(Pomiary Automatyka Robotyka nr 4/2003)

MODELOWANIE FIZYCZNE

Zbigniew Mrozek

Streszczenie

Najnowsze pakiety symulacyjne realizują zasadę modelowanie fizycznego i wykorzystują modele zgodne ze standardem języka Modelica. Elementy modelu fizycznego odpowiadają elementom rzeczywistego systemu, a w miejscu łączenia tych elementów wymaga się spełniania praw fizyki. Autor pokazuje użyteczność tego podejścia na przykładzie modelu napędu elektrycznego.

Abstract

New simulation packages are based on physical modeling principle and are build on free Modelica language models. Components of physical model describe topology of original system. Physical phenomena are defined inside and on terminals of the component. An example of electric drive structure model shows effectivity of this approach.

Autor, dr inż Zbigniew Mrozek, Wydział Inżynierii Elektrycznej I Komputerowej, Politechnika Krakowska, E-mail: *zbigniew.mrozek@pk.edu.pl*

1. WSTĘP

Pakiety uniwersalne oprogramowania symulacyjnego umożliwiają symulację praktycznie dowolnego systemu, jeśli jego model może być przedstawiony w postaci układu równań różniczkowych lub różniczkowo-algebraicznych. Jeśli modele są zadawane w postaci schematów blokowych, to o możliwości użycia takiego pakietu decyduje dostępność potrzebnych bloków w bibliotekach lub możliwość samodzielnego ich zbudowania. Połączenia bloków służą do jednokierunkowego przekazywania informacji.

Istotą **modelowania fizycznego** jest wymóg spełniania praw fizyki (np. prawa zachowania energii) w miejscu połączenia elementów modelu. Jest to pewna analogia do grafu wiązań (Bond graf), a zarazem istotna różnica w porównaniu do schematów blokowych. Modelowanie fizyczne omówiono na przykładzie języka Modelica [3,8] i pakietu symulacyjnego Dymola [2]. Modelica jest nowym, obiektowo zorientowanym językiem do modelowania systemów. Modele i biblioteki elementów modeli przygotowane zgodnie ze specyfikacją Modeliki, mogą być używane w dowolnym środowisku oprogramowania symulacyjnego, zintegrowanym z Modeliką (aktualnie Dymola i MathModelica). Stanowi to nową jakość w zakresie modelowania i symulacji.

Celem artykułu jest zachęcenie czytelników do wykorzystania możliwości tworzenia modeli fizycznych, szczególnie w tych dziedzinach, dla których oferowane są bezpłatne biblioteki modeli (rozdział 3.3). W artykule objaśniono graficzny i tekstowy sposób tworzenia modeli na przykładzie regulatora PID. Następnie omawia się dostępne biblioteki modeli i objaśnia się zalety modelowania fizycznego na przykładzie modelowania elektrycznego układu napędowego.

2. SCHEMATY BLOKOWE I MODELE FIZYCZNE

Modele są budowane w celu lepszego zrozumienia struktury i działania rzeczywistego lub projektowanego systemu. Są one potrzebne, gdyż urządzenia przemysłowe, a nawet wyroby powszechnego użytku, są coraz bardziej złożone. W większości przypadków tworzy się wiele modeli,

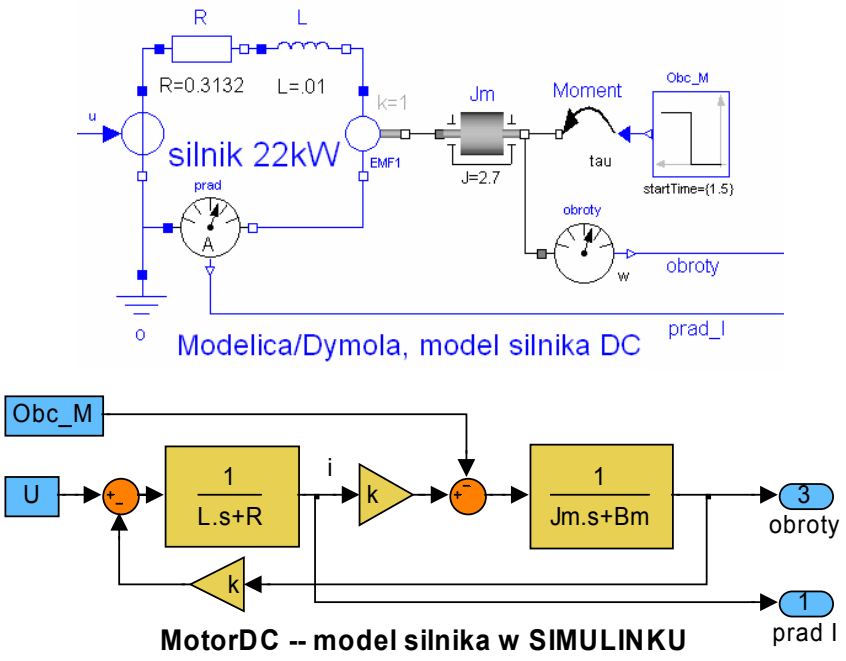
ujmujących w różny sposób te same lub różne elementy systemu. Booch, Rumbaugh i Jacobson [1] podają zasady modelowania systemów. Najważniejsze z nich to:

