INSTRUKCJA DO ĆWICZEŃ LABORATORYJNYCH Z PODSTAW AUTOMATYKI W PROGRAMIE MATLAB

dr inż. GRZEGORZ MZYK

Spis treści

1 Informacje wstępne

Instrukcja dotyczy obsługi programu Matlab w zakresie laboratorium z przedmiotu "Podstawy automatyki (2)" na III roku kierunku AiR. Proponowany plan spotkań jest udostępniony na stronie internetowej *http://diuna.ict.pwr.wroc.pl/grmz* w folderze *Dydaktyka* jako plik *palab.pdf*. Zakłada się następujące wymagania odnośnie stanowiska komputerowego:

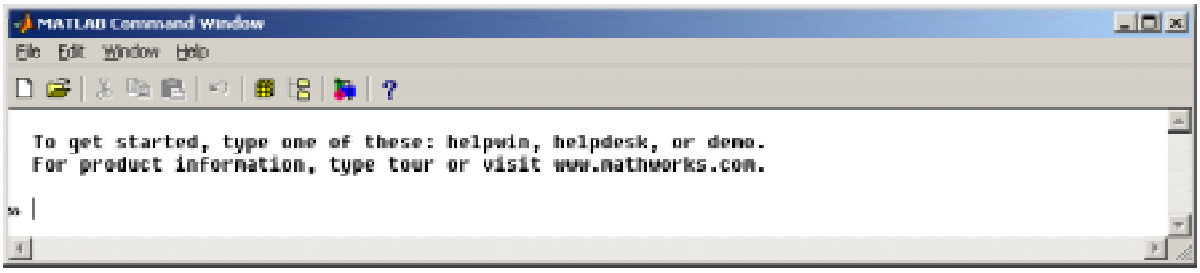
- pamięć *RAM*: 64MB, lub większa;
- system operacyjny: *Windows 95*, lub nowszy;
- zainstalowany *Matlab* w wersji 5.2, z pakietem *Control Toolbox* oraz nakładką *Simulink*.

2 Środowisko *Matlab*a

Standardowa instalacja programu Matlab znajduje się w folderze *c:\matlab* i zajmuje około 400MB. Plikiem uruchomieniowym jest *c:\matlab\bin\matlab.exe* ("ikona na pulpicie").

2.1 Linia poleceń, komunikacja z systemem operacyjnym

Uruchomienie *c:\matlab\bin\matlab.exe* powoduje otwarcie okna z linią poleceń (ang. *command line*)



Rys. 1: Okno główne

Przykładowe polecenia:

pwd – wyświetl aktualny katalog (print working directory);

cd nowykat – zmień katalog na *nowykat*;

who – wyświetla listę wszystkich zdefiniowanych w przestrzeni roboczej (ang. *WorkSpace*) zmiennych;

clear – czyści przestrzeń roboczą (wartości i etykiety wszystkich zmiennych zostają wykasowane).

Przestrzeń roboczą (wszystkie zmienne i ich wartości) można zapisaś w pliku z rozszerzeniem **.mat* wydając z menu */File* polecenie */SaveWorkspaceAs*. Analogicznie można otworzyć zapamiętane środowisko poprzez */File/LoadWorkspace*.

2.2 Definiowanie i modyfikacja zmiennych, obszar roboczy (*WorkSpace*)

Zdefiniowanie nowej zmiennej, lub modyfikacja już istniejącej następuje poprzez proste przypisanie jej wartości, np.

x=8; – definiuje zmienną o nazwie *x* (jeśli wcześniej taka nie istniała) i nadaje jej wartość *8*, średnik na końcu wyłącza "echo".

Aktualną wartość zmiennej można odczytać wpisując jej nazwę (bez średnika), zatem polecenie

x – powoduje wyświetlenie wartości *8*.

Dla skalarów dostępne są operatory: *, /, +, – i ^ (potęgowanie).

