Laboratorium 1

Rozwiązywanie równań różniczkowych z niezerowymi warunkami początkowymi

1. Cel ćwiczenia

- Zapoznanie się z metodami symbolicznego i numerycznego rozwiązywania równań różniczkowych w Matlabie,
- Wykorzystanie Simulinka do tworzenia modelu równania różniczkowego w postaci schematu blokowego
- Archiwizacja otrzymanych rozwiązań
- Publikacja wyników

2. Wprowadzenie teoretyczne

a) Symboliczne rozwiązywanie równań różniczkowych – **`dsolve()`**

Rozwiązanie równania różniczkowego przy użyciu zmiennych symbolicznych polega na obliczeniach wykonywanych na wyrażeniach matematycznych, a nie na liczbach (rozwiązanie numeryczne), w wyniku czego otrzymujemy również wyrażenie matematyczne. Przy pomocy zmiennych symbolicznych oraz przez wykorzystanie funkcji **`dsolve()`** możliwe jest rozwiązanie równania różniczkowego dowolnego rzędu.

W rozwiązaniu symbolicznym równania różniczkowego najważniejsza jest zmienna **`D`** (duże D), która określa różniczkę pierwszego stopnia ($D = {d}/{dt}$), podobnie **`D2`** oznacza różniczkę drugiego stopnia ($D2 = {d^2}/{dt^2}$) itd. Funkcja **`dsolve()`** domyślnie różniczkuje po czasie.

Składnia funkcji:

```
dsolve('rownanie_rozniczkowe','warunek1' , 'warunek2');
```

Za pomocą funkcji **`dsolve()`** możliwe jest również rozwiązanie układu równań różniczkowych jak i określenie warunków początkowych. Kolejne równania podajemy po przecinkach, każde napisane pomiędzy parą apostrofów. Bezpośrednio po napisaniu równań, umieszczamy warunki początkowe w apostrofach, również oddzielone przecinkami.

Składnia funkcji:

```
dsolve('rownanie1' , 'rownanie2' , … , 'warunek1' , 'warunek2');
```

Przykład 1:

Rozwiązać równanie różniczkowe:

$$\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 2x = 0 \text{ przy war. pocz. } x(0) = 0, \frac{dx}{dt}(0) = 2$$

Wykorzystując funkcję `dsolve()`.

Rozwiązanie: Tworzymy m-plik o nazwie rozw1.m

```
%% Rozwiązanie równania różniczkowego metodą dsolve()
% Równanie różniczkowe: x"+3x'+2x=0, x(0)=0, x'(0)=2
%%
clear all;  % czyszczenie pamięci
close all;  % zamykanie okien
clc;        % czyszczenie okna komend Matlaba
%%
tic     % uruchomienie zegara mierzącego czas wykonywania programu
syms x y;   % definiowanie zmiennych symbolicznych
y=dsolve('D2x+3*Dx+2*x=0' , 'x(0)=0' , 'Dx(0)=2');% wprowadzenie
% parametrów równania i warunków początkowych
%%
pretty(y)       % wypisanie rozwiązania w oknie komend Matlaba
%%
t=0:0.01:9.99;  % zdefiniowanie wektora czasowego t
w=subs(y);      % podstawienie wektora czasowego t do równania y
%%
plot(t,w);  % wykres rozwiązania w funkcji czasu
title('Wykres rozwiązania rownania rozniczkowego');%tytułu wykresu
xlabel('Czas [s]'); % podpisanie osi X
ylabel('Amplituda sygnału'); % podpisanie osi Y
grid;       % naniesienie siatki na wykres
a = toc     % wyświetlenie czasu działania programu i zapisanie go
% w zmiennej a
```

b) Rozwiązanie numeryczne – funkcja **ode**xx**()**

MATLAB zawiera funkcje rozwiązujące zagadnienie początkowe dla równań różniczkowych zwyczajnych za pomocą np. par metod Rungego-Kutty rzędu 2 i 3 (funkcja **ode23()**) oraz rzędu 4 i 5 (funkcja **ode45()**). Istnieją jeszcze inne metody typu **ode** jednak te dwie są najczęściej stosowane i generują bardzo dobre wyniki. Metoda **ode23()** od **ode45()** różni się między innymi szybkością oraz dokładnością wyniku (**ode45()** jest wolniejsze, ale dokładniejsze).

Funkcje te rozwiązują zagadnienia początkowe dla układów równań zwyczajnych postaci

$$\frac{dx}{dt} = F(t,x), \qquad x(t_0) = x_0$$

Oznacza to, że równanie różniczkowe *n*-tego rzędu należy najpierw zamienić na *n* równań różniczkowych pierwszego rzędu.

