

MR(A)C - komentár k bonusovej úlohe

Obsah

1	Časť prvá	2
1.1	Úloha prvá, bod prvý	2
1.1.1	Bod druhý	4
1.1.2	Bod tretí	5
1.2	Úloha druhá	5
1.2.1	Bod prvý	5
1.2.2	Bod druhý	6
1.2.3	Bod tretí	7

1 Časť prvá

V zadaní sa hovorí:

Uvažujme riadený systém, ktorý pracuje v pásme danom dvomi pracovnými bodmi. V týchto dvoch hraničných pracovných bodoch prenosová funkcia systému nie je rovnaká, vyskytujú sa mierne rozdiely v hodnotách koeficientov jednotlivých polynómov, pričom stupne polynómov sú zhodné, konkrétne:

$$G_{OP_1} = 0,1659 \frac{s + 22}{s^2 + 3,1423s + 2,6539} \quad (1)$$

$$G_{OP_2} = 0,1669 \frac{s + 20,7618}{s^2 + 2,3422s + 2,7293} \quad (2)$$

Mimochodom, ide o prenosové funkcie zodpovedajúce dynamiky „laboratórnych procesov - motorčekov“, ktoré sú v laboratóriu D330. Sú identifikované pre okolie rôznych pracovných bodov, teda raz sú otáčky motora nízke, raz vysoké - nemá význam tu hovoriť o fyzikálnych veličinách, pretože je to merané (a ovládané) len ako napäťové rozsahy o až 10 V (azda si niektorý čitateľ spomína). Možno nemá význam o tom vôbec hovoriť.

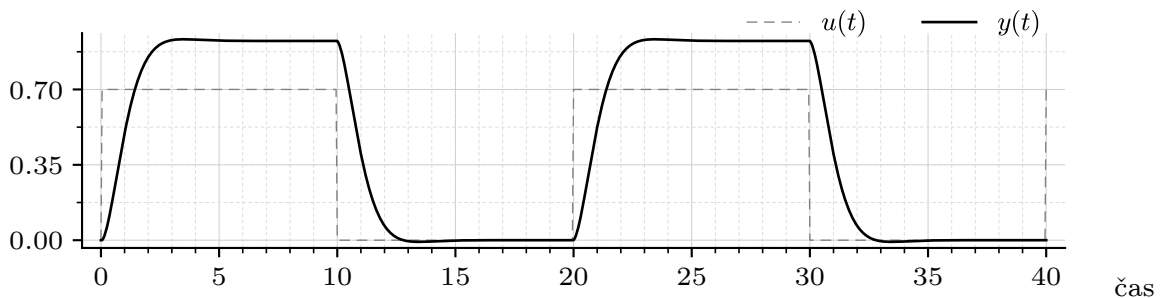
1.1 Úloha prvá, bod prvý

Určte *nominálnu* prenosovú funkciu sústavy tak, že jej koeficienty sú priemery hodnôt oboch prenosových funkcií (1) a (2).

$$G_n(s) = 0,1664 \frac{s + 21,3809}{s^2 + 2,7423s + 2,6916} \quad (3)$$

Uvedené je prenosová funkcia systému 2. rádu (dva póly, jedna nula).

Pre azda ešte lepšiu predstavu o riadenom systéme, nakreslime priebeh výstupnej veličiny pre isté skokové priebehy vstupnej veličiny - viď obr. 1.



Obr. 1: Priebeh výstupnej veličiny systému (3) pre isté skokové priebehy vstupnej veličiny. Naozaj len mimochodom: čas je reálne v sekundách, a jednotky zobrazených veličín sú volty (via meracia karta).

