczkowego I rzędu:

x_i = INT (DX_i, $X0_i$)

DX_i = INT ($D2X_i$, $DX0_i$)

b/ AREX - do zapisu wyrażenia na drugą pochodną:

$D2X_i$ = AREX (1/M_i * (P_i * COS(NI * T) - ...))

W praktyce często stosuje się dodatkowe elementy AREX do modelowania wspólnych członów występujących w poszczególnych równaniach opisujących badany układ lub w przypadku gdy wyrażenie nie da się zapisać w jednym wierszu programu. Z przypadkiem takim mamy do czynienia w kolejnych przykładzie.

Przykład 3.

Badany jest układ mechaniczny, w którym występują drgania

parametryczne, opisany układem równań różniczkowych:

$$\ddot{x}_1 + \omega_1^2 x_1 + \beta_{11}\dot{x}_1 + \beta_{12}\dot{x}_2 + (\lambda_{11} x_1 + \lambda_{12} x_2)\cos 2\nu t = 0$$

$$\ddot{x}_2 + \omega_2^2 x_2 + \beta_{21}\dot{x}_1 + \beta_{22}\dot{x}_2 + (\lambda_{21} x_1 + \lambda_{22} x_2)\cos 2\nu t = 0$$

gdzie:

$x_i, \dot{x}_i, \ddot{x}_i$ - współrzędne główne i ich pochodne $i = 1,2$,

β_{ij}, λ_{ij} - parametry układu $(i,j = 1,2)$,

ω_1, ω_2 - stałe odpowiadające częstościom własnym układu liniowego,

2ν - częstość wymuszenia parametrycznego.

Model symulacyjny dla badanego układu przedstawiony jest na rys.3.9. Łatwo zauważyć, że zbudowanie modelu symulacyjnego

```
SYSTEM

        ANALOG X1,DX1,D2X1,X2,DX2,D2X2
        ANALOG BR1,BR2,LR1,LR2
        PARAMETER  OM1,OM2,B11,B12,B21,B22
        PARAMETER LAM11,LAM12,LAM21,LAM22,NI

DYNAMIC

        X1=INT(DX1,.0)
        DX1=INT(D2X1,.0)
        WYM=AREX(COS(2*NI*T))
        BR1=AREX(B11*DX1+B12*DX2)
        LR1=AREX((LAM11*X1+LAM12*X2)*WYM)
        D2X1=AREX(-OM1*OM1*X1-BR1-LR1)
        X2=INT(DX2,.0)
        DX2=INT(D2X2,.0)
        BR2=AREX(B21*DX1+B22*DX2)
        LR2=AREX((LAM21*X1+LAM12*X2)*WYM)
        D2X2=AREX(-OM2*OM2*X2-BR2-LR2)
```

Rys.3.9. Model symulacyjny dla przykładu 3.

w oparciu o równoważny schemat analogowy wymagałoby dużej ilości elementów liczących, a co za tym idzie, znacznie wydłużyłoby program.

3.2. Wprowadzanie danych liczbowych.

W języku LAHYSS istnieją trzy sposoby wprowadzania danych liczbowych dla symulowanego modelu badanego układu dynamicznego:

1/ wprowadzenie poprzez segment CONTROL,
2/ wprowadzenie programowe poprzez segment NUMERICAL,
3/ wprowadzenie ręczne.

Wybór sposobu wprowadzania danych zależy przede wszystkim od roli jaką spełnia parametr w symulowanym algorytmie. Dla parametrów rzadko zmienianych /stałych/ w badanym układzie najwygodniej jest określić jego wartość na początku segmentu CONTROL. W początkowym stadium badań symulacyjnych, gdy nie znamy jeszcze przedziałów zmienności interesujących nas parametrów, korzystne jest ręczne wprowadzanie wartości parametrów. Natomiast w przypadku, gdy parametr zmieniany jest zgodnie z zadanym algorytmem badania układu /np. przy obliczeniach optymalizacyjnych/, lub gdy należy przeprowadzić obliczenia dla wielu wartości liczbowych wybranych parametrów, stosujemy wprowadzanie programowe.

Sposób realizacji poszczególnych metod wprowadzania danych omówimy na przykładzie, w którym wykorzystamy wszystkie przedstawione metody.

Przykład 4.

Rozważmy układ opisany już w przykładzie 3.
Przedmiotem badań jest określenie obszarów, w których odpowiedź układu jest stateczna. Obszary wyznaczane będą na płaszczyźnie (γ, λ_1), przy założeniu:

$$\lambda_{11} = \lambda_{22} = \lambda_1,$$

$$\lambda_{12} = \lambda_{21} = \lambda_2.$$

```
CONTROL

        LET(OM1=1.0)
        LET(OM2=2.0)
        DIGITAL

NUMERICAL

        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'

        ACCEPT *,B11,B12,B21,B22,AL2,AL1MIN,AL1MAX
        LAM12=AL2
        LAM21=AL2

        AL1=AL1MIN
10      IF(AL1.GT.AL1MAX) GO TO 30

        DO 20  J=0,20
        NI=J*0.1
20      CALL RUN(20.0)

        AL1=AL1+0.1
        LAM11=AL1
        LAM22=AL1
        GO TO 10

30      CONTINUE
```

Rys.3.10. Fragment programu symulacyjnego realizujący wprowadzanie danych /przykład 4/.

dla różnych wartości parametrów β_{ij}. Parametry ω_i są stałe dla wszystkich wykresów. W przedstawionym na rys.3.10 fragmencie programu zawarte są tylko instrukcje wyprowadzania i obliczania parametrów, pominięto natomiast sposób wyznaczania charakteru odpowiedzi oraz instrukcje sterujące rejestarcją wyników.

Parametry ω_1 i ω_2 jako stałe określane są w segmencie CONTROL /instrukcja LET/ , pozostałe wprowadzane są programowo w segmencie NUMERICAL. Parametry β_{ij} /B11, B12, B21, B22/ oraz λ_2 (AL2) i granice zmienności $\lambda_{1min}, \lambda_{1max}$ parametru λ_1 / AL1MIN, AL2MAX/ wczy-tywane są w formacie swobodnym z terminala użytkownika /[30]/.

Parametr λ_2 /AL2/ oraz y /NI/ zmieniane są w petlach programowych procedury EXEC segmentu NUMERICAL z stałym krokiem 0.1 w przedziałach

$$\lambda_{1min} \leq \lambda_1 \leq \lambda_{1max} \qquad \text{oraz}$$
$$0 \leq y \leq 2 ,$$

przy czym do zmian parametru y użyto pętli DO, natomiast dla λ_1 ciągu instrukcji FORTRAN-u.

3.3. Wykorzystanie urządzeń peryferyjnych.

Urządzenia peryferyjne opisane w p.2.3. oraz rozkazy sterujące tymi urządzeniami opisane w p.2.4 zapewniają zarówno możliwość pełnej dokumentacji przeprowadzonych badań symulacyjnych jak i realizację złożonych algorytmów, wykorzystujących urządzenia peryferyjne jako pamięć przebiegów analogowych. W języku LAHYSS istnieje również możliwość wyprowadzania wyników w postaci wydruków, których format określany jest przez użytkownika zgodnie z zasadami FORTRAN-u /[1] , [30]/. Podane dalej przykłady ilustrują różne sposoby wykorzystania urządzeń peryferyjnych i rejestracji wyników symulacji.

Przykład 5.

