

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
2	Časť druhá	14
2.1	Úloha prvá	14
2.1.1	Bod prvý	14
2.1.2	Bod druhý	14
2.1.3	Bod tretí	14
2.1.4	Bod štvrtý	15
2.1.5	Bod piaty	15
2.1.6	Bod šiesty	17
2.1.7	Bod siedmi	17
2.1.8	Bod ôsmy až desiaty	17
2.2	Dodatok (prevažne o nastavovaní rýchlosti adaptácie)	21
3	Časť navyše (s relatívnym stupňom 2)	25
3.1	Celkový pohľad na úlohu	25
3.1.1	Riadený systém z pohľadu návrhu riadiaceho systému	25

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 0 až 10 V (azda si niektorý čitateľ spomína). Možno nemá význam o tom vôbec hovoriť.

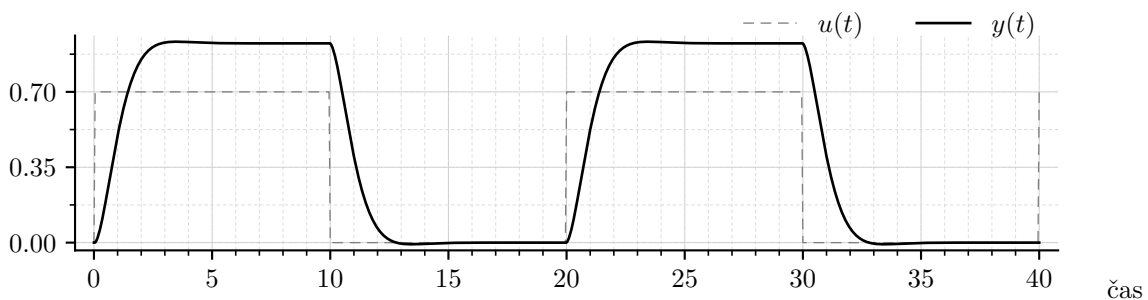
1.1 Úloha prvá, bod prvý

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:

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

```
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 # Vytvorime 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 velky 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 hlada 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 (stavovych velicin) chceme aj
91         # vystupnu velicinu y

```

```

92
93
94
95     u_log[idx,:] = sig_dummy_ext[idx,:]
96     # v tejto simulácii 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((period*
127         period_time + period_time)/sim_T_s)):
128         lastValue = period_tab[:,1][(period*period_time)/sim_T_s]
129         try:
130             sig_vysl[idx] = lastValue
131         except:
132             break
133
134 sig_dummy_ext = sig_vysl
135
136 # Teraz mozme "spustit" simulaciu:
137
138 t_log, x_log, y_log, u_log, = fcn_simSch1(
139     sim_t_start,
140     sim_T_s,
141     sim_finalIndex,
142     sig_dummy_ext,
143 )
144
145 # Tu by mohlo byt kreslenie obrazku, ale to tu neuvedieme.
146 # Jednoducho nieco v zmysle plot(t_log, y_log) a podobne...

```

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

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

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:

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

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

Vypočítajme:

Výpis kódu 2: Výpočet ideálnych parametrov zákona riadenia

```

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

```

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 zvoľme

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

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

$$\ddot{z}(t) = -a_1 \dot{z}(t) - a_0 z(t) + u(t) = -a_1 x_2(t) - a_0 x_1(t) + u(t) \quad (25)$$

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 zrejme...

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 zrejme 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()`:

Výpis kódu 3: dotx_m

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

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“):

Výpis kódu 4: x_m

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

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

Výpis kódu 5: y_m

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

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:

$$y_{\nu_1}(s) = \Theta_1^* \frac{1}{(s + \lambda)} u(s) \quad (37)$$

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 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

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

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

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

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

a označíme

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

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

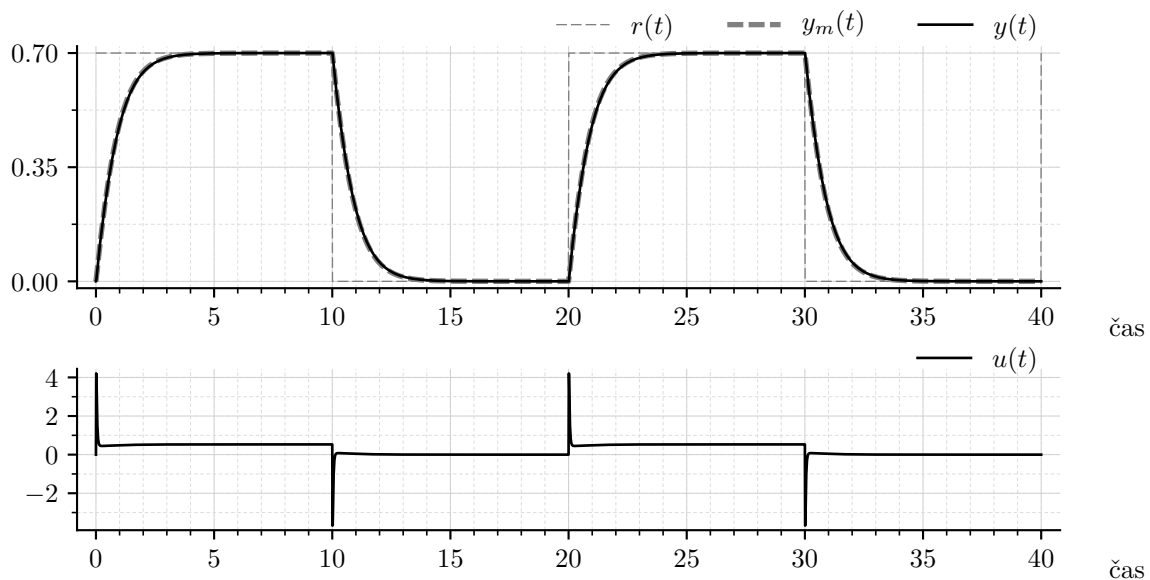
$$u(t) = \Theta_c^* D X(t) + \Theta_4^* r(t) \quad (45)$$

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

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



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

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 $x(t)$ (pochopiteľne) 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.

Výpis kódu 6: Simulačná schéma uzavretého regulačného obvodu

```
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     # Riadeny system
14
15     A = np.array([[0, 1], [-2.6916, -2.7423]])
16     b = np.array([[0], [1]])
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(-1,1))
92
93     # -----
94     # Pomocne filtre:
95
96     dotnu1 = fcn_LTIS(nu1_log[idx-1,:], 0, Lambda_pom, q_pom,
97     u_log[idx-1,:])
98     nu1_log[idx,:] = nu1_log[idx-1,:] + dotnu1 * T_s
99
100     dotnu2 = fcn_LTIS(nu2_log[idx-1,:], 0, Lambda_pom, q_pom,
101     y_log[idx-1,:])
102     nu2_log[idx,:] = nu2_log[idx-1,:] + dotnu2 * T_s
103
104     # -----
105     # Vektor omega:
106
107     omega = np.array([y_log[idx,:],
108     nu1_log[idx-1,:].reshape(-1,1),
109     nu2_log[idx-1,:].reshape(-1,1),
110     ref_sig,
111     ])
112
113     # -----
114     # Vektor Theta:
115
116     Theta = np.array([[-4.5535],
117     [-18.3809],
118     [11.8071],
119     [6.0096],
120     ])
121
122     # -----
123     u_log[idx,:] = np.dot(Theta.T, omega)[0]
124
125     return [t_log, x_log, y_log, u_log, y_m_log]
126
127 # -----
128 # Nastavenie simulacie
129
130 sim_t_start = 0
131 sim_t_final = 40
132 sim_T_s = 0.01
133 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
134
135 # Preddefinovany signal (pouzity ako referencny signal)
136
137 period_time = 20
138 period_tab = np.array([[0, 0.7],
139     [10, 0],
140     ])
141
142 sig_vysl = np.zeros([sim_finalIndex, 1])
143
144 for period in range(int(sim_t_final/period_time) + 1):
145     for idx in range(int((period*period_time)/sim_T_s), int((period*
146     period_time + period_time)/sim_T_s)):
147         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
148     period_time))<=idx*sim_T_s ][-1]
149         try:
150             sig_vysl[idx] = lastValue
151         except:
152             break
153
154 sig_dummy_ext = sig_vysl
155
156 # Spustenie simulacie
157
158 t_log, x_log, y_log, u_log, y_m_log = fcn_simSch2(
159     sim_t_start,
160     sim_T_s,
161     sim_finalIndex,
162     sig_dummy_ext,
163     )
164
165 # Tu by bolo kreslenie obrazkov...

