Sprawozdanie z laboratorium Modelowanie Systemów Dynamicznych

Temat: Rozwiązywanie ODE w MATLABIE

Marceli Jach, wydział EAIiIB, Automatyka i Robotyka, grupa 4. 30.11.2022r.

Cel ćwiczenia:

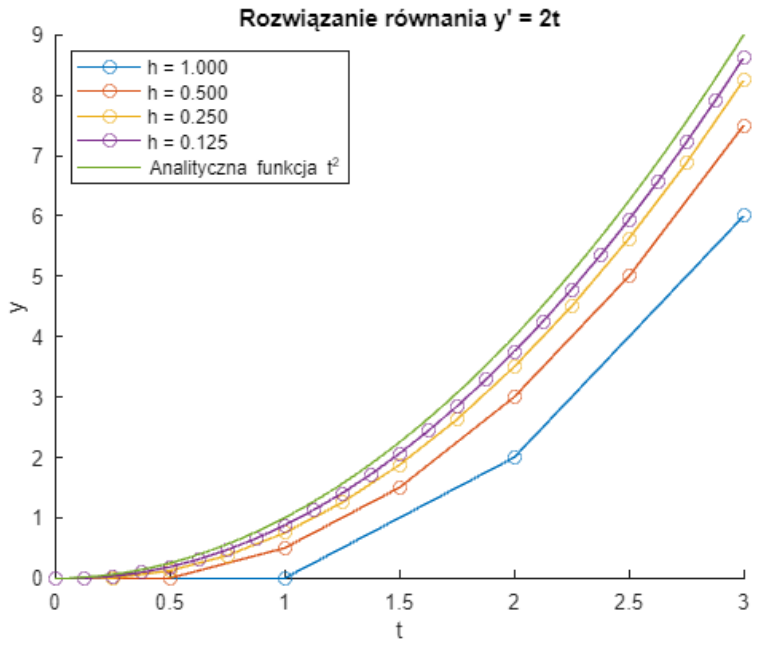
Celem ćwiczenia było wykonanie trzech zadań związanych z rozwiązywaniem równań różniczkowych zwyczajnych (Ordinary Differential Equion) w środowisku MATLAB®. Dzięki możliwości rozwiązywania ODE, możliwe jest modelowanie różnych systemów dynamicznych za pomocą samego kodu MATLAB – nie ma potrzeby aby korzystać z Simulinka.

Zadanie 1.

Zadanie pierwsze polegało na napisaniu skryptu rozwiązującego równanie y’ = 2t metodą Eulera, w zakresie [0,3], dla warunku początkowego $y_0 = 0$. Rozwiązanie należało nanieść na wykres dla odpowiedniego kroku [1, 0.5, 0.25, 0.125]. Oprócz tego należało nanieść rozwiązanie analityczne, dla porównania dokładności.

Kod realizujący zadanie:

```
%Zadanie 1
h = [1,0.5,0.25,0.125];
f = @(t) 2*t;
N = 3;
figure
hold on
for i = h
    t = 0:i:N;
    y(1) = 0;
    for j = 1:N/i
        dy = f(t(j));
        y(j+1) = y(j) +dy*i;
    end
    plot(t,y,"o-", "DisplayName",sprintf("h = %.3f",i))
end
fplot(@(t) t.^2,[0,3],"DisplayName","Analityczna funkcja t^2")
legend("Location","northwest")
title("Rozwiązanie równania y' = 2t")
xlabel("t")
ylabel('y')
```



Rysunek 1. Wykres przedstawiający rozwiązania równania y' = 2t metodą Eulera dla różnych wartości kroków

Można zauważyć że dokładność takiego rozwiązania jest proporcjonalna do gęstości kroku. Niestety, znaczne zwiększenie kroku niesie ze sobą wzrost złożoności obliczeniowej, przez co taka metoda nie jest najwydajniejszą. Dużo lepszym pomysłem, byłoby skorzystanie z tzw. solverów, czyli metod numerycznych rozwiązujących zagadnienie początkowe dla równań różniczkowych zwyczajnych.

Zadanie 2.

Zadanie to polegało na rozwiązaniu zadania pierwszego za pomocą solvera ode45 i porównanie wyników z metodą Eulera. Środowisko Matlab udostępnia wiele typów solverów. Każdy z nich ma takie samo zadanie, natomiast niektóre mogą być w różnych przypadkach wydajniejsze poprzez korzystanie z różnych metod obliczania rozwiązania. W naszym przypadku wykorzystamy solver ode45.

Kod realizujący zadanie:

```
fun = @(t,y) 2*t;
[t,y] = ode45(fun,[0,3],0);
figure
plot(t,y,'r')
title("Rozwiazanie rownania y' = 2t za pomocą solvera")
xlabel("t")
ylabel('y')
```

Rysunek 2. Wykres przedstawiający rozwiązanie równania y' = 2t z wykorzystaniem solvera ode45

Porównując wykres z zadania 1 do powyższego wykresu, można zauważyć że rozwiązanie z wykorzystaniem ode45 jest dokładniejsze niż wszystkie poprzednie opcje. Narzędzie to nie wymaga od nas zbyt wiele kodu i łączy w sobie szybkość i dokładność co czyni je dobrym narzędziem do celów modelowania systemów dynamicznych.