Simulované priebehy vznikli s využitím nasledujúceho kódu (Python) - viď výpis kódu 1:

```
1 import numpy as np
2
3 from scipy.integrate import odeint
4
5 # odeint je ODE solver a tu nim budeme riesit
6 # Linear Time Invariant System, a jeho zapis ako (maticova)
7 # sustava diferencialnych rovníc je - vid nasled. funkciu,
8 # pricom metoda "matmul" je samozrejme maticove nasobenie
9
10 def fcn_LTIS(x, t, A, b, u):
11
12     dotx = np.matmul(A, x) + np.matmul(b, u)
13
```

```

14     return dotx
15
16     # Vytvorme teraz funkciu, ktora bude realizovat simulacnu schemu.
17     # Argumentami funkcie su parametre suvisiace s casom
18     # a vopred dane (zname) signaly.
19
20     def fcn_simSch1(t_start, T_s, finalIndex, sig_dummy_ext):
21
22         # -----
23         # Parametre riadeného systému
24
25         A = np.array([[0, 1], [-2.6916, -2.7423]])
26         b = np.array([[0], [1]])
27         c = np.array([[3.5578], [0.1664]])
28
29         # -----
30         # Do pola t_log sa bude logovat cas. Pole ma finalIndex
31         # riadkov a 1 stlpec a je plne nul. Potom sa na prvu
32         # poziciu (index 0) zapise hodnota t_start
33
34         t_log = np.zeros([finalIndex, 1])
35         t_log[0,:] = t_start
36
37         # -----
38         # Zaciatočne podmienky pre stavový vektor nech su x_0
39         # co je vektor rovnako veľky ako vektor b
40
41         x_0 = np.zeros(b.shape[0])
42
43         # Stavový vektor sa bude logovat do pola x_log s prislusnym
44         # pocetom stlpcov (detto y_log pre vyst. velicinu)
45
46         x_log = np.zeros([finalIndex, len(x_0)])
47         x_log[0,:] = x_0
48
49         y_log = np.zeros([finalIndex, 1])
50         y_log[0,:] = np.dot(c.T, x_0.reshape(-1,1))
51
52         # -----
53
54         u_log = np.zeros([finalIndex, 1])
55         u_log[0,:] = 0
56
57         # -----
58         # Jedna iteracia for cyklu je posun v case o T_s.
59         # ODE solver hľada riesenie pre casovy rozsah timespan.
60         # Pred danou iteraciou pozname vsetko z predchadzajúcej
61         # iteracie (idx-1)
62         # Pocas iteracie si _log-ujeme "vysledky"
63
64         timespan = np.zeros(2)
65         for idx in range(1, int(finalIndex)):
66
67             timespan[0] = t_log[idx-1,:]
68             timespan[1] = t_log[idx-1,:] + T_s
69
70             t_log[idx,:] = timespan[-1]
71             # posledny prvok v poly je zapisany (logovany)
72
73             # -----
74             # solver odeint pouzije fcn_LTIS, zaciatočne podmienky
75             # stavu su z predch. iteracie (x_log[idx-1,:]), riesi
76             # na casovom rozsahu timespan a dalej (do fcn_LTIS) sa
77             # posunu uvedene parametre/hodnoty (args)
78
79             odeOut = odeint(fcn_LTIS,
80                             x_log[idx-1,:],
81                             timespan,
82                             args=(A, b, u_log[idx-1,:])
83                             )
84
85             x_log[idx,:] = odeOut[-1,:]
86             # odeOut obsahuje hodnoty stavu x pre cely timespan,
87             # ale zapisujeme len poslednu hodnotu stavu x
88
89             y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))
90             # okrem stavu (stavových velicin) chceme aj

```

```

91         # vystupnu velicinu y
92
93         # -----
94
95         u_log[idx,:] = sig_dummy_ext[idx,:]
96         # v tejto simulacii len citame "externy" signal
97         # a pouzivame ho ako vstup do systemu
98
99
100     return [t_log, x_log, y_log, u_log, ]
101
102
103
104     # Vytvorme teraz vsetko potrebne pre "spustenie" simulacie,
105     # teda pre zavolanie prave vytvorenej funkcie fcn_simSch1.
106     # Hovorme tomu "nastavenie simulacie". Casove nastavenie:
107
108     sim_t_start = 0
109     sim_t_final = 40
110     sim_T_s = 0.05
111     sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
112
113
114     # Dalej je potrebne vytvorit (vopred znamy) signal.
115     # Co sa tu deje ponechajme bez komentara, ale vysledkom
116     # je proste "signal" pouzitelny v simulacii...
117
118     period_time = 20
119     period_tab = np.array([[0, 0.7],
120                            [10, 0],
121                            ])
122
123     sig_vysl = np.zeros([sim_finalIndex, 1])
124
125     for period in range(int(sim_t_final/period_time) + 1):
126         for idx in range( int((period*period_time)/sim_T_s), int((
127             period*period_time + period_time)/sim_T_s)):
128             lastValue = period_tab[:,1][(period_tab[:,0] + (period*
129                 period_time))<=idx*sim_T_s ][-1]
130             try:
131                 sig_vysl[idx] = lastValue
132             except:
133                 break
134
135     sig_dummy_ext = sig_vysl
136
137     # Teraz mozme "spustit" simulaciu:
138
139     t_log, x_log, y_log, u_log, = fcn_simSch1(
140         sim_t_start,
141         sim_T_s,
142         sim_finalIndex,
143         sig_dummy_ext,
144     )
145
146     # Tu by mohlo byt kreslenie obrazku, ale to tu neuvedieme.
147     # Jednoducho nieco v zmysle plot(t_log, y_log) a podobne...

```

Výpis kódu 1: Základná simulačná schéma (áno, blbo sa to odtiaľ kopíruje, nie, nie je to zámer)

1.1.1 Bod druhý

Pre nominálnu prenosovú funkciu sústavy určte polynómy Z_p , R_p a zosilnenie k_p pričom

$$\frac{y(s)}{u(s)} = k_p \frac{Z_p(s)}{R_p(s)} \quad (4)$$

kde $Z_p(s)$ je monický polynóm stupňa m , $R_p(s)$ je monický polynóm stupňa n a k_p je tzv. *vysokofrekvenčné zosilnenie sústavy*. Relatívny stupeň sústavy je $n^* = n - m$.

$$Z_p(s) = s + 21,3809 \quad (5a)$$

$$R_p(s) = s^2 + 2,7423s + 2,6916 \quad (5b)$$

$$k_p = 0,1664 \quad (5c)$$

$Z_p(s)$ je monický polynóm, pretože pri najvyššej mocnine „premennej“ (operátor s) je koeficient 1. Rovnako polynóm $R_p(s)$ je monický. Relatívny stupeň prenosovej funkcie je $n^* = 2 - 1 = 1$

1.1.2 Bod tretí

Zistite, či polynóm $Z_p(s)$ je Hurwitzov.