2.3 Operacje na macierzach

Należy mieć świadomość, że *Matlab* (nazwa od ang. *Matrix Laboratory*) w całości jest oparty na macierzach. Wszelkie sygnały (ich dyskretne odpowiedniki) są przechowywane właśnie jako macierze (ew. wektory).

Macierz definiuje się wykorzystując nawiasy kwadratowe [ ], znak przecinek lub spację, jako separator elemetów w danym wierszu i średnik jako separator wierszy, np. polecenie definiujące macierz A postaci $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ wygląda następująco

A=[1,2;3,4]

Do poszczególnych elementów macierzy odwołujemy się przez nawiasy okrągłe, np.

A(i,j)– oznacza element w i-tym wierszu i j-tej kolumnie macierzy A, może on stanowić *lvalue*

Macierz można definiować blokami, np. polecenie

B=[A,A]

utworzy macierz B postaci $\begin{bmatrix} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \end{bmatrix}$.

Sekwencje uporządkowane (np. kolejne elementy ciągu arytmetycznego) uzyskuje się przez konwencję *min:[krok]:max*, np. symbol *1:3:10* oznacza wektor $\begin{bmatrix} 1 & 4 & 7 & 10 \end{bmatrix}$, a polecenie

C=[1:3,1:2:5] – definiuje macierz C postaci $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \end{bmatrix}$.

W rozumieniu macierzowym dostępne są operatory *, +, – oraz ^. Mnożenie i dzielenie tablic (element po elemencie) wykonuję się z użyciem operatorów .* (kropka-gwiazdka) i ./ (kropka-slash).

2.4 Podstawowe funkcje algebry liniowej

Przykłady:

eye(n)– definiuje macierz jednostkową o rozmiarze $n \times n$;

zeros(n,m) – definiuje macierz o rozmiarze $n \times m$ złożoną z samych zer;

ones(n,m) – definiuje macierz o rozmiarze $n \times m$ złożoną z samych jedynek;

det(A) – wylicza wyznacznik macierzy A;

inv(A) – wyznacza macierz odwrotną do A: A^{-1}; równoważna postać polecenia to : *A^(-1)*;

trace(A) – wylicza ślad macierzy A;

lu(A) – wyznacza rozkład LU macierzy A;

chol(A) – wyznacza rozkład Cholesky'ego macierzy A;

svd(A) – wyznacza rozkład względem wartości szczególnych macierzy A;

eig(A) – wyznaczenie wartości własnych macierzy A;

rand(n,m) – generuje macierz losową o rozmiarze $n \times m$ i elementach z rozkładu jednostajnego $U[0,1]$;

randn(n,m) – generuje macierz losową o rozmiarze $n \times m$ i elementach z rozkładu normalnego $N(0,1)$;

3 Tworzenie i uruchamianie *M*-skryptów

Ciąg poleceń można umieścić w osobnym pliku (skrypcie) z rozszerzeniem **.m*. W tym celu należy wybrać z menu */File* polecenie */New/M-file*, co spowoduje otwarcie okna edytora.

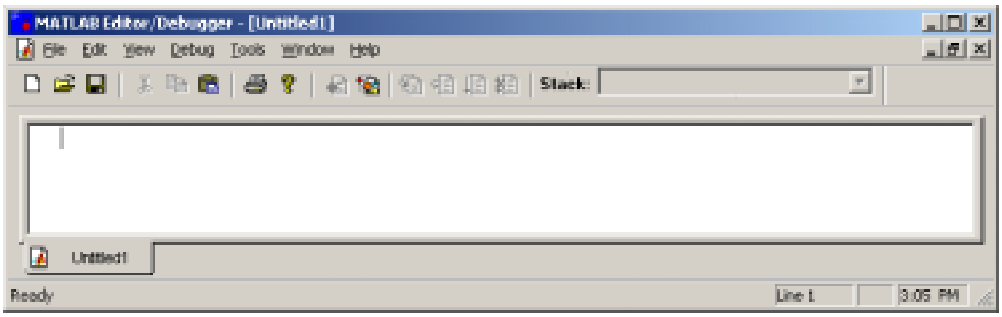
Dostępne są oczywiście instrukcje warunkowe (if ...) oraz iteracyjne (while, for). Przykładowy skrypt:

%Sekwancyjne wypełnienie wektora liczbami losowymi

for i=1:10 W(i)=i;

Znak % służy do wstawiania komentarzy.