```

2 Časť druhá

2.1 Úloha prvá

Pre nominálnu prenosovú funkciu riadeného systému navrhnete adaptívne riadenie s referenčným modelom so vstupno-výstupnou štruktúrou zákona riadenia (merateľný je len výstup riadeného systému a samozrejme vstup). Pre odvodenie zákona adaptácie použijete priamu Lyapunovovu metódu. Nech referenčný model je v tvare

$$W_m(s) = \frac{s+3}{s^2+3.5s+3} \quad (50)$$

2.1.1 Bod prvý

Určte zákon riadenia, ktorý sa bude používať v adaptívnom riadiacom systéme. Viď (11).

Avšak! Hovoríme o *adaptívnom* riadiacom systéme. Prečo sa „adaptujeme“? Pretože nepoznáme hodnoty parametrov riadeného systému. A keďže tieto nepoznáme, nedokážeme vypočítať (ideálne) parametre zákona riadenia, ktorými sú Θ_1^* , Θ_2^* , Θ_3^* a Θ_4^* (nedokážeme to v „adaptívnom prípade“).

Preto, v adaptívnom riadiacom systéme sa používa zákon riadenia, kde sú ideálne („hviezdičkované“) parametre zákona riadenia nahradené ich odhadmi. Tieto odhady sa „adaptujú“ (priebežne identifikujú) a teda sa menia v čase. Preto píšeme (v tomto prípade), že adaptovanými parametrami zákona riadenia sú $\Theta_1(t)$, $\Theta_2(t)$, $\Theta_3(t)$ a $\Theta_4(t)$. Takže v *adaptívnom* riadiacom systéme bude zákon riadenia:

$$u(t) = \Theta_1(t) \left[\frac{1}{(s+\lambda)} \right] u(t) + \Theta_2(t) \left[\frac{1}{(s+\lambda)} \right] y(t) + \Theta_3(t)y(t) + \Theta_4(t)r(t) \quad (51)$$

2.1.2 Bod druhý

Vypočítajte ideálne parametre zákona riadenia.

Urobili sme tak v časti 1.2.2.

2.1.3 Bod tretí

Zistite, či $W_m(s)$ je striktné pozitívne reálna (SPR) prenosová funkcia.

Máme prenosovú funkciu

$$W_m(s) = \frac{s+3}{s^2+3.5s+3} \quad (52)$$

V prvom bode nás zaujíma, či $W_m(s)$ je reálna pre všetky reálne s . Ak za s dosadíme reálne číslo, potom $W_m(s)$ je reálne číslo.

Ďalej nás zaujíma, či menovateľ $W_m(s)$ má korene v ľavej polrovine komplexnej roviny alebo má reálne korene na imaginárnej osi. Korene polynómu $s^2+3.5s+3$ sú $s_1 = -2$ a $s_2 = -1.5$. Takže táto podmienka je splnená.

A nakoniec je otázka, či $\Re\{W_m(j\omega)\} \geq 0$ pre všetky reálne ω . Vykonajme teda $W_m(s) \rightarrow W_m(j\omega)$:

$$W_m(j\omega) = \frac{j\omega+3}{(j\omega)^2+3.5j\omega+3} \quad (53)$$

Toto je (vlastne) nejaké komplexné číslo. Samotná ω nech je reálne číslo. Upravme.

$$W_m(j\omega) = \frac{j\omega+3}{(j\omega)^2+3.5j\omega+3} \quad (54a)$$

$$= \frac{j\omega+3}{-\omega^2+3.5j\omega+3} \quad (54b)$$

$$= \frac{j\omega+3}{(3-\omega^2)+j3.5\omega} \quad (54c)$$

Je potrebné získať reálnu časť komplexného čísla $W_m(j\omega)$. Pre vyjadrenie reálnej časti je výhodné využiť, že platí:

$$\frac{a + jb}{c + jd} = \frac{(a + jb)(c - jd)}{(c + jd)(c - jd)} = \frac{(a + jb)(c - jd)}{c^2 + d^2}$$

Takže:

$$W_m(j\omega) = \frac{(j\omega + 3)}{((3 - \omega^2) + j3,5\omega)} \frac{((3 - \omega^2) - j3,5\omega)}{((3 - \omega^2) - j3,5\omega)} \quad (55a)$$

$$= \frac{(j\omega + 3)((3 - \omega^2) - j3,5\omega)}{(3 - \omega^2)^2 + (3,5\omega)^2} \quad (55b)$$

$$= \frac{9 - 3\omega^2 - j10,5\omega + j3\omega - j\omega^3 + 3,5\omega^2}{(3 - \omega^2)^2 + (3,5\omega)^2} \quad (55c)$$

Reálna časť z toho je

$$\Re\{W_m(j\omega)\} = \frac{9 + 0,5\omega^2}{(3 - \omega^2)^2 + (3,5\omega)^2} \quad (56)$$

a teda platí $\Re\{W_m(j\omega)\} \geq 0 \forall \omega \in \mathbb{R}$. Prenosová funkcia $W_m(s)$ je SPR.

2.1.4 Bod štvrtý

Napište rovnicu výstupnej adaptačnej odchýlky e_1 .

Výstupná adaptačná odchýlka je, samozrejme, $e_1(t) = y(t) - y_m(t)$. Tu sa však myslí písanie rovnice, ktorá opisuje dynamiku adaptačnej odchýlky. Teda

$$e_1(t) = [W_m(s)] \frac{1}{\Theta_4^*} (\theta^T(t) \omega(t)) \quad (57)$$

kde pre podrobný význam jednotlivých symbolov odkazujeme čitateľa na študijný materiál k príslušnej téme. V skratke, je jasné, že $W_m(s)$ je prenosová funkcia referenčného modelu, Θ_4^* je jeden y ideálnych parametrov zákona riadenia, $\theta^T(t)$ je vektor pozostávajúci z odchýlok medzi ideálnymi parametrami zákona riadenia a ich odhadmi a $\omega(t)$ je signálny vektor zákona riadenia.

Mimochodom, toto je veľmi veľmi dôležitá rovnica celkovo v oblasti akejkoľvek priebežnej identifikácie parametrov (alebo adaptácie ak chcete). Podrobnosti sú ďaleko nad rámec tohto textu. . .

2.1.5 Bod piaty

Pre systém diferenciálnych rovníc $(\dot{e}, \dot{\theta})$, kde $\dot{\theta}$ sa najskôr uvažuje vo všeobecnom tvare (na začiatku odvodu sa uvažuje len všeob. funkcia f) zvolte kandidáta na Lyapunovovu funkciu a odvoďte (skonkretizujte) predpis (pravú stranu) pre $\dot{\theta}$.

Ako je čitateľovi iste známe, rovnicu opisujúcu dynamiku adaptačnej odchýlky je možné zapísať aj pomocou nejakého stavového vektora, konkrétne sa v učebnom texte uvádza

$$\dot{e}(t) = A_c e(t) + \overline{B}_c \frac{1}{\Theta_4^*} (\theta^T(t) \omega(t)) \quad (58a)$$

$$e_1(t) = C_c^T e(t) \quad (58b)$$

pričom $W_m(s) = C_c^T (sI - A_c)^{-1} \overline{B}_c$ a pre ďalšie podrobnosti viď príslušný učebný text.

Tu vieme, čo je signál $e_1(t)$ a aj ho vieme merať/získať. Vonkoncom nevieme merať/získať stavový vektor $e(t)$. Ale vieme, že existuje. Teoreticky.

Nič nebráni zostaveniu systému diferenciálnych rovníc v tvare

$$\dot{e}(t) = A_c e(t) + \overline{B}_c \frac{1}{\Theta_4^*} (\theta^T(t) \omega(t)) \quad (59a)$$

$$\dot{\theta}(t) = f(e_1(t), \omega(t)) \quad (59b)$$

a tento systém má, ako je uvedené v učebnom texte, veľmi výhodné vlastnosti vzhľadom na možnosti adaptácie (priebežnej identifikácie) pre dosiahnutie cieľa riadenia.

Inými slovami, ak systém diferenciálnych rovníc (59) bude stabilný (plus pár miliónov detailov detailov), tak stavový vektor $e(t)$ sa bude asymptoticky (s časom) blížiť k nule. Je potom jasné, že aj adaptačná odchýlka $e_1(t)$ sa bude blížiť k nule, čo je cieľ riadenia. Podrobnosti prečo to tak je, sa najlepšie ukazujú na prípade MRAC stavového (ako sme spomenuli v jednom odstavci vyššie - v časti 1.2.3).