Je Hurwitzov polynóm totiž znamená, že „polynóm je stabilný“, a tým sa myslí, že korene polynómu sú v ľavej polrovine komplexnej roviny. Koreň polynómu $Z_p(s) = s + 21,3809$ je $s = -21,3809$ čo je na reálnej osi v záporných číslach, a teda v ľavej polrovine komplexnej roviny.

1.2 Úloha druhá

Vyriešte MRC problém pre nominálnu prenosovú funkciu sústavy, uvažujte referenčný model daný prenosovou funkciou v tvare

$$W_m(s) = \frac{s + 3}{s^2 + 3.5s + 3} \quad (6)$$

Referenčný model je daný prenosovou funkciou v tvare

$$\frac{y_m(s)}{r(s)} = W_m(s) = k_m \frac{Z_m(s)}{R_m(s)} \quad (7)$$

kde k_m je vysokofrekvenčné zosilnenie referenčného modelu, $Z_m(s)$ monický Hurwitzov polynóm stupňa m_m , $R_m(s)$ monický Hurwitzov polynóm stupňa n_m , pričom relatívny stupeň $n_m^* = n_m - m_m = n^*$.

Riešením MRC problému je taký zákon riadenia u , ktorý zabezpečí, že výstup sústavy y sleduje výstup referenčného modelu y_m pri danom referenčnom signály (vstupe referenčného modelu) r . Všeobecný tvar zákona riadenia, ktorý rieši MRC problém je

$$u = \Theta_1^T \frac{\alpha(s)}{\Lambda(s)} u + \Theta_2^T \frac{\alpha(s)}{\Lambda(s)} y + \Theta_3^* y + \Theta_4^* r \quad (8)$$

kde $\alpha(s)$ je vektor obsahujúci mocniny s , $\alpha(s) = [s^{n-2}, \dots, s, 1]^T$ ak $n \geq 2$, inak $\alpha(s) = 0$. Vektory $\Theta_1^* \in \mathbb{R}^{n-1}$, $\Theta_2^* \in \mathbb{R}^{n-1}$ a skaláry $\Theta_3^* \in \mathbb{R}^1$, $\Theta_4^* \in \mathbb{R}^1$ sú konštantné parametre zákona riadenia, ktorých hodnoty hľadáme. $\Lambda(s)$ je ľubovoľný monický Hurwitzov polynóm stupňa $n - 1$ obsahujúci $Z_m(s)$ ako faktor

$$\Lambda(s) = \Lambda_0(s) Z_m(s) \quad (9)$$

a teda aj $\Lambda_0(s)$ je ľubovoľný monický Hurwitzov polynóm zodpovedajúceho stupňa.

Všimnime si, že dosť podstatnou vlastnosťou referenčného modelu je, že má rovnaký relatívny stupeň ako riadený systém, teda $n_m^* = n_m - m_m = n^*$. Referenčný model nemusí mať rovnaký rád ako riadený systém. Musí však mať rovnaký relatívny stupeň. Plyní to z požiadaviek (predpokladov) pri riešení MRC problému (úlohy riadenia s referenčným modelom) vo všeobecnosti.

1.2.1 Bod prvý

Na základe všeobecného tvaru zákona riadenia (8) určte zákon riadenia pre uvažovaný konkrétny prípad.

$\alpha(s)$ je vektor, ktorého dĺžka závisí od rádu riadeného systému (zjavne má dĺžku $n - 1$). V tomto prípade $n = 2$, čo spĺňa $n \geq 2$. Preto $\alpha(s) = [s^{2-2}]^T = [s^0]^T = [1]^T = 1$.

A teda $\alpha(s)$ bude v tomto prípade jednoducho číslo 1. Je to známa vec, nie je to parametrom zákona riadenia.

Vektory $\Theta_1^* \in \mathbb{R}^{n-1}$, $\Theta_2^* \in \mathbb{R}^{n-1}$ budú mať v tomto prípade tiež len jeden prvok (sú dĺžky $n-1$) a teda len rovno píšme čísla Θ_1^* a Θ_2^* . K tomu Θ_3^* , Θ_4^* sú vždy len skaláre. V tomto prípade sú tieto štyri čísla parametrami zákona riadenia. Tieto parametre sú predmetom výpočtu/hľadania, ak chceme použiť tento konkrétny zákon riadenia.

Polynóm $\Lambda(s)$ nie je „neznámou“. Nie je to parameter zákona riadenia v tom pravom zmysle. Je ľubovoľný ak sú dodržané uvedené podmienky/predpoklady. Pri jeho voľbe je užitočné uvažovať o tom, že tento polynóm možno interpretovať vzhľadom na *pozorovateľ stavu* a jeho dynamické vlastnosti. Súvislosť pozorovateľa stavu s tu používaným zákonom riadenia bola uvedená v učebnom texte. Akokoľvek, ak $\Lambda(s)$ spĺňa dané podmienky je to ok. V tomto prípade (vlastne nie je veľa ľubovôle):

$$\Lambda(s) = Z_m(s) = s + \lambda \quad (10)$$

kde $\lambda = 3$, pretože $Z_m(s) = s + 3$.

Práve sme určili zákon riadenia pre uvažovaný konkrétny prípad:

$$u(s) = \Theta_1^* \frac{1}{(s + \lambda)} u(s) + \Theta_2^* \frac{1}{(s + \lambda)} y(s) + \Theta_3^* y(s) + \Theta_4^* r(s) \quad (11)$$

1.2.2 Bod druhý

Vypočítajte parametre Θ_1^* , Θ_2^* , Θ_3^* , Θ_4^* .