Uruchomienie skryptu polega na wybraniu w menu */Tools* edytora polecenia */Run*. Możliwe jest również uruchomienie poprzez wpisanie nazwy *M-skryptu* w linii poleceń *Matlab*a (ustawiając poleceniem *cd* odpowiednią ścieżkę dostępu).



Rys. 2: Edytor skryptów

4 Funkcje pakietu *Control Toolbox*

Pakiet *Control Toolbox* to dodatkowy, tematyczny zestaw funkcji. Jego obsługa nie pociąga za sobą żadnych dodatkowych wymagań od użytkownika. Definiowanie systemu liniowego, którego transmitancja jest funkcją wymierną zmiennej s opiera się na utworzeniu dwóch wektorów wierszowych, zawierających odpowiednio współczynniki wielomianu licznika i mianownika transmitancji, np. systemowi całkującemu o transmitancji $\frac{1}{s}$ odpowiadają wektory [1] (licznik) i $[1, 0]$ (mianownik). Istnieje zestaw funkcji, które na podstawie wspomnianych wektorów tworzą system. Przykładem może być funkcja jest funkcja *tf()* (ang. transfer function). Do najczęściej wykorzystywanych funkcji zawartych w *Control Toolbox* należą:

bode(sys) – wyznacza charakterystyki Bode'go systemu *sys*;

feedback(sys1,sys2) – łączy dwa systemy (*sys1* z ujemnym sprzężeniem zwrotnym *sys2*);

freqresp(sys, w) – wyznacza odpowiedź systemu *sys* w dziedzinie częstotliwości ($Re(K(j\omega))$ oraz $\mathrm{Im}(K(j\omega))$) dla wartości pulsacji zadanych w wektorze w;

gensig(typ,okres) – generator standardowych sygnałów, np. *typ='sin'*, *okres=1* – generuje falę sinusoidalną o okresie 1s;

impulse(sys) – wyznacza odpowiedź impulsową systemu *sys*;

kalman() – projektuje estymator *Kalmana*

lsim(sys,u,t) – symuluje system *sys* przy dowolnym pobudzeniu zawartym w wektorze u, (t – wektor czasu);

ltiview(typ,sys) – wykreśla charakterystykę typu *typ* (np. *'step'*, *'impulse'*, *'nyquist'*) systemu *sys*

parallel(sys1,sys2) – połączenie równoległe systemów *sys1* i *sys2*;

series(sys1,sys2) – połączenie szeregowe systemów *sys1* i *sys2*;

ss(A,B,C,D) – tworzy system liniowy na podstawie opisu w przestrzeni stanów: $x' = Ax + Bu$, $y = Cx + Du$;

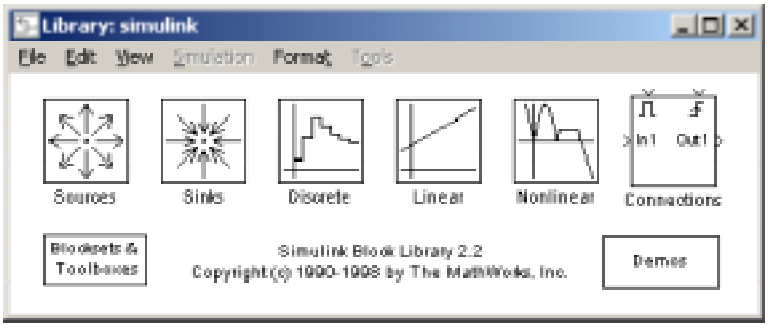
tf(L,M) – tworzy system liniowy na podstawie wektorów współczynników transmitancji licznika L i mianownika M.

