

# Wprowadzenie do programowania obiektowego w Matlabie.

## Sterowanie temperaturą pomieszczenia.\*

Krzysztof Arent<sup>†</sup>

## 1 Wprowadzenie

Celem tego ćwiczenia jest nabycie podstawowych umiejętności w zakresie programowania obiektowego w Matlabie i następnie ich wykorzystanie w obszarze teorii sterowania. Ćwiczenie składa się z dwóch części. Najpierw należy zaimplementować bazę danych dla liniowych, stacjonarnych systemów dynamicznych z ćwiczenia 1-go, ale tym razem na bazie programowania obiektowego. Następnie, nabyte umiejętności należy zastosować do badań numerycznych nad zagadnieniem regulacji temperatury w pomieszczeniu [1]

## 2 Baza danych

### 2.1 Czynności wstępne

Rozpakuj archiwum `zadanie_2.zip`, załączone do tego ćwiczenia, gdzieś wewnątrz kartoteki `~/matlab`. Otrzymana podkartoteka `zadanie_2` zawiera sześć plików, które można podzielić na trzy grupy:

**klasy Matlaba** : `SystemDynamiczny.m`, `BazaDanych.m`;

**skrypty Matlaba** : `SystemDynamicznyDrajwer.m`, `BazaDanychDrajwer.m`;

**pozostałe pliki** : `Contents.m`, `baza_z.txt`.

Zauważ, że `zadanie_2` spełnia minimalne wymagania dla toolboku Matlaba.

### 2.2 Klasa `DynamicalSystem`

Przeanalizuj kod z listingu poniżej. Następnie odpowiedz na kilka pytań.

---

```
1 classdef SystemDynamiczny
2     % SYSTEMDYNAMICZNY jest klasa , ktora reprezentuje system dynamiczny
3     % liniowy, ciagly w czasie, stacjonarny, SISO,
4     % typu wejscie/stan/wyjscie
```

---

\*Ćwiczenie laboratoryjne do kursu Teoria sterowania (W12AIR-SM0007, W12AIR-SM0723).

© K.Arent, 2023. Wszelkie prawa zastrzeżone

<sup>†</sup>Katedra Cybernetyki i Robotyki, Wydział Elektroniki, Fotoniki i Mikrosystemów, Politechnika Wrocławska

```

5
6 properties
7     nazwa="bezNazwy";
8     A=[0];
9     B=[0];
10    C=[0];
11    D=[0];
12    u=@(t)0
13 end
14 properties (Dependent=true)
15     M
16 end
17
18 methods
19     function obj=SystemDynamiczny(val0, val1, val2, val3, val4, val5)
20         % SYSTEMDYNAMICZNY jest konstruktorem parametrycznym.
21         % Umożliwia on utworzenie obiektu tej klasy, którego
22         % pola mają przypisane następujące wartości:
23         % nazwa=val0, A=val1, B=val2, C=val3, D=val4, u=val5
24         if nargin==6
25             obj.nazwa=val0;
26             obj.A=val1;
27             obj.B=val2;
28             obj.C=val3;
29             obj.D=val4;
30             obj.u=val5;
31         end
32     end
33     function m=get.M(obj)
34         % Funkcja jest ściśle powiązana z polem zależnym M. Jej ciało
35         % zawiera odwzorowanie wejście/stan systemu dynamicznego
36         % reprezentowanego przez klasę. Zmienna m jest uchwyttem funkcji
37         % stowarzyszonej z odwzorowaniem.
38         m=@(t,x) obj.A*x+obj.B*obj.u(t);
39     end
40     function [x,y,u,t]=trajektoria(obj, tk, x0)
41         % TRAJEKTORIA zwraca wartości stanu, wyjścia i wejścia,
42         % przypisane zmiennym x,y,u odpowiednio, które odpowiadają
43         % chwilom czasu zapisanym pod zmienną t, z zakresu od 0 do tk,
44         % dla warunku początkowego x0.
45         [t,x]=ode45(obj.M,[0,tk],x0);
46         y=(obj.C*x'+obj.D*obj.u(t'))';
47         u=obj.u(t');
48     end
49     function dsplots(obj,tk,x0)
50         % DSPLOTS wykresla przebiegi stanu x, wyjścia y i wejścia u
51         % jako funkcji czasu na przedziale [0,tk]
52         % dla warunku początkowego x0
53         [x,y,u,t]=obj.trajektoria(tk, x0);
54         subplot(3,1,1); ...
55             plot(t,y), grid on, xlabel('t'), ylabel('y'), title(obj.nazwa);
56         subplot(3,1,2); ...