Pre prehľadnosť označme

$$\frac{y(s)}{u(s)} = k_p \frac{s + b_0}{s^2 + a_1 s + a_0} \quad (12)$$

a

$$W_m(s) = k_m \frac{s + b_{0m}}{s^2 + a_{1m} s + a_{0m}} \quad (13)$$

Potom je možné ukázať (a v učebnom texte je to ukázané), že uzavretý regulačný obvod sa bude zhodovať s referenčným modelom ak bude platiť:

$$\begin{bmatrix} -1 & 0 & -k_p \\ -a_1 & -k_p & -(k_p b_{0m} + k_p b_0) \\ -a_0 & -k_p b_0 & -k_p b_0 b_{0m} \end{bmatrix} \begin{bmatrix} \Theta_1^* \\ \Theta_2^* \\ \Theta_3^* \end{bmatrix} = \begin{bmatrix} a_{1m} + b_0 - b_{0m} - a_1 \\ a_{0m} + b_0 a_{1m} - a_1 b_{0m} - a_0 \\ b_0 a_{0m} - a_0 b_{0m} \end{bmatrix} \quad (14a)$$

$$\Theta_4^* = \frac{k_m}{k_p} \quad (14b)$$

Vypočítajte:

```

1  kp = 0.1664
2  b0 = 21.3809
3  a1 = 2.7423
4  a0 = 2.6916
5
6  km = 1
7  b0m = 3
8  a1m = 3.5
9  a0m = 3
10
11 M = np.array([[ -1,  0, -kp],
12               [ -a1, -kp, -(kp*b0m + kp*b0)],
13               [ -a0, -kp*b0, -kp*b0*b0m],
14               ])
15
16 N = np.array([[a1m + b0 - b0m - a1],
17               [a0m + b0*a1m - a1*b0m - a0],
18               [b0*a0m - a0*b0m],
19               ])
20
21 Theta_c = np.linalg.solve(M,N)
22 # Theta_c je vektor obsahujuci Theta_1, Theta_2 a Theta_3
23
24 Theta_4 = km/kp

```

Výpis kódu 2: Výpočet ideálnych parametrov zákona riadenia

Teda:

$$\begin{bmatrix} \Theta_1^* \\ \Theta_2^* \\ \Theta_3^* \end{bmatrix} = \begin{bmatrix} -18,3809 \\ 11,8071 \\ -4,5535 \end{bmatrix} \quad (15a)$$

$$\Theta_4^* = 6,0096 \quad (15b)$$

1.2.3 Bod tretí

Zostavte simulačnú schému uzavretého regulačného obvodu a overte vypočítané parametre zákona riadenia.

Ako realizovať numerickú simuláciu riadeného systému sme (nepriamo) ukázali v predchádzajúcom. Riadený systém je daný ako prenosová funkcia. Tú je možné previesť na opis systému v stavovom priestore a potom je možné použiť ODE solver (tak ako bolo ukázané).

O prevode prenosovej funkcie na opis v stavovom priestore

Mimochodom, prevod z prenosovej funkcie na stavový opis nie je jednoznačný. Záleží na voľbe stavových veličín (stavového priestoru). Tu si dovoľme uviesť voľbu stavových veličín tak, že výsledkom je opis systému v tzv. normálnej forme riaditeľnosti.

Prenosovú funkciu riadeného systému, ktorou sa tu zaoberáme, je možné, vo všeobecnosti, napísať v tvare

$$\frac{y(s)}{u(s)} = \frac{b_1 s + b_0}{s^2 + a_1 s + a_0} \quad (16)$$

pričom tu „nesedí“ označovanie a b_0 nie je to isté b_0 ako pred tým. Tu nám ide o tvar vo všeobecnosti, a ten ostal zachovaný... (snáď je to pre čitateľa dostatočne jasné)

Otázka je ako túto prenosovú funkciu previesť na opis v stavovom priestore - ako zvoliť stavové veličiny. Pre prípad, keď je v čitateli len konštanta (systém nemá nuly), je voľba stavových veličín značne intuitívna. Preto napíšme prenosovú funkciu (16) ako dve prenosové funkcie v sérii nasledovne

$$\frac{z(s)}{u(s)} = \frac{1}{s^2 + a_1 s + a_0} \quad (17)$$

$$\frac{y(s)}{z(s)} = b_1 s + b_0 \quad (18)$$

kde sme zaviedli pomocnú veličinu z .