W układzie omówionym w przykładzie 2 /p.3.1/ należy zarejestrować przebiegi czasowe sygnału x(t) oraz wykres na płaszczyźnie fazowej $(x, \dot{x})$. Zadanie to realizuje program pokazany na

```
PROGRAM (UKLAD NIELINIOWY)

SYSTEM
        Deklaracje sygnałów i parametrów

DYNAMIC
        Opis modelu układu dynamicznego

PERIPHERAL

        REXT (1,X)
        REXT (2,X,DX)

CONTROL
        Wprowadzenie danych liczbowych

        ACCESS (1,0.01)
        HEAD(1,'WYKRES CZASOWY')
        ACCESS (1,0.01)
        HEAD(2,'WYKRES FAZOWY')
        RUN(10.0)
        PLOT(1)
        PLOT(2)

END
```

Rys.3.11. Program symulacyjny do przykładu 5.

rys.3.11. W programie tym pominięto instrukcje nieistotne z punktu widzenia wykorzystania urządzeń peryferyjnych, zastępując je opisem wykonanym pismem odręcznym.

Wyniki symulacji wyprowadzone będą w postaci odpowiednich wykresów zarówno na drukarkę wierszową /rozkaz RUN/, jak i graph plotter /rozkaz PLOT/ . Wykresy opisane będą nagłówkami odpowiednio:

```
PROGRAM  UKŁAD NIELINIOWY                    REXT 1
WYKRES CZASOWY
i
PROGRAM  UKŁAD NIELINIOWY                    REXY 2
WYKRES FAZOWY
```

Skalowanie wykresów odbędzie się automatycznie /nie użyto rozkazu SCALE/ . Ten sposób skalowania jest wykorzystywany w przy-padkach, gdy:

1/ Nie znamy wartości ekstremalnych maksymalnej i minimalnej rejestrowanych sygnałów,

2/ przebiegi rejestrowane są dla wartości parametrów o dużym przedziale zmienności i zależy nam bardziej na dokładnym zbadaniu charakteru przebiegów niż na ich porównaniu; skalowanie automatyczne zapewnia bowiem "rozciągnięcie" przebiegów na całą płaszczyznę wykresu.

Przykład 6.

Przy modelowaniu algorytmów iteracyjnych często stosowany jest następujący sposób rejestracji wyników: jeden wybrany sygnał rejestrowany jest dla wszystkich przebiegów iteracyjnych, natomiast pozostałe interesujące użytkownika wielkości rejestrowane są jednokrotnie, po zakończeniu obliczeń iteracyjnych. Taka rejestracja wyników zapewnia możliwość oceny zbieżności algorytmu iteracyjnego przy oszczędnym wykorzystaniu pamięci i większej szybkości obliczeń symulacyjnych. Pełna dokumentacja wyników wykonywana jest tylko dla wielkości wyznaczonych przez algorytm iteracyjny.

Załóżmy, że modelujemy algorytm iteracyjny badania układu dynamicznego, w którym interesują nas przebiegi czasowe sygnałów X1, X2, X3 oraz wykres fazowy na płaszczyźnie (X2, DX2).

```
PROGRAM  (ALGORYTM  ITERACYJNY)
SYSTEM
        deklaracje sygnałów i parametrów
DYNAMIC
        Opis modelu badanego układu
PERIPHERAL

        REXT (1,DX2)
        REXT (2,X1,X2,X3)
        REXY (3,X2,DX2)

CONTROL
        wprowadzenie danych liczbowych
        DIGITAL

NUMERICAL

        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'
        wczytanie parametrów dla obliczeń iteracyjnych
        CALL ACCESS(1,0.01)

10      CALL COMPUTE (TK)
        TYPE *, 'DX2K=',DX2
        IF(ABS(DX2-DXK).LE.EPS) GO TO 20
        obliczenia nowych wartości parametrów dla kolejnej iteracji
        GO TO 10
20      CALL PLOT(1)
        CALL CLOSER (1)
        CALL ACCESS(2,0.01)
        CALL ACCESS(3,0.01)
        CALL SOLVE(TK)
        CALL PLOT(2)
        CALL PLOT(3)
        RETURN
        END

END
```

Rys.3.12. Program symulacyjny algorytmu iteracyjnego /przykład 6/.

Warunkiem zakończenia obliczeń iteracyjnych jest osiągnięcie przez sygnał DX2 wartości DXK po czasie TK sekund, interesują nas również przebiegi czasowe sygnału DX2 w poszczególnych iteracjach. Na rys.3.12 pokazany jest program realizujący to zadanie.

Sterowanie obliczeniami i urządzeniami peryferyjnymi odbywa się z segmentu NUMERICAL, do którego sterowanie przekazane zostało rozkazem DIGITAL w segmencie CONTROL. Działanie programu jest następujące:

1/ wprowadzanie danych liczbowych,

2/ włączenie urządzenia REXT1, na którym rejestrowane będą przebiegi sygnału DX2 /urządzenia REXT2 i REXY3 pozostają wyłączone/,

3/ wykonanie jednego przebiegu symulacyjnego,

4/ wyprowadzenie na monitor końcowej wartości sygnału DX2,

5/ Sprawdzenie warunku zakończenia obliczeń iteracyjnych i, jeśli spełniony, przejście do p-tu 7,

6/ obliczenie nowych parametrów dla kolejnej iteracji i przejście do wykonania p-tu 9,

7/ wykonanie wykresów na graph-plotterze dla rejestrowanych na urządzeniu REXT1 przebiegów sygnału DX2 dla wszystkich iteracji i odłączenie urządzenia REXT1,

8/ włączenie urządzeń REXT2 i REXY3,

9/ wykonanie jednego przebiegu symulacyjnego dla ost anio obliczonych wartości parametrów,

10/ wykonanie wykresów na graph-plotterze dla przebiegów zarejestrowanych na urządzeniach REXT2 i REXY3.

Przykład 7.

W przykładzie przedstawimy sposób wykorzystania urządzeń peryferyjnych do rozwiązania zadania identyfikacji układu dynamicznego.

Załóżmy, że na urządzeniu INPUT1 zarejestrowany został przebieg czasowy XR na wyjściu identyfikowanego układu, którego struktura jest znana. Należy dobrać parametry układu tak, by przebieg na jego wyjściu X był zgodny z przebiegiem XR. Jako kryterium oceny zgodności przebiegów przyjmujemy zależność:

```
PROGRAM (IDENTYFIKACJA)

SYSTEM
        Deklaracje sygnałów i parametrów
DYNAMIC
        Opis struktury identyfikowanego układu
C       OBLICZANIE  WARTOSCI KRYTERIUM

        EPS=SUM(XR,-X)
        Q=INT(EPS*EPS,.0)

PERIPHERAL
        Deklaracje urządzeń rejestracji wyników
        INPUT(1,XR)

CONTROL

        DIGITAL

NUMERICAL

        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'
        Wprowadzenie danych liczbowych
        CALL ACCESS (1,0.01)
10      CALL BACK (1)
        CALL COMPUTE (TK)
        IF (Q.LE.QMIN) GO TO 20
        Obliczenie nowych parametrów

        GO TO 10
20      CONTINUE
        Wyprowadzenie wyników symulacji
        RETURN
        END

END
```

Rys.3.13. Program symulacyjny dla zadania identyfikacji /przykład 7/

$$Q = \int_0^{t_k} (XR(t) - x(t))^2 \, dt = \min.$$

Na rys.3.13 pokazany jest program symulacji takiego zadania.