Zatem równanie $x'' + 3x' + 2x = 0$ należy zmienić na:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -3x_2 - 2x_1 \end{cases}$$

Składnia funkcji:

```
[T,X]=ode23('nazwa_funkcji_F_od_t_x',t0:dt:tk,x0);
[T,X]=ode45('nazwa_funkcji_F_od_t_x',t0:dt:tk,x0);
```

Gdzie:

- `'nazwa_funkcji_F_od_t_x'` – nazwa pliku funkcyjnego zawierającego *n* równań różniczkowych pierwszego rzędu,
- `t0:dt:tk` – wektor czasu o początku w chwili `t0`, kroku `dt` i końcu symulacji w chwili `tk`,

- `x0` – wektor zawierający wszystkie warunki początkowe równania różniczkowego zaczynając od x(0) a kończąc na $x^{(n-1)}(0)$
- `T` – zwracany przez funkcję wektor czasu symulacji
- `X` – macierz zawierająca *n* wektorów, z których każdy jest kolejną pochodną rozwiązania różniczkowego, czyli pierwsza kolumna macierzy `X` reprezentuje równanie x=F(t), druga kolumna to $^{dx}/_{dt}$=F(t) itd. Każda kolumna zawiera określone wartości dla kolejnych chwil czasowych wektora `T`.

Przykład 2:

Rozwiązać równanie różniczkowe z przykładu 1 wykorzystując funkcję **`ode45()`**

Rozwiązanie: Aby rozwiązać to zadanie należy utworzyć dwa osobne pliki. Najpierw stworzyć oddzielny plik funkcyjny, który będzie wywoływany z funkcji **`ode45()`**. Następnie plik główny, w którym umieszczone są wszystkie podstawowe komendy.

Plik funkcyjny:

```
function xdot=rrode(t,x) % funkcja o nazwie rrode, do której
% dostarczane są zmienne t oraz x, natomiast zwracana jest wartość
% xdot, plik należy zapisać pod nazwą rrode.m

xdot=zeros(2,1);    % zdefiniowanie obszaru pamięci przy użyciu
% macierzy wypełnionej zerami o rozmiarze 2x1, bez tego program
% zwróci błąd, gdyż w pierwszym kroku nie będzie wiedział skąd
% odczytać i gdzie zapisać dane

xdot(1)=x(2);   % definiujemy x1' = x2
xdot(2)=(-3*x(2)-2*x(1));   % definiujemy x2' = -3*x2 - 2*x1
```

Główny plik programu:

```
%% Rozwiązanie równia różniczkowego przy pomocy komendy ode45()
% Równanie różniczkowe takie jak w przykładzie 1
%%
tic;
t0 = 0;      % określamy czas początkowy na 0 sekund
dt = 0.01;   % krok symulacji wynosi 0.01 sekundy
tk = 9.99;   % czas trwania symulacji
x01 = 0;     % pierwszy warunek początkowy x(0) = 0
x02 = 2;     % drugi warunek początkowy x'(0) = 2
x0 = [x01 x02];   % łączymy warunki początkowe w jeden wektor
%%
[T,X] = ode45('rrode',t0:dt:tk,x0); % wywołanie funkcji ode45()
% do rozwiązania równania różniczkowego zapisanego w pliku rrode.m
%%
plot(T,X(:,1),'b'); % wyrysowanie pierwszej kolumny macierzy X
% w funkcji T opcja 'b' oznacza, że wykres będzie niebieski
title('Rozwiązanie r. r. metodą ode45()'); % nadanie tytułu
xlabel('Czas [s]'); % podpisanie osi X
ylabel ('Amplituda sygnału'); % podpisanie osi Y
grid;      % naniesienie siatki na wykres
b=toc      % wyświetlenie czasu działania programu i zapisanie
           % go w zmiennej b
```

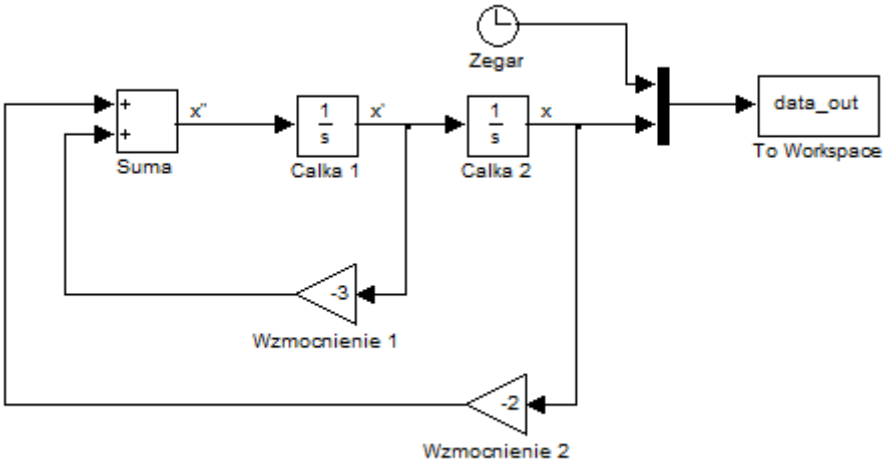
Uwaga: W odróżnieniu od większości języków programowania, w Matlabie, funkcje definiuje się zawsze w osobnych plikach. Dzięki temu są one dostępne dla innych programów. Nazwa funkcji musi być identyczna z nazwą pliku, w którym jest zapisana.