Prvú prenosovú funkciu (17) možno prepísať na diferenciálnu rovnicu druhého rádu v tvare

$$\ddot{z}(t) + a_1 \dot{z}(t) + a_0 z(t) = u(t) \quad (19)$$

Túto je možné previesť na sústavu diferenciálnych rovníc prvého rádu - voľbou stavových veličín. Napríklad nech

$$x_1(t) = z(t) \quad (20)$$

kde $x_1(t)$ je prvá stavová veličina. Potom platí

$$\dot{x}_1(t) = \dot{z}(t) \quad (21)$$

Druhú stavovú veličinu zvolíme

$$x_2(t) = \dot{z}(t) \quad (22)$$

a teda

$$\dot{x}_2(t) = \ddot{z}(t) \quad (23)$$

V tomto bode môžeme ľahko písať

$$\dot{x}_1(t) = x_2(t) \quad (24)$$

To je prvá diferenciálna rovnica! Obsahuje len novo zavedené stavové veličiny ($x_1(t)$ a $x_2(t)$). Druhá diferenciálna rovnica je vlastne (23). Avšak, vieme signál $\ddot{z}(t)$ vyjadriť len pomocou novo zavedených stavových veličín? Vieme. Z (19) je zrejmé, že

$$\ddot{z}(t) = -a_1\dot{z}(t) - a_0z(t) + u(t) = -a_1x_2(t) - a_0x_1(t) + u(t) \quad (25)$$

takže (23) je

$$\dot{x}_2(t) = -a_1x_2(t) - a_0x_1(t) + u(t) \quad (26)$$

a to je druhá diferenciálna rovnica...

Obe rovnice spolu:

$$\dot{x}_1(t) = x_2(t) \quad (27)$$

$$\dot{x}_2(t) = -a_1x_2(t) - a_0x_1(t) + u(t) \quad (28)$$

A v maticovom zápise:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \quad (29)$$

Vráťme sa k prenosovej funkcii (18). Túto možno napísať ako diferenciálnu rovnicu v tvare

$$y(t) = b_1\dot{z}(t) + b_0z(t) \quad (30)$$

Avšak, my sme už urobili voľbu takú, že $\dot{z}(t) = x_2(t)$ a $z(t) = x_1(t)$. Takže diferenciálnu rovnicu (30) môžeme písať ako

$$y(t) = b_1x_2(t) + b_0x_1(t) \quad (31)$$

alebo v maticovom tvare

$$y(t) = [b_0 \quad b_1] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad (32)$$

Celý systém s novo zavedenými stavovými veličinami teda je v tvare

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \quad (33)$$

$$y(t) = [b_0 \quad b_1] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad (34)$$

a ak označíme stavový vektor ako $x(t) = [x_1(t) \quad x_2(t)]^T$, potom je systém v známom tvare

$$\dot{x}(t) = Ax(t) + bu(t) \quad (35a)$$

$$y(t) = c^T x(t) \quad (35b)$$

kde matica A a vektory b a c sú zrejmé...

Sústava diferenciálnych rovníc (35) je vo vhodnom tvare pre potreby ODE solvera. V skripte (vo výpise kódu 1) je takáto sústava realizovaná (podľa požiadaviek ODE solvera) funkciou `fcn_LTIS()` uvedenej na riadku 10.

O referenčnom modeli

Prenosová funkcia referenčného modelu je v tvare (13). Ako túto prenosovú funkciu previesť do tvaru sústavy diferenciálnych rovníc by malo byť zrejmé z predchádzajúceho textu.

Vzhľadom na princíp fungovania tu zostavovanej simulačnej schémy (funkcia `fcn_simSch1` vo výpise kódu 1, riadok 20), je možné realizovať numerickú simuláciu aj jednoduchšie ako to typicky predpokladá ODE solver. Pomocou jednoduchej sumácie.

Potrebuje poznať „prírastok k stavovému vektoru“ v každej iterácii (`for` cyklu). Tento „prírastok“ je daný veľkosťou zmeny stavového vektora (v čase) a dĺžkou času, počas ktorého táto zmena platí.

Poznáme veľkosť zmeny stavového vektora? Áno, doslova: $\dot{x}(t) = Ax(t) + bu(t)$. V prípade referenčného modelu by sme však označili jednotlivé prvky špecifickejšie, teda $\dot{x}_m(t) = A_m x_m(t) + b_m r(t)$ (vstupom RM je samozrejme $r(t)$). Toto môžeme dokonca implementovať s pomocou už existujúcej funkcie `fcn_LTIS()`:

```
1 dotx_m = fcn_LTIS(x_m_log[idx-1,:], 0, A_m, b_m, ref_sig)
```

Výpis kódu 3: `dotx_m`

Ako dlho bude „trvať“ táto zmena? V simulačnej schéme `fcn_simSch1` to určuje parameter (argument) `T_s`.

„Prírastok k stavovému vektoru“ potom je `dotx_m * T_s`. A tento prírastok je potrebné pripočítať k predchádzajúcej („starej“) hodnote stavového vektora, teda obyčajná sumácia („numerický integrál“):

```
1 x_m_log[idx,:] = x_m_log[idx-1,:] + dotx_m * T_s
```

Výpis kódu 4: `x_m`

Tým sme získali hodnotu stavového vektora v aktuálnom kroku („novú“ hodnotu). My však v tomto prípade potrebujeme výstupnú veličinu (nie stavový vektor), teda

```
1 y_m_log[idx,:] = np.dot(c_m.T, x_m_log[idx,:].reshape(-1,1))
```

Výpis kódu 5: `y_m`

kde by mali byť jednotlivé premenné (a funkcie) viac-menej už čitateľovi jasné... Snáď len toľko, že `c_m.T` je transponované pole `c_m` a že `reshape(-1,1)` je metóda, ktorá zmení tvar poľa na toľko riadkov koľko treba (prvý argument `-1`) a práve jeden stĺpec (druhý argument `1`).

O zákone riadenia

Zákon riadenia má v tomto prípade tvar - viď (11). To je však zápis vo frekvenčnej oblasti s operátorom s .

