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	15
2.1.4	Bod štvrtý	15
2.1.5	Bod piaty	16
2.1.6	Bod šiesty	17
2.1.7	Bod siedmi	17
2.1.8	Bod ôsmy až desiaty	18

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:

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

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

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.

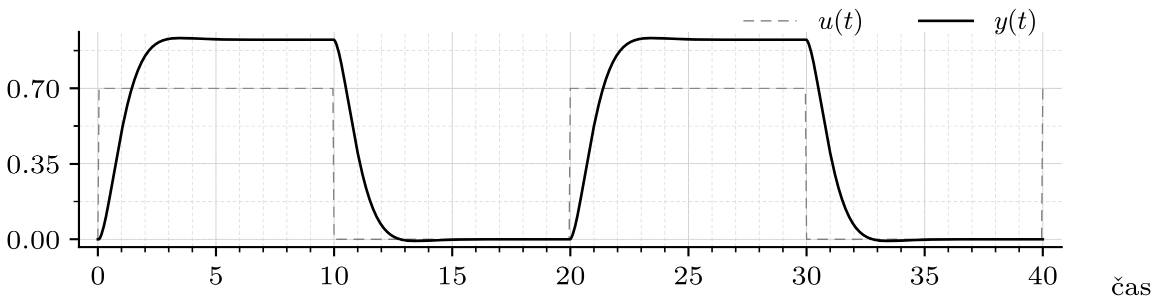
1.1 Úloha prvá, bod prvý

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

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

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:

```
import numpy as np

from scipy.integrate import odeint

# odeint je ODE solver a tu nim budeme riesit
# Linear Time Invariant System, a jeho zapis ako (maticova)
# sustava diferencialnych rovnic je - vid nasled. funkciu,
# pricom metoda "matmul" je samozrejme maticove nasobenie

def fcn_LTIS(x, t, A, b, u):

    dotx = np.matmul(A, x) + np.matmul(b, u)