Jasným (viac-menej) kandidátom na Lyapunovovu funkciu pre systém (59) teda je (si dovoľíme nepísať signály ako funkcie času, teda napr. píšeme len e nie $e(t)$)

$$V = e^T P e + \left| \frac{1}{\Theta_4^*} \right| \theta^T \Gamma^{-1} \theta \quad (60)$$

Cesta potom vedie do bodu, kde máme

$$\dot{V} = e^T (-Q) e + 2 (e^T P \bar{B}_c) \frac{1}{\Theta_4^*} \theta^T \omega + 2 \left| \frac{1}{\Theta_4^*} \right| \theta^T \Gamma^{-1} \dot{\theta} \quad (61)$$

a chceme zabezpečiť aby časová derivácia \dot{V} bola záporne definitná. Na pravej strane rovnice (61), člen $e^T (-Q) e$ je záporne definitný (vždy menší ako nula). Ak by na pravej strane rovnice (61) zvyšné dva členy neboli platilo by požadované $\dot{V} \leq 0$. Inými slovami žiadame:

$$0 = 2 (e^T P \bar{B}_c) \frac{1}{\Theta_4^*} \theta^T \omega + 2 \left| \frac{1}{\Theta_4^*} \right| \theta^T \Gamma^{-1} \dot{\theta} \quad (62a)$$

$$2 \left| \frac{1}{\Theta_4^*} \right| \theta^T \Gamma^{-1} \dot{\theta} = -2 (e^T P \bar{B}_c) \frac{1}{\Theta_4^*} \theta^T \omega \quad (62b)$$

$$\left| \frac{1}{\Theta_4^*} \right| \Gamma^{-1} \dot{\theta} = - (e^T P \bar{B}_c) \text{sign} \left(\frac{1}{\Theta_4^*} \right) \left| \frac{1}{\Theta_4^*} \right| \omega \quad (62c)$$

$$\Gamma^{-1} \dot{\theta} = -\text{sign} \left(\frac{1}{\Theta_4^*} \right) (e^T P \bar{B}_c) \omega \quad (62d)$$

$$\dot{\theta} = -\text{sign} \left(\frac{1}{\Theta_4^*} \right) (e^T P \bar{B}_c) \Gamma \omega \quad (62e)$$

Tu sme, v princípe, práve získali hľadaný zákon adaptácie, teda predpis, ktorý hovorí ako sa mení vektor θ . Avšak, obsahuje vektor e . Ten nemáme. Všetko uvedené je len teoretické.

Tu sa využije veľmi významná vlastnosť, že prenosová funkcia $W_m(s)$ je SPR. Pripomeňme, že pre ňu platí $W_m(s) = C_c^T (sI - A_c)^{-1} \bar{B}_c$ a s využitím tejto reprezentácie referenčného modelu je možné využiť Meyerovu-Kalmanovu-Yakubovichovu Lemmu (vetu). Podľa nej platí

$$\begin{aligned} A_c^T P + P A_c &= -Q \\ P \bar{B}_c &= C_c \end{aligned}$$

kde by sme mali byť oboznámený čo konkrétne teraz sú uvedené matice...

Keďže platí $P \bar{B}_c = C_c$, tak máme

$$\dot{\theta} = -\text{sign} \left(\frac{1}{\Theta_4^*} \right) (e^T P \bar{B}_c) \Gamma \omega \quad (63a)$$

$$\dot{\theta} = -\text{sign} \left(\frac{1}{\Theta_4^*} \right) (e^T C_c) \Gamma \omega \quad (63b)$$

Výraz $e^T C_c$ je skalár. Takže $e^T C_c = C_c^T e$. A potom je jasné, že $C_c^T e = e_1$. Preto môžeme písať

$$\dot{\theta} = -\text{sign} \left(\frac{1}{\Theta_4^*} \right) e_1 \Gamma \omega \quad (64)$$

Toto je stále hľadaný zákon adaptácie. Ale už neobsahuje nedostupný (len teoretický) vektor e ale dostupný signál e_1 , čo je jednoducho výstupná adaptačná odchýlka.

2.1.6 Bod šiesty

Určte zákon adaptácie, ktorý sa bude používať v adaptívnom riadiacom systéme.

Práve sme tak urobili v predchádzajúcom bode. Zákon adaptácie je

$$\dot{\Theta}(t) = -\text{sign}\left(\frac{1}{\Theta_4^*}\right) e_1(t) \Gamma \omega(t) \quad (65)$$

Pripomeňme, že signálny vektor zákona riadenia v tomto prípade je

$$\omega(t) = \begin{bmatrix} y(t) \\ \nu_1(t) \\ \nu_2(t) \\ r(t) \end{bmatrix} \quad (66)$$

takže zákon adaptácie detailnejšie napísaný je

$$\dot{\Theta}(t) = -\text{sign}\left(\frac{1}{\Theta_4^*}\right) e_1(t) \begin{bmatrix} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & 0 \\ 0 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & \gamma_4 \end{bmatrix} \begin{bmatrix} y(t) \\ \nu_1(t) \\ \nu_2(t) \\ r(t) \end{bmatrix} \quad (67)$$

kde sme ako príklad zvolili maticu Γ diagonálnu s kladnými číslami $\gamma_1, \dots, \gamma_4$ na diagonále.

2.1.7 Bod siedmi

Zvoľte Γ (jednoducho, zvoľte všetky ľubovoľne voliteľné prvky zákona/zákonov adaptácie).

Voľba matice Γ ako diagonálnej matice v prvom rade spĺňa podmienky pre túto maticu (má byť symetrická a kladne definitná). Má to však aj praktický význam. Matica Γ je hlavným nástrojom ako ovplyvniť vlastnosť zákona adaptácie, ktorá sa nazýva rýchlosť adaptácie. Ak je navyše diagonálna, vieme relatívne nezávisle ovplyvňovať rýchlosť adaptácie jednotlivých prvkov vektora adaptovaných parametrov. Uvažujme teda o matici Γ v tvare

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & 0 \\ 0 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & \gamma_4 \end{bmatrix} \quad (68)$$

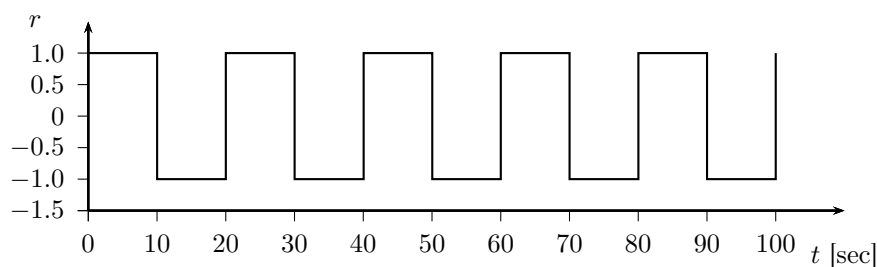
V tomto prípade máme vektor adaptovaných parametrov v tvare

$$\Theta(t) = \begin{bmatrix} \Theta_3(t) \\ \Theta_1(t) \\ \Theta_2(t) \\ \Theta_4(t) \end{bmatrix} \quad (69)$$

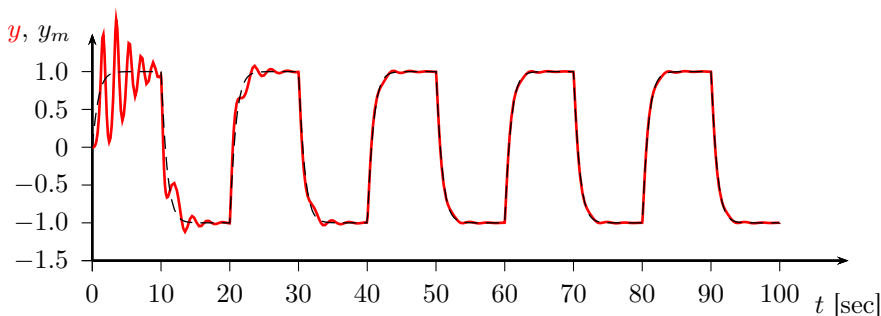
Ak by sme chceli ovplyvniť rýchlosť s akou sa mení napr. parameter $\Theta_2(t)$, je to možné urobiť zmenou veľkosti čísla γ_3 v matici Γ . Ak zvýšime γ_3 , potom $\dot{\Theta}_2(t)$ bude dosahovať v princípe vyššie hodnoty, a naopak...

2.1.8 Bod ôsmy až desiaty

- Začiatkové hodnoty adaptovaných parametrov zvoľte nulové.
- Zostavte adaptívny riadiaci systém (simulačnú schému) a pridajte ho k simulovanej sústave.
- Použite obdĺžnikový referenčný signál r ako na Obr. 3. Vzorové výsledky simulácie sú na Obr. 4.