Zadanie 3

Zadanie to polegało na obliczeniu równań różniczkowych opisujących urządzenie do hamowania lądujących samolotów które wystąpiło na poprzednich zajęciach laboratoryjnych. Nie będę tutaj opisywać budowy tego urządzenia, ani w jaki sposób działa, ponieważ opisałem to już w sprawozdaniu poprzednim. Zamiast tego zamieszczę jedynie oznaczenia dla zmiennych stanu oraz równania stanu. Zadanie należało wykonać wykorzystując poznany dzisiaj solver ode45.

$$\begin{cases} x_1 = x \\ x_2 = \dot{x} \\ x_3 = y_2 \\ x_4 = \dot{y_2} \\ x_5 = y_3 \\ x_6 = \dot{y_3} \end{cases}$$

Rysunek 3. Zmienne stanu.

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -2f_{k1}sin\theta/m_1 \\ \dot{x_3} = x_4 \\ \dot{x_4} = (2f_{k1} - f_{k2})/m_2 \\ \dot{x_5} = x_6 \\ \dot{x_6} = (f_{k2} - f_b)/m_3 \end{cases}$$

Rysunek 4. Równania stanu

W celu wykonania tego zadania, należało najpierw napisać funkcję w Matlabie która będzie implementować równania stanu, czyli wyliczać pochodne zmiennych stanu (de facto będzie opisywać działanie hamownika).

Kod funkcji:

```
function Dx = hamownik(t,x)
h = 42;       %[m]
k2 = 303600;
k1 = 54700;
m3 = 200;
m2 = 450.28;
m1 = 14000;
% 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)/(h+y1);
if y1 >= 2*x(3)
    Fk1 = k1*(y1-2*x(3));
else
    Fk1 = 0;
end
if x(3) >= x(5)
    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;
Dx = [dx(1);dx(2);dx(3);dx(4);dx(5);dx(6)];
```

Zaimplementowana w taki sposób funkcja będzie zwracać pochodne zmiennych stanu. Mając to na uwadze będzie można przekazać uchwyt do tej funkcji do solvera ode45, który otrzymując oprócz tego zakres czasu w którym chcemy wyliczyć te zmienne oraz warunki początkowe, zwróci nam rozwiązanie. Warto jednak mieć na uwadze to, że przyspieszenie nie jest zmienną stanu. W takim razie obliczenia wykonane przez solver nie pozwolą nam narysować wykresu przyspieszenia. W związku z tym musimy zdefiniować dodatkową funkcję wyjść która będzie działać jako funkcja pomocnicza do ode45.

Kod funkcji pomocniczej:

```
function status = hamownik_out(t,x,flag)


global w3
h = 42;       %[m]
k1 = 54.7e3;  %[N/m]
m1 = 14000;   %[kg]

if strcmp(flag,'init')
    w3 = 0;
elseif isempty(flag)
    y1 = sqrt(x(1)^2+h^2)-h;
    sin_alfa = x(1)/(h+y1);

    if y1 >= 2*x(3)
        Fk1 = k1*(y1-2*x(3));
    else
        Fk1 = 0;
    end

    w3 = [w3;-2*Fk1*sin_alfa/m1];
end
status = 0;
```

Funkcja ta na podstawie aktualnych zmiennych stanów oraz flag przekazywanych do niej przez funkcje ode45 oblicza aktualne przyspieszenie, oraz przypisuje je do zmiennej globalnej w3.

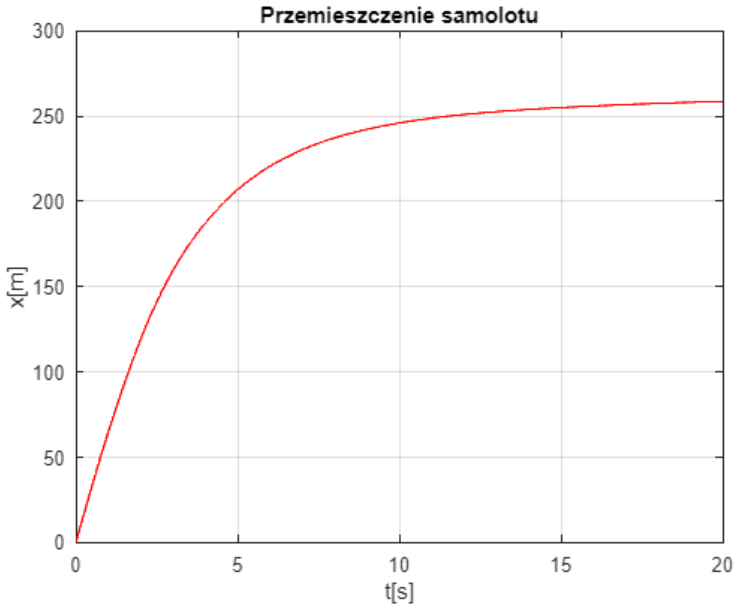
Mając do dyspozycji te dwie funkcje, jesteśmy w stanie napisać kod który będzie z nich korzystał, oraz wyrysuje dla nas wykresy przemieszczenia, prędkości oraz przyspieszenia samolotu. Dzięki temu będziemy mogli sprawdzić poprawność implementacji.