Na programie tym umieszczono w sposób jawny, tylko instrukcje istotne dla omawianego algorytmu. W sekcji DYNAMIC segmentu SYSTEM pokazano tylko sposób realizacji obliczenia kryterium użytego do oceny zgodności przebiegów, zaś w sekcji PERIPHERAL tylko deklarację elementu INPUT, w którym zarejestrowany jest przebieg XR w identyfikowanym układzie. Element INPUT pracuje w przedstawionym programie jak magnetofon, który jest przewijany /instrukcja CALL BACK(1)/ przed każdym przebiegiem symulacyjnym. Przy korzystaniu z tego elementu należy pamiętać o zapewnieniu zgodności odstępu próbkowania czytania rejestrowanych sygnałów określanego instrukcją ACCESS, z odstępem jaki był użyty do rejestracji sygnałów. W kolejnym przykładzie pokażemy jeden z wielu możliwych sposobów rejestracji sygnałów dla urządzenia INPUT.

Przykład 8.

Przy rozpatrywaniu procesu zużycia szyn kolejowych spotykamy się z następującym zjawiskiem /ze zrozumiałych względów pominiemy opis matematyczny procesu/ . Przy najechaniu na złącze między szynami prędkość kątowa koła wagonu ulega zaburzeniu. Powoduje to ślizganie się koła po szynie i jej zużywanie. Powstałe w wyniku "wyrwy" powodują dodatkowe zaburzenia przy przejazdach kolejnych kół i, co za tym idzie, zwiększone zużycie.

Przy badaniu tego typu procesu zachodzi konieczność rejestracji przebiegu zużycia przy przejeździe koła przez złącze i wprowadzenie tego przebiegu jako wymuszenia przy następnym przejeździe.

Na rys.3.14 pokazany jest program wykorzystujący elementy INPUT i OUTPUT do tego celu. Sygnał PROF odpowiada profilowi szyny przed przejazdem koła, zaś sygnał NPROF profilowi po przejeździe. Sygnał PROF jest czytany z urządzenia INPUT, a nowo powstały profil rejestrowany jest w urządzeniu OUTPUT. Po zakończeniu jednego przebiegu symulacyjnego następuje przepisanie zawartości pamięci urządzenia OUTPUT do urządzenia INPUT

/rozkaz COPY/, co pozwala na badanie następnego przejazdu. Poszczególne profile rejestrowane są również na urządzeniu REXT w ten sposób, że przepisywane są na nie przebiegi zarejestrowane przez urządzenie OUTPUT. Przepisanie to następuje po każdym przebiegu symulacyjnym /rozkaz APPEND/.

```
PERIPHERAL

        INPUT (1,PROF)
        OUTPUT (2,NPROF)
        REXT (3,NPROF)

CONTROL

        DIGITAL

NUMERICAL

        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'

*       wprowadzanie danych liczbowych

        CALL ACCESS (1,DTOUT)
        CALL ACCESS (2,DTOUT)

10      CALL COMPUTE (TK)
        CALL COPY (2,1)
        CALL BACK (1)
        CALL APPEND (2,3)
        IF ( koniec badan      ) GO TO 20
        GO TO 10

20      CALL PLOT(3)

        RETURN
        END
```

Rys.3.14. Fragment programu symulacyjnego do przykładu 8.

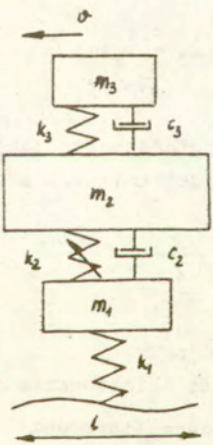
4. Przykłady programów użytkowych.

W rozdziale przedstawione będą dwa przykłady pełnych programów użytkowych napisanych w języku LAHYSS. Pierwszy program /4.1/ służy do wyznaczania charakterystyk amplitudowo-częstościowych modelu pojazdu samochodowego. Wyznaczanie tych charakterystyk stanowi jedno z podstawowych narzędzi badań własności dynamicznych układów. W p.4.2. przedstawiono program symulacyjny rozwiązania zadania brzegowego. Program ten ilustruje na prostym przykładzie sposób realizacji algorytmu iteracyjnego.

Przedstawione przykłady zawierają: krótkie omówienie teoretyczne zadania, tekst programu symulacyjnego napisanego w języku LAHYSS oraz wyniki w postaci wykresów wykonywanych przez graphplotter. Ze względów technicznych /trudności w powieleniu/ nie została zamieszczona dokumentacja działania programów, uzyskana na drukarce wierszowej.

4.1. Wyznaczanie charakterystyk amplitudowo-częstotliwościowych.

Przedmiotem badań jest uproszczony model pojazdu samochodowego poruszającego się po falistej nawierzchni /rys.4.1./.



Rys.4.1. Uproszczony model pojazdu samochodowego.

Oznaczenia:

m_1 - masa zawieszenia (nieresorowana),
m_2 - masa podwozia pojazdu (resorowana),
m_3 - masa kierowcy wraz z fotelem,
k_1 - stała sprężystość opon,
k_2, c_2 - stała sprężystości i tłumienia amortyzatorów,
k_3, c_3 - stała sprężystości i tłumienia fotela kierowcy.

Dla sprężystości amortyzatora przyjęto nieliniową charakterystykę typu $k_2 (x + \mu x^3)$.

Model ten opisany jest układem trzech nieliniowych równań różniczkowych zwyczajnych drugiego rzędu:

$$m_1\ddot{x}_1 + k_1x_1 + k_2(x_1 - x_2) + k_2\mu(x_1 - x_2)^3 + c_2(\dot{x}_1 - \dot{x}_2) =$$

$$= bk_1 \cos\omega t$$

$$m_2\ddot{x}_2 + k_2(x_2 - x_1) + k_2\mu(x_2 - x_1)^3 + c_2(\dot{x}_2 - \dot{x}_1) +$$

$$+ k_3(x_2 - x_3) + c_3(\dot{x}_2 - \dot{x}_3) = 0$$

$$m_3\ddot{x}_3 + k_3(x_3 - x_2) + c_2(\dot{x}_3 - \dot{x}_2) = 0 ,$$

gdzie:

b - amplituda pofałdowania jezdni,
ω - częstość wymuszenia wyznaczana z wzoru:

$$\omega = \frac{2\pi v}{l}$$

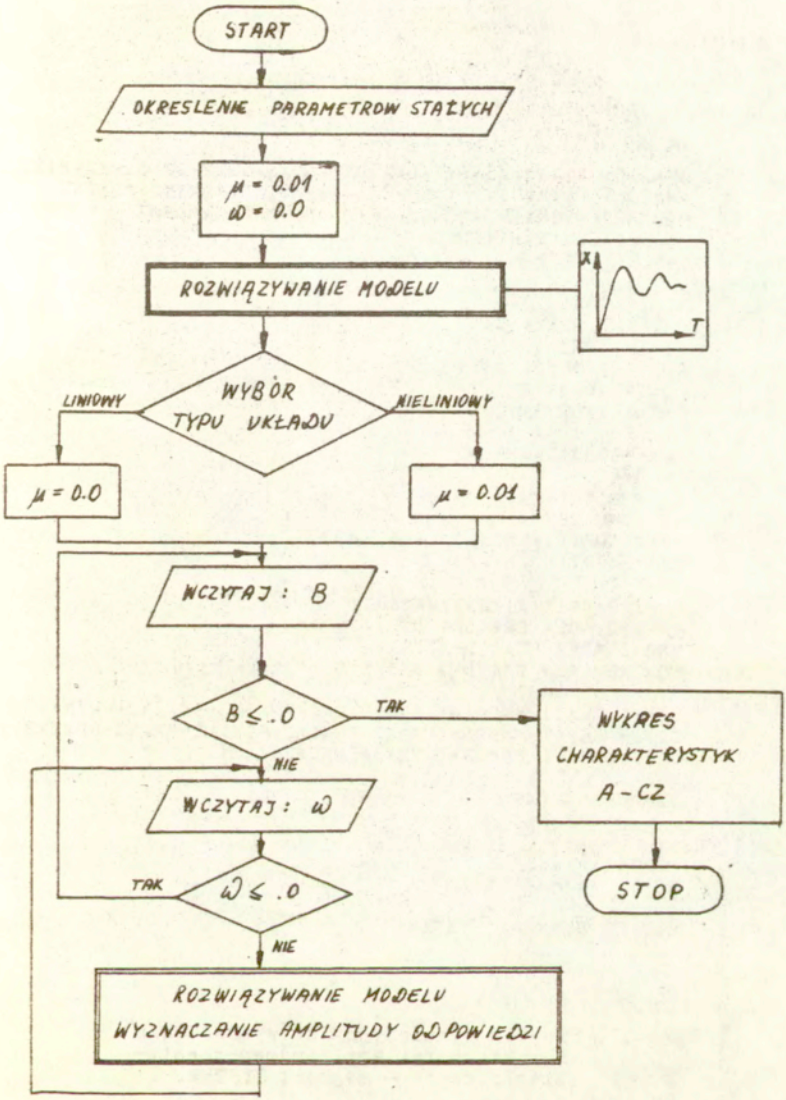
gdzie:

v - prędkość pojazdu,
l - długość fali pofałdowania jezdni,

Przyjęto następujące dane liczbowe:

m_1 = 700 kg m_2 = 600 kg m_3 = 100 kg

k_1 = 390000 N/m k_2 = 36000 N/m k_3 = 10000 N/m



Rys. 4.2. Algorytm badania modelu pojazdu samochodowego.

```
PROGRAM(POJAZD)

SYSTEM
        ANALOG  X1,X2,X3,DX1,DX2,DX3,D2X1,D2X2,D2X3,NX2,X1X2
        ANALOG  PWYM,PX1,PX1X2,PX2X3,PNX1X2,PDX1X2,PDX2X3
        PARAMETER  K1,K2,K3,C2,C3,MASA1,MASA2,MASA3
        PARAMETER  B,MI,OMEGA

DYNAMIC
        X1=INT(DX1,.0)
        DX1=INT(D2X1,.0)
        X2=INT(DX2,.0)
        DX2=INT(D2X2,.0)
        X3=INT(DX3,.0)
        DX3=INT(D2X3,.0)

        PWYM=GCOS(B,OMEGA)

        PX1=POT(K1,X1)
        NX2=INW(X2)
        X1X2=SUM(X1,NX2)
        PX1X2=POT(K2,X1X2)

        PNX1X2=AREX(X1X2**3*K2*MI)
        PDX1X2=AREX(C2*(DX1-DX2))
        PX2X3=AREX(K3*(X2-X3))
        PDX2X3=AREX(C3*(DX2-DX3))

        D2X1=AREX(1/MASA1*(-PX1-PX1X2-PNX1X2-PDX1X2+PWYM))
        D2X2=AREX(1/MASA2*(PX1X2+PNX1X2+PDX1X2-PX2X3-PDX2X3))
        D2X3=AREX(1/MASA3*(PX2X3+PDX2X3))

        X3MAX=XMAX(X3)

PERIPHERAL
        REXT(1,X1,X3)
        REXY(2,X3MAX)

C       END OF SEGMENT SYSTEM
```

Rys.4.3. Program symulacyjny do wyznaczania charakterystyk amplitudowo-częstotliwościowych - segment SYSTEM.

```
CONTROL

        RUNGE(4,.001)
        LET(MASA1,100.)
        MASA2=600
        MASA3=100
        K1=390 000
        K2=36000
        K3=10000
        C2=1200
        C3=400
        ACCESS(1,0.01)
        TEXT(1,'MODEL NIELINIOWY')
        MI=.01
        OMEGA=.0
        B=.02
        RUN(10.0)
        CLOSER(1)
        DIGITAL

C       END OF SEGMENT CONTROL

NUMERICAL

        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'
        TK=10.0
        CALL ACCESS(2,TK)
C
C       WYBOR UKLADU : 1 - UKLAD LINIOWY ; 2 - NIELINIOWY
C
        ACCEPT *,ITYP
        GOTO (10,20), ITYP
10      MI=.0
        CALL TEXT(2,'UKLAD LINIOWY')
        GOTO 100
20      MI=.01
        CALL TEXT(2,'UKLAD NIELINIOWY')
100     ACCEPT *,B
        IF(B.LE.0.0) GOTO 200
110     ACCEPT *,OMEGA
        IF(OMEGA.LE0.0) GOTO 100
        CALL PREPARE
        CALL INIT(X3MAX,.0)
        CALL SOLVE(TK)
        GOTO 110
200     CALL PLOT(2)
        END

C       END OF SEGMENT NUMERICAL

END
```

Rys.4.4. Program symulacyjny do wyznaczania charakterystyk amplitudowo-częstotliwościowych - segmenty CONTROL i NUMERICAL.

$c_2 = 1200$ Ns/m $\qquad$ $c_3 = 400$ Ns/m

$\mu = 0.01$
$l = 4$ m .

Przedmiotem badań jest znalezienie wpływu amplitudy wymuszenia b pochodzącej od nierówności drogi na zachowanie się samochodu przy różnych prędkościach v . W tym celu należy wyznaczyć charakterystyki amplitudowo-częstotliwościowe /krzywe rezonansowe/ układu, na których przedstawiona jest zależność amplitudy odpowiedzi układu dla wybranego punktu w funkcji częstości wymuszenia.

Dla amplitudy wymuszenia przyjęto przedział zmienności:

$b = 0 - 2$ cm, zaś prędkości

$v = 0 - 15$ km/h, co odpowiada zakresowi częstości

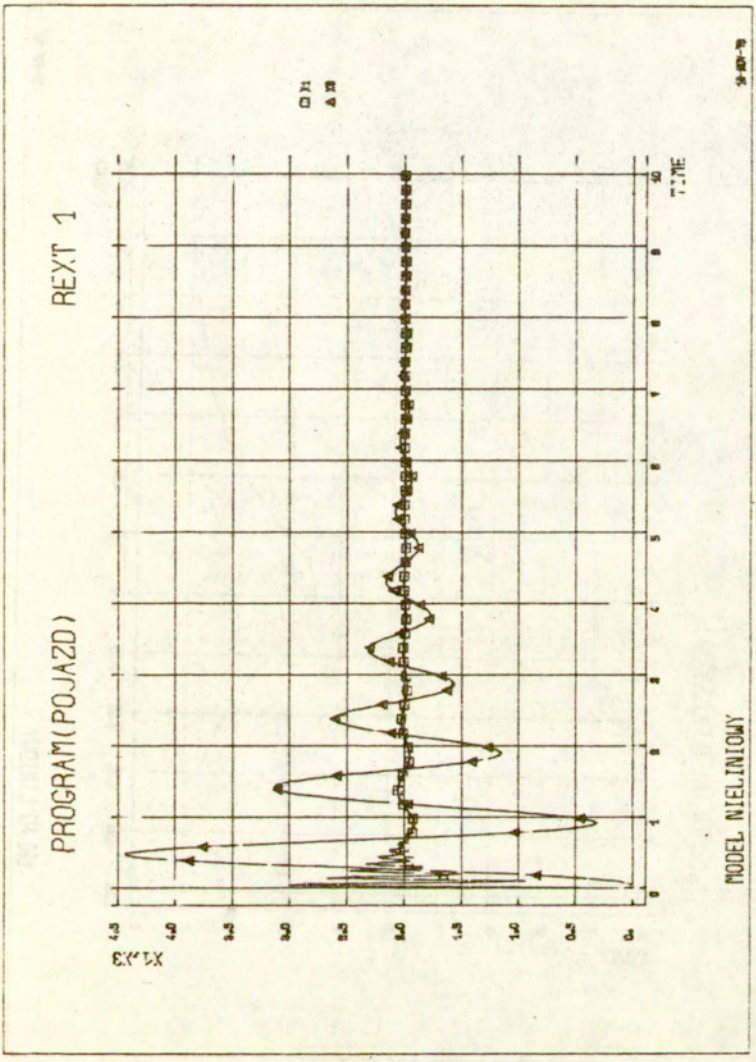
$\omega = 0 - 10$ rd/s.

Na rys.4.2. pokazany jest algorytm hybrydowy dla tak postawionego zadania. Składa się on z dwóch etapów.

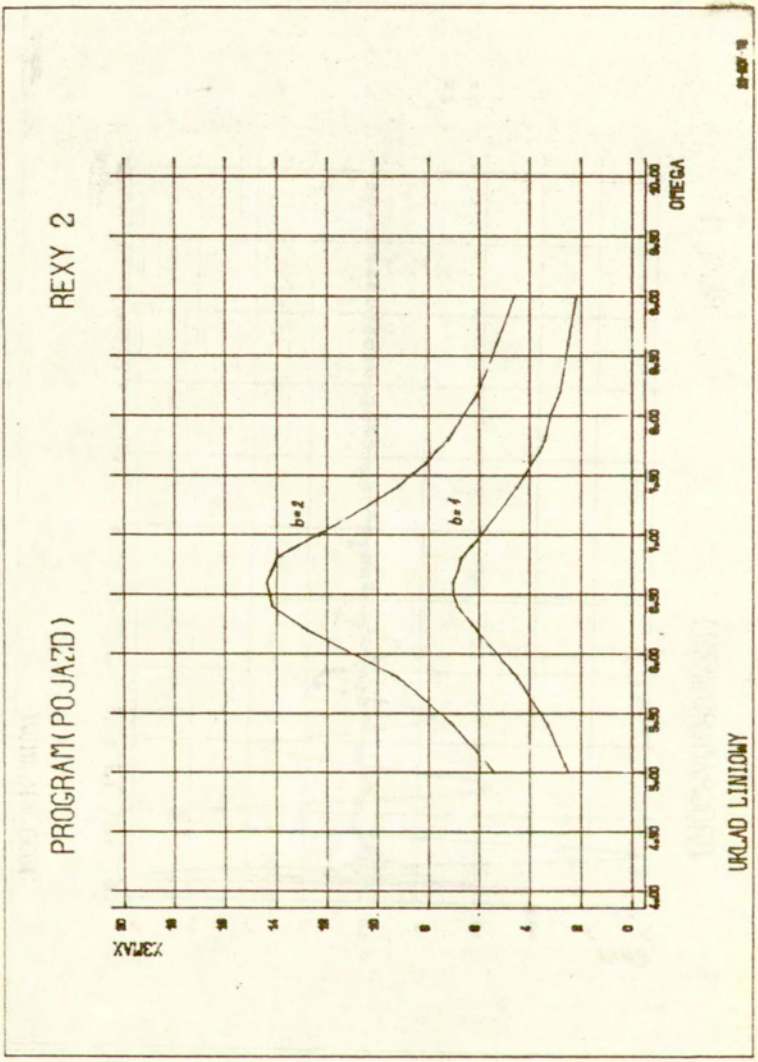
Pierwszy polega na wyznaczeniu czasu zanikania przebiegów nieustalonych w samochodzie. Jest to konieczne do określenia czasu obserwacji układu przy pomiarach amplitud, których można dokonać tylko w stanie ustalonym odpowiedzi na wymuszenie harmoniczne.

Drugi etap służy do właściwego wyznaczania charakterystyk amplitudowo-częstotliwościowych układu. Algorytm przewiduje możliwość wyboru typu modelu pojazdu samochodowego. Układ liniowy powstaje przez pominięcie nieliniowej części charakterystyki sprężyny k_2 /$\mu = 0$/ . Wartości parametrów b i ω zczytywane są z terminala, przy czym wprowadzenie ujemnej ω powoduje przejście do czytania kolejnego b . Wprowadzenie ujemnego b powoduje zakończenie symulacji i wykonanie wykresu charakterystyk amplitudowych na podstawie wyników zarejestrowanych w poszczególnych przebiegach.

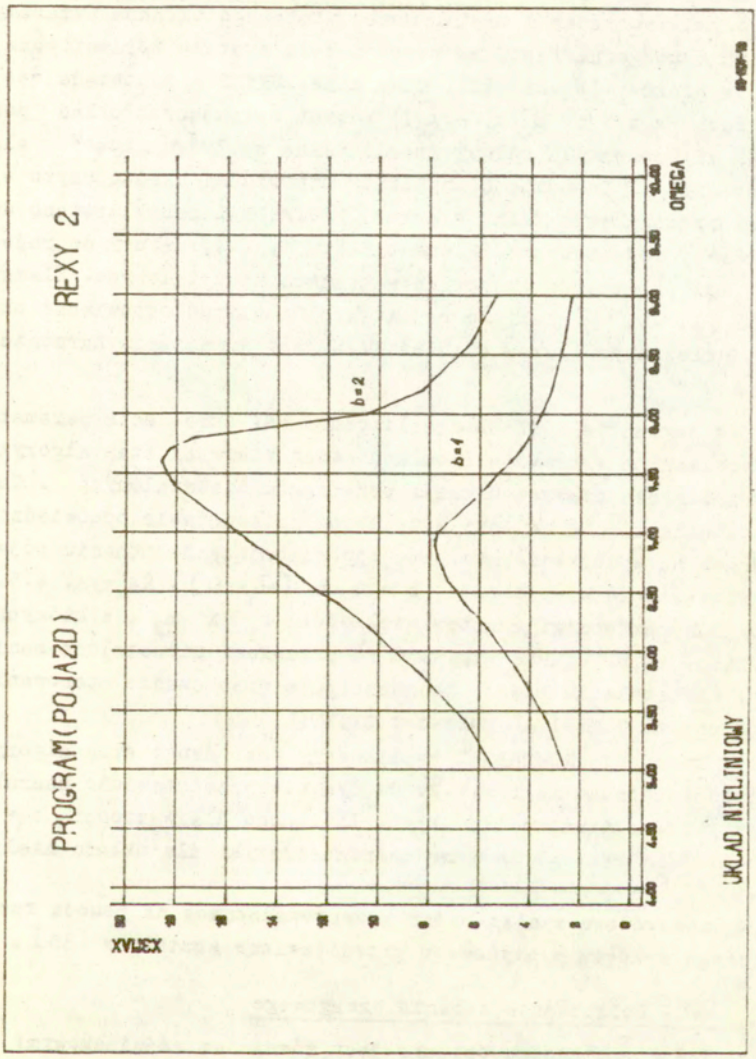
Na rys. 4.3. i 4.4. przedstawiono tabulogram programu w języku LAHYSS realizującego przedstawiony algorytm. W segmencie SYSTEM /rys.4.3./ zapisany jest model badanego pojazdu.