Obr. 3: Referenčný signál r



Obr. 4: Výsledok simulácie

Nasledujúci výpis kódu 7 obsahuje funkciu `fcn_simSch3`. Je postavená na rovnakých princípoch ako sme tu už viac krát uviedli. Rozdielom oproti predchádzajúcej simulácii vo výpise kódu 6 je pridanie zákona adaptácie (nahradenie pevne stanovených hodnôt vektora Θ).

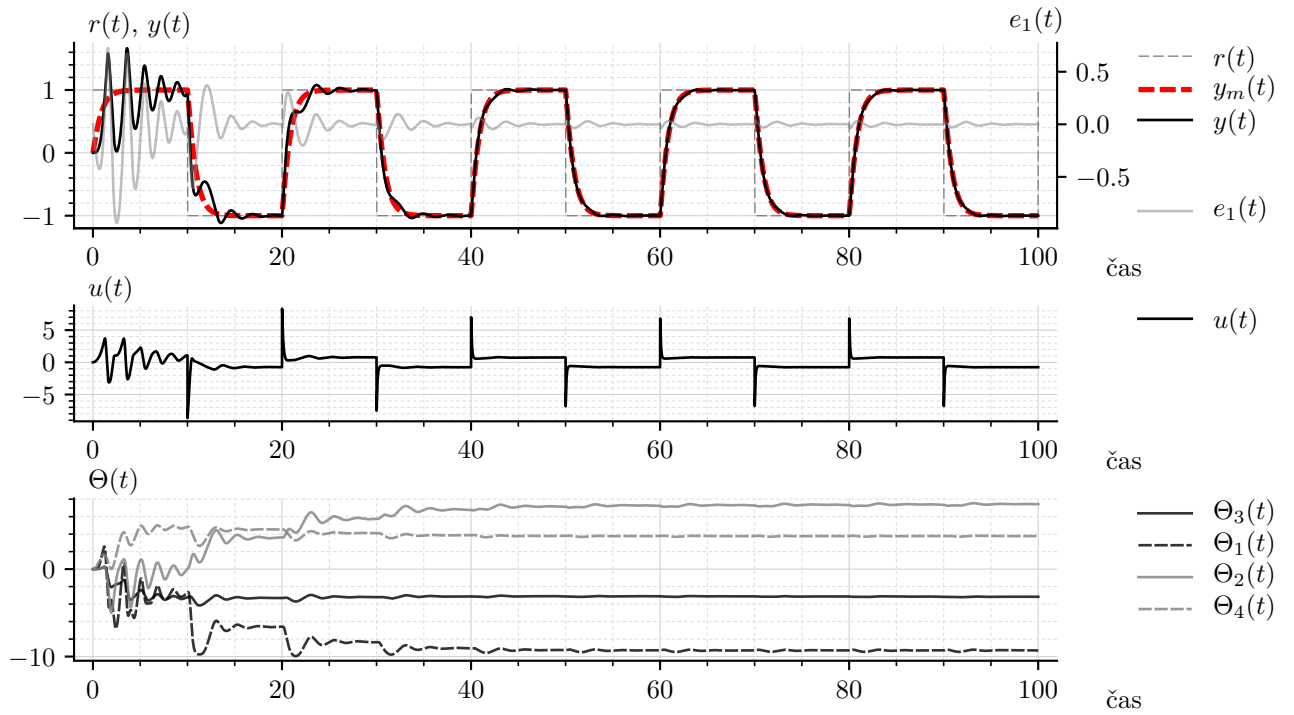
Výsledky simulácie pre uvedené zadanie sú na obr. 5.

Výpis kódu 7: Simulačná schéma adaptívneho riadiaceho systému

```

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_simSch3(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]])
17     c = np.array([[3.5578], [0.1664]])
18
19     # -----
20     # Referenčný 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     # Pomocné 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)])

```



Obr. 5: Výsledok simulácie

```

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     Theta_log = np.zeros([finalIndex, 4])
60
61     # -----
62     u_log = np.zeros([finalIndex, 1])
63     u_log[0,:] = 0
64
65     # -----
66     timespan = np.zeros(2)
67     for idx in range(1, int(finalIndex)):
68
69         timespan[0] = t_log[idx-1,:]
70         timespan[1] = t_log[idx-1,:] + T_s
71
72         t_log[idx,:] = timespan[-1]
73
74         # -----
75
76         odeOut = odeint(fcn_LTIS,
77                         x_log[idx-1,:],
78                         timespan,
79                         args=(A, b, u_log[idx-1,:])
80                         )
81
82         x_log[idx,:] = odeOut[-1,:]
83         y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))
84
85         # -----
86         # Referencny model:

```

```

87
88     ref_sig = sig_dummy_ext[idx-1, :]
89
90     dotx_m = fcn_LTIS(x_m_log[idx-1,:], 0, A_m, b_m, ref_sig)
91
92     x_m_log[idx,:] = x_m_log[idx-1,:] + dotx_m * T_s
93
94     y_m_log[idx,:] = np.dot(c_m.T, x_m_log[idx,:].reshape(-1,1))
95
96     # -----
97     # Adaptacna odchylka
98
99     adaptError = y_log[idx-1,:] - y_m_log[idx-1,:]
100
101     # -----
102     # Pomocne filtre
103
104     dotnu1 = fcn_LTIS(nu1_log[idx-1,:], 0, Lambda_pom, q_pom,
105 u_log[idx-1,:])
106     nu1_log[idx,:] = nu1_log[idx-1,:] + dotnu1 * T_s
107
108     dotnu2 = fcn_LTIS(nu2_log[idx-1,:], 0, Lambda_pom, q_pom,
109 y_log[idx-1,:])
110     nu2_log[idx,:] = nu2_log[idx-1,:] + dotnu2 * T_s
111
112     # -----
113     # Vektor omega:
114
115     omega = np.array([y_log[idx,:],
116                      nu1_log[idx-1,:].reshape(-1,1),
117                      nu2_log[idx-1,:].reshape(-1,1),
118                      ref_sig,
119                      ])
120
121     # -----
122     # Zakon adaptacie
123
124     Gamma = np.diag(np.array([5, 50, 50, 5]))
125
126     dotTheta = - np.matmul(Gamma, omega) * adaptError
127
128     Theta_log[idx,:] = Theta_log[idx-1,:] + dotTheta.reshape
129 (1,-1) * T_s
130
131     # -----
132     # Zakon ZakonRiadenia
133
134     u_log[idx,:] = np.dot(Theta_log[idx-1,:].T, omega)[0]
135
136     return [t_log, x_log, y_log, u_log, y_m_log, Theta_log]
137
138 # -----
139 # Nastavenie simulacie
140
141 sim_t_start = 0
142 sim_t_final = 100
143 sim_T_s = 0.01
144 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
145
146 # Preddefinovany signal (pouzity ako referencny signal)
147
148 period_time = 20
149 period_tab = np.array([[0, 1],
150                        [10, -1],
151                        ])
152
153 sig_vysl = np.zeros([sim_finalIndex, 1])
154
155 for period in range(int(sim_t_final/period_time) + 1):
156     for idx in range( int((period*period_time)/sim_T_s), int((period*
157 period_time + period_time)/sim_T_s)):
158         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
159 period_time))<=idx*sim_T_s ][-1]
160         try:
161             sig_vysl[idx] = lastValue
162         except:
163             break
164
165 sig_dummy_ext = sig_vysl
166
167 # Spustenie simulacie

```

```

163
164 t_log, x_log, y_log, u_log, y_m_log, Theta_log = fcn_simSch3(
165     sim_t_start,
166     sim_T_s,
167     sim_finalIndex,
168     sig_dummy_ext,
169 )
170
171 # Tu by bolo kreslenie obrazkov...

```

2.2 Dodatok (prevažne o nastavovaní rýchlosti adaptácie)

Vo svetle toho, že v tomto príklade je použitý model reálneho systému – jednosmerného motora, uvažujme realistickejšiu simuláciu. Zmysluplné okolie pracovného bodu je $\pm 0,7$ [V], teda má význam žiadať od výstupnej veličiny $y(t)$ maximálne hodnoty $\pm 0,7$ a akčný zásah môže byť v rozsahu cca ± 5 (tiež vo voltoch, ale to sú už prílišné podrobnosti keďže čitateľ zrejme nepozná detaily k riadenému systému).

Prípád 1

Preto, použijeme rovnaké nastavenie adaptívneho riadiaceho systému ako vo „vzorovej“ (umelo vymyslenej) simulácii a v prvom rade zmeňme priebeh referenčného signálu. Výsledok je na obr. 6.