```

---

## Aspekty programowania

1. Czy powyższy kod implementuje klasę w Matlabie? Uzasadnij odpowiedź.
2. Czy ma miejsce dziedziczenie w powyższym kodzie?
3. Czy ma miejsce hermetyzacja w powyższym kodzie?
4. Czy ma miejsce przeciążenie w powyższym kodzie?

## Aspekty teorii sterowania

Rozważmy reprezentację liniowego, stacjonarnego systemu dynamicznego ze zmiennymi stanu,

$$\begin{aligned}\dot{x} &= Ax + Bu, \\ y &= Cx + Du.\end{aligned}\tag{1}$$

1. Wskaż linijki w kodzie powyżej, gdzie są przechowywane wartości parametrów (1).
2. Wskaż linijki w kodzie powyżej, gdzie jest zaimplementowane odwzorowanie wejście/stan/wyjście zdefiniowane przez (1).
3. Wskaż linijki w kodzie powyżej, gdzie są obliczane sygnały występujące w (1).

## 2.3 SystemDynamicznyDrajwer

- Przeanalizuj skrypt Matlaba, którego kod jest zamieszczony poniżej. Ten skrypt jest powiązany z klasą DynamicalSystem i został zaprojektowany do badania właściwości tej klasy.
- Napisz brakujący kod w poniższym skrypcie, implementujący *ZADANIE\_1* ÷ *ZADANIE\_6*.

---

```
1 %% SYSTEMDYNAMICZNYDRAJWER – skrypt do testowania klasy SystemDynamiczny
2
3 %% pomoc
4 help SystemDynamiczny
5 help SystemDynamiczny. trajektoria
6 help SystemDynamiczny. dsplots
7
8 %% czynnosci wstepne
9 clear all
10 close all
11
12 % zmienne pomocnicze
13 f=0; % licznik wykresow
14 tk=100; % chwila koncowa symulacji
15
16 % parametry liniowego systemu dynamicznego
17 nazwa="demo";
18 A=[0,-1;1 -2];
19 B=[3;1];
20 C=[0,1];
```

```

21 D=[0];
22
23 % uchwyt do wybranych funkcji wejsciowych
24 sigm=0.005;
25 impulse=@(t)exp(-t.^2/(2*sigm^2))/(sigm*sqrt(2*pi)); % gaussian function
26 omega=0.1;
27 rectangular=@(t)sign(sin(omega*t));
28 zero=@(t)0;
29 % ZADANIE 1: zdefiniuj uchwyt to funkcji skokowej
30
31
32
33 %% tworzenie obiektu klasy SystemDynamiczny z nastepujacymi
34 % parametrami: nazwa, A, B, C, D, zero
35 ds=SystemDynamiczny(nazwa,A,B,C,D,zero);
36
37 %% trajektoria systemu ds (przebiegi y, x, u w czasie) dla x0=[10; 0];
38 % na przedziale czasu [0,tk] na osobnym rysunku nr f.
39 x0=[10;0];
40 f=f+1; figure(f);
41 ds.dplots(tk,x0);
42
43 %% odpowiedz systemu ds na pobudzenie zerowe, na przedziale czasu [0,tk] dla
44 % x0=[0;0] (wykresy czasowe y,x,u) na osobnym rysunku nr f.
45 x0=[0;0];
46 f=f+1; figure(f);
47 ds.dplots(tk,x0);
48
49 %% odpowiedz systemu ds na pobudzenie impulsowe
50 % ZADANIE 2: wykresl odpowiedz systemu ds na pobudzenie impulsowe
51 % na przedziale czasu [0,tk] dla x0=[0;0] (przebiegi czasowe y,x,u)
52 % na oddzielnym rysunku nr f.
53
54
55
56 %% odpowiedz systemu ds na pobudzenie skokowe
57 % ZADANIE 3: wykresl odpowiedz systemu ds na pobudzenie skokowe
58 % na przedziale czasu [0,tk] dla x0=[0;0] (przebiegi czasowe y,x,u)
59 % na oddzielnym rysunku nr f.
60
61
62
63 %% odpowiedz systemu ds na pobudzenie sygnałem prostokatnym
64 % ZADANIE 4: wykresl odpowiedz systemu ds na pobudzenie sygnałem prostokatnym
65 % na przedziale czasu [0,tk] dla x0=[0;0] (przebiegi czasowe y,x,u)
66 % na oddzielnym rysunku nr f.
67
68
69
70 %% odpowiedz systemu ds na pobudzenie sygnałem prostokatnym
71 % ZADANIE 5: wykresl odpowiedz systemu ds na pobudzenie sygnałem prostokatnym
72 % na przedziale czasu [0,tk] dla x0=[0;0] (przebiegi czasowe y,x,u)
73 % na oddzielnym rysunku nr f. Tym razem parametr omega w definicji