```

```python
    return dotx

# Vytvorme teraz funkciu, ktora bude realizovat simulacnu schemu.
# Argumentami funkcie su parametre suvisiace s casom
# a vopred dane (zname) signaly.

def fcn_simSch1(t_start, T_s, finalIndex, sig_dummy_ext):

    #-------------------------------------------
    # Parametre riadeneho systemu

    A = np.array([[0, 1], [-2.6916, -2.7423]])
    b = np.array([[0], [1]])
    c = np.array([[3.5578], [0.1664]])

    #------------------------------------------------------
    # Do pola t_log sa bude logovat cas. Pole ma finalIndex
    # riadkov a 1 stlpec a je plne nul. Potom sa na prvu
    # poziciu (index 0) zapise hodnota t_start

    t_log = np.zeros([finalIndex, 1])
    t_log[0,:] = t_start

    #-------------------------------------------
    # Zaciatocne podmienky pre stavovy vektor nech su x_0
    # co je vektor rovnako velky ako vektor b

    x_0 = np.zeros(b.shape[0])

    # Stavovy vektor sa bude logovat do pola x_log s prislusnym
    # poctom stlpcov (detto y_log pre vyst. velicinu)

    x_log = np.zeros([finalIndex, len(x_0)])
    x_log[0,:] = x_0

    y_log = np.zeros([finalIndex, 1])
    y_log[0,:] = np.dot(c.T, x_0.reshape(-1,1))

    #--------------------------

    u_log = np.zeros([finalIndex, 1])
    u_log[0,:] = 0

    #------------------------------------------------------
    # Jedna iteracia for cyklu je posun v case o T_s.
    # ODE solver hlada riesenie pre casovy rozsah timespan.
    # Pred danou iteraciou pozname vsetko z predchadzajucej
    # iteracie (idx-1)
    # Pocas iteracie si _log-ujeme "vysledky"

    timespan = np.zeros(2)
    for idx in range(1, int(finalIndex)):

        timespan[0] = t_log[idx-1,:]
        timespan[1] = t_log[idx-1,:] + T_s

        t_log[idx,:] = timespan[-1]
        # posledny prvok v poly je zapisany (logovany)

        # ------------
        # solver odeint pouzije fcn_LTIS, zaciatocne podmienky
        # stavu su z predch. iteracie (x_log[idx-1,:]), riesi
        # na casovom rozsahu timespan a dalej (do fcn_LTIS) sa
        # posunu uvedene parametre/hodnoty (args)

        odeOut = odeint(fcn_LTIS,
                        x_log[idx-1,:],
                        timespan,
                        args=(A, b, u_log[idx-1,:])
                        )

        x_log[idx,:] = odeOut[-1,:]
        # odeOut obsahuje hodnoty stavu x pre cely timespan,
        # ale zapisujeme len poslednu hodnotu stavu x

        y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))
        # okrem stavu (stavovych velicin) chceme aj
```

```
        # vystupnu velicinu y

        # -------------------------

        u_log[idx,:] = sig_dummy_ext[idx,:]
        # v tejto simulacii len citame "externy" signal
        # a pouzivame ho ako vstup do systemu


    return [t_log, x_log, y_log, u_log, ]



# Vytvorme teraz vsetko potrebne pre "spustenie" simulacie,
# teda pre zavolanie prave vytvorenej funkcie fcn_simSch1.
# Hovorme tomu "nastavenie simulacie". Casove nastavenie:

sim_t_start = 0
sim_t_final = 40
sim_T_s = 0.05
sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)


# Dalej je potrebne vytvorit (vopred znamy) signal.
# Co sa tu deje ponechajme bez komentara, ale vysledkom
# je proste "signal" pouzitelny v simulacii...

period_time = 20
period_tab = np.array([[0, 0.7],
                       [10, 0],
                      ])

sig_vysl = np.zeros([sim_finalIndex, 1])

for period in range(int(sim_t_final/period_time) + 1):
    for idx in range( int((period*period_time)/sim_T_s), int((period*period_time + period_time)/sim_T_s)):
        lastValue = period_tab[:,1][(period_tab[:,0] + (period*period_time))<=idx*sim_T_s ][-1]
        try:
            sig_vysl[idx] = lastValue
        except:
            break

sig_dummy_ext = sig_vysl


# Teraz mozme "spustit" simulaciu:

t_log, x_log, y_log, u_log, = fcn_simSch1(
                                    sim_t_start,
                                    sim_T_s,
                                    sim_finalIndex,
                                    sig_dummy_ext,
                                    )

# Tu by mohlo byt kreslenie obrazku, ale to tu neuvedieme.
# 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

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

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 $n^* = n - m$.

$$Z_p(s) = s + 21,3809 \tag{5a}$$
$$R_p(s) = s^2 + 2,7423s + 2,6916 \tag{5b}$$
$$k_p = 0,1664 \tag{5c}$$

$Z_p(s)$ 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

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

Referenčný model je daný prenosovou funkciou v tvare

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

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

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

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

$$\Lambda(s) = \Lambda_0(s) Z_m(s) \tag{9}$$

a teda aj $\Lambda_0(s)$ je ľubovolný 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. Plynie 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 $\alpha(s) = \left[s^{2-2}\right]^\mathsf{T} = \left[s^0\right]^\mathsf{T} = \left[1\right]^\mathsf{T} = 1$.

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

Vektory $\Theta_1^\star \in \mathbb{R}^{n-1}$, $\Theta_2^\star \in \mathbb{R}^{n-1}$ budú mať v tomto prípade tiež len jeden prvok (sú dĺžky $n-1$) a teda len rovno píšme čísla $\Theta_1^\star$ a $\Theta_2^\star$. K tomu $\Theta_3^\star$, $\Theta_4^\star$ 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):

$$\Lambda(s) = Z_m(s) = s + \lambda \tag{10}$$

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

Práve sme určili zákon riadenia pre uvažovaný konkrétny prípad:

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

1.2.2 Bod druhý

Vypočítajte parametre $\Theta_1^\star$, $\Theta_2^\star$, $\Theta_3^\star$, $\Theta_4^\star$.

Pre prehľadnosť označme

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

a

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

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

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

$$\Theta_4^\star = \frac{k_m}{k_p} \tag{14b}$$

Vypočítajme:

```
kp = 0.1664
b0 = 21.3809
a1 = 2.7423
a0 = 2.6916

km = 1
b0m = 3
a1m = 3.5
a0m = 3

M = np.array([[-1,  0, -kp],
              [-a1, -kp, -(kp*b0m + kp*b0)],
              [-a0, -kp*b0, -kp*b0*b0m],
             ])

N = np.array([[a1m + b0 - b0m - a1],
              [a0m + b0*a1m - a1*b0m - a0],
              [b0*a0m - a0*b0m],
             ])

Theta_c = np.linalg.solve(M,N)
# Theta_c je vektor obsahujuci Theta_1, Theta_2 a Theta_3