Priebeh veličín riadeného systému, výstupnej aj vstupnej, je v poriadku z hľadiska rozsahov (v okolí pracovných bodov to zodpovedá reálnemu systému (motorčeku)). Povedzme však, že rýchlosť adaptácie (výsledok na obr. 6) je nevyhovujúca. Nech je požiadavka taká, že v čase 100 má byť riadiaci systém už adaptovaný. Je teda potrebné upraviť voliteľný parameter zákona adaptácie, ktorý vo veľkej miere určuje rýchlosť adaptácie. Tým ja matica Γ . V tomto „vzorovom“ prípade vidíme, že sa uvažuje

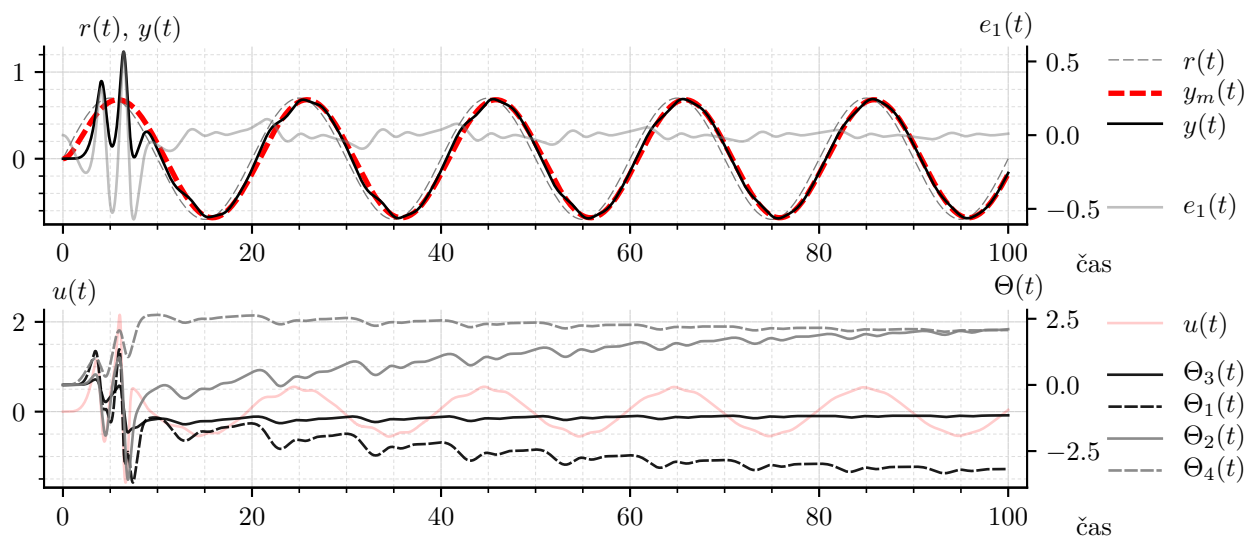
```
Gamma = np.diag(np.array([5, 50, 50, 5]))
```

Prípád 2

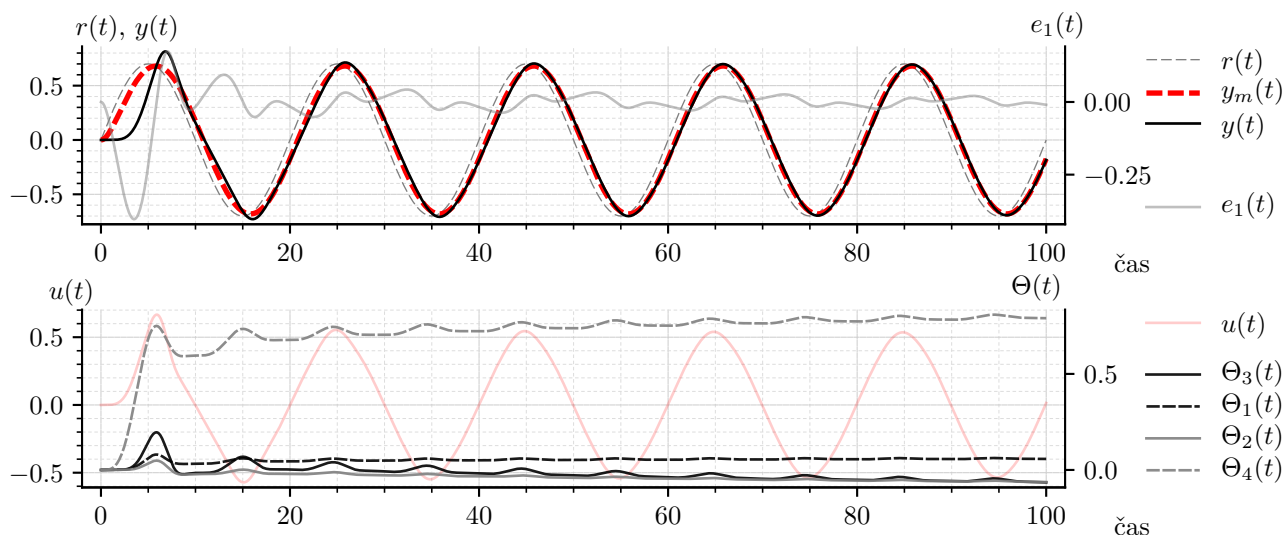
Výslednú rýchlosť adaptácie však neovplyvňuje len matica Γ ale samozrejme všetky prvky v zákone adaptácie. Teda signálny vektor ω a samozrejme adaptačná odchýlka. Všetky tieto signály - ich charakter a vplyv na rýchlosť adaptácie, závisia od toho ako veľmi tzv. *vybudíme* systém².

Pod vybudením sa myslí navodenie takých podmienok, aby sa prejavili takpovediac všetky vlastnosti systému. V tu uvažovanej schéme je viac menej jedinou možnosťou ako „vybudiť“ riadený systém je riadiť ho tak, že sledujeme relatívne „zložitý“ referenčný signál. Naopak, ak referenčný signál je „jednoduchý“, tak je ho

²Tu si dovoľme smerovať čitateľa na pojem *persistent excitation* v adaptívnych systémoch alebo pri priebežnej identifikácii vo všeobecnosti #UTFG #nooffense.



Obr. 6



Obr. 7

aj jednoduché sledovať, a takpovediac blok adaptácie ľahko nájde vhodné parametre zákona riadenia (tak aby $y(t)$ sledovalo $y_m(t)$)³.

Jednoduchým (v princípe najjednoduchším) referenčným signálom je sínusoida („obsahuje“ len jednu frekvenciu, obdĺžnikový signál „obsahuje“ nekonečne veľa frekvencií). Vyskúšajme teda $r(t)$ ako harmonický signál s periódou 20 [časových jednotiek - sekúnd v realite] a amplitúdou 0,7 [voltov v realite]. Pritom pre začiatok uvažujme

```
Gamma = np.diag(np.array([1, 1, 1, 1]))
```

Výsledok je na obr. 7. Adaptačná odchýlka sa s časom znižuje, ale povedzme, že nie dostatočne rýchlo.

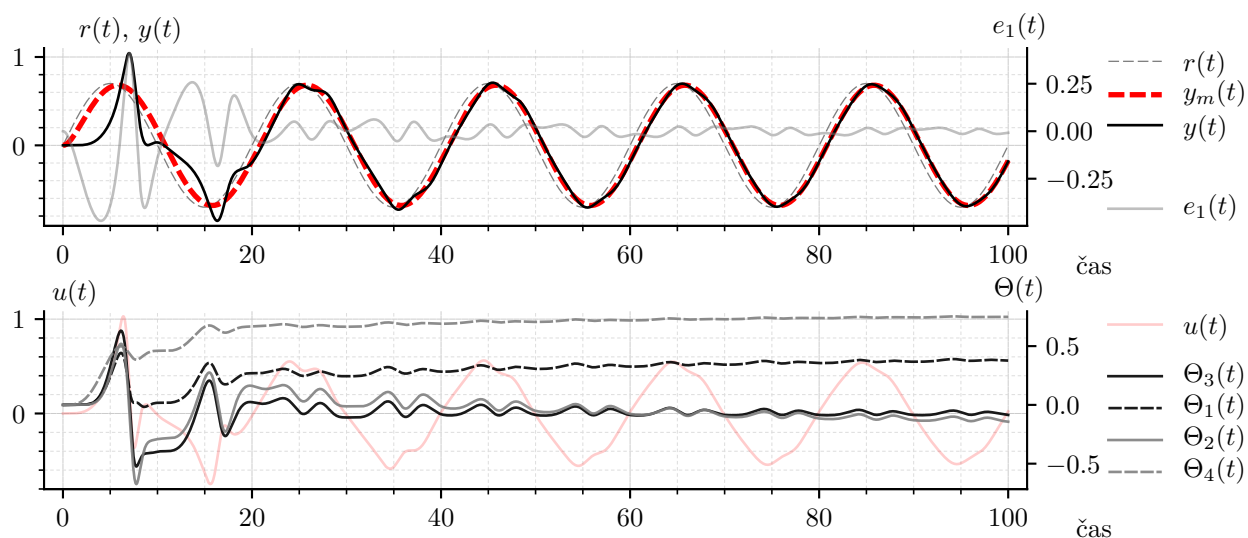
Prípad 3

Na priebehu adaptovaných parametrov (vektor $\Theta(t)$) je vidieť, že najrýchlejšie zmeny boli pri parametri Θ_4 . Skúsme docieľiť, aby sa aj ostatné parametre menili rýchlejšie - prípadne, aby sa parameter Θ_4 menil pomalšie. Jednoducho, aby adaptácia všetkých parametrov prebiehala „relatívne rovnako rýchlo“.