c) Rozwiązanie równań różniczkowych przy użyciu pakietu Simulink

Pakiet Simulink uruchamia się poprzez wpisanie komendy **`simulink`** w oknie komend Matlaba lub poprzez kliknięcie ikony  na pasku narzędzi.

Tak jak w przypadku rozwiązania numerycznego należy zamienić równanie różniczkowe *n*-tego rzędu na *n* równań różniczkowych pierwszego rzędu.

Na podstawie powyższego układu równań tworzymy schemat blokowy:



Warunki początkowe wstawia się w blokach reprezentujących całki (1/s), warunek x'(0) w bloku „Całka 1", warunek x(0) w bloku „Całka 2".

Aby eksportować wyniki do przestrzeni roboczej Matlaba należy użyć bloku „To Workspace" (Simulink => Sinks). W celu zapisu danych w postaci macierzy w bloku tym należy zmienić ustawienia opcji „Save format" ze „Structure" na „Array". Dzięki temu w przestrzeni roboczej Matlaba pojawi się macierz o nazwie takiej samej jak nazwa bloku.

Aby uzyskać wyniki takie jak w przykładzie 1 i 2, w parametrach symulacji trzeba ustawić maksymalny krok symulacji 0.01 sek. (Simulation => Configuration parameters => Max step size).

Model należy zapisać w pliku model_lab1.mdl.

Przykład 3:

M-plik uruchamiający symulację:

```
%% Rozwiązanie równania różniczkowego przy użyciu schematu
blokowego z Simulinka
%
% Schemat blokowy równania różniczkowego:
%
% <<model.png>> - plik z grafiką o nazwie model.png powinien się
znajdować w tym samym katalogu co publikowany dokument
%%
tic          % uruchomienie zegara mierzącego czas pracy programu
sim('model_lab1.mdl');   % uruchomienie modelu o nazwie
                         % model_lab1.mdl z poziomu m-pliku
```

```
plot(data_out(:,1),data_out(:,2),'g');  % wyrysowanie poprawnego
% wyniku w kolorze zielonym
xlabel('Czas [s]');          % podpisanie osi X
ylabel('Amplituda sygnalu');% podpisanie osi Y
title('Wykres rozwiazania rownania rozniczkowego');%nadanie tytułu
grid;       % naniesienie siatki na wykres
c = toc     % wyświetlenie czasu działania programu i zapisanie
            % go w zmiennej c
```

Uwaga: W Simulinku również można uzyskać wykres wpinając w schemat blok „Scope" w miejsce „To Workspace", jednak takiego wykresu nie można w żaden sposób zapisać poza użyciem klawisza Print Screen i dalszej obróbce w zewnętrznym programie graficznym.

3. Zapisywanie i publikowanie wyników

a) Zapisywanie

W celu zapisania wszystkich macierzy znajdujących się w przestrzeni roboczej Matlaba należy użyć skrótu klawiszowego Ctrl+s w oknie przestrzeni roboczej, następnie należy wpisać nazwę pliku z danymi o rozszerzeniu *.mat. Można również kliknąć w ikonę 

Aby wczytać dane z dysku należy użyć w oknie komend (lub m-pliku) polecenia **`load`** lub przy użyciu ikony 

Składnia:

```
load MojeDane.mat
```

b) Publikowanie

Publikowanie wyników w Matlabie pozwala w prosty i przejrzysty sposób stworzyć gotowy dokument typu html, doc, pdf lub inne. Służy do tego komenda **`publish`**, najłatwiej ją wywołać klikając w m-pliku na File => Publish (opcje publikacji znajdują się w File => Publish Configuration for… => Edit Publish Configuration for…).

Podstawową zaletą publikowania jest to, że jednym kliknięciem dobrze napisany kod zmienia się w gotowy do prezentacji dokument jak np. raport lub sprawozdanie.