```

```

74 % uchwytu dla sygnału prostokątnego ma być większy niż w ZADANIU 4
75
76
77
78 %% trajektoria systemu ds w przestrzeni stanu
79 % ZADANIE 6: wykreśl w przestrzeni stanu trajektorie stanu otrzymana
80 % w efekcie odpowiedzi systemu na pobudzenie sygnałem prostokątnym
81 % na przedziale czasu  $[0, t_k]$  dla  $x_0=[0;0]$  na oddzielnym rysunku f.

```

---

## 2.4 Klasa BazaDanych

1. Przeanalizuj kod w listingu poniżej. Czy ma w nim miejsce dziedziczenie? **Opcjonalnie:** przy okazji uzasadniania odpowiedzi na to pytanie zapoznaj się z klasą *handle* w dokumentacji Matlab'a i z odpowiedzią na post *Changing variable (properties in class) by method* w MatlabCentral, zob. <https://www.mathworks.com/matlabcentral/answers/431064-changing-variable-properties-in-class-by-method>
2. Napisz brakujący kod w m-pliku implementującym klasę *BazaDanych*.
3. Użyj matlabowego skryptu *BazaDanychDrajwer.m* do weryfikacji poprawności uzyskanego kodu *BazaDanych.m*.

---

```

1  classdef BazaDanych < handle
2  % BAZADANYCH – klasa reprezentuje baze danych dla systemów dynamicznych
3  % ze zmiennymi stanu, które są liniowe, stacjonarne, SISO
4
5      properties
6          DaneNazwaPliku="noFile.txt";
7          Dane=SystemDynamiczny;
8          count=0;
9      end
10
11     methods
12         function obj = BazaDanych(bdNazwaPliku)
13             % BAZADANYCH jest konstruktorem parametrycznym dla tej klasy
14             % bdNazwaPliku – nazwa pliku z bazy systemów dynamicznych
15             if nargin==1
16                 obj.DaneNazwaPliku = bdNazwaPliku;
17                 [fid, message]=fopen(obj.DaneNazwaPliku, 'r');
18                 if fid==1
19                     disp(message);
20                 else
21                     obj.count=0;
22                     while feof(fid) == 0
23                         nazwa=fgetl(fid);
24                         A=eval(fgetl(fid));
25                         B=eval(fgetl(fid));
26                         C=eval(fgetl(fid));
27                         D=eval(fgetl(fid));
28                         system=SystemDynamiczny(nazwa,A,B,C,D,...
29                             @(t)exp(-t.^2/(2*0.1^2))/(0.1*sqrt(2*pi)));
30                         obj.count=obj.count+1;

```

```

31         obj.Dane(obj.count)=system;
32     end
33     st=fclose(fid);
34 end
35 end
36 end
37
38 function wyszukaj-BD(obj,string)
39 % WYSZUKAJBD(string) – metoda, która wyświetla na ekranie
40 % nazwy wszystkich systemów w bazie danych, zawierających
41 % w sobie ciąg znaków string.
42
43 end
44
45 function sortuj_N(obj,macierz)
46 % SORTUJN('matrix') – metoda, która wyświetla na ekranie
47 % nazwy systemów z bazy danych według rosnących wartości
48 % ||macierz||_2, gdzie macierz przyjmuje wartości ze zbioru
49 % {'A', 'B', 'C', 'D'}.
50     if obj.count > 0
51         for ind=1:obj.count
52             normM(ind)=norm(eval(['obj.Dane(ind).',macierz]),2);
53         end
54         while any(normM >= 0)
55             aux=find(normM == max(normM));
56             fprintf('%s\t%f\n', obj.Dane(aux(1)).nazwa, normM(aux(1)) );
57             normM(aux(1))=-1;
58         end
59     else
60         fprintf('baza_danych_jest_pusta\n');
61     end
62 end
63
64 function s_stabilne(obj)
65 % S-STABILNE – metoda, która wyświetla na ekranie nazwy
66 % systemów z bazy danych, które są asymptotycznie stabilne.
67
68 end
69
70 function wykresy(obj, nazwa, tk)
71 % WYKRESY – metoda, która wyświetla rysunek z dwoma
72 % wykresami, jeden nad drugim. Pierwszy wykres przedstawia
73 % wyjście systemu nazwa na przedziale czasu [0,tk] przy założeniu
74 % zerowych warunków początkowych i pobudzeniu impulsowym.
75 % Drugi, dolny wykres, przedstawia trajektorie w przestrzeni stanu,
76 % o ile liczba zmiennych stanu jest równa 2 lub 3. Jeżeli ten
77 % warunek nie jest spełniony to w miejscu dolnego wykresu powinien
78 % się pojawić komunikat o treści: dim x > 0 lub dim x < 2
79
80     if obj.count > 0
81         indf=0;
82         for ind=1:obj.count
83             if ~isempty(strmatch(nazwa, obj.Dane(ind).nazwa))