Kod programu:

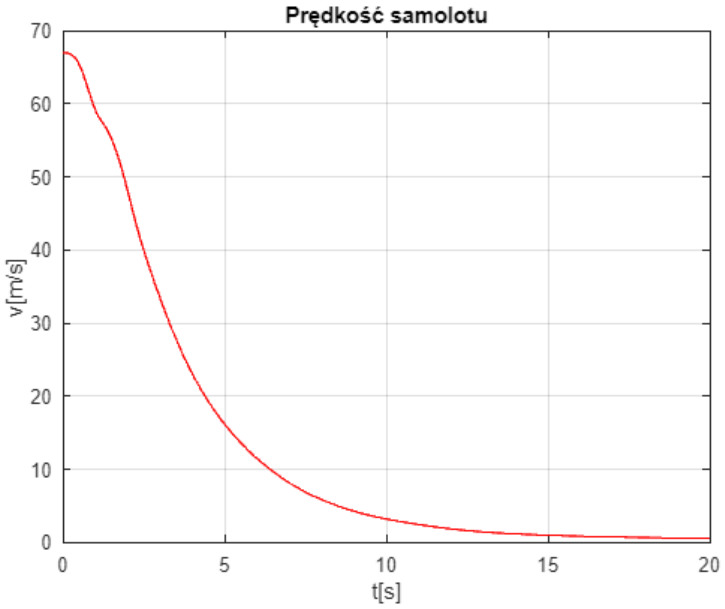
```
global w3
options = odeset('OutputFcn',@hamownik_out,'Refine',1);
[T,Y] = ode45(@hamownik,[0 20],[0 67 0 0 0 0],options);

figure
plot(T,Y(:,1),'r')
title("Przemieszczenie samolotu")
xlabel("t[s]"); ylabel("x[m]"); grid on;
figure
plot(T,Y(:,2),'r')
title("Prędkość samolotu")
xlabel("t[s]"); ylabel("v[m/s]"); grid on;
figure
plot(T,w3,'r')
title("Przemieszczenie samolotu")
xlabel("t[s]"); ylabel("a[m/s^2]"); grid on;
```

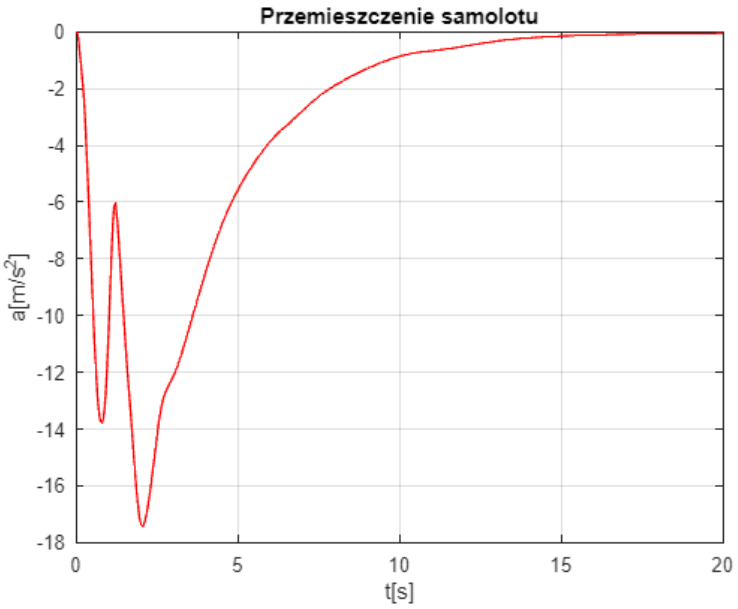
Powyższy kod powoduje zwrócenie poniższych wykresów:



Rysunek 5. Wykres przedstawiający przemieszczenie samolotu względem miejsca rozpoczęcia hamowania.



Rysunek 6. Wykres przedstawiający prędkość samolotu od momentu rozpoczęcia hamowania.



Rysunek 7. Wykres przedstawiający przyspieszenie samolotu od momentu rozpoczęcia hamowania.

Wykresy pokrywają się z tymi wyznaczonymi na poprzednich zajęciach laboratoryjnych. Na tej podstawie można stwierdzić że obliczenia zostały wykonane poprawnie.

Wnioski:

Rozwiązywanie równań różniczkowych jest podstawą modelowania systemów dynamicznych. Aby sprawnie wykonywać modele, należy również sprawnie obliczać rozwiązania takich równań. Umożliwiają to różne metody numeryczne zaimplementowane w środowiska inżynierskie takie jak chociażby MATLAB®. Wykorzystywana podczas dzisiejszych zajęć funkcja ode45 znacznie przyspiesza tą część wykonywania modelu i umożliwia otrzymanie wystarczająco dokładnego wyniku. Chcąc obliczyć równanie różniczkowe zwyczajne w przyszłości, z całą pewnością skorzystam z solverów udostępnianych przez MATLAB®.