Organizácia predmetu

Adaptívne riadenie (LS, ak.r. 2020/2021)

Cieľ predmetu:
Študenti po absolvovaní predmetu získajú vedomosti o najvýznamnejších metódach a algoritmoch používaných v oblasti adaptívneho riadenia procesov. Absolventi predmetu získajú vedomosti týkajúce sa odvodenia a analýzy vlastností vybraných algoritmov priameho adaptívneho riadenia a nepriameho adaptívneho riadenia. Získajú poznatky o základných princípoch vybraných heuristických adaptívnych regulátorov, komerčných adaptívnych regulátorov, a princípoch využitia adaptácie pri fuzzy riadiacich systémoch.

Zodpovedný za predmet: prof. Ing. Ján Murgaš, PhD.

Predmet patrí medzi povinné predmety a študent po absolvovaní získa 7 kreditov. Týždenný rozsah predmetu: prednášky: 2 h, cvičenia: 2 h

Predmet zabezpečuje:

Ing. Marián Tárník, PhD. (prednášky, cvičenia)

Google Classroom v rámci STUBA GSuite:

https://classroom.google.com/c/MjcOMTgOMDkxOTQ5?cjc=tmbis5s

Kód triedy: `tmbis5s`

GitHub:

https://github.com/PracovnyBod/ADRIA

Podmienky absolvovania predmetu:

1. Aktívna účasť na vyučovacom procese.
2. Počas semestra je možné získať max. 60 bodov, pričom pre splnenie podmienok pre vykonanie skúšky je potrebných 33,6 bodu.
3. Je potrebná účasť na záverečnej skúške, je možné získať max. 40 bodov.

Priebežné hodnotenie študentov dennej prezenčnej formy štúdia počas semestra:

- Priebežná práca/účasť na cvičeniach: 12 bodov
- Vypracovanie referátov (zadaní): 48 bodov, konkrétne:
  - Referát prvý: 13 bodov
  - Referát druhý: 15 bodov
  - Referát tretí: 20 bodov

Priebežné hodnotenie študentov dennej dištančnej formy štúdia počas semestra:

- Vypracovanie referátov (zadaní): 60 bodov, konkrétne:
  - Referát nultý: 12 bodov
  - Referát prvý: 13 bodov
  - Referát druhý: 15 bodov
  - Referát tretí: 20 bodov

Harmonogram semestra pre študentov dennej prezenčnej formy štúdia

Týždeň	Prednáška	Cvičenie
1.	Úvod, stabilita systémov, adaptívna stabilizácia.	Adaptívna stabilizácia, priestor pre otázky...
2.	Samonastavujúci sa regulátor: rekurzívna metóda najmenších štvorcov. (a ukážky STR pre ďalšie príklady)	Samonastavujúci sa regulátor: rekurzívna metóda najmenších štvorcov (reprodukcia vzorového príkladu). Implementácia pomocou rôznych nástrojov
3.	Samonastavujúci sa regulátor (info k cv), riadenie s referenčným modelom.	Samonastavujúci sa regulátor: metóda rozmiestňovania pólov; dokončenie adapt. riadiaceho systému (reprodukcia vzorového príkladu). Priestor pre otázky (a ukážky STR pre ďalšie príklady).
4.	MRAC gradientný.	MRAC gradientný. Referát prvý, odovzdanie do troch týždňov (13b).
5.	MRAC stavový.	MRAC stavový (viac menej pokračovanie prednášky).
6.	MRAC stavový a zovšeobecnenie riadenia s referenčným modelom - MRC problém.	MRAC stavový (priestor pre otázky), Referát druhý, odovzdanie do troch týždňov (15b).
7.	MRAC vstupno-výstupný pre $n^\star = 1$	Extra priestor pre otázky, prípadné dokončenie predchádzajúcich úloh
8.	MRAC vstupno-výstupný pre $n^\star = 1$ a pre $n^\star = 2$.	MRAC vstupno-výstupný pre $n^\star = 1$, Referát tretí, odovzdanie do štyroch týždňov (20b).
9.	MRAC vstupno-výstupný pre $n^\star = 2$ - príklady.	Extra priestor pre otázky, prípadné dokončenie predchádzajúcich úloh
10.	Rôzne.	Extra priestor pre otázky, prípadné dokončenie predchádzajúcich úloh, AR pre kyvadlo (kyvadlo ako riadený systém)
11.	Zhrnutie pred skúškou.	Zhrnutie pred skúškou.
12.	Rezerva	Rezerva
13.	nič	nič

Harmonogram semestra pre študentov dennej dištančnej formy štúdia

Týždeň	Konzultácia
1.	
2.	Konzultácia k téme Samonastavujúci sa regulátor, Referát nultý (12b)
3.	
4.	Konzultácia k téme MRAC gradientný
5.	
6.	Konzultácia k téme MRAC stavový
7.	
8.	Konzultácia k rôznym predchádzajúcim témam
9.	
10.	Konzultácia k téme MRAC vstupno-výstupný
11.	
12.	Konzultácia k záverečnej skúške

Literatúra

[1] K. J. Åström and R. M. Murray. *Feedback Systems*. Princeton University Press, 2008.

[2] K.J. Åström and B. Wittenmark. *Adaptive Cotrol, 2nd edition*. Addison-Wesley, 1995.