5 Nakładka graficzna *Simulink*

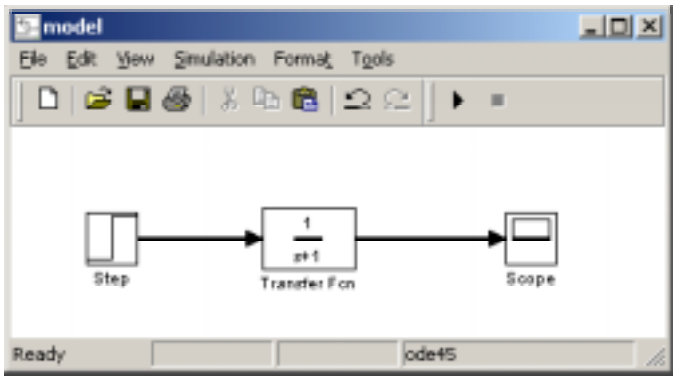
Nakładka *Simulink* ułatwia tworzenie modeli oraz zwalnia nas w znacznym stopniu od znajomości składni funkcji pakietu *Control Toolbox*. Uruchomienie nakładki odbywa się poleceniem *simulink*, co powoduje otwarcie okna zawierającego zbiór zasobników (Rys. 3). Oprócz niego otwiera się także czyste okno edytora graficznego modeli (Rys. 4).

5.1 Edytor graficzny

Budowa schematów blokowych odbywa się na zasadzie "przeciągania myszą" bloków zasobników do edytora graficznego (Rys. 4). Schemat (model) można zapisać w pliku z rozszerzeniem **.mdl*. Połączenia pomiędzy poszczególnymi blokami następują również poprzez "przeciąganie". Węzły połączeniowe uzyskujemy przystrzymując klawisz *Ctrl*.



Rys. 3: Zasobnik nakładki Simulink



Rys. 4: Graficzny edytor modeli (schematów symulacji)

Symulację gotowego modelu uruchamia się poleceniem *Ctrl+T* (*/Start* w menu */Simulation*). Okno ustawień parametrów symulacji wywołuje się poleceniem Ctrl+E (*/Parameters* w menu */Simulation*).

Własności i parametry elementów zawartych w schemacie zmienia się po dwukrotnym kliknięciu na poszczególne obiekty (patrz przykłady na Rys. 6–8).

Po załadowaniu pliku **.mat* do obszaru roboczego (patrz punkt 2.1) można zastosować funkcję *plot()* w celu uzyskania wykresu.

5.2 Widoczność danych w *Matlab*ie

Wartości wszystkich zmiennych *Matlab*a są widoczne w *Simulink*u, zaś wartości wszystkich zmiennych *Simulink*a są widoczne w *Matlab*ie.

5.3 Źródła sygnałów (*Sources*)

Z punktu widzenia tematyki laboratorium najbardziej przydatne będą następujące bloki generujące (Rys. 9):

step – skok jednostkowy;

ramp – sygnał liniowo narastający;

sine wave – sinusoida;

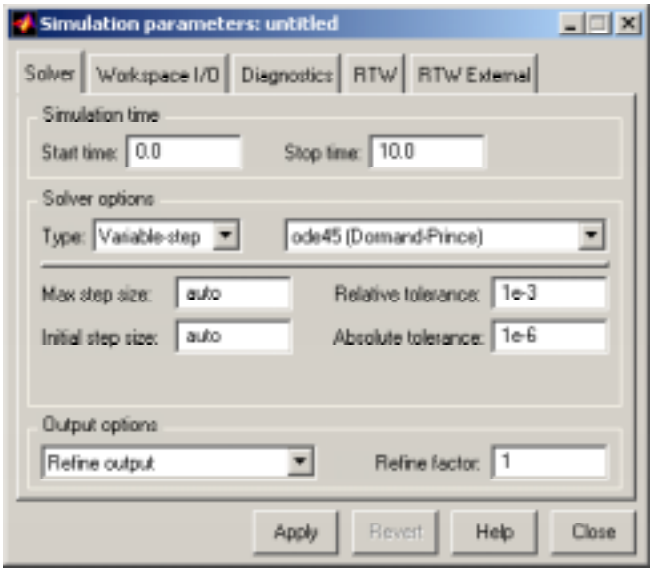
from file – sygnał (wektor) zawarty w wyspecyfikowanym pliku;

