Laboratorium 7

Łukasz Janusz, grupa 3

Wstęp

Na zajęciach zapoznałem się z możliwością wykorzystania S-funkcji do tworzenia modeli w simulinku. W ramach ćwiczenia utworzyłem model symetrycznego układu hamowania samolotu na lotniskowcu korzystając z S-funkcji.

Stworzona S-funkcja

```
 function [sys, x0, str, ts] = Janusz (t, x, u,flag)
 % Type "open sfunctmpl" in command to open new S-Function
 % Save it as your own file,like "example" in this case;
 % Change the function name as your file name
 % Goto line 114, there you will define the differential eq in "function
sys=mdlDerivatives(t,x,u)"
 % Goto line 82, to define the continous states, inputs from simulink block and
output in simulink in "function [sys,x0,str,ts]=mdlInitializeSizes"
 % Goto line 94, for Initial conditions of differential eq in "function
[sys,x0,str,ts]=mdlInitializeSizes"
 % Goto line 151, where you will declare the outputs in "function
sys=mdlOutputs(t,x,u)"

 %
 % The following outlines the general structure of an S-function.
 %
 switch flag

   %%%%%%%%%%%%%%%%%%
   % Initialization %
   %%%%%%%%%%%%%%%%%%
   case 0
     [sys,x0,str,ts]=mdlInitializeSizes;

   %%%%%%%%%%%%%%%
   % Derivatives %
   %%%%%%%%%%%%%%%
   case 1
     sys=mdlDerivatives(t,x,u);

   %%%%%%%%%%
   % Update %
   %%%%%%%%%%
   case 2
```