Rys. 4.5. Przebiegi nieustalone w pojeździe.



Rys. 4.6. Charakterystyki amplitudowe dla układu liniowego.



Rys. 4.7. Charakterystyki amplitudowe dla układu nieliniowego.

Ze względu na ograniczone miejsce, nie będziemy przedstawiać metodyki tworzenia i zapisu modelu badanego układu. Zwracamy jedynie uwagę na fakt, że część modelu została zorientowana na zapis blokowy (bloki PX1, NX2, X1X2, PX1X2 - odpowiadające członom $k_1 x_1$ i $k_2(x_1 - x_2)$) równań opisujących układ podczas gdy pozostałe człony zrealizowane są "równaniowo" element AREX . Do pomiaru amplitudy odpowiedzi układu użyto elementu pomiarowego XMAX. W sekcji PERIPHERAL zadeklarowano dwa elementy rejestrujące. Element REXT (1, ...) służy do rejestracji przebiegów czasowych w stanach nieustalonych. Element REXY (2, ...) służy do rejestracji amplitud odpowiedzi układu, zmierzonych przez element XMAX przy wymuszeniu harmonicznym.

W segmencie CONTROL zrealizowane są: określenie parametrów niezmiennych w trakcie symulacji oraz pierwszy etap algorytmu wyznaczenie czasu zanikania przebiegów nieustalonych . Czas ustalania się odpowiedzi wyznaczono na podstawie odpowiedzi układu na wymuszenie skokowe, odpowiadające najechaniu pojazdu na przeszkodę o wysokości b = 2 cm ($\omega = 0$) . Na rys. 4.5. pokazano przebiegi czasowe wielkości x_1 i x_3 , z których wynika, że po czasie tk = 10 s przebiegi nieustalone zanikają. Po ustaleniu czasu tk następuje przekazanie sterowania segmentowi NUMERICAL rozkazem DIGITAL.

W segmencie NUMERICAL realizowany jest drugi etap algorytmu przedstawionego na rys.4.2. Na rys.4.6 przedstawiono charakterystyki amplitudowe dla układu liniowego dla wartości b = 1,2 cm, zaś na rys.4.7 te same charakterystyki dla układu nieliniowego.

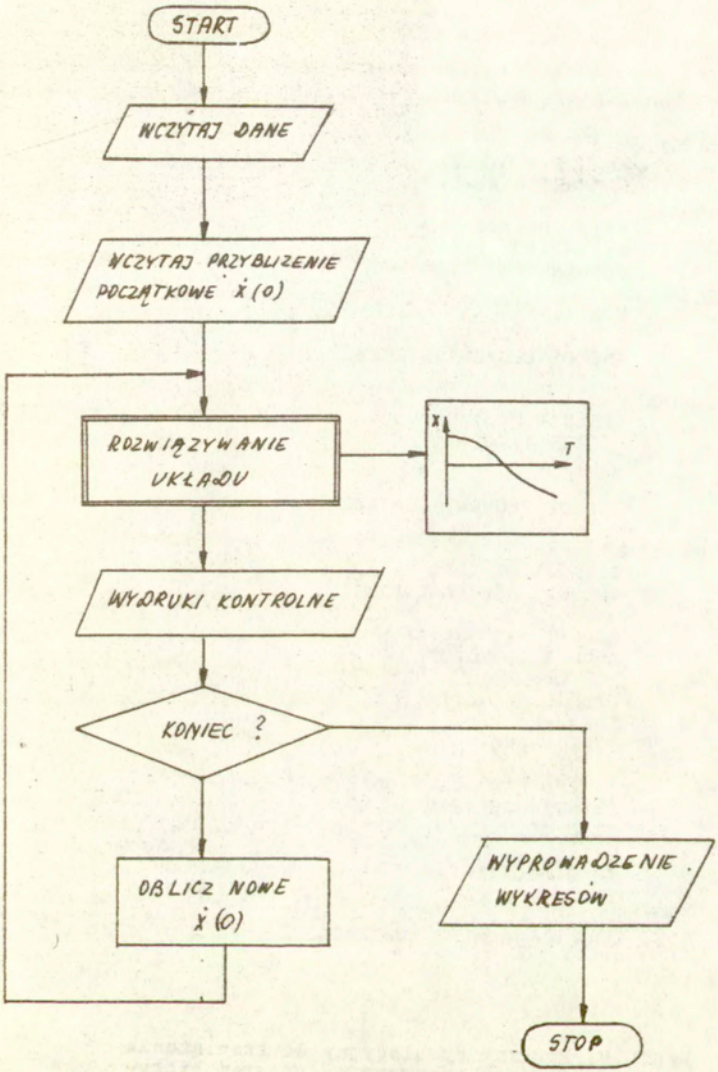
Szczegółowe wyniki badań przeprowadzonych za pomocą rzeczywistego sprzętu hybrydowego przedstawione zostały w [28] .

4.2. Rozwiązanie zadania brzegowego

Zadanie brzegowe opisane jest równaniem różniczkowym:

$$\ddot{x} + l\dot{x} + kx = 0 \text{ , z warunkami brzegowymi:}$$

$$x(0) = x0 \quad , \qquad x(tk) = xk.$$



Rys. 4.8. Algorytm hybrydowy rozwiązania zadania brzegowego.

```
PROGRAM(ZADANIE BRZEGOWE)

SYSTEM
        ANALOG X,DX,DZX
        PARAMETER XO,DXO
DYNAMIC
        X=INT(DX,XO)
        DX=INT(D2X,DXO)
        D2X=INW(DX)
PERIPHERAL
        REXT(1,X)

C       END OF SEGMENT SYSTEM

CONTROL
        RUNGE(4,0.01)
        ACCESS(1,.02)
        DIGITAL

C       END OF SEGMENT CONTROL

NUMERICAL
        SUBROUTINE EXEC
        INCLUDE 'ZMIENNE.FTN'

        ACCEPT *,TK,XO,XK
        ACCEPT *,DXO,WSP

100     CALL COMPUTE(TK)
        ERR=X-XK
        TYPE *,DXO,X

        IF(ABS(ERR).LE.0.001) GOTO 200
        DXO=DXO-WSP*ERR
        GOTO 100

200     CALL PLOT(1)
        END

C       END OF SEGMENT NUMERICAL

END
```

Rys.4.9. Program symulacyjny do rozwiązania zadania brzegowego /program hybrydowy/.