```

```

84         indf=indf+1; figure(indf);
85         n=size(obj.Dane(ind).A,1); x0=zeros(n,1);
86         [x,y,u,t]=obj.Dane(ind).trajektoria(tk, x0);
87         subplot(211); ...
88             plot(t,y), grid on, ...
89             xlabel('t'), ylabel('y'), ...
90             title(obj.Dane(ind).nazwa);
91         m=size(obj.Dane(ind).A,1);
92         if m==3
93             subplot(212); ...
94             plot3(x(:,1)',x(:,1)', x(:,1)'), ...
95             grid on, ...
96             xlabel('x_1'),ylabel('x_2'),zlabel('x_3');
97         else if m==2
98             subplot(212); ...
99             plot(x(:,1)',x(:,2)'), grid on, ...
100             xlabel('x_1'),ylabel('x_2');
101         else if m==1
102             subplot(212); ...
103                 text(0.5,0.5,'dim_x<2');
104             else
105                 subplot(212); ...
106                 text(0.5,0.5,'dim_x>3');
107             end
108         end
109     end
110 end
111 end
112 end
113 end
114
115 function zapisz_bd(obj, nazwa_pliku)
116     % ZAPISZ-BD(nazwa_pliku) – metoda, która zapiszuje zawartosc
117     % bazy danych na pliku typu ASCII
118
119 end
120
121 function zawartosc_bd(obj)
122     % ZAWARTOSC-BD – metoda, która wyswietla na ekranie
123     % nazwy wszystkich systemow dynamicznych przechowywanych
124     % w bazie danych
125
126     %more on;
127     for ind=1:obj.count
128         fprintf(' %s\n', obj.Dane(ind).nazwa);
129     end
130     %more off;
131 end
132
133 function wczytaj_k(obj)
134     % WCZYTAJK – metoda, która umożliwia dolaczenie nowego systemu
135     % dynamicznego do bazy danych przy uzyciu klawiatury
136

```

```

137     end
138
139     function wczytaj_P(obj,nazwaPliku)
140         % WCZYTAJP(nazwaPliku) – metoda, która z pliku nazwaPliku,
141         % typu ASCII, wczytuje parametry systemów dynamicznych i przypisuje
142         % je odpowiednim polom zmiennej obj.Dane.
143
144         obj.DaneNazwaPliku=nazwaPliku;
145         [fid,message]=fopen(obj.DaneNazwaPliku,'r');
146         if fid==-1
147             disp(message);
148         else
149             while feof(fid) == 0
150                 obj.count=obj.count+1;
151                 obj.Dane(obj.count).nazwa=fgetl(fid);
152                 obj.Dane(obj.count).A=eval(fgetl(fid));
153                 obj.Dane(obj.count).B=eval(fgetl(fid));
154                 obj.Dane(obj.count).C=eval(fgetl(fid));
155                 obj.Dane(obj.count).D=eval(fgetl(fid));
156                 obj.Dane(obj.count).u=@(t)exp(-t.^2/(2*0.1^2))/(0.1*sqrt(2*pi));
157             end
158             st=fclose(fid);
159         end
160     end
161 end
162 end

```

---

### 3 Regulacja temperatury pomieszczenia

Wykonaj ćwiczenie symulacyjne pt. Regulacja temperatury pomieszczenia (ang. *A Temperature Control of a Container* z [1], rozdział A.2). W tym celu napisz skrypt w Matlabie, `temperatura.m`, podobny do `DynamicalSystemDriver.m`, który wykorzystuje klasę `DynamicalSystem`. Kod skryptu `temperatura.m` powinien być implementacją rozwiązania zadania symulacyjnego.

Poniżej zamieszczono szereg wskazówek ułatwiających wykonanie zadania.