1) wybór modelu ma duży wpływ na to, w jaki sposób problem będzie badany i jaki kształt przyjmie rozwiązanie,
2) model może być opracowany na różnych poziomach szczegółowości,

Zasada pierwsza stanowi, że rodzaj wybranego modelu decyduje zarówno o sposobie projektowania, jak też o końcowym rezultacie projektu. Przykładowo wybór modelu statycznego uniemożliwia zaprojektowania układu sterowania w stanach przejściowych. Zasada druga stanowi, że szczegółowość modelu należy dostosować do celu, w jakim model ma być użyty. Szkodliwa jest zarówno nadmierna szczegółowość modelu, jak też jego nadmierne uproszczenie.

Podczas projektowanie należy utworzyć modele wszystkich elementów systemu, niezależnie od ich natury fizycznej. **Efektywne metody analityczne** projektowania złożonych systemów, zbudowanych z elementów o różnej naturze **nie istnieją lub nie są znane**. W tej sytuacji zbudowanie modelu przybliżonego i symulacja są zazwyczaj istotnymi i niezbędnymi elementami projektowania. Dostępne narzędzia do modelowania i symulacji w większości nie są dostosowane do potrzeb projektowania systemów interdyscyplinarnych.

Aby porównać schematy blokowe z modelem fizycznym, pokazano dwa warianty modelu silnika elektrycznego prądu stałego. Obciążeniem silnika jest moment bezwładności `Jm` i moment obrotowy obciążenia `tau=Obc_M`. W górnej część rysunku 2.1 przedstawiono model fizyczny zrealizowany w języku Modelica, z użyciem pakietu symulacyjnego Dymola. **Obciążenie** jest tu **dołączone do wału**, na zewnątrz modelu silnika. Dodatkowe elementy (na przykład przekładnia, tarcie suche, tarcie wiskotyczne, luz czy też sprężystość wału mogą być pobrane z biblioteki i dołączone do wału wybranego elementu).



Rysunek 2.1 Model fizyczny silnika prądu stałego i części mechanicznej napędu w języku Modelica (rysunek górny) oraz model silnika w SIMULINKU: (rysunek dolny,[10]).

W dolnej części rysunku 2.1 zamieszczono model tego samego silnika w SIMULINKU [7,10]. W tym modelu tłumienie `Bm` oraz moment bezwładności `Jm` zostały włączone do równań modelu silnika elektrycznego. Umieszczenie dodatkowych elementów może wymagać ingerencji w model wewnętrzny silnika. Parametr $k=k_m \cdot \Phi_N$ [N·m/A] jest stałą konstrukcyjną silnika. W SIMULINKU sygnały są przekazywane pomiędzy blokami tylko w jedną stronę, co powoduje konieczność utworzenia dodatkowego wejścia dla momentu obciążenia silnika. Ponadto do modelu silnika są wprowadzane parametry przekładni i maszyny roboczej (przełożenie, moment bezwładności itp.), co jest niezbędne dla obliczenia prędkości obrotowej silnika. Pogarsza to czytelność schematów blokowych SIMULINKA i utrudnia dalszą rozbudowę modelu – na przykład w celu uwzględnienia

luzów, tarcia lub elastyczności układu przeniesienia napędu.

Model fizyczny odwzorowuje strukturę połączeń elementów rzeczywistego systemu. Jest przez to łatwiejszy w przygotowaniu, a ewentualne błędy mogą być skuteczniej wyeliminowane. Przykładowo, potrzebną do układu sprzężenia zwrotnego wartość prądu pobiera się z obwodu, w którym umieszczono rezystancję `R` i indukcyjność `L` obwodu głównego silnika. Wał modelu silnika jest jednocześnie jego wyjściem i wejściem mechanicznym. Z wału pobierana jest moc do napędzania maszyn roboczych, a zarazem przykładany jest tu moment obciążenia, który oddziaływuje na prędkość obrotową silnika. Również z wału pobiera się wartość prędkości obrotowej do pętli sprzężenia zwrotnego.