Theta_4 = km/kp
```

Výpis kódu 2: Výpočet ideálnych parametrov zákona riadenia

Teda:

$$\begin{bmatrix} \Theta_1^\star \\ \Theta_2^\star \\ \Theta_3^\star \end{bmatrix} = \begin{bmatrix} -18,3809 \\ 11,8071 \\ -4,5535 \end{bmatrix} \tag{15a}$$

$$\Theta_4^\star = 6,0096 \tag{15b}$$

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 dovolíme 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

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

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

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

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

kde sme zaviedli pomocnú veličinu z.

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

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

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

$$x_1(t) = z(t) \tag{20}$$

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

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

Druhú stavovú veličinu zvoľme

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

a teda

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

V tomto bode môžeme ľahko písať

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

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

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

takže (23) je

$$\dot{x}_2(t) = -a_1 x_2(t) - a_0 x_1(t) + u(t) \tag{26}$$

a to je druhá diferenciálna rovnica...

Obe rovnice spolu:

$$\dot{x}_1(t) = x_2(t) \tag{27}$$
$$\dot{x}_2(t) = -a_1 x_2(t) - a_0 x_1(t) + u(t) \tag{28}$$

A v maticovom zápise:

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

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

$$y(t) = b_1\dot{z}(t) + b_0 z(t) \tag{30}$$

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

$$y(t) = b_1 x_2(t) + b_0 x_1(t) \tag{31}$$

alebo v maticovom tvare

$$y(t) = \begin{bmatrix}b_0 & b_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} \tag{32}$$

Celý systém s novo zavedenými stavovými veličinami teda je v tvare

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -a_0 & -a_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}u(t) \tag{33}$$
$$y(t) = \begin{bmatrix}b_0 & b_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} \tag{34}$$

a ak označíme stavový vektor ako $x(t) = \begin{bmatrix}x_1(t) & x_2(t)\end{bmatrix}^\mathsf{T}$, potom je systém v známom tvare

$$\dot{x}(t) = Ax(t) + bu(t) \tag{35a}$$
$$y(t) = c^\mathsf{T} x(t) \tag{35b}$$

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.

Potrebujeme 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()`:

```
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“):

```
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

```
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. Dovolíme si preto písať

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

kde sme zmiešali písanie operátora s a času t.

Autor pozná takýto (alebo podobný) spôsob zápisu práve z literatúry o klasickom adaptívnom riadení[1]. Tu nie je cieľom porušovať matematické vzťahy a podobne. Tu je cieľom zjednodušene 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:

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

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

[1] viď 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 $\Theta_1^\star$. Uvedený samostatný dynamický systém je možné vyjadriť aj opisom v stavovom priestore. Pre tento konkrétny príklad

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

$$y_{\nu_1}(t) = \Theta_1^\star \nu_1(t) \tag{38b}$$

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

$$\Theta_1^\star \nu_1(t) \tag{39}$$

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

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

$$\Theta_2^\star \nu_2(t) \tag{40}$$

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

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

Zákon riadenia (36) je teda možné zapísať v tvare

$$u(t) = \Theta_1^\star \nu_1(t) + \Theta_2^\star \nu_2(t) + \Theta_3^\star y(t) + \Theta_4^\star r(t) \tag{42}$$

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

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

a označme

$$u(t) = \Theta^{\star\mathsf{T}} \omega \tag{44}$$

Vieme vyrobiť všetky signály v signálnom vektore ω? Vieme. Poznáme parametre vo vektore $\Theta^\star$? 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 akéhokoľvek) 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

$$u(t) = \Theta_c^{\star\mathsf{T}} DX(t) + \Theta_4^\star r(t) \tag{45}$$

kde $\Theta_c^\star = \begin{bmatrix}\Theta_3^\star & \Theta_1^{\star\mathsf{T}} & \Theta_2^{\star\mathsf{T}}\end{bmatrix}^\mathsf{T}$; $\Theta_4^\star$ sú parametre zákona riadenia a maticu