#### Wskazówki

**część 1** Przekształć układ równań różniczkowych na stronie 387 in [1] (równania (5) w polskim tłumaczeniu) do następującej postaci:

$$\underbrace{\begin{bmatrix} \frac{dT_p}{dt} \\ \frac{dT_{sz}}{dt} \\ \frac{dT_{Hg}}{dt} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} ??? & ??? & ??? \\ ??? & ??? & ??? \\ ??? & ??? & ??? \end{bmatrix}}_A \underbrace{\begin{bmatrix} T_p \\ T_{sz} \\ T_{Hg} \end{bmatrix}}_x + \underbrace{\begin{bmatrix} ??? & ??? \\ ??? & ??? \\ ??? & ??? \end{bmatrix}}_B \begin{bmatrix} u \\ T_o \end{bmatrix}, \quad (2)$$

$$\underbrace{h}_{\dot{y}} = \underbrace{\begin{bmatrix} ??? & ??? & ??? \end{bmatrix}}_C \underbrace{\begin{bmatrix} T_p \\ T_{sz} \\ T_{Hg} \end{bmatrix}}_x. \quad (3)$$



Zauważ, że (2) można lekko zmodyfikować:

$$\dot{x} = Ax + B_1 u + B_2 T_o, \quad (4)$$

gdzie  $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$ .

**część 2** Ponieważ  $u \equiv 0$  dlatego użyteczna forma równań (2), (3) dla tej części ćwiczenia jest następująca:

$$\begin{aligned} \dot{x} &= Ax + B_2 T_o, \\ h &= Cx. \end{aligned} \quad (5)$$

**część 3** Ponieważ  $T_o \equiv 0$  i  $u = K_p h$  to równania (2), (3) mogą być zapisane następująco:

$$\begin{aligned} \dot{x} &= Ax + B_1 u, \\ h &= Cx, \\ u &= K_p h. \end{aligned} \quad (6)$$

Wówczas pierwsze dwa równania w (6) mogą być przekształcone do postaci:

$$\begin{aligned} \dot{x} &= \bar{A}x, \quad x(0) = x_0, \\ h &= Cx. \end{aligned} \quad (7)$$

gdzie  $\bar{A} = A + B_1 K_p C$ . Postać (7) jest przydatna dla obecnej części zadania.

**część 4** W tej części zadania przydatne są następujące równania:

$$\begin{aligned} \dot{x} &= \bar{A}x + B_2 T_o, \\ h &= Cx. \end{aligned} \quad (8)$$

Warto mieć na uwadze, że  $T_p = [1 \ 0 \ 0]x$ .

**część 5** W tym przypadku równania układu sprzężenia zwrotnego są następujące:

$$\begin{aligned} \dot{x} &= Ax + B_1 u + B_2 T_o \\ h &= Cx \\ \dot{z} &= h \\ u &= K_p h + K_I z \end{aligned} \quad (9)$$

Powyższe równania można zapisać w zwartej formie:

$$\begin{aligned} \underbrace{\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix}}_{\dot{\tilde{x}}} &= \underbrace{\begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}}_{\bar{A}} \underbrace{\begin{bmatrix} x \\ z \end{bmatrix}}_{\tilde{x}} + \underbrace{\begin{bmatrix} B_1 & B_2 \\ 0 & 0 \end{bmatrix}}_{\bar{B}} \begin{bmatrix} u \\ T_o \end{bmatrix} \\ h &= \underbrace{\begin{bmatrix} C & 0 \end{bmatrix}}_{\bar{C}} \begin{bmatrix} x \\ z \end{bmatrix} \\ u &= \underbrace{\begin{bmatrix} K_P C & K_I \end{bmatrix}}_{\bar{K}} \begin{bmatrix} x \\ z \end{bmatrix} \end{aligned} \quad (10)$$

Niech  $\tilde{B} = [\tilde{B}_1 \ \tilde{B}_2]$ . Wówczas przydatna postać równań do badań symulacyjnych w tej części zadania jest następująca:

$$\begin{aligned}\dot{\tilde{x}} &= \hat{A}\tilde{x} + \tilde{B}_2 T_o \\ h &= \tilde{C}\tilde{x}\end{aligned}\tag{11}$$

gdzie  $\hat{A} = \tilde{A} + \tilde{B}_1 \tilde{K}$ .

## 4 Sprawozdanie

Zawartość kartoteki `zadanie_2` z m-plikiem `temperatura.m` powinna być skompresowana i zapisana na pliku `.zip`, załączonym do sprawozdania. Sprawozdanie, w formacie `.pdf`, powinno zawierać zapisy sesji ze skryptami

- `SystemDynamicznyDrajwer.m`,
- `BazaDanychDrajwer.m`,
- `temperatura.m`,

otrzymanymi przy pomocy funkcji `diary` i `print`.

## Literatura

- [1] J. W. Polderman and J. C. Willems. *Introduction to Mathematical Systems Theory. A Behavioral Approach*. Springer New York, NY, 1998.