3. MODELICA I PAKIET DYMOLA

Standaryzacja modeli symulacyjnych ułatwia tworzenie modeli złożonych, zawierających elementy i podsystemy o różnej naturze fizycznej, pochodzące z różnych pakietów symulacyjnych. Potrzeby te przez długi czas nie miały wsparcia ze strony wiodących producentów. Powodem mogła być obawa o utratę części dotychczasowych rynków zbytu i zmniejszenia dochodów. Sytuacja zmieniła się, gdy realizację tego problemu podjęto w ramach projektu SiE-WG [6], finansowanego przez Wspólnotę Europejską. Rezultatem prac grupy utworzonej w ramach SiE-WG było ogłoszenie we wrześniu 1997 specyfikacji języka Modelica, przeznaczonego do modelowania fizycznego. Język Modelica został stworzony zbiorowym wysiłkiem naukowców i praktyków z przemysłu oraz twórców kilku wcześniejszych, obiektowo-zorientowanych języków jak: Allan, Dymola (stara wersja), NMF, ObjectMat, Omola , SIDOPS+, Smile. Prócz wymienionych wyżej, wykorzystano idee zawarte w językach ASCEND, gPROMS, U.L.M i VHDL-AMS. Na podstawie powyższych wzorów przyjęto obiektowo zorientowaną syntaktykę i semantykę dla Modeliki [3]. Autor tej pracy uczestniczył we wczesnym etapie prac SiE-WG [6] podczas odbywania stażu na uniwersytecie w Gandawie (Belgia).

Aktualnie rozwój Modeliki przebiega następująco:

1. Powołano niedochodową organizację: *Modelica Association.* Jest ona właścicielem praw do znaku towarowego i logo Modeliki, specyfikacji języka i bibliotek standardowych. Jej zadaniem jest propagowanie i promocja języka Modelica.
2. Specyfikacja języka, biblioteki standardowe i narzędzia są udostępnione w Internecie i mogą być wykorzystane i rozwijane przez wszystkich zainteresowanych. Są one dostępne nieodpłatnie dla celów badawczych, rozwojowych i przemysłowych pod adresem *http://www.Modelica.org/*,
3. Oprogramowanie symulacyjne oraz biblioteki i translatory oparte o Modelikę są rozwijane przez uczelnie i jednostki badawcze, a także przez niezależne firmy komercyjne, które mogą się zrzeszać w *Modelica Association* [8]. Część tych prac jest finansowana przez Wspólnotę Europejską [12].

3.1. MODELICA JAKO JĘZYK MODELOWANIA MATEMATYCZNEGO

Pakiet Modelica akceptuje równania różniczkowe (ODE: *ordinary differential equations*) i równania różniczkowo-algebraiczne (DAE: *differential algebraic equations*), opisujące zjawiska zachodzące w modelowanym urządzeniu, w postaci niejawnej.

$$f(t, dx/dt, x, y) = 0 \tag{1}$$

gdzie:

x – wektor zmiennych dynamicznych (ich pochodna *dx/dt* występuje we wzorze (1))
y – wektor zmiennych algebraicznych (pochodna nie występuje we wzorze (1))
t – czas

Zaletą takiego podejścia jest możliwość wykorzystania tego samego modelu niezależnie od sposobu pobudzania modelowanego systemu. Umożliwia to późniejsze przypisanie dowolnych zmiennych modelu do wejść i do wyjść systemu. W literaturze [2-4] takie podejście określa się jako deklaratywne lub bezprzyczynowe (*acausal, non-causal*).

Modele przygotowane w języku Modelica mogą być wykorzystana przez dowolny pakiet symulacyjny po dokonaniu odpowiednich przekształceń symbolicznych równań różniczkowych (ODE) i różniczkowo-algebraicznych (DAE). Potrzebna jest też analiza struktur dziedziczenia klas modeli i połączeń bloków w schematach graficznych oraz wiele innych operacji. Nie jest to problem banalny, gdyż Modelica akceptuje równania w postaci uwikłanej, nadokreślone i źle uwarunkowane.

Powyższe problemy zostały rozwiązane przez niektórych producentów oprogramowania:

- aktualna wersja pakietu symulacyjnego Dymola z firmy Dynasim AB jest w pełni zintegrowana z Modeliką. Dymolę opisano w dalszej części pracy.
- MathModelica z firmy MathCore wraz z Modeliką jest pakietem symulacyjnym, które może być zintegrowane z pakietami Mathematica oraz Microsoft Visio.

Dodatkowo, na jednym ze szwedzkich uniwersytetów prowadzone są prace nad stworzeniem ogólnie dostępnego (*open source*) i bezpłatnego środowiska do modelowania i symulacji w Modelice [4].