Tu však potrebujeme zákon riadenia v časovej oblasti. Dovoľme si preto písať

$$u(t) = \Theta_1^* \left[\frac{1}{(s + \lambda)} \right] u(t) + \Theta_2^* \left[\frac{1}{(s + \lambda)} \right] y(t) + \Theta_3^* y(t) + \Theta_4^* r(t) \quad (36)$$

kde sme zmiešali písanie operátora s a času t .

Autor pozná takýto (alebo podobný) spôsob zápisu práve z literatúry o klasickom adaptívnom riadení¹. Tu nie je cieľom porušovať matematické vzťahy a podobne. Tu je cieľom zjednodušiť vyjadriť praktický zápis, akým je (36), vo vzťahu k pôvodnej teórii daného zákona riadenia (kde mimochodom je takýto zápis veľmi užitočný)

Autor sa však mnoho krát stretáva s nevôľou čitateľov/poslucháčov prijať uvedený spôsob zápisu, prípadne si to vyžaduje dodatočné vysvetľovanie.

Akokoľvek, čo vlastne predstavuje napríklad prvý člen na pravej strane rovnice (36)? Ešte lepšie, prvý člen na pravej strane rovnice (11)? Označme prvý člen na pravej strane rovnice (11) takto:

$$y_{v1}(s) = \Theta_1^* \frac{1}{(s + \lambda)} u(s) \quad (37)$$

Je to teda samostatný dynamický systém daný prenosovou funkciou. Vstupom je v tomto prípade signál $u(s)$. Tento signál je „filtrovaný“ vždy známym filtrom, ktorého

¹viď napr. knihu G. Tao., Adaptive control design and analysis. John Wiley & Sons, Inc., 2003.

vlastnosti sú dané polynómom $\Lambda(s)$. Vznikne „nový signál“ (prefiltrované u) a tento je potom vynásobený parametrom zákona riadenia, v tomto prípade Θ_1^* . Uvedený samostatný dynamický systém je možné vyjadriť aj opisom v stavovom priestore. Pre tento konkrétny príklad

$$\dot{\nu}_1(t) = -\lambda\nu_1(t) + u(t) \quad (38a)$$

$$y_{\nu_1}(t) = \Theta_1^*\nu_1(t) \quad (38b)$$

kde sme zaviedli pomocnú stavovú veličinu $\nu_1(t)$. Vieme aj reálne generovať (vyrobiť) tento pomocný signál $\nu_1(t)$? Je to jednoducho stavová veličina lineárneho dynamického systému, v princípe rovnakého ako je referenčný model, alebo model riadeného systému. Takže vieme reálne generovať pomocný signál $\nu_1(t)$. Prvý člen zákona riadenia (36) teda nahrádza výraz

$$\Theta_1^*\nu_1(t) \quad (39)$$

pričom $\nu_1(t)$ je signál, ktorý vieme vyrobiť...

Druhý člen zákona riadenia (36), analogicky, nahrádza výraz

$$\Theta_2^*\nu_2(t) \quad (40)$$

kde $\nu_2(t)$ je daný diferenciálnou rovnicou

$$\dot{\nu}_2(t) = -\lambda\nu_2(t) + y(t) \quad (41)$$

Zákon riadenia (36) je teda možné zapísať v tvare

$$u(t) = \Theta_1^*\nu_1(t) + \Theta_2^*\nu_2(t) + \Theta_3^*y(t) + \Theta_4^*r(t) \quad (42)$$

Čo sú parametre a čo sú signály v tomto zákone riadenia? Rozdelíme ich do vektorov. Vektora parametrov a vektora signálov

$$u(t) = \begin{bmatrix} \Theta_1^* & \Theta_2^* & \Theta_3^* & \Theta_4^* \end{bmatrix} \begin{bmatrix} \nu_1(t) \\ \nu_2(t) \\ y(t) \\ r(t) \end{bmatrix} \quad (43)$$

a označme

$$u(t) = \Theta^{*\top}\omega \quad (44)$$

Vieme vyrobiť všetky signály v signálnom vektore ω ? Vieme. Poznáme parametre vo vektore Θ^* ? Poznáme.

V tejto chvíli nič nebráni zostaveniu simulačnej schémy uzavretého regulačného obvodu.

O formálnej súvislosti s MRAC stavovým

Pre lepšiu konzistenciu uvedeného s učebným textom je vhodné zmeniť poradie prvkov v signálnom vektore ω . Učebný text totiž pracuje s myšlienkou čo najviac pripodobniť odvodenie „MRAC vstupno-výstupného“ k odvodeniu, ktoré definuje „MRAC stavový“. Dôvodom je, že ak máme k dispozícii stavový opis systému práve v normálnej forme riaditeľnosti, potom odvodenie (viac-menej akéhokolvek) zákona riadenia je najjednoduchšie možné po formálnej stránke. Najpriamočiarejšie. Bez nutnosti formulácie predpokladov navyše, formálnych konštrukcií a podobne.

Inými slovami, ak máme dostupný postup návrhu (odvodenie) nejakého riadiaceho systému, v tomto prípade Adaptívneho riadenia s referenčným modelom s využitím Lyapunovovej teórie stability, potom je veľmi výhodné snažiť sa tento postup uplatniť vo všeobecnosti.