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

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

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

Všimnime si ale, že

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

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

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

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

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

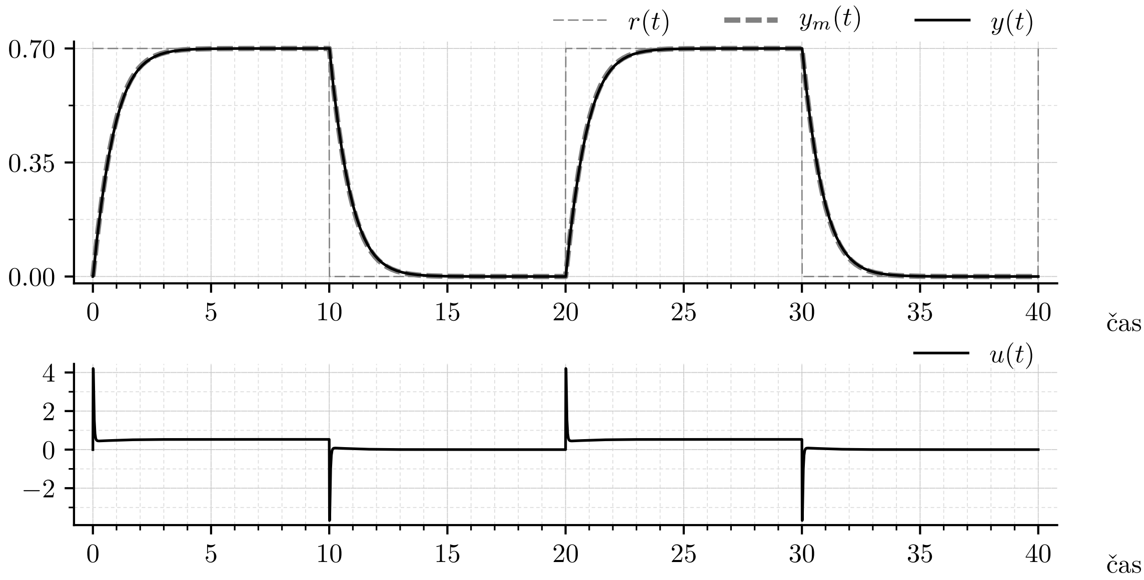
$$\Theta = \begin{bmatrix} \Theta_3^\star \\ \Theta_1^\star \\ \Theta_2^\star \\ \Theta_4^\star \end{bmatrix} \tag{49}$$

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.

```python
import numpy as np

from scipy.integrate import odeint

def fcn_LTIS(x, t, A, b, u):
    dotx = np.matmul(A, x) + np.matmul(b, u)
    return dotx


def fcn_simSch2(t_start, T_s, finalIndex, sig_dummy_ext):

    #------------------------------------------------
    # Riadeny system

    A = np.array([[0, 1], [-2.6916, -2.7423]])
    b = np.array([[0], [1]])
```



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

```
    c = np.array([[3.5578], [0.1664]])

    #------------------------------
    # Referencny model

    A_m = np.array([[0, 1], [-3.0, -3.5]])
    b_m = np.array([[0], [1]])
    c_m = np.array([[3], [1]])

    #------------------------------
    # Pomocne filtre

    Lambda_pom = np.array([[-3]])
    q_pom = np.array([[1]])

    #-------------------------------------------------------
    t_log = np.zeros([finalIndex, 1])
    t_log[0,:] = t_start

    #-------------------------------------------
    x_0 = np.zeros(b.shape[0])

    x_log = np.zeros([finalIndex, len(x_0)])
    x_log[0,:] = x_0

    y_log = np.zeros([finalIndex, 1])
    y_log[0,:] = np.dot(c.T, x_0.reshape(-1,1))

    #------------------------------
    x_m_0 = np.zeros(b_m.shape[0])

    x_m_log = np.zeros([finalIndex, len(x_m_0)])
    x_m_log[0,:] = x_m_0

    y_m_log = np.zeros([finalIndex, 1])
    y_m_log[0,:] = np.dot(c_m.T, x_m_0.reshape(-1,1))

    #------------------------------
    nu1_log = np.zeros([finalIndex, q_pom.shape[0]])
    nu2_log = np.zeros([finalIndex, q_pom.shape[0]])

    #------------------------------
    u_log = np.zeros([finalIndex, 1])
    u_log[0,:] = 0

    #-------------------------------------------------------
```

```
    timespan = np.zeros(2)
    for idx in range(1, int(finalIndex)):

        timespan[0] = t_log[idx-1,:]
        timespan[1] = t_log[idx-1,:] + T_s

        t_log[idx,:] = timespan[-1]

        # ------------

        odeOut = odeint(fcn_LTIS,
                        x_log[idx-1,:],
                        timespan,
                        args=(A, b, u_log[idx-1,:])
                        )

        x_log[idx,:] = odeOut[-1,:]
        y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))

        # -------------------------
        # Referencny model:

        ref_sig = sig_dummy_ext[idx-1, :]

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

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

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

        # -------------------------
        # Pomocne filtre:

        dotnu1 = fcn_LTIS(nu1_log[idx-1,:], 0, Lambda_pom, q_pom, u_log[idx-1,:])
        nu1_log[idx,:] = nu1_log[idx-1,:] + dotnu1 * T_s

        dotnu2 = fcn_LTIS(nu2_log[idx-1,:], 0, Lambda_pom, q_pom, y_log[idx-1,:])
        nu2_log[idx,:] = nu2_log[idx-1,:] + dotnu2 * T_s

        # -------------------------
        # Vektor omega:

        omega = np.array([y_log[idx,:],
                          nu1_log[idx-1,:].reshape(-1,1),
                          nu2_log[idx-1,:].reshape(-1,1),
                          ref_sig,
                        ])

        # -------------------------
        # Vektor Theta:

        Theta = np.array([[-4.5535],
                          [-18.3809],
                          [11.8071],
                          [6.0096],
                        ])

        # -------------------------

        u_log[idx,:] = np.dot(Theta.T, omega)[0]

    return [t_log, x_log, y_log, u_log, y_m_log]

# --------------------------------------------------
# Nastavenie simulacie

sim_t_start = 0
sim_t_final = 40
sim_T_s = 0.01
sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)

# Preddefinovany signal (pouzity ako referencny signal)

period_time = 20
```

```python
period_tab = np.array([[0, 0.7],
                       [10, 0],
                      ])

sig_vysl = np.zeros([sim_finalIndex, 1])

for period in range(int(sim_t_final/period_time) + 1):
    for idx in range( int((period*period_time)/sim_T_s), int((
   period*period_time + period_time)/sim_T_s)):
        lastValue = period_tab[:,1][(period_tab[:,0] + (period*
   period_time))<=idx*sim_T_s ][-1]
        try:
            sig_vysl[idx] = lastValue
        except:
            break

sig_dummy_ext = sig_vysl

# Spustenie simulacie

t_log, x_log, y_log, u_log, y_m_log = fcn_simSch2(
    sim_t_start,
    sim_T_s,
    sim_finalIndex,
    sig_dummy_ext,
    )

# Tu by bolo kreslenie obrazkov...
```