SIMULINK [7, 10] jest zbudowanym na bazie MATLABA interaktywnym pakietem do modelowania i symulacji ciągłych oraz dyskretnych modeli dynamicznych. Umożliwia tworzenie wielopoziomowych systemów w postaci schematów blokowych. MATLAB/SIMULINK i Dymola uzupełniają się wzajemnie. Dymola generuje pliki, które mogą być eksportowane do pakietu MATLAB/SIMULINK. Pozwala to na wykorzystanie bogatych bibliotek obu pakietów oraz ich zastosowanie do prototypowania w czasie rzeczywistym.

3.2. TWORZENIE KLAS MODELI

Modelica nie oferuje wygodnego graficznego interfejsu użytkownika. Do tworzenia modeli, symulacji i prototypowania można użyć pakietów Dymola lub MathModelica. Każda klasa elementów modelu może być tworzona i opisana w trybie graficznym lub tekstowym, na różnych poziomach abstrakcji:

— najczęściej tworzy się schematy strukturalne (*component diagram*). Model (podobnie jak w SIMULINKU) jest budowany w trybie graficznym, poprzez łączenie pomiędzy sobą ikon elementów pobranych z okien licznych bibliotek Modeliki.

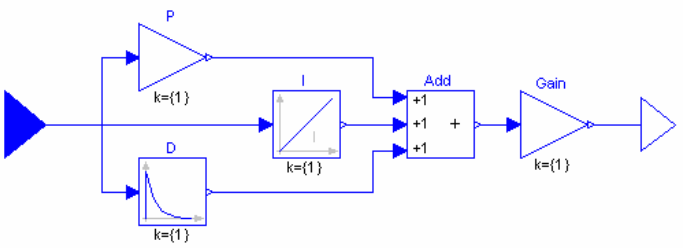
— nowy model (lub tylko te elementy modelu, których brak w bibliotekach) można też stworzyć w trybie tekstowym, podając równania opisujące zachowanie modelowanych elementów i deklarując użyte zmienne, stałe i parametry. Jeśli w modelu użyto kilku elementów, należy opisać ich wzajemne połączenia

Dodatkowo każdy element modelu (a dokładniej każda klasa elementów modelu) może być uzupełniony o warstwę ikon i warstwę dokumentacji. Na ikonie zaznacza się umowne wejścia i wyjścia modelu. Często działają one w obie strony. Ikony ukrywają budowę wewnętrzną modelu i są wykorzystywane przy budowaniu schematów strukturalnych.

Autorzy pakietu Dymola zdawali sobie sprawę, że użytkownikom i programistom zazwyczaj brakuje czasu na przygotowanie i aktualizowanie dokumentacji. Wyposażyli więc Dymolę w generator HTML, który dla każdej klasy modelu tworzy automatycznie jej dokumentację. Dokumentacja może zawierać ikonę, topologię połączeń elementów modelu, tabelkę z parametrami modelu (dla każdego parametru: nazwa, wartość domyślna i opis) oraz definicję przepisaną z warstwy klas edytora modelu (lista wykorzystanych klas i równania), a nawet odwołania do dokumentacji fabrycznej pakietu. Warstwa dokumentacji Edytora Modeli daje użytkownikowi możliwość uszczegółowienia dokumentacji HTML poprzez wstawienia dodatkowych opisów i ilustracji. Dodatkowe opisy oraz linki do dodatkowych rysunków należy przygotować w języku HTML. Dokumentację można oglądać korzystając z dowolnej przeglądarki internetowej.

3.2.1. Schematy strukturalne

Na rysunku 3.1 pokazano schemat strukturalny regulatora PID. Regulator składa się z odpowiednio połączonych ikon elementów składowych, dostępnych także w bibliotece Modeliki. W ten sposób tworzy się zdecydowaną większość modeli. Podobny sposób tworzenia modelu jest stosowany w SIMULINKU.



Rysunek 3.1. Regulator PID, zbudowany z elementów pobranych z biblioteki pakietu Modelica

3.2.2. Tryb tekstowy modelowania

Schematy strukturalne są automatycznie przekształcane do postaci tekstowej przed ich zapisaniem w pliku dyskowym komputera. Tryb tekstowy może być także użyty do tworzenia nowych modeli poprzez matematyczne opisanie zjawisk zachodzące w rzeczywistym obiekcie lub przez tekstowe opisanie topologii wzajemnych połączeń istniejących elementów modelu. Przykładowo, dołączenie do sumatora „Add" z rysunku 3.1 trzech elementów (proporcjonalny `P`, całkujący `I` oraz różniczkujący `D`) tworzących regulator PID, jest opisane poniżej w trzech kolejnych liniach „connect":

```
equation
  connect(P.outPort, Add.inPort1);
  connect(I.outPort, Add.inPort2);
  connect(D.outPort, Add.inPort3);
….
```