random number – generator liczb losowych.

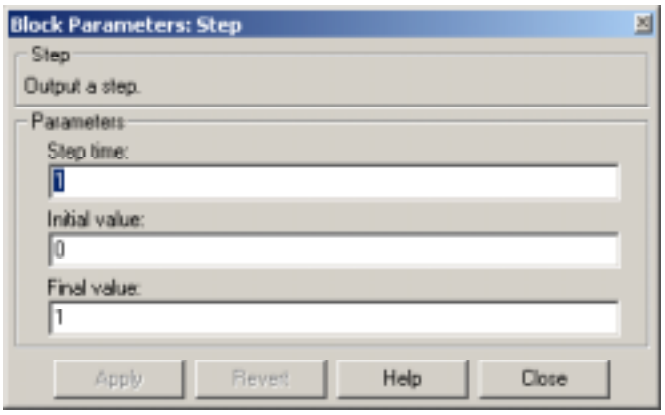
5.4 Rejestratory sygnałów (*Sinks*)

Wśród bloków rejestrujących (Rys. 10) wykorzystujemy:

scope – rejestrator jednowymiarowy (w dziedzinie czasu);



Rys. 5: Parametry symulacji



Rys. 6: Własności bloku źródłowego STEP

xy graph – rejestrator dwuwymiarowy
to file – zapis do pliku.

5.5 Elementy liniowe z czasem ciągłym (*Linear*)

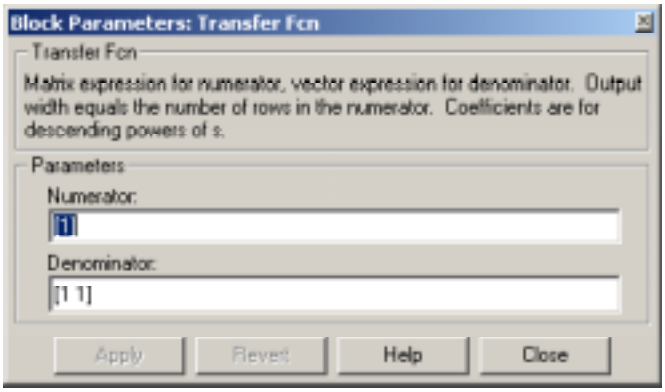
Najbardziej przydatny będzie blok *Tranfer Function* (patrz Rys 11)

5.6 Elementy nieliniowe (*Nonlinear*)

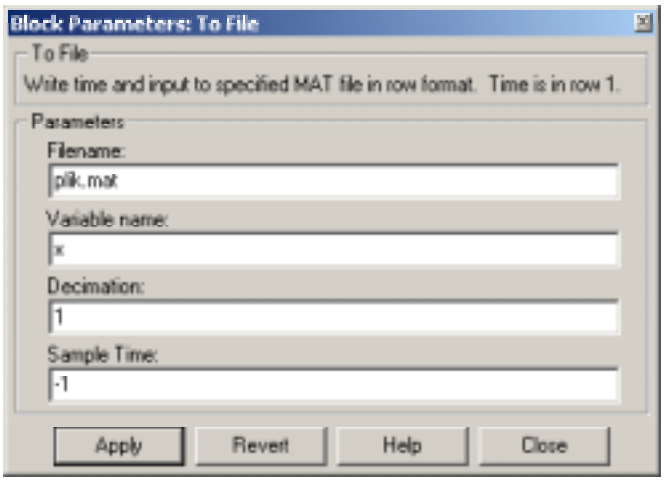
Z zasobnika *Nonlinear* (Rys. 12) bardzo przydatne będą bloki:
Transport Delay – opóźnienie;
Matlab Function – realizacja dowolnego wzoru matematycznego (domyślna nazwa sygnału wejściowego: *u*).