Výpis kódu 6: Simulačná schéma uzavretého regulačného obvodu

2 Časť druhá

2.1 Úloha prvá

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

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

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 $\Theta_1^\star$, $\Theta_2^\star$, $\Theta_3^\star$ a $\Theta_4^\star$ (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:

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

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 striktne pozitívne reálna (SPR) prenosová funkcia.

Máme prenosovú funkciu

$$W_m(s) = \frac{s+3}{s^2+3,5s+3} \tag{52}$$

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 $W_m(s) \rightarrow W_m(j\omega)$:

$$W_m(j\omega) = \frac{j\omega+3}{(j\omega)^2+3,5j\omega+3} \tag{53}$$

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

$$\begin{aligned} W_m(j\omega) &= \frac{j\omega+3}{(j\omega)^2+3,5j\omega+3} && \text{(54a)} \\ &= \frac{j\omega+3}{-\omega^2+3,5j\omega+3} && \text{(54b)} \\ &= \frac{j\omega+3}{(3-\omega^2)+j3,5\omega} && \text{(54c)} \end{aligned}$$

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:

$$\begin{aligned} W_m(j\omega) &= \frac{(j\omega+3)}{((3-\omega^2)+j3,5\omega)} \frac{\left((3-\omega^2)-j3,5\omega\right)}{((3-\omega^2)-j3,5\omega)} && \text{(55a)} \\ &= \frac{(j\omega+3)\left((3-\omega^2)-j3,5\omega\right)}{(3-\omega^2)^2+(3,5\omega)^2} && \text{(55b)} \\ &= \frac{9-3\omega^2-j10,5\omega+j3\omega-j\omega^3+3,5\omega^2}{(3-\omega^2)^2+(3,5\omega)^2} && \text{(55c)} \end{aligned}$$

Reálna časť z toho je

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

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

Napíšte 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

$$e_1(t) = [W_m(s)] \frac{1}{\Theta_4^\star} \left(\theta^\mathsf{T}(t)\ \omega(t)\right) \tag{57}$$

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, $\Theta_4^\star$ je jeden y ideálnych parametrov zákona riadenia, $\theta^\mathsf{T}(t)$ 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 odvodenia sa uvažuje len všeob. funkcia f) zvoľte 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

$$\dot{e}(t) = A_c e(t) + \overline{B}_c \frac{1}{\Theta_4^\star} \left(\theta^\mathsf{T}(t)\omega(t)\right) \tag{58a}$$

$$e_1(t) = C_c^\mathsf{T} e(t) \tag{58b}$$

pričom $W_m(s) = C_c^\mathsf{T} \left(sI - A_c\right)^{-1} \overline{B}_c$ 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 diferenciálny rovníc v tvare

$$\dot{e}(t) = A_c e(t) + \overline{B}_c \frac{1}{\Theta_4^\star} \left(\theta^\mathsf{T}(t)\omega(t)\right) \tag{59a}$$

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

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 dovolíme nepísať signály ako funkcie času, teda napr. píšeme len e nie $e(t)$)

$$V = e^\mathsf{T} P e + \left|\frac{1}{\Theta_4^\star}\right| \theta^\mathsf{T} \Gamma^{-1} \theta \tag{60}$$

Cesta potom vedie do bodu, kde máme

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

a chceme zabezpečiť aby časová derivácia $\dot{V}$ bola záporne definitná. Na pravej strane rovnice (61), člen $e^\mathsf{T}\left(-Q\right) e$ 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:

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

$$2 \left|\frac{1}{\Theta_4^\star}\right| \theta^\mathsf{T} G^{-1} \dot{\theta} = -2 \left(e^\mathsf{T} P \overline{B}_c\right) \frac{1}{\Theta_4^\star} \Theta^\mathsf{T} \omega \tag{62b}$$

$$\left|\frac{1}{\Theta_4^\star}\right| \Gamma^{-1} \dot{\theta} = -\left(e^\mathsf{T} P \overline{B}_c\right) \texttt{sign}\left(\frac{1}{\Theta_4^\star}\right) \left|\frac{1}{\Theta_4^\star}\right| \omega \tag{62c}$$

$$\Gamma^{-1} \dot{\theta} = -\texttt{sign}\left(\frac{1}{\Theta_4^\star}\right) \left(e^\mathsf{T} P \overline{B}_c\right) \omega \tag{62d}$$

$$\dot{\theta} = -\texttt{sign}\left(\frac{1}{\Theta_4^\star}\right) \left(e^\mathsf{T} P \overline{B}_c\right) \Gamma \omega \tag{62e}$$

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 $W_m(s) = C_c^\mathsf{T} \left(sI - A_c \right)^{-1} \overline{B}_c$ a s využitím tejto reprezentácie referenčného modelu je možné využiť Meyerovu-Kalmanovu-Yakubovichovu Lemmu (vetu). Podľa nej platí

$$A_c^\mathsf{T} P + P A_c = -Q$$
$$P\overline{B}_c = C_c$$

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

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

$$\dot{\theta} = -\mathtt{sign}\left(\frac{1}{\Theta_4^\star}\right) \left(e^\mathsf{T} P \overline{B}_c\right) \Gamma \omega \tag{63a}$$

$$\dot{\theta} = -\mathtt{sign}\left(\frac{1}{\Theta_4^\star}\right) \left(e^\mathsf{T} C_c\right) \Gamma \omega \tag{63b}$$

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

$$\dot{\theta} = -\mathtt{sign}\left(\frac{1}{\Theta_4^\star}\right) e_1 \Gamma \omega \tag{64}$$

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

$$\dot{\Theta}(t) = -\mathtt{sign}\left(\frac{1}{\Theta_4^\star}\right) e_1(t)\ \Gamma\ \omega(t) \tag{65}$$

Pripomeňme, že signálny vektor zákona riadenia v tomto prípade je

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

takže zákon adaptácie detailnejšie napísaný je

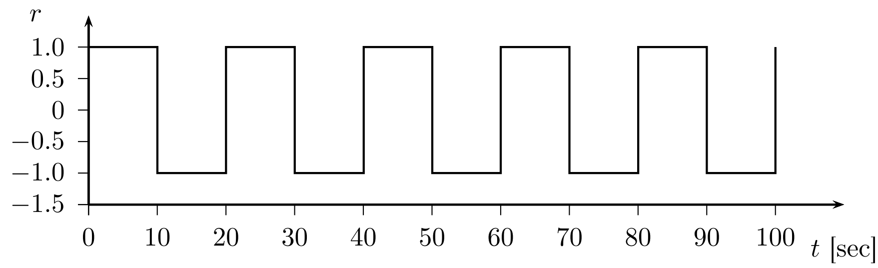
$$\dot{\Theta}(t) = -\mathtt{sign}\left(\frac{1}{\Theta_4^\star}\right) e_1(t) \begin{bmatrix} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & 0 \\ 0 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & \gamma_4 \end{bmatrix} \begin{bmatrix} y(t) \\ \nu_1(t) \\ \nu_2(t) \\ r(t) \end{bmatrix} \tag{67}$$

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

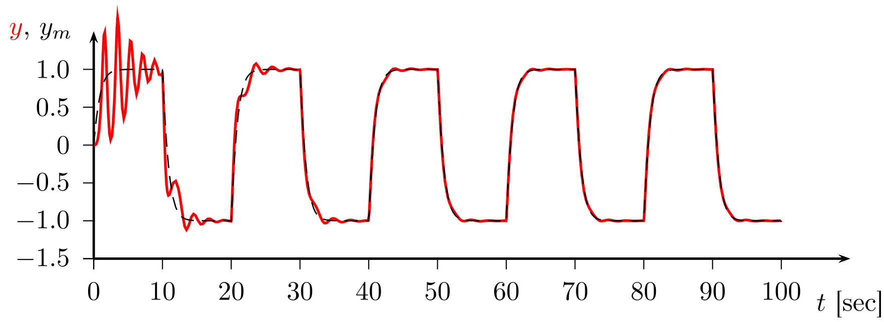
2.1.7 Bod siedmi

Zvoľte Γ (jednoducho, zvoľte všetky ľubovolne 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



Obr. 3: Referenčný sigál r



Obr. 4: Výsledok simulácie

rýchlosť adaptácie jednotlivých prvkov vektora adaptovaných parametrov. Uvažujme teda o matici Γ v tvare

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

V tomto prípade máme vektor adaptovaných parametrov v tvare

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

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...