Element całkujący (`Integrator I`) regulatora PID opisano w postaci równania:

```
equation
  for i in 1:size(k, 1) loop
    der(y[i]) = k[i]*u[i];
  end for;
end Integrator;
```

Integrator działa na zmiennych wektorowych: i-ty sygnał u[i] na wejściu bloku całkującego jest pochodną sygnału wyjściowego y[i]. W podanym powyżej układzie równań nie ma operacji całkowania. Powyższy zapis jest zgodny z notacją MATLABA: w nawiasach kwadratowych podano numery elementów wektora lub macierzy.

Zamiast opisanego wyżej regulatora PID, zaleca się stosowanie umieszczonego w tej samej bibliotece LimPID, z ograniczeniem wartości sygnału wyjściowego. Oba warianty regulatorów PID znajdują się w bibliotece *Modelica.Blocks.Continuous*.

Nowe klasy modeli można tworzyć wykorzystując dotychczasowy dorobek (*reuse*) poprzez dziedziczenie i modyfikację istniejących klas modeli własnych i klas pobranych z bibliotek. Można też podać równania i zaprojektować ikony dla nowych, własnych modeli. Równania uwikłane będą automatycznie przekształcone do postaci jawnej i rozwiązane symbolicznie. Jeśli jest to niemożliwe, to równania są rozwiązywane numerycznie.

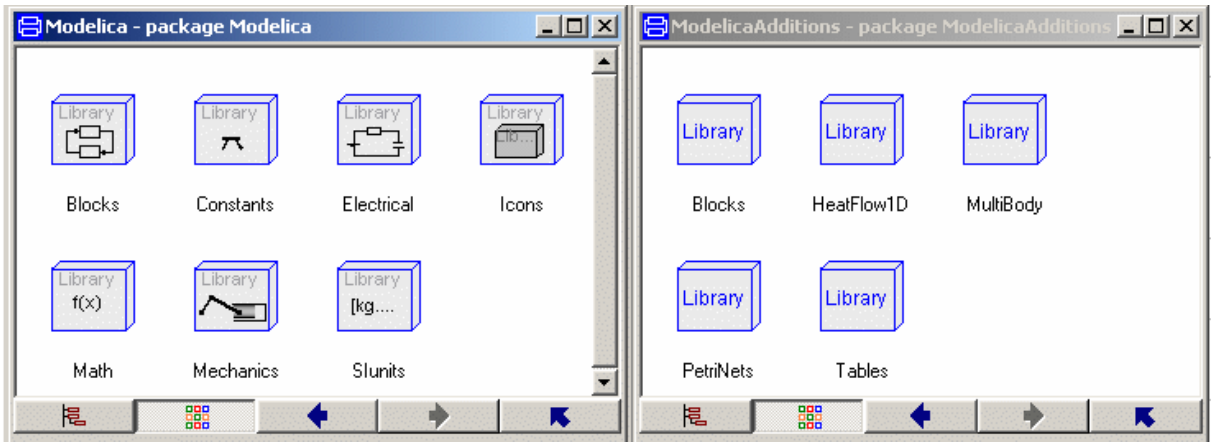
3.3. BIBLIOTEKI JĘZYKA MODELICA

Modelica jest dostępna w Internecie bezpłatnie wraz z dwoma pakietami bibliotecznymi: standardowym i dodatkowym (*additions*). Ponadto dostępne są nieodpłatnie biblioteki opracowane przez użytkowników Modeliki. Aktualny ich wykaz podano w tabeli 1:

Tabela 1 Bezpłatne biblioteki opracowane przez użytkowników Modeliki

Nazwa biblioteki	Opis
Thermofluid	Ciecze i wymiana ciepła
HyLibLight	Elementy hydrauliczne
ObjectStab	Systemy energetyczne
ExtendedPetriNet	Sieci Petriego
QSSFluidFlow	Stany ustalone przepływu cieczy
SystemDynamics	Dynamika systemów
FuzzyControl	Systemy rozmyte
ATplus	Budynki inteligentne

Jeszcze inne biblioteki można zakupić jako opcje pakietów Dymola lub MathModelica. Biblioteki Modeliki mają zazwyczaj wielopoziomową strukturę hierarchiczną. Poniżej pokazano okna biblioteki standardowej i dodatkowej (additions), a w nich ikony z nazwami dołączonych bibliotek specjalistycznych.