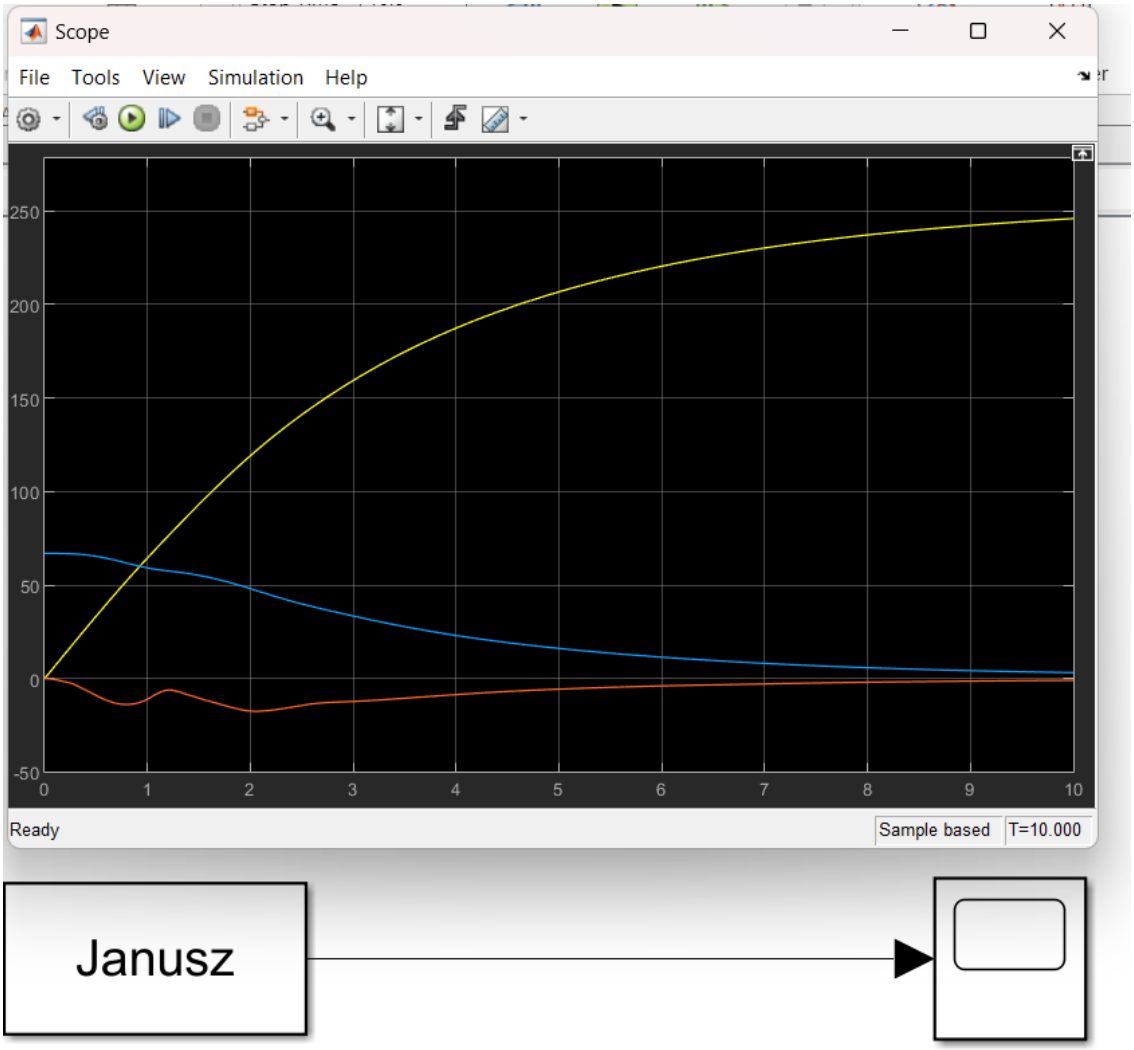
```
    sys=mdlUpdate(t,x,u);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3
    sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%
  case 4
    sys=mdlGetTimeOfNextVarHit(t,x,u);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%
  case 9
    sys=mdlTerminate(t,x,u);

  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end sfuntmpl

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts]=mdlInitializeSizes

%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
% Note that in this example, the values are hard coded.  This is not a
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
%
```

```
sizes = simsizes;

sizes.NumContStates  = 6;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

%
% initialize the initial conditions
%
x0  = [0 67 0 0 0 0];

%
% str is always an empty matrix
%
str = [];

%
% initialize the array of sample times
%
ts  = [0 0];

% end mdlInitializeSizes

%
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
%
function sys=mdlDerivatives(t,x,u)

    h = 42;       %[m]
    m1 = 14000;   %[kg]
    m2 = 450.28;  %[kg]
    m3 = 200;     %[kg]
    k1 = 54700;   %[N]
    k2 = 303600;  %[N]

    % interpolacja
    wezlyF3 = [0 10 20 30 40 50 60 70 80 90 94 98 102 104 107 120];
```

```
    wartosciF3 = [833 400 160 320 520 520 660 830 1070 1600 2100 2800 4100 5000
9000 9000];
    funF3 = interp1(wezlyF3,wartosciF3,x(5),'pchip');
    Fb = funF3*x(6)^2;

    % zmienne stanu
    y1 = sqrt(x(1)^2 + h^2) - h;
    sin_theta = x(1)/sqrt(x(1)^2 + h^2);
    if y1 - 2*x(3) >= 0
        Fk1 = k1 * (y1 - 2*x(3));
    else
        Fk1 = 0;
    end
    if x(3) - x(5) >= 0
        Fk2 = k2 * (x(3) - x(5));
    else
        Fk2 = 0;
    end
    dx(1) = x(2);
    dx(2) = (-2 * Fk1 * sin_theta) / m1;
    dx(3) = x(4);
    dx(4) = (2 * Fk1 - Fk2) / m2;
    dx(5) = x(6);
    dx(6) = (Fk2 - Fb) / m3;
sys = [dx(1);dx(2);dx(3);dx(4);dx(5);dx(6)];

%
%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
%

function sys=mdlOutputs(t,x,u)
    h = 42;
    m1 = 14000;
    k1 = 54700;
    y1 = sqrt(x(1)^2 + h^2) - h;
    sin_theta = x(1)/sqrt(x(1)^2 + h^2);
    if y1 - 2*x(3) >= 0
        Fk1 = k1 * (y1 - 2*x(3));
    else
        Fk1 = 0;
    end
    acc = (-2 * Fk1 * sin_theta) / m1;
```

```
sys = [x(1) x(2), acc];

%
%=============================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
%

function sys=mdlUpdate(t,x,u)
sys = [];


%
%=============================================================================
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.  Note that the result is
% absolute time.  Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=============================================================================
%

function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;


%
%=============================================================================
% mdlTerminate
% Perform any end of simulation tasks.
%=============================================================================
%


function sys=mdlTerminate(t,x,u)

sys = [];
```

Wynik



Podsumowanie

Jak widać, uzyskany wynik jest identyczny z wynikiem uzyskanym na poprzednich zajęciach. Dzięki temu wiem, że moja funkcja działa poprawnie i nie wymaga edycji. Z perspektywy simulinka, implementacja modelu jest bardzo prosta, jednakże pierwszy raz spotkałem się z tworzeniem S-funkcji, przez co nie uważam tego za najprzyjemniejszy sposób wykonania zadania. Bardzo pomocną rzeczą okazały się być komentarze zostawione w szablonie funkcji, dzięki nim wiedziałem, gdzie jakie dane wpisać.