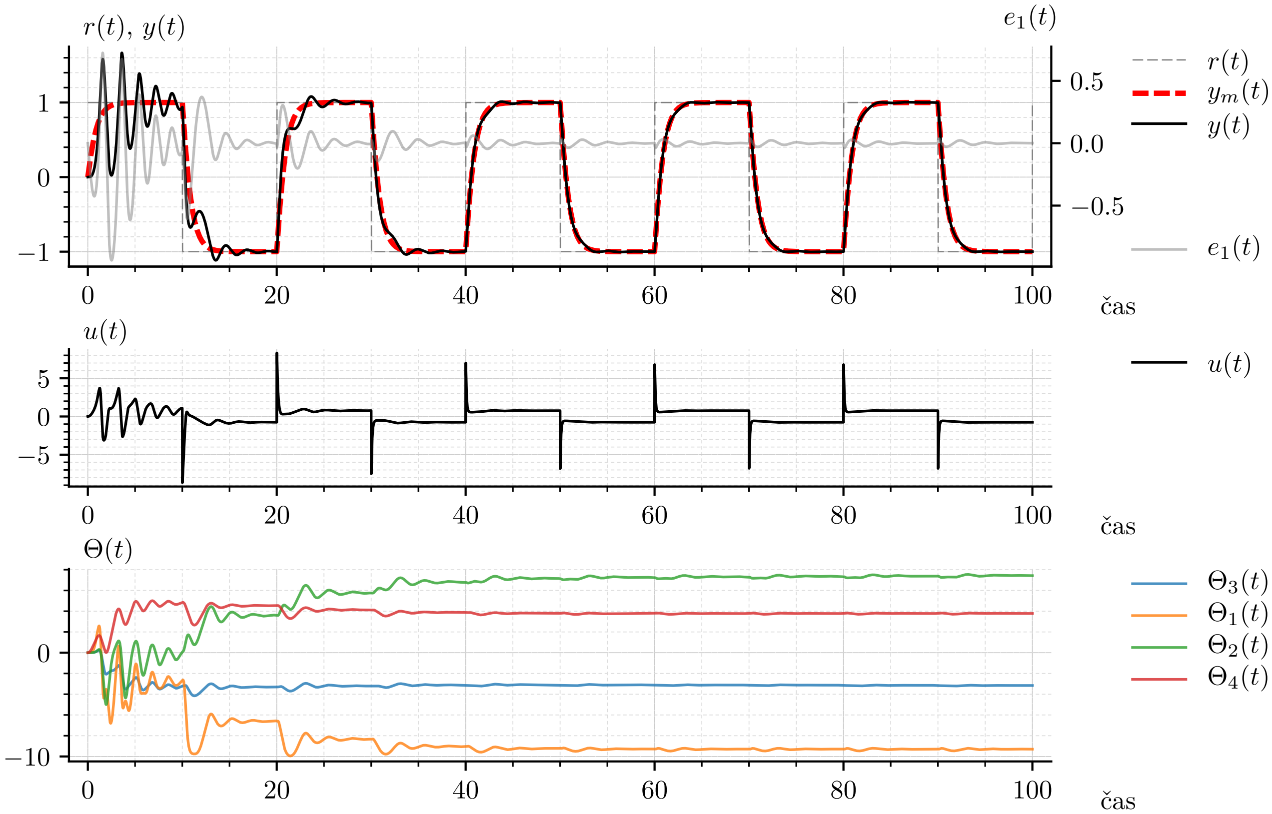
2.1.8 Bod ôsmy až desiaty

- Začiatočné 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.

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 `Theta`).