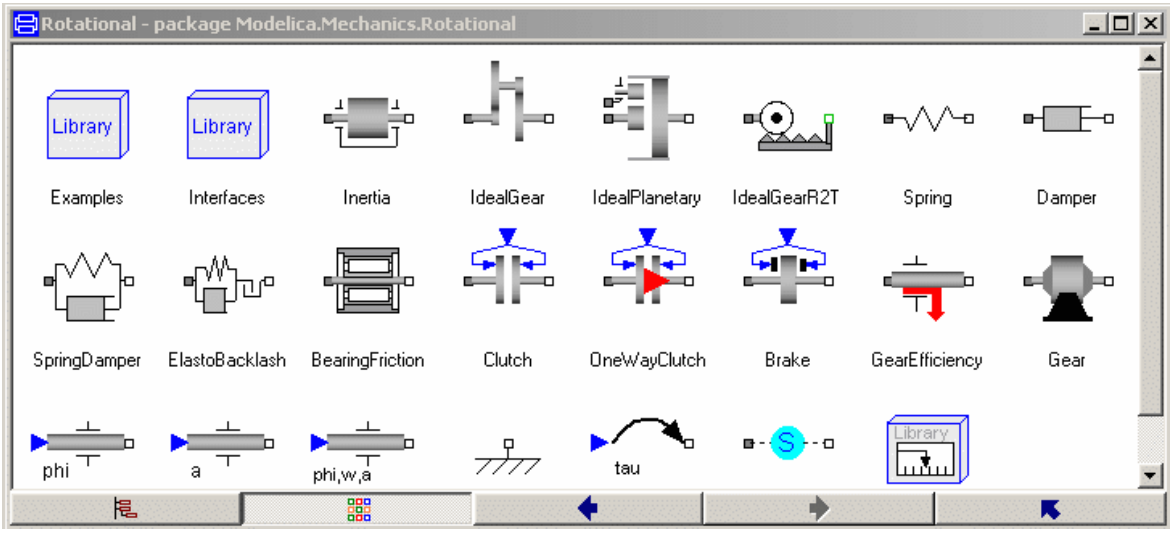
Rysunek 3.2. Biblioteki pakietu Modelica: standardowa i dodatkowa

W Modelice stworzono odrębną biblioteke jednostek legalnych jednostek miar układu SI. (biblioteka SIunits). Fragment zawartości tej biblioteki, definiujący jednostki masy i gęstości pokazano poniżej.

```
// Mechanics (chapter 3 of ISO 31-1992)
  type Mass = Real (
      final quantity="Mass",
      final unit="kg",
      min=0);
  type Density = Real (
      final quantity="Density",
      final unit="kg/m3",
      displayUnit="g/cm3",
      min=0);
```

Zwraca się uwagę, że prócz jednostki układu SI używanej do obliczeń (na przykład gęstość jest wyrażana w kg/m^3), dopuszcza się użycie innej jednostki (g/cm^3) do prezentacji wyników obliczeń. Dodatkowo, w linii `min=0` określono, że masa i gęstość nie może mieć wartości ujemnej.

Poniżej pokazano zawartość biblioteki mechanicznej z pakietu podstawowego Modeliki. Biblioteka mechaniczna zawiera: elementy przesuwne (*Translational Sublibrary*) oraz pokazane na rysunku 3.3: elementy obrotowe (*Rotational Sublibrary*}. Większość elementów tej biblioteki nie posiada strzałek, co oznacza że przepływ energii może odbywać się w dowolnym kierunku. Mają one wpisane w swojej definicji równania, które zgodnie z zasadami modelowania fizycznego obowiązują w miejscu połączenia wałów. Zazwyczaj jest to wymóg równości kątów obrotu `phi` oraz zerowanie się sumy momentów `tau`.

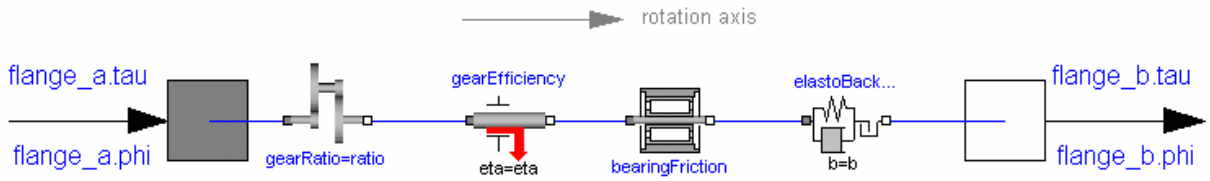


Rysunek 3.3. Biblioteka podstawowa: bloki elementów obrotowych *Modelica.Mechanics.Rotational*