Rozwiązanie zadania polega na znalezieniu takiej wartości prędkości $\dot{x}(0) = dx0$, aby zadanie sprowadzić do zadania początkowego w postaci:

$$\ddot{x} + l\dot{x} + kx = 0, \text{ z warunkami początkowymi:}$$

$$x(0) = x0,$$
$$\dot{x}(0) = dx0.$$

Na rys.4.8 pokazano schemat blokowy algorytmu rozwiązania zadania brzegowego. Jest to typowy algorytm iteracyjny, często występujący przy modelowaniu układów dynamicznych. Wyjaśnienia wymagają jedynie bloki: KONIEC i OBLICZ NOWE X 0 . Ustalenie prawidłowego warunku zakończenia iteracji jest bardzo ważne, bowiem zły warunek może doprowadzić do "zapętlenia się" obliczeń. W programie przyjęto warunek:

$$| x(tk) - xk | \leq 0.001.$$

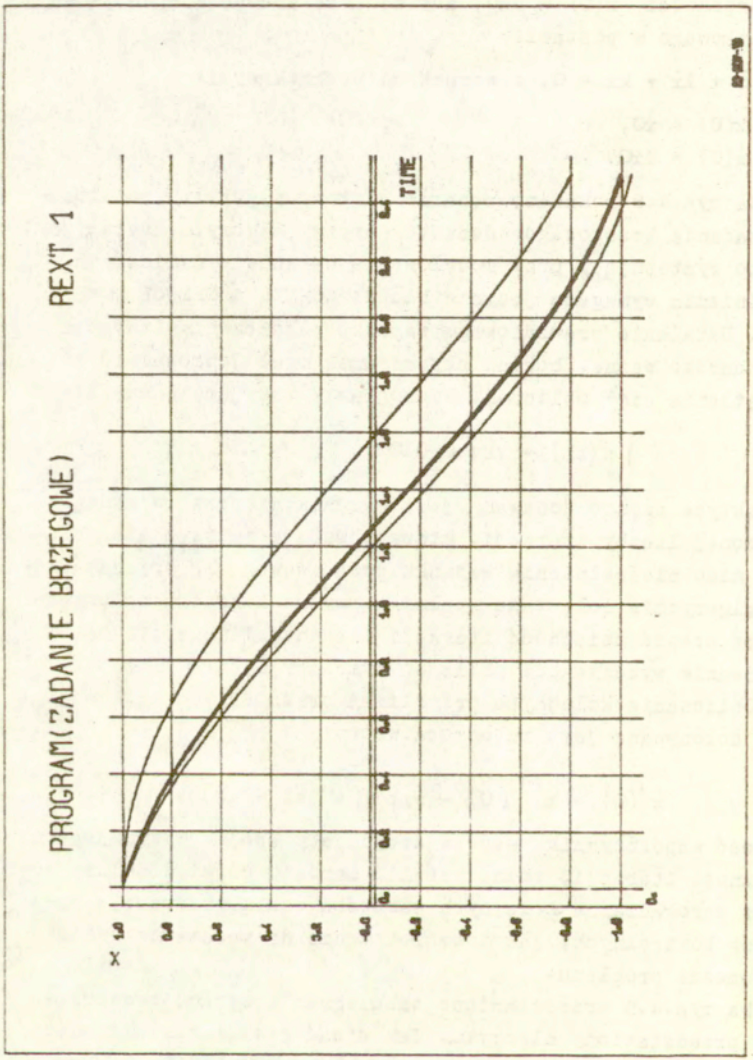
W praktyce często dodawany jest warunek nieprzekroczenia założonej liczby iteracji, który powoduje zatrzymanie obliczeń mimo niespełnienia warunku podstawowego. W przedstawionym algorytmie rolę taką spełniają wydruki kontrolne pozwalające ocenić zbieżność iteracji i ewentualnie systemowe przerwanie wykonywania obliczeń symulacyjnych.

Obliczanie kolejnych przybliżeń prędkości początkowej x 0 dokonywane jest za pomocą wzoru:

$$\dot{x}^{i}(0) = \dot{x}^{i-1}(0) - wsp \cdot (x(tk) - xk) .$$

Wartość współczynnika wsp, w decydujący sposób wpływającą na zbieżność iteracji, można ustalić zarówno doświadczalnie poprzez wprowadzanie kolejnych wartości wsp i obserwacje wydruków kontrolnych, jak i teoretycznie na podstawie ogólnej znajomości problemu.

Na rys.4.9 przedstawiono tabulogram programu modelującego przedstawiony algorytm. Jak widać sterowanie segmentu SYSTEM odbywa się całkowicie z segmentu NUMERICAL. W segmencie tym sprawdzany jest warunek zakończenia iteracji oraz obliczane są kolejne wartości szukanej, wartości $\dot{x}(0)$. Dla uproszczenia zapisu sekcji DYNAMIC przyjęto wartości:



Rys. 4.10. Przebiegi czasowe dla poszczególnych iteracji.

l = .0
k = 1.0.

Na rys.4.10 przedstawiono wykresy czasowe rozwiązania układu dla kolejnych wartości $\dot{x}(0)$, dla następujących danych:

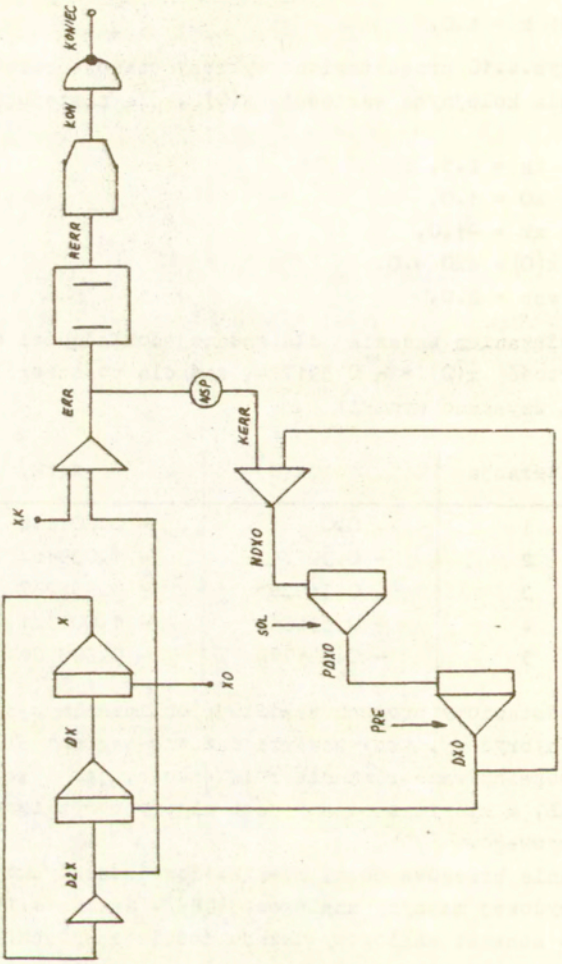
tk = 2.5,
x0 = 1.0,
xk = -1.0,
$\dot{x}(0)$ = dx0 =.0.
wsp = 2.0.

Rozwiązaniem zadania dla zadanej dokładności obliczeń jest wartość $\dot{x}(0)$ = - 0.331774, zaś dla poszczególnych iteracji uzyskano wyniki:

iteracja	$\dot{x}(0)$	x(tk)
1	0.0	- 0.801144
2	- 0.397713	- 1.03916
3	- 0.319385	- 0.992287
4	- 0.334812	- 1.001521
5	- 0.331774	- 0.999701

Przedstawiony program realizuje obliczenia w sposób całkowicie hybrydowy, tzn. zawiera zarówno segment SYSTEM opisujący rozwiązywane równanie różniczkowe, jak i segment NUMERICAL, w którym zapisany jest algorytm rozwiązania zadania brzegowego.