Ak tento postup máme pri predpoklade, že sú dostupné práve tie stavové veličiny, ktoré vedú na opis riadeného systému práve v normálnej forme riaditeľnosti, potom by bolo výhodné aby sme „novú úlohu“ formálne previedli do tvaru, ktorý zodpovedá známemu postupu.

To sa deje pri odvodení „MRAC vstupno-výstupného“ práve vtedy, keď sa zavedie pojem *doplnená sústava*, alebo *doplnený riadený systém*. To má čitateľ možnosť vidieť v učebnom texte. „Doplnkom“ sú práve signály, ktoré sme tu označili ako $\nu_1(t)$ a $\nu_2(t)$.

Výsledkom je, že zákon riadenia sa uvažuje v tvare

$$u(t) = \Theta_c^* D X(t) + \Theta_4^* r(t) \quad (45)$$

kde $\Theta_c^* = [\Theta_3^* \quad \Theta_1^{*\top} \quad \Theta_2^{*\top}]^\top$; Θ_4^* sú parametre zákona riadenia a maticu

$$D = \begin{bmatrix} c^\top & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}$$

sme zaviedli práve preto aby sme v zákone riadenia (45) mohli priamo písať doplnený stavový vektor $X(t)$, teda

$$X(t) = \begin{bmatrix} x(t) \\ \nu_1(t) \\ \nu_2(t) \end{bmatrix} \quad (46)$$

Všimnime si ale, že

$$DX(t) = \begin{bmatrix} y(t) \\ \nu_1(t) \\ \nu_2(t) \end{bmatrix} \quad (47)$$

čo je veľmi dôležité, pretože signál $y(t)$ máme, ale signál $x(t)$ (pochopteľne) nemáme.

Pre prípad (45) teda môžeme zaviesť signálny vektor ω v poradí:

$$\omega = \begin{bmatrix} DX(t) \\ r(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ \nu_1(t) \\ \nu_2(t) \\ r(t) \end{bmatrix} \quad (48)$$

Ak raz zvolíme poradie signálov vo vektore ω , potom je tým jednoznačne určené aj poradie signálov vo vektore parametrov, v tomto prípade

$$\Theta = \begin{bmatrix} \Theta_3^* \\ \Theta_1^* \\ \Theta_2^* \\ \Theta_4^* \end{bmatrix} \quad (49)$$

O simulačnej „schéme“ (skôr o „simulačnom skripte“)

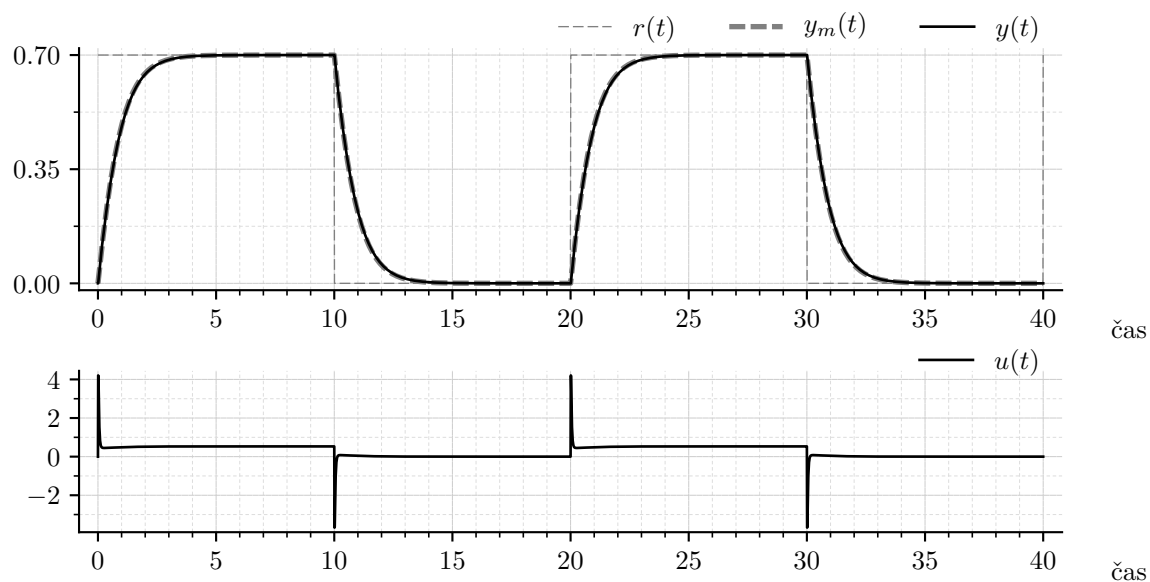
Nasledujúci výpis kódu 6 obsahuje funkciu `fcn_simSch2`. Je postavená na rovnakých princípoch ako sme videli v kóde 1. Navyše však obsahuje aj realizáciu neadaptívnej verzie riadiaceho systému, ktorý používa tu diskutovaný zákon riadenia.