W przedstawionych przykładach zastosowano podstawową składnię wykorzystywaną do formatowania gotowych dokumentów.

`%%` - podwójny znak komentarza oznacza nową komórkę. Pierwsza komórka w m-pliku zawiera tytuł i opis dokumentu.

Przykład:

```
%% Sprawozdanie z Laboratorium nr 1
% Imię i Nazwisko, nr indeksu, grupa
%
% Celem ćwiczenia jest rozwiązanie równania różniczkowego trzema
metodami:
%
% # Metodą symboliczną - dsolve()
% # Metodą numeryczną - ode45()
% # Przy użyciu schematu blokowego w Simulinku
```

Ważne jest, aby kolejne linie zaczynały się od pojedynczego znaku komentarza, w ten sposób w końcowym dokumencie znajdzie się ciągły tekst.

Użyty znak # oznacza, że jest to element numerowany. Aby wypunktować obiekt należy użyć znaku *.

Każde kolejne użycie `%%` tworzy nową komórkę, dodając po spacji za tym znakiem tekst, tworzy się nowy rozdział w dokumencie. Jeżeli nie zostanie dodany żaden tekst, to tworzy się przerwę w kodzie programu. Dzięki czemu umieszczając ten znak np. po komendzie plot(), w gotowym dokumencie zostanie w tym miejscu wstawiony wykres.

W przykładzie 3 zastosowano składnię:

```
% <<model.png>>
```

Linia ta służy do wstawienia w miejsce tego kodu grafiki o nazwie model.png. Jednak plik ten powinien znajdować się w katalogu, w którym publikowany jest dokument.

Więcej informacji o publikowaniu można znaleźć na stronie:
http://www.mathworks.com/help/matlab/matlab_prog/marking-up-matlab-comments-for-publishing.html

4. Przebieg ćwiczenia:

Rozwiązać równania różniczkowe:

a) $x'' + 2x' + 2x = 2$ dla $x'(0) = 1$ $x(0) = 0.5$
b) $2x'' + 3x' + x = 6$ dla $x'(0) = -1$ $x(0) = 2$
c) $x'' + x' + 3x = 3$ dla $x'(0) = 2$ $x(0) = 1$
d) $x'' + 2x' + 5x = 2$ dla $x'(0) = 1$ $x(0) = 2$
e) $4x'' + 2x' + 4x = 4$ dla $x'(0) = 1$ $x(0) = 1$
f) $2x'' + 2x' + 8x = 2$ dla $x'(0) = -1$ $x(0) = 1$
g) $6x'' + 3x' + 6x = 3$ dla $x'(0) = 0$ $x(0) = 1$

wykorzystując wszystkie metody poznane na zajęciach. Wyniki należy porównać **na jednym wykresie**, pamiętając o ustawieniu identycznego kroku symulacji w każdej metodzie.

Dodatek

1. Nałożenie kilku przebiegów na jednym wykresie.

a) przy użyciu tylko komendy **`plot()`**

Składnia programu:

```
plot(t1,x1,'r',t2,x2,'g',t3,x3,'b');
```

b) przy użyciu komendy **`hold on`**

Składnia programu:

```
plot(t1,x1,'r');
hold on;
plot(t2,x2,'b');
plot(t3,x3,'g');
hold off;
```

komenda **`hold on`** nie pozwala nadpisać wykresu do momentu użycia komendy **`hold off`**. Dzięki temu wywołania kolejnych funkcji **`plot()`** może być oddzielone fragmentami dowolnego kodu.

2. Ograniczenie zakresu osi X oraz Y.

Ograniczenie takie jest przydatne, jeżeli chce się pokazać tylko istotny wycinek wykresu, służą do tego komendy **`xlim()`** oraz **`ylim()`**.

Składnia programu:

```
plot(t,x,'r');
xlim([x1 x2]); % x1 wartość początkowa, x2 końcowa, x1<x2
```

```
ylim([y1 y2]); % y1 wartość początkowa, y2 końcowa, y1<y2
```

3. Ustawianie pozycji i rozmiaru okna wykresu.

Do ustawienia pozycji i rozmiaru okna wykresu (figury) należy użyć następującej składni programu:

```
set(gcf,'position',[x1 y1 x2 y2]); % (x1,y1) – pozycja lewego
% dolnego roku okna, (x2,y2) – pozycja prawego górnego rogu okna
```

Jeżeli w programie jest używane tylko jedno okno do wyrysowania wszystkich wykresów, to daną linię kodu umieszcza się bezpośrednio pod pierwszym wywołaniem komendy plot(). Jeżeli w programie jest wiele okien, to pod każdym powinna znajdować się ta komenda.