Modelica jest obiektowo-zorientowana i każdy element modelu ma pola, w których przechowywane są parametry wybranego bloku, istotne z punktu widzenia symulacji. Przypisanie parametrom potrzebnych wartości jest wykonywane w sposób dogodny dla użytkownika, w większości na podstawie typowych danych katalogowych wybranego urządzenia. Na przykład dla modelu przekładni `Gear` (rys. 3.3, 3.4) określa się następujące parametry:

- Przełożenie (*ratio*),
- Sprawność (*gear efficiency*), uwzględniająca tylko straty na tarcie w zębach przekładni,
- tarcie w łożyskach: mokre (*friction_pos*) i suche (*peak*),
- elastyczność (c), tłumienie (d) i luzy (b).

Jeśli niektórym parametrom nie nadano wartości, to przyjęte zostaną wartości domyślne. W razie potrzeby po dowolnej stronie przekładni, do wału_a lub do wału_b, może być dołączona ikona modelu inercji.



Rysunek 3.4. Model przekładni Gear z biblioteki *Modelica.Mechanics.Rotational*

W omawianym modelu zwraca uwagę rozdzielenie tarcia suchego i mokrego w łożyskach oraz tarcia w zębach przekładni. Umożliwia to poprawną symulację przy uruchamianiu i zatrzymaniu przekładni oraz przy pracy z dowolną prędkością.

3.4. DYMOLA

W grupie programów przydatnych do modelowania złożonych urządzeń mechatronicznych należy zwrócić uwagę na MATLAB/SIMULINK [7,10] i Dymolę [2] *(dynamic modeling language)* – obiektowo zorientowany pakiet do modelowania i symulacji obiektów fizycznych. Dymola wykorzystuje modele w **standardzie Modeliki**. Ułatwia to symulowanie układów o mieszanej naturze fizycznej: mechanicznych, elektrycznych, termodynamicznych, chemicznych oraz innych, opisanych równaniami lub modelami Modeliki. Dymola jest oferowana dla Windows 98/2000/NT/ME oraz dla Unix i Linux.

Połączenie elementów Modeliki na schemacie strukturalnym służy nie tylko do jednokierunkowej transmisji informacji (jak w SIMULINKU), lecz realizuje **prawa fizyki** obowiązujące w miejscu połączenia. Wbudowany do Dymoli translator może dokonać symbolicznego przekształcenia ponad

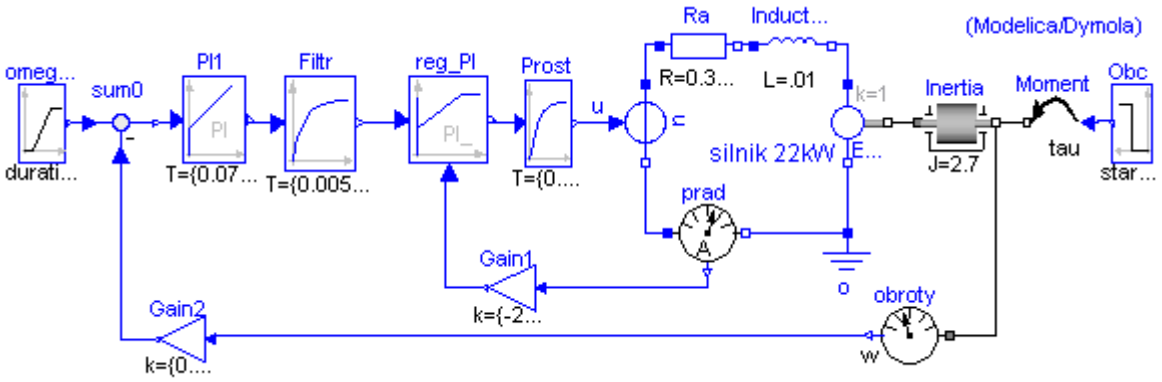
100 000 równań dla potrzeb symulacji. Edytor graficzny pozwala na przeglądanie, edycję i tworzenie nowych modeli. Dymola wspomaga **symulację w czasie rzeczywistym** (również HiL, *hardware in the loop*), ale może też wygenerować pliki dla MATLABA i SIMULINKA. Przeniesione z Dymoli podsystemy mogą być wykorzystane do prototypowania w czasie rzeczywistym i do projektowania sterowników cyfrowych [5,11,13]. Połączenie Modeliki z SIMULINKIEM (poprzez Dymolę) ułatwia prototypowanie bardzo dużych i skomplikowanych systemów. Wyniki obliczeń Dymoli są zapisywane w formacie umożliwiającym ich przetwarzanie w **środowisku MATLABA** [7,10].