Overenie správnosti vypočítaných parametrov zákona riadenia je realizované grafickým porovnaním priebehu výstupu referenčného modelu a výstupnej veličiny riadeného systému - viď obr. 2.

```

1 import numpy as np
2
3 from scipy.integrate import odeint
4
5 def fcn_LTIS(x, t, A, b, u):
6     dotx = np.matmul(A, x) + np.matmul(b, u)
7     return dotx
8
9
10 def fcn_simSch2(t_start, T_s, finalIndex, sig_dummy_ext):
11
12     # -----
13     # Riadený systém
14
15     A = np.array([[0, 1], [-2.6916, -2.7423]])
16     b = np.array([[0], [1]])

```



Obr. 2: Porovnanie priebehu výstupu referenčného modelu a výstupnej veličiny riadeného systému

```

17  c = np.array([[3.5578], [0.1664]])
18
19  # -----
20  # Referencny model
21
22  A_m = np.array([[0, 1], [-3.0, -3.5]])
23  b_m = np.array([[0], [1]])
24  c_m = np.array([[3], [1]])
25
26  # -----
27  # Pomocne filtre
28
29  Lambda_pom = np.array([[ -3]])
30  q_pom = np.array([[1]])
31
32  # -----
33  t_log = np.zeros([finalIndex, 1])
34  t_log[0,:] = t_start
35
36  # -----
37  x_0 = np.zeros(b.shape[0])
38
39  x_log = np.zeros([finalIndex, len(x_0)])
40  x_log[0,:] = x_0
41
42  y_log = np.zeros([finalIndex, 1])
43  y_log[0,:] = np.dot(c.T, x_0.reshape(-1,1))
44
45  # -----
46  x_m_0 = np.zeros(b_m.shape[0])
47
48  x_m_log = np.zeros([finalIndex, len(x_m_0)])
49  x_m_log[0,:] = x_m_0
50
51  y_m_log = np.zeros([finalIndex, 1])
52  y_m_log[0,:] = np.dot(c_m.T, x_m_0.reshape(-1,1))
53
54  # -----
55  nu1_log = np.zeros([finalIndex, q_pom.shape[0]])
56  nu2_log = np.zeros([finalIndex, q_pom.shape[0]])
57
58  # -----
59  u_log = np.zeros([finalIndex, 1])
60  u_log[0,:] = 0
61
62  # -----

```

```

63     timespan = np.zeros(2)
64     for idx in range(1, int(finalIndex)):
65
66         timespan[0] = t_log[idx-1,:]
67         timespan[1] = t_log[idx-1,:] + T_s
68
69         t_log[idx,:] = timespan[-1]
70
71         # -----
72
73         odeOut = odeint(fcn_LTIS,
74                        x_log[idx-1,:],
75                        timespan,
76                        args=(A, b, u_log[idx-1,:])
77                        )
78
79         x_log[idx,:] = odeOut[-1,:]
80         y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))
81
82         # -----
83         # Referencny model:
84
85         ref_sig = sig_dummy_ext[idx-1, :]
86
87         dotx_m = fcn_LTIS(x_m_log[idx-1,:], 0, A_m, b_m, ref_sig)
88
89         x_m_log[idx,:] = x_m_log[idx-1,:] + dotx_m * T_s
90
91         y_m_log[idx,:] = np.dot(c_m.T, x_m_log[idx,:].reshape
92         (-1,1))
93
94         # -----
95         # Pomocne filtre:
96
97         dotnu1 = fcn_LTIS(nu1_log[idx-1,:], 0, Lambda_pom, q_pom,
98         u_log[idx-1,:])
99         nu1_log[idx,:] = nu1_log[idx-1,:] + dotnu1 * T_s
100
101         dotnu2 = fcn_LTIS(nu2_log[idx-1,:], 0, Lambda_pom, q_pom,
102         y_log[idx-1,:])
103         nu2_log[idx,:] = nu2_log[idx-1,:] + dotnu2 * T_s
104
105         # -----
106         # Vektor omega:
107
108         omega = np.array([y_log[idx,:],
109         nu1_log[idx-1,:].reshape(-1,1),
110         nu2_log[idx-1,:].reshape(-1,1),
111         ref_sig,
112         ])
113
114         # -----
115         # Vektor Theta:
116
117         Theta = np.array([[ -4.5535],
118         [ -18.3809],
119         [ 11.8071],
120         [ 6.0096],
121         ])
122
123         # -----
124         u_log[idx,:] = np.dot(Theta.T, omega)[0]
125
126     return [t_log, x_log, y_log, u_log, y_m_log]
127
128 # -----
129 # Nastavenie simulacie
130
131 sim_t_start = 0
132 sim_t_final = 40
133 sim_T_s = 0.01
134 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
135
136 # Preddefinovany signal (pouzity ako referencny signal)
137
138 period_time = 20

```

```

137 period_tab = np.array([[0, 0.7],
138                        [10, 0],
139                        ])
140
141 sig_vysl = np.zeros([sim_finalIndex, 1])
142
143 for period in range(int(sim_t_final/period_time) + 1):
144     for idx in range( int((period*period_time)/sim_T_s), int((
145         period*period_time + period_time)/sim_T_s)):
146         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
147             period_time))<=idx*sim_T_s ][-1]
148         try:
149             sig_vysl[idx] = lastValue
150         except:
151             break
152
153 sig_dummy_ext = sig_vysl
154
155 # Spustenie simulacie
156 t_log, x_log, y_log, u_log, y_m_log = fcn_simSch2(
157     sim_t_start,
158     sim_T_s,
159     sim_finalIndex,
160     sig_dummy_ext,
161 )
162 # Tu by bolo kreslenie obrazkov...

```

Výpis kódu 6: Simulačná schéma uzavretého regulačného obvodu