Zadanie brzegowe można również rozwiązać tylko przy użyciu hybrydowej maszyny analogowej([8]). Na rys.4.11. przedstawiono schemat analogowy układu modelującego cały algorytm, zaś na rys.4.12 odpowiadający mu program napisany w języku LAHYSS. Algorytm został zrealizowany całkowicie za pomocą elementów liczących MA, stąd brak w programie segmentu NUMERICAL. Ten sposób rozwiązania zadania za pomocą języka LAHYSS jest mniej dogodny od poprzedniego z dwóch względów. Po pierwsze program jest mniej czytelny i wymaga dobrej znajomości programowania hybrydowej maszyny analogowej. Po dru-



Rys. 4.11. Schemat analogowy rozwiązania zadania brzegowego.

```
PROGRAM(ZADANIE BRZEGOWE - 2)
C       PROGRAM BEZ SEGMENTU NUMERICAL

SYSTEM
        ANALOG X,DX,D2X,ERR,KERR,AERR,DX0,NDX0,DXOP
        LOGIC KON,KONIEC
        PARAMETER X0,XK,WSP,TK

DYNAMIC
        X=INT(DX,X0)
        DX=INT(D2X,DX0)
        D2X=INW(X)

        ERR=SUM(D2X,XK)
        KERR=POT(WSP,ERR)
        AERR=ABS(ERR)

        NDX0=SUM(SUM(DX0,KERR)
        PDX0=AMEM(NDX0,SOL)
        DX0=AMEM(PDX0,PRE)

        KON=COMP(AERR,-.001)
        KONIEC=NOT(KON)

PERIPHERAL
        REXT(1,X)

C       END OF SEGMENT SYSTEM

CONTROL
        RUNGE(4,.01)
        ACCESS(1,.02)
        LET(TK,2.5)
        LET(X0,1.0)
        LET(XK,-1.0)
        LET(WSP,2.0)

        ITER(TK,KONIEC,10)
        PLOT(1)

C       END OF SEGMENT CONTROL

END
```

Rys.4.12. Program symulacyjny do rozwiązania zadania brzegowego /program analogowy/.

gie, duża liczba elementów w sekcji DYNAMIC powoduje prawie trzykrotne wydłużenie czasu obliczeń poszczególnych przebiegów, w porównaniu z omówionym już programem hybrydowym /rys. 4.9/.

5. Literatura.

[1] BAŃKOWSKI J. , FIJAŁKOWSKI K. , Programowanie w języku FORTRAN, WNT, Warszawa 1979.

[2] BEKEY G.A., KARPLUS W.J., Hybrid Computation. John Willey, N.Y. 1968.

[3] BRENNAN R.D., SANO H., Pactolus: A Digital Analog Simulation Program for IBM 1620. Proceedings of Fall Joint Computer Conference, 1964.

[4] Continous System Modeling Program /CSMP/. Application Description. IBM Data Service Center, 1970.

[5] Fortran IV - Plus. User's Guide. Digital Equipment Corporation, 1975.

[6] FRELELC B., Krótki opis języka CSDP. Prace Instytutu Automatyki PW, Warszawa 1976.

[7] GORDON G., System Simulation. Prentice Hall, Inc., Englewood Cliffs, N.Y.1969.

[8] HAKES J., KOHOUT P., Aplikace a systemy hybridnich pocitacu. AAT, Praha 1971.

[9] Hellmann W., Zastosowanie analogowych maszyn matematycznych. WNT, Warszawa 1970.

[10] HYTRAN Simulation Language /HSL/ Programming Manual, Electronic Associates, Inc. 1967.

[11] IAS User's Guide. Digital Equipment Corporation, 1976.

[12] JAMES M.L., SMITH G.M., WOLFORD J.C., Applied Numerical Methods for Digital Computation with Fortran and CSMP. Harper and Rou Publisher, N.Y. 1977.

[13] KACKI E., STARZAK S., Język symulacyjny AM-F-1. Wydawnictwo Politechniki Łódzkiej, Łódź 1972.

[14] KACKI E., WOŹNIAKOWSKI H., Modelowanie analogowe, hybrydowe oraz cyfrowa symulacja maszyn analogowych. PWN, Warszawa 1979.

[15] KACKI E., WOŹNIAKOWSKI H., System dydaktyczny oparty na modelowaniu układów ciągłych. Wydawnictwa Politechniki Wrocławskiej, Wrocław 1978.

[16] KĄCKI E., Zastosowanie języków symulacyjnych w pracach naukowo-badawczych. II Krajowa Konferencja Informatyków, Poznań 1973.

[17] KLAWE M., MAZURAK L., Prace Instytutu Energetyki w dziedzinie modelowania cyfrowego w oparciu o języki symulacyjne. AICA Conference, Gliwice 1972.

[18] KOCHENBURGER R.J., Computer Simulation of Dynamic Systems, Prentice Hall, Inc. Englewood Cliffs, N.Y. 1972.

[19] LEVINE L., Metody stosowania maszyn analogowych. WNT, Warszawa 1969.

[20] LINDE M., Porównanie przydatności matematycznych maszyn hybrydowych z symulacyjnymi językami programowania do modelowania układów automatycznego sterowania procesami dynamicznymi. AICA Conference, Gliwice 1972.

[21] MADEJ W., MARDAL W., MULAWA A., CEMMA - 2 - język do symulacji procesów ciągłych z automatyczną optymalizacją parametrów układu. Prace IMM, z.2, Warszawa 1971.

[22] MADEJ W., Język symulacyjny CEMMA. Podręcznik programowania. Prace IMM, Warszawa 1968.

[23] MĘDRZYCKI J., Technika analogowa i hybrydowa, WNT, Warszawa 1974.

[24] MISNIAKIEWICZ K., Język specjalizowany do symulacji układów dynamicznych na maszynie cyfrowej ODRA 1204. Prace Zakładu Systemów Automatyki Kompleksowej PAN, Gliwice 1973.

[25] MIMIC. Umass User's Manual. University of Massachusetts, Amherst 1970.

[26] MOSZYŃSKI K., Rozwiązywanie równań różniczkowych zwyczajnych na maszynach cyfrowych. WNT, Warszawa 1971.

[27] NEVRIVA P., Simulace ridicich systemu na cislicovem pocitaci. SNTL, Praha 1975.

[28] NIEZGODZKI P., SZEMPLIŃSKA-STUPNICKA W., Program analogowy i układ pomiarowo rejestrujący do automatycznego wykreślania rodziny charakterystyk amplitudowo-częstotliwościowych. Prace IPPT PAN, Warszawa 4/1978.

[29] ORŁOWSKI H., HAWRYLUK J., Modelowanie cyfrowe. WNT. Warszawa 1971.

[30] PDP-11 Fortran. Language Reference Manual. Digital Equipment Corporation, 1975.

[31] Shannon R.E., System simulation. The Art and Science. Prentice Hall, Inc, Englewood Cliffs, N.Y.1975.

[32] SKVOR V., URBAN V., Hybrydni analogovy pocitac MEDA 41TC. Programovani. AAT, Praha 1975.

[33] SZOPLIŃSKI Z., Elektroniczna technika analogowa. WNT Warszawa 1969.

[34] The SCi Continous System Simulation Language /CSSL/. Simulation 1967, Vol.9, No 6.

[35] TRACZYK W., Układy cyfrowe automatyki. WNT, Warszawa 1973.

[36] TURSKI W.M., Podstawy użytkowania maszyn cyfrowych w ośrodkach naukowo-technicznych. PWN, Warszawa 1973.

Spis treści