Výsledky simulácie pre uvedené zadanie sú na obr. 5.

```
import numpy as np

from scipy.integrate import odeint

```



Obr. 5: Výsledok simulácie

```
def fcn_LTIS(x, t, A, b, u):
    dotx = np.matmul(A, x) + np.matmul(b, u)
    return dotx


def fcn_simSch3(t_start, T_s, finalIndex, sig_dummy_ext):

    #-------------------------------------------
    # Riadeny system

    A = np.array([[0, 1], [-2.6916, -2.7423]])
    b = np.array([[0], [1]])
    c = np.array([[3.5578], [0.1664]])

    #----------------------------
    # Referencny model

    A_m = np.array([[0, 1], [-3.0, -3.5]])
    b_m = np.array([[0], [1]])
    c_m = np.array([[3], [1]])

    #----------------------------
    # Pomocne filtre

    Lambda_pom = np.array([[-3]])
    q_pom = np.array([[1]])

    #--------------------------------------------------------
    t_log = np.zeros([finalIndex, 1])
    t_log[0,:] = t_start

    #-------------------------------------------
    x_0 = np.zeros(b.shape[0])

    x_log = np.zeros([finalIndex, len(x_0)])
    x_log[0,:] = x_0

    y_log = np.zeros([finalIndex, 1])
    y_log[0,:] = np.dot(c.T, x_0.reshape(-1,1))
```

```python
    #------------------------------
    x_m_0 = np.zeros(b_m.shape[0])

    x_m_log = np.zeros([finalIndex, len(x_m_0)])
    x_m_log[0,:] = x_m_0

    y_m_log = np.zeros([finalIndex, 1])
    y_m_log[0,:] = np.dot(c_m.T, x_m_0.reshape(-1,1))

    #------------------------------
    nu1_log = np.zeros([finalIndex, q_pom.shape[0]])
    nu2_log = np.zeros([finalIndex, q_pom.shape[0]])

    #------------------------------
    Theta_log = np.zeros([finalIndex, 4])

    #------------------------------
    u_log = np.zeros([finalIndex, 1])
    u_log[0,:] = 0

    #---------------------------------------------------------------
    timespan = np.zeros(2)
    for idx in range(1, int(finalIndex)):

        timespan[0] = t_log[idx-1,:]
        timespan[1] = t_log[idx-1,:] + T_s

        t_log[idx,:] = timespan[-1]

        # ------------

        odeOut = odeint(fcn_LTIS,
                        x_log[idx-1,:],
                        timespan,
                        args=(A, b, u_log[idx-1,:])
                        )

        x_log[idx,:] = odeOut[-1,:]
        y_log[idx,:] = np.dot(c.T, x_log[idx,:].reshape(-1,1))

        # -------------------------
        # Referencny model:

        ref_sig = sig_dummy_ext[idx-1, :]

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

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

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

        # -------------------------
        # Adaptacna odchylka

        adaptError = y_log[idx-1,:] - y_m_log[idx-1,:]

        # -------------------------
        # Pomocne filtre

        dotnu1 = fcn_LTIS(nu1_log[idx-1,:], 0, Lambda_pom, q_pom, u_log[idx-1,:])
        nu1_log[idx,:] = nu1_log[idx-1,:] + dotnu1 * T_s

        dotnu2 = fcn_LTIS(nu2_log[idx-1,:], 0, Lambda_pom, q_pom, y_log[idx-1,:])
        nu2_log[idx,:] = nu2_log[idx-1,:] + dotnu2 * T_s

        # -------------------------
        # Vektor omega:

        omega = np.array([y_log[idx,:],
                          nu1_log[idx-1,:].reshape(-1,1),
                          nu2_log[idx-1,:].reshape(-1,1),
                          ref_sig,
                        ])

```

```
        # ------------------------------
        # Zakon adaptacie

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

        dotTheta = - np.matmul(Gamma, omega) * adaptError

        Theta_log[idx,:] = Theta_log[idx-1,:] + dotTheta.reshape(1,-1) * T_s

        # ------------------------------
        # Zakon ZakonRiadenia

        u_log[idx,:] = np.dot(Theta_log[idx-1,:].T, omega)[0]

    return [t_log, x_log, y_log, u_log, y_m_log, Theta_log]

# ------------------------------------------------------------
# Nastavenie simulacie

sim_t_start = 0
sim_t_final = 100
sim_T_s = 0.01
sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)

# Preddefinovany signal (pouzity ako referencny signal)

period_time = 20
period_tab = np.array([[0, 1],
                        [10, -1],
                      ])

sig_vysl = np.zeros([sim_finalIndex, 1])

for period in range(int(sim_t_final/period_time) + 1):
    for idx in range( int((period*period_time)/sim_T_s), int((period*period_time + period_time)/sim_T_s)):
        lastValue = period_tab[:,1][(period_tab[:,0] + (period*period_time))<=idx*sim_T_s ][-1]
        try:
            sig_vysl[idx] = lastValue
        except:
            break

sig_dummy_ext = sig_vysl

# Spustenie simulacie

t_log, x_log, y_log, u_log, y_m_log, Theta_log = fcn_simSch3(
    sim_t_start,
    sim_T_s,
    sim_finalIndex,
    sig_dummy_ext,
    )

# Tu by bolo kreslenie obrazkov...
```

Výpis kódu 7: Simulačná schéma adaptívneho riadiaceho systému