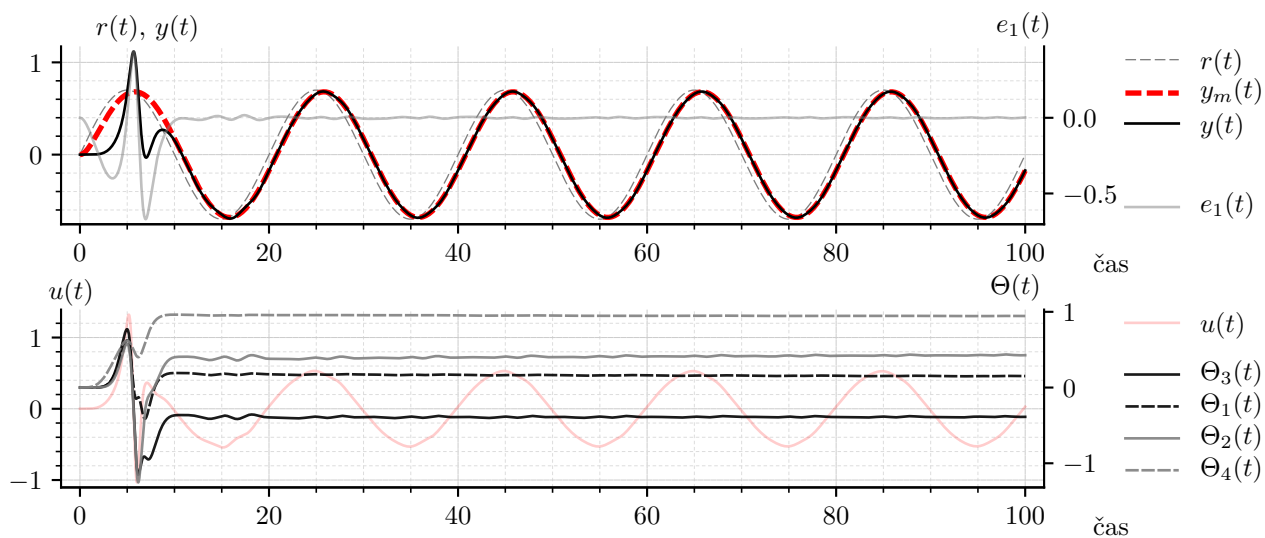
S pomocou cieľavedomého experimentovania a kvalifikovaného odhadu⁴ sme, dajme tomu, dospeli k nasledujúcim hodnotám v matici Γ :

³Mimochodom, referenčný signál by sme si v praxi pravdepodobne nemohli len tak zvoliť, ak, tak azda pre „fázu učenia/adaptácie“ a podobne. Aj toto treba mať na pamäti pri uvažovaní o tu uvedených poznámkach.

⁴Nie, na toto neexistuje univerzálny návod.



Obr. 8



Obr. 9

```
Gamma = np.diag(np.array([3, 5, 10, 0.5]))
```

Výsledok je na obr. 8. Je možné konštatovať, že všetky parametre sa menili (na začiatku určite) rovnako rýchlo. Celková rýchlosť zmenšovania sa adaptačnej odchýlky je však pri tom relatívne rovnaká ako v predchádzajúvom prípade.

Prípad 4

Ak by sme teraz chceli zvýšiť celkovú rýchlosť adaptácie, je možné urobiť napr.:

```
Gamma = np.diag(np.array([3, 5, 10, 0.5])*2)
```

Výsledok je na obr. 9.

Prípad 5

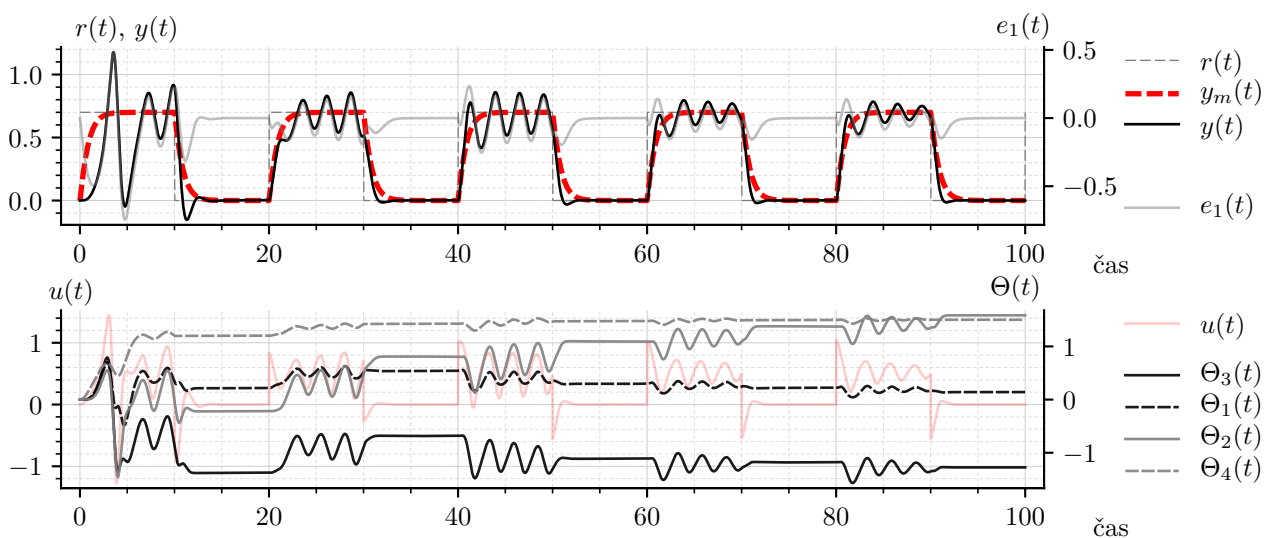
Vyskúšajme s touto voľbou Γ zložitejší referenčný signál ako v prípade 1. Výsledok je na obr. 10. Rýchlosť zmeny jednotlivých adaptovaných parametrov je relatívne rovnaká, avšak celková rýchlosť adaptácie nie je dostatočná.

Prípad 6

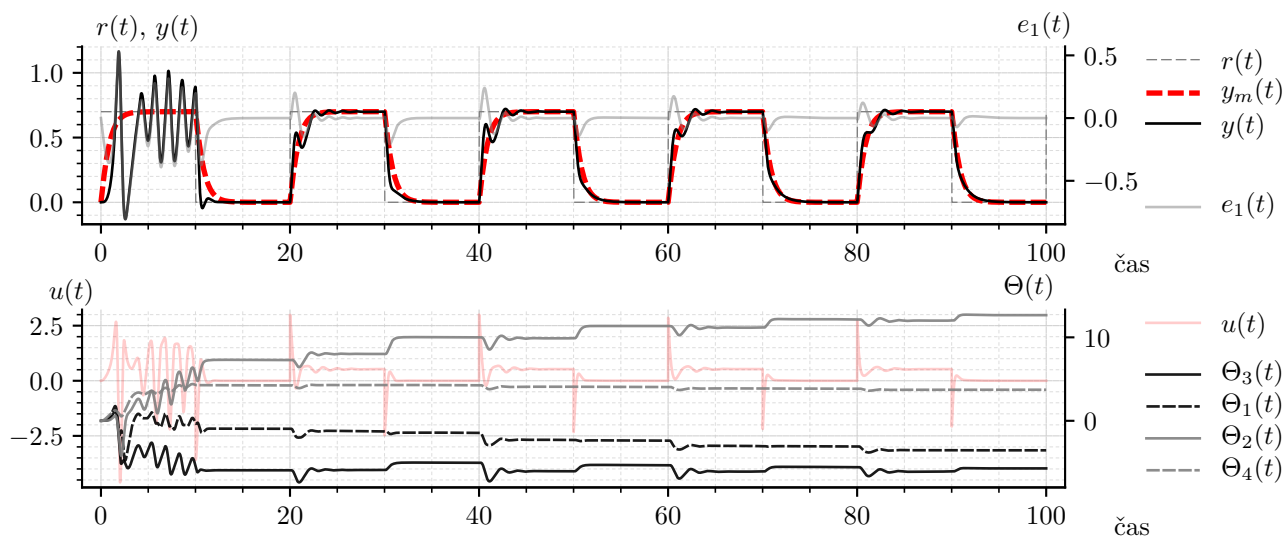
Preto:

```
Gamma = np.diag(np.array([3, 5, 10, 0.5])*10)
```

Výsledok je na obr. 11. Rýchlosť adaptácie sme jednoznačne zvýšili a mohli by sme ju zvyšovať aj viac.