5.7 Systemy z czasem dyskretnym (*Discrete*)

Patrz Rys. 13



Rys. 7: Własności bloku liniowego TRANSFER FUNCTION



Rys. 8: Własności rejestratora TO FILE

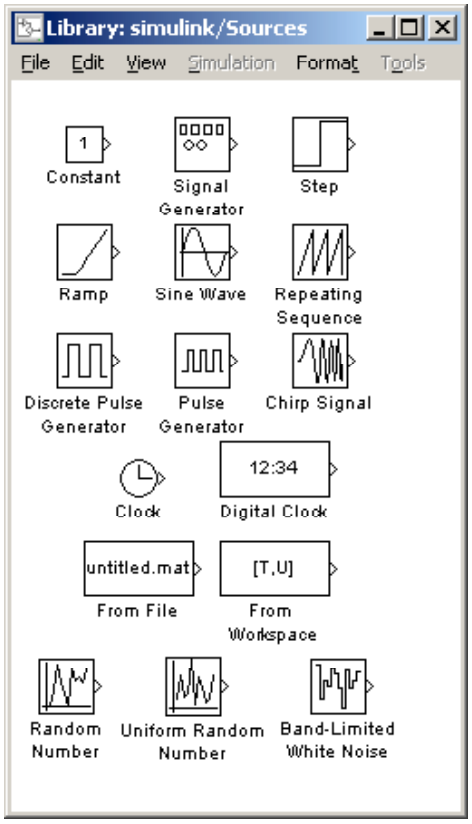
6 Korzystanie z *HELP*a

Pomoc można uzyskać wydając w oknie głównym *Matlab*a polecenie *help nazwa_funkcji.* Najwygodniej jest jednak korzystać z lokalnych zbiorów w formacie HTML, wybierając polecenie */HelpDesk* z menu */Help.*

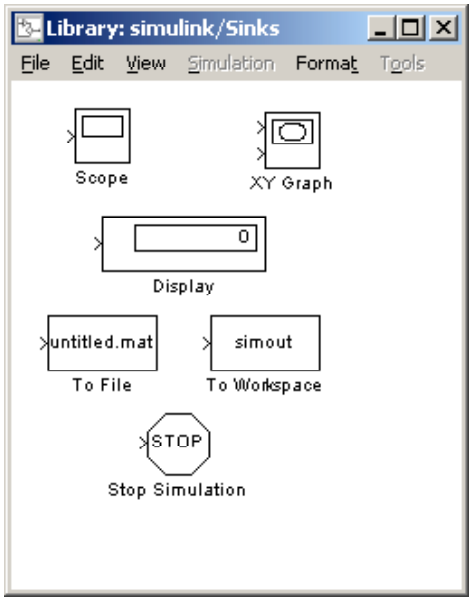
LITERATURA

[1] A. Zalewski, R. Cegieła, "Matab - obliczenia numeryczne i ich zastosowania", wyd. Nakom, Poznań, 1997.

