

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 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.

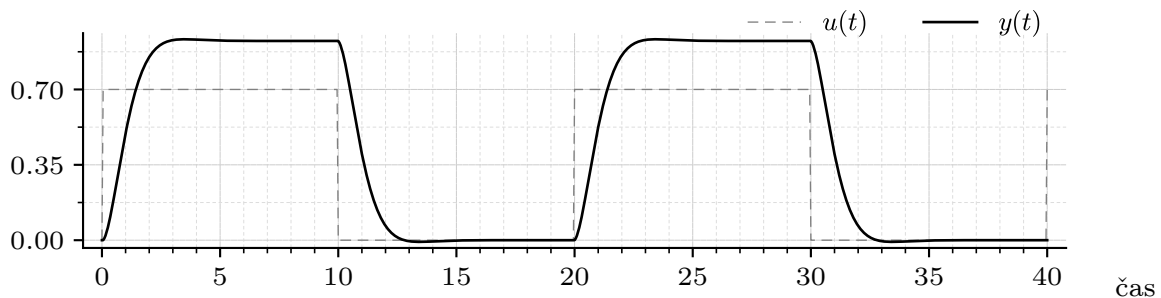
1.1 Úloha prvá, bod prvý

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 2:

```
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     # poctom 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 predchadzajucej
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 (stavovych 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“ 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:

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

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