Obr. 10

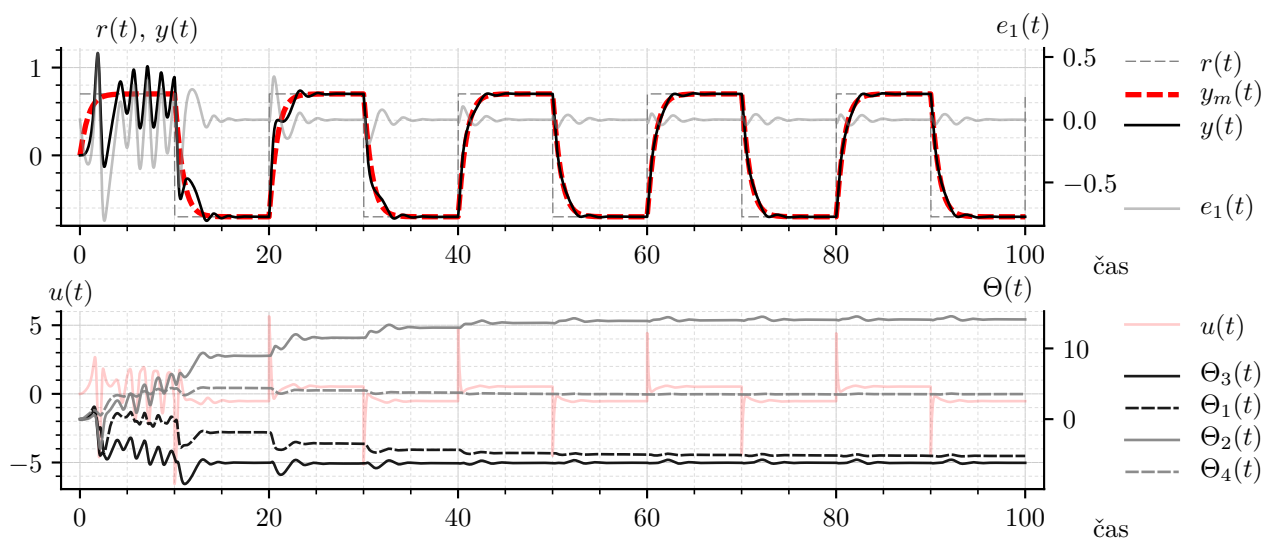


Obr. 11

Prípád 7

Všimnime si ale, že sme uvažovali signál $r(t)$ len v kladných hodnotách. „Nevybudili“ sme systém v celom potenciálnom rozsahu okolia pracovného bodu. Skúsme preto ponechať aktuálne nastavenie Γ ale zmeniť $r(t)$ - viď obr. 12.

Zmena $r(t)$, v zmysle, že je viac „budiaci“, sa jednoznačne prejavila v rýchlosti ustálenia sa adaptovaných parametrov a mierne aj na rýchlosti zmenšovania sa adaptačnej odchýlky.



Obr. 12

3 Časť navyše (s relatívnym stupňom 2)

V predchádzajúcom sme sa zaoberali prípadom, keď model riadeného systému bola prenosová funkcia, ktorej relatívny stupeň bol $n^* = 1$. V tejto časti je príklad, keď riadený systém má význam modelovať prenosovou funkciou s relatívnym stupňom $n^* = 2$.

3.1 Celkový pohľad na úlohu

Nech riadeným systémom je kyvadlo a nech riadenou veličinou je poloha (uhol) kyvadla. Zároveň, nech riadený systém je daný diferenciálnou rovnicou opisujúcou dynamiku rotačného pohybu kyvadla. Rovnica je v tvare

$$ml^2\ddot{\varphi}(t) + \beta\dot{\varphi}(t) + mgl \sin \varphi(t) = u(t) \quad (70)$$

kde hmotný bod s hmotnosťou m [kg] pripevnený na rameno so zanedbateľnou hmotnosťou a dĺžkou l [m] kmitá (otáča sa okolo osi). Kmity sú tlmené viskóznym trením s koeficientom β [kg m² s⁻¹]. Uhol medzi zvislicou a ramenom kyvadla je označený φ [rad] a gravitačné zrýchlenie je $g = 9,81$ [m s⁻²]. Signál $u(t)$ [kg m² s⁻²] je externý moment sily pôsobiaci na rameno kyvadla, $\dot{\varphi}(t)$ [rad s⁻¹] je uhlová rýchlosť a $\ddot{\varphi}(t)$ [rad s⁻²] je uhlové zrýchlenie ramena kyvadla. Číselné hodnoty parametrov kyvadla sú nasledovné:

$$\begin{aligned} m &= 1 \quad [\text{kg}] \\ l &= 1 \quad [\text{m}] \\ \beta &= 2 \cdot 0,5 \cdot \sqrt{\frac{g}{l}} \quad [\text{kg m}^2 \text{ s}^{-1}] \end{aligned}$$

Cieľom je riadiť polohu kyvadla v okolí rôznych pracovných bodov. Uvažujme však, že pracovné body sú z intervalu polôh kyvadla o až 90 stupňov. Konkrétna voľba pracovných bodov ako aj voľba veľkosti ich okolia sa ponechávajú na čitateľa.

Nech dynamika výstupnej veličiny riadeného systému v ktoromkoľvek pracovnom bode je daná referenčným modelom, ktorým je lineárny dynamický systém s pólmí $p_1 = -2$ a $p_2 = -1$. Samozrejme je, že tento lineárny dynamický systém bude mať jednotkové statické zosilnenie.

Referenčný model

Ak vieme, že RM je druhého rádu, a tiež vlastne vieme (aj keď sme to ešte explicitne neukázali), že sa zaoberáme riadeným systémom, ktorého (lineárny) model má relatívny stupeň $n^* = 2$, potom z toho plynie, že referenčný model má tvar

$$W_m(s) = \frac{3}{s^2 + 3s + 2} \quad (71)$$

3.1.1 Riadený systém z pohľadu návrhu riadiaceho systému

Pre lepšiu predstavu o riadenom systéme ho opíšme v stavovom priestore. Voľbou stavových veličín $x_1(t) = \varphi(t)$ a $x_2(t) = \dot{\varphi}(t)$ máme

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l} \sin(x_1(t)) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u(t) \quad (72a)$$

$$y(t) = x_1(t) \quad (72b)$$

kde sme pre prehľadnosť v nasledujúcom zaviedli aj výstupnú veličinu $y(t) = \varphi(t)$.

Toto je, zjavne, nelineárny časovo-invariantný systém druhého rádu. Tu sa však zaoberáme takou triedou riadiacich systémov, ktoré predpokladajú lineárny model riadeného systému. Jeho parametre môžu byť neznáme (môžu sa meniť), ale musí byť lineárny.

Linearizácia v okolí pracovného bodu

Uvažovaný riadený systém je možné linearizovať v okolí pracovného bodu.

V prvom rade potrebujeme poznať pracovný bod. Pre zvolenú hodnotu (ustálenú) na vstupe systému, označme ju u_{PB} , potrebujeme poznať prislúchajúcu hodnotu (ustálenú) na výstupe, označme ju y_{PB} .

Tu máme k dispozícii analytický opis riadeného systému (70). V ustálenom stave (časové derivácie nulové) máme

$$(mgl) \sin(y_{PB}) = u_{PB} \quad (73a)$$

$$y_{PB} = \arcsin\left(\frac{1}{(mgl)} u_{PB}\right) \quad (73b)$$

Toto je, samozrejme, prevodová charakteristika riadeného systému. Nie je to priamka. To znamená, že v jednom pracovnom bode (v okolí jedného pracovného bodu) bude statické zosilnenie (sklon prevodovej charakteristiky) iné, ako v inom pracovnom bode (v okolí iného pracovného bodu).