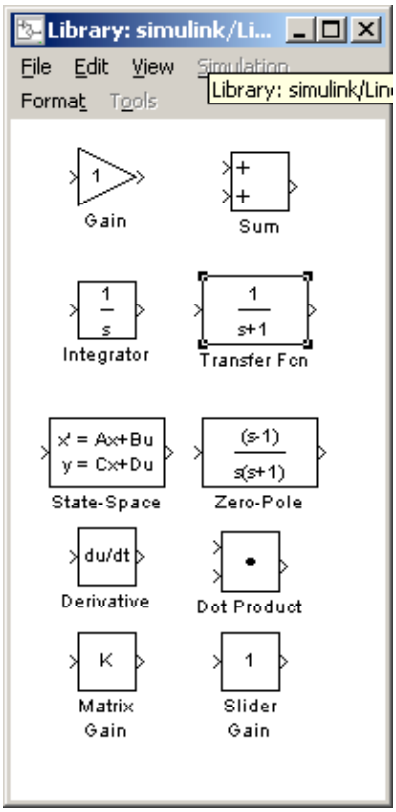
[2] J. Brzózka, "Ćwiczenia z automatyki w Matlabie i Simulinku", wyd. Mikom, Warszawa, 1997.



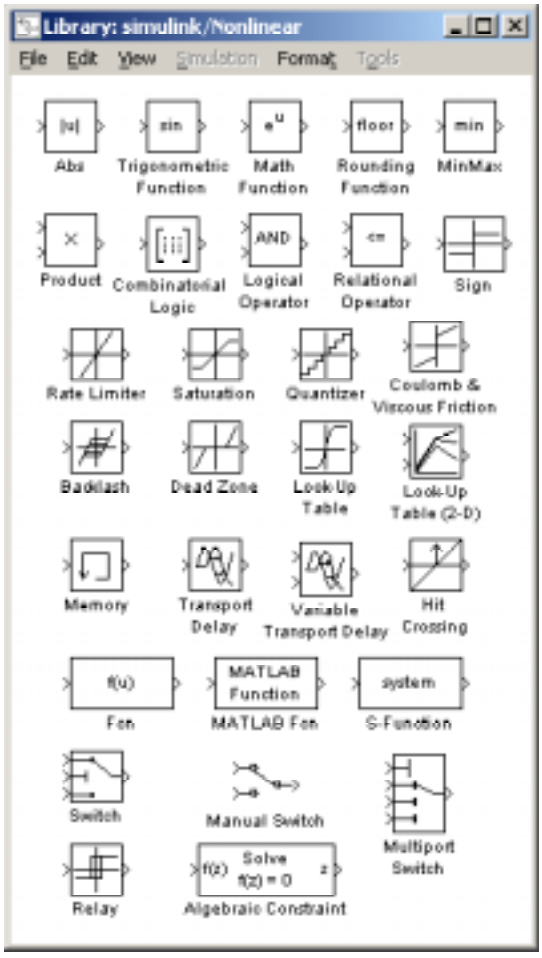
Rys. 9: Bloki generujące sygnały



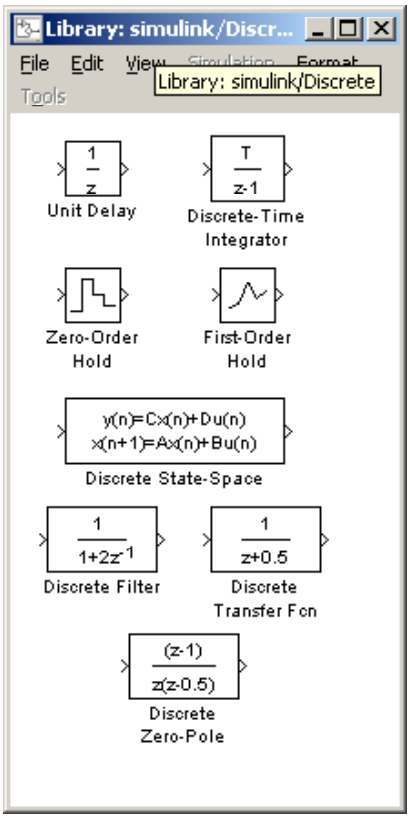
Rys. 10: Bloki odbiorcze – rejestrujące sygnały



Rys. 11: Modele liniowe z czasem ciągłym



Rys. 12: Modele nieliniowe



Rys. 13: Modele z czasem dyskretnym