Opcjami rozszerzającymi Dymolę, dostępnymi za dopłatą, są: trójwymiarowa animacja, interfejs do SIMULINKA oraz symulacja w czasie rzeczywistym. Prócz bezpłatnych bibliotek z Modeliki, do Dymoli można dokupić biblioteki **HyLib** (Hydraulics library) i **Powertrain** (układy przekazywania mocy w pojazdach).

Dymola daje możliwość dokonania wyboru metody całkowania dla równań różniczkowych (ODE) i różniczkowo-algebraicznych (DAE), dobrze i źle uwarunkowanych oraz dla równań nadokreślonych {ODAE, *overdetermined* DAE). Do symulacji **systemów dyskretnych** oraz w czasie rzeczywistym wykorzystywane są stałokrokowe metody całkowania. Podczas symulacji można zmieniać parametry modelu, krok i metodę całkowania. Można też powtórzyć obliczenia dla innych warunków początkowych i dla nowych wartości parametrów – bez powtórnej translacji i bez potrzeby powtórnego kompilowania wygenerowanego przez Dymolę kodu.

3.5. MODEL FIZYCZNY UKŁADU NAPĘDOWEGO

Modelowanie fizyczne zilustrowano przykładem modelu strukturalnego układu napędowego z silnikiem prądu stałego i sterowaniem z dwoma pętlami sprzężenia zwrotnego: stabilizacji prądu silnika i prędkości obrotowej, jak opisano w artykule [9].



Rysunek 3.5 Model fizyczny elektrycznego układu napędowego

WNIOSKI I UWAGI KOŃCOWE

Pakiety wyspecjalizowane, z małymi wyjątkami, są efektywne tylko we własnej dziedzinie zastosowań. Nie radzą one sobie z modelowaniem systemów o innej naturze. Przykładowo symulator obwodów elektronicznych nie jest dostosowany do modelowania pomp i rozdzielaczy hydraulicznych. Autor uważa, że modelowanie fizyczne zyska w ciągu kilku lat powszechne uznanie zarówno wśród naukowców jak i praktyków w przemyśle. Istotną zaletą pakietu symulacyjnego Dymola jest jego integracja ze środowiskiem MATLABA, co znacznie powiększa zakres zastosowań obu tych pakietów. MATLAB i SIMULINK będą również umożliwiać modelowanie fizyczne. Aktualnie jest to realizowane przez dodatkowe moduły SimMechanics i SimPowerSystems (systemy elektryczne, mechaniczne i sterowanie).

LITERATURA

[1] Booch G., Rumbaugh J., Jacobson I. *The Unified Modelling Language User Guide*, Addison Wesley, 1999

[2] *Dymola, Dynamic Modelling Laboratory, www.dynasim.se*

[3] Elmqvist H., Mattsson S.E., Otter M., *Object-Oriented and Hybrid Modeling in Modelica*. ADPM 2000, Dortmund Germany, 2000; Journal Européen des systemes automatisés, 35,1/2001, pp. 1 a X. 2000

[4] Fritzson P., et al..:*The Open Source Modelica Projec*t, Proceedings of the 2nd International Modelica Conference, , The Modelica Association 2002

[5] Grega Wojciech, Sterowanie cyfrowe w czasie rzeczywistym, Wydawnictwa Wydziału EAIiE AGH Kraków, 1999.

[6] Kerckoffs E.J.H., Vangheluwe H.L., Vansteenkiste G.C., Geril P., ESPIRIT IT Basic Research Working Group 8467 "Simulation in Europe – Working Group (SiE-WG), Progress Report 1995.

[7] MATLAB, SIMULINK, STATEFLOW. *http://www.mathworks.com*

[8] Modelica®, *A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, Version 2.0, http://www.Modelica.org/*

[9] Mrozek B., *Regulatory rozmyte dla napędu prądu stałego*, PAR nr 2, pp 16, 2003

[10] Mrozek B, Mrozek Z, *MATLAB 6, poradnik użytkownika* Wydawnictwo PLJ, Warszawa 2001.

[11] Mrozek Z, *Komputerowo wspomagane projektowanie systemów mechatronicznych*, Zeszyty Naukowe PK, Kraków 2002,

[12] RealSim: *Realtime Simulation for Multi-Physics Systems projekt, IST-1999-11979*, *http://www.cordis.lu/ist/projects/99-11979.htm*

[13] Uhl T. (ed.), Bojko T., Mrozek Z., Petko M., Szwabowski W., Korendo Z., Bogacz M., *Wybrane problemy projektowania mechatronicznego*, Wydawnictwo Katedry Robotyki i Dynamiki Maszyn, AGH Kraków 1999.