Mimochodom, zvolme si veľkosť okolia pracovného bodu. Dobrou pomôckou je grafické zobrazenie prevodovej charakteristiky. To sa ponecháva na čitateľa (autor je lenivý). Tu si zvolme okolie z veľkosťou ± 3 [°] (cca 0,0524 [rad]) na strane výstupnej veličiny a budme s ním spokojný v tom zmysle, že pri takýchto malých odchýlkach si prevodová charakteristika zachováva prakticky rovnaký sklon ako v pracovnom bode.

Ďalej je potrebné zaviesť veličiny, ktoré budú takpovediac odchýlkami odhodnot v pracovnom bode. Konkrétne

$$\Delta u(t) = u(t) - u_{PB} \quad (74a)$$

$$\Delta y(t) = y(t) - y_{PB} \quad (74b)$$

kde sme teda definovali Δ odchýlky od pracovného bodu. To znamená, že pôvodné veličiny sú

$$u(t) = u_{PB} + \Delta u(t) \quad (75a)$$

$$y(t) = y_{PB} + \Delta y(t) \quad (75b)$$

$$x_1(t) = x_{1PB} + \Delta x_1(t) \quad (75c)$$

$$x_2(t) = x_{2PB} + \Delta x_2(t) \quad (75d)$$

kde sme rovnako yaviedli „odchýlkové veličiny“ aj pre stavové veličiny so systému (72). Dosadíme v (72), čo v prvom rade znamená, že

$$\begin{bmatrix} \frac{d}{dt} (x_{1PB} + \Delta x_1(t)) \\ \frac{d}{dt} (x_{2PB} + \Delta x_2(t)) \end{bmatrix} = \begin{bmatrix} \Delta \dot{x}_1(t) \\ \Delta \dot{x}_2(t) \end{bmatrix} \quad (76)$$

pretože x_{1PB} a x_{2PB} su len čísla nezávislé od času. Potom na základe (72) môžeme písať

$$\Delta \dot{x}_1(t) = (x_{2PB} + \Delta x_2(t)) \quad (77a)$$

$$\Delta \dot{x}_2(t) = -\frac{\beta}{ml^2} (x_{2PB} + \Delta x_2(t)) - \frac{g}{l} \sin((x_{1PB} + \Delta x_1(t))) + \frac{1}{ml^2} (u_{PB} + \Delta u(t)) \quad (77b)$$

$$\Delta y(t) = \Delta x_1(t) \quad (77c)$$

kde sme zohľadnili, že musí platiť $\Delta y(t) = \Delta x_1(t)$, keďže máme $y(t) = x_1(t)$. Rovnice (77) opisujú stále to isté nelineárne kyvadlo, avšak pomocou iných („odchýlkových“) veličín. Je to užitočné preto, že tento „nový“ systém má ustálený stav x_e (ekvilíbrio) v bode

$$x_e = \begin{bmatrix} \Delta x_{e1} \\ \Delta x_{e2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (78)$$

a to samozrejme pri $\Delta u(t) = 0$. V tomto „novom“ ustálenom stave, očividne, pre pôvodné veličiny platí, že $u(t) = u_{PB}$ a $y(t) = y_{PB}$. Teda systém je ustálený v pracovnom bode. V tomto „novom“ ekvilibriu, je možné systém štandardne linearizovať. Výsledkom je

$$\begin{bmatrix} \Delta \dot{x}_1(t) \\ \Delta \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} \cos(y_{PB}) & -\frac{\beta}{m l^2} \end{bmatrix} \begin{bmatrix} \Delta x_1(t) \\ \Delta x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m l^2} \end{bmatrix} \Delta u(t) \quad (79a)$$

$$\Delta y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_1(t) \\ \Delta x_2(t) \end{bmatrix} \quad (79b)$$

Toto je lineárny dynamický systém, ktorý je možné zapísať v tvare prenosovej funkcie

$$\frac{\Delta y(s)}{\Delta u(s)} = \frac{k_p}{s^2 + a_1 s + a_0} \quad (80)$$

kde

$$k_p = \frac{1}{m l^2} \quad (81a)$$

$$a_0 = \frac{g}{l} \cos(y_{PB}) = \frac{g}{l} \cos\left(\arcsin\left(\frac{1}{(mgl)} u_{PB}\right)\right) \quad (81b)$$

$$a_1 = \frac{\beta}{m l^2} \quad (81c)$$

Je teda zrejmé, že statické a dynamické vlastnosti tohto lineárneho systému sú závislé od pracovného bodu.

Ilustrácia potreby adaptácie

Z uvedeného je zrejmé, že statické a dynamické vlastnosti tohto lineárneho systému sú závislé od pracovného bodu. Ak by sme navrhli riadiaci systém, tak, že sa splní cieľ riadenia v okolí jedného pracovného bodu, potom v inom pracovnom bode (v jeho okolí) by tento nespĺňal cieľ riadenia.

Ukážme zmenu vlastností (statických aj dynamických) riadeného systému (nelineárneho kavydla), tak, že budeme robiť skokové zmeny v okolí jedného pracovného bodu, a potom v okolí iného pracovného bodu.

Zvoľme pracovné body

$$u_{PB1} = 5 \quad y_{PB1} = 0,5348 \text{ [rad]} = 30,64 \text{ [deg]} \quad (82a)$$

$$u_{PB2} = 9 \quad y_{PB2} = 1,1616 \text{ [rad]} = 66,55 \text{ [deg]} \quad (82b)$$

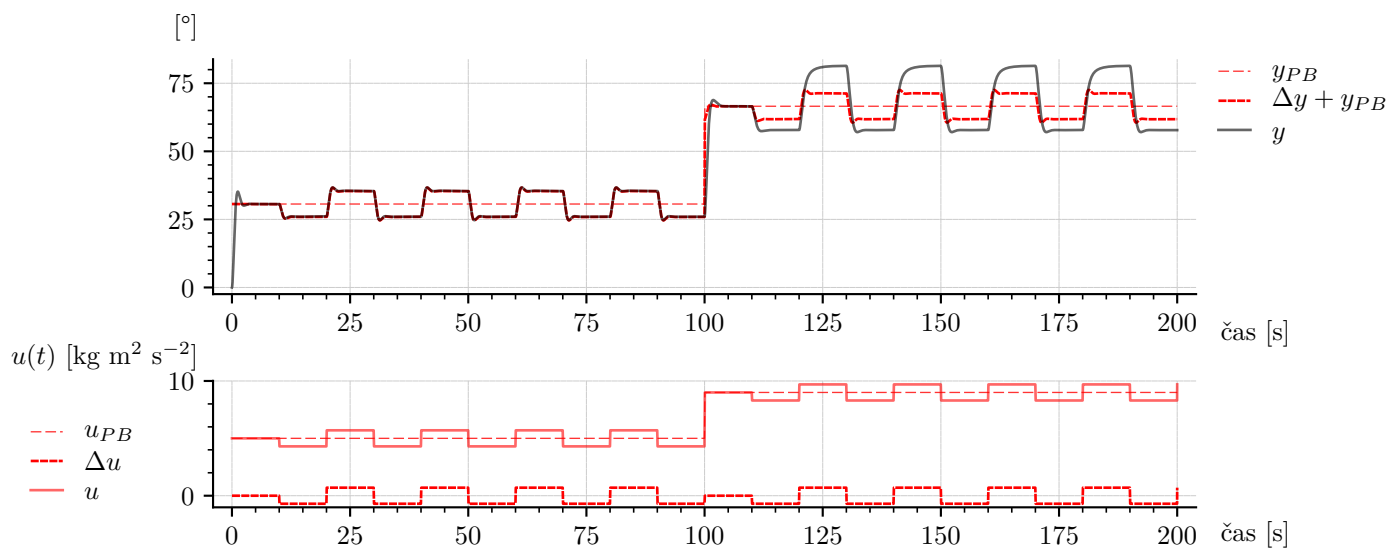
Pre prvý pracovný bod (u_{PB1}, y_{PB1}) vypočítajme aj parametre lineárneho modelu (platného len v okolí pracovného bodu)

$$k_p = 1 \quad (83a)$$

$$a_0 = 8.44 \quad (83b)$$

$$a_1 = 3.13 \quad (83c)$$

Simulujme teraz aj nelineárny systém, aj lineárny systém (celý čas s parametrami (83)), pričom vstupom nech sú skokové zmeny v okolí zvolených pracovných bodov. Výsledok je na obr. 13. V prvom pracovnom bode sa lineárny a nelineárny systém zhodujú ale v druhom už nie. Aby sa zhodoval, musel by lineárny model zmeniť (prispôbiť) parametre. Inými slovami, parametre modelu riadeného systému sa menia v závislosti od pracovného bodu. Preto je potrebné mať adaptívny riadiaci systém...



Obr. 13: Porovnanie výstupov lineárneho a nelineárneho systému v okolí rôznych pracovných bodov.

to be continued...