[3] H. Butler. *Model Reference Adaptive Control: From theory to practice*. Prentice Hall International (UK) Ltd., 1992.

[4] P. Ioannou and B. Fidan. *Adaptive Control Tutorial*. Society for Industrial and Applied Mathematics, USA., 2006.

[5] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, Inc, 1996.

[6] Lennart Ljung. *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[7] R. Monopoli. Model reference adaptive control with an augmented error signal. *IEEE Transactions on Automatic Control*, 19(5):474 – 484, oct 1974.

[8] J. Murgaš and I. Hejda. *Adaptívne riadenie technologických procesov*. Slovenská technická univerzita v Bratislave, 1993.

[9] K. S. Narendra and A. M. Annaswamy. *Stable adaptive systems*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[10] K. S. Narendra, Y.-H. Lin, and L. S. Valavani. Stable adaptive controller design, part ii: Proof of stability. *IEEE Transactions on Automatic Control*, 25(3):440 – 448, jun 1980.

[11] K. S. Narendra and L. S. Valavani. Stable adaptive controller design—direct control. *IEEE Transactions on Automatic Control*, 23(4):570 – 583, aug 1978.

[12] K. M. Passino and S. Yurkovich. *Fuzzy Control*. Addison Wesley Longman, Inc., 1998.

[13] S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall., 1994.

[14] G. Tao. *Adaptive control design and analysis*. John Wiley & Sons, Inc., 2003.

[15] M. Tárník. Direct model reference adaptive control of small laboratory dc motor. *posterus.sk*, 4(1), 2011.

Veľmi stručný prehľad Adaptívneho riadenia

Obsah

1 História a nedávna história

Názvy *Adaptívny systém* a *Adaptívne riadenie* boli v publikáciách prvý krát použité okolo roku 1950. Motiváciou pre vývoj adaptívnych regulátorov bol návrh autopilota pre vysokovýkonné experimentálne lietadlá na začiatku 50-tych rokov.

Lietadlo operuje vo veľkom rozsahu rýchlostí a výšok. Komplexná dynamika lietadla môže byť pre daný pracovný bod (rýchlosť, výška) aproximovaná lineárnym modelom v tvare

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1a}$$
$$y(t) = Cx(t) + Du(t) \tag{1b}$$

pričom A, B, C a D sú funkciou pracovného bodu, ktorý sa mení v čase (teda A, B, C a D sú funkciou času).

Zjednodušene, tá istá výchylka „smerovky“ (alebo kormidla na lodi) spôsobí pri rôznych rýchlostiach iný moment sily (ktorý otáča lietadlo alebo loď). Teda mení sa parameter riadeného systému, ktorý môže byť opísaný napríklad prenosovou funkciou, ktorej vstupom je výchylka kormidla a výstupom je kurz (natočenie) lode.

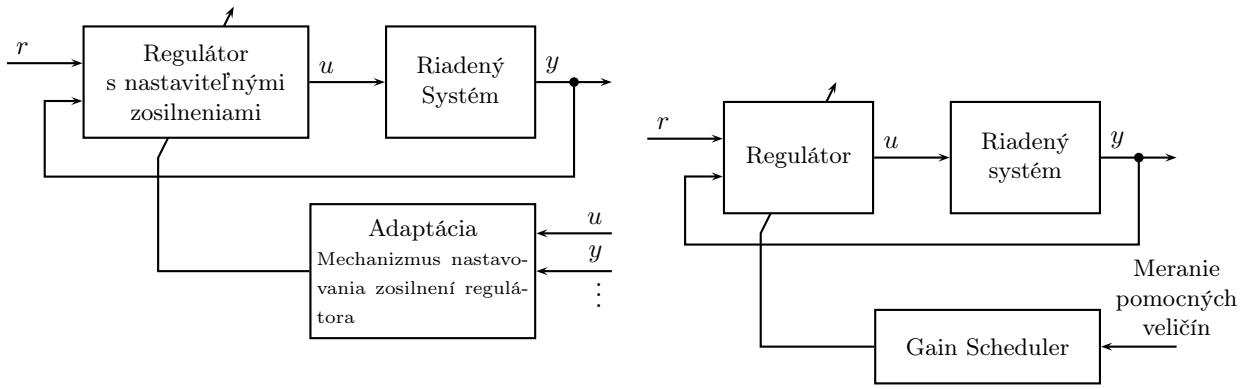
Toľko veľmi stručné zhrnutie inak veľmi rozsiahlej histórie Adaptívneho riadenia.

Ako príklad súčasného/nedávneho využitia Adaptívneho riadenia možno uviesť opäť systémy riadenia letu. V správe *The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), IEEE Control Systems Society, 2011, available at www.ieeecss.org.* je uvedený takýto príklad.

Názvom *The Joint Direct Attack Munition* sa označuje navádzací systém, ktorý konvertuje nenavádzané bomby na *smart* muníciu schopnú činnosti v každom počasí. V tomto systéme sa uplatňuje robustné a adaptívne riadenie. Navzájom sa dopĺňajú a kombinujú.

Tento príklad uvádzame najmä preto, aby sme čitateľa upozornili na uvedenú správu. Medzičasom (2014) je dostupná aj druhá edícia tejto správy, dostupné na: http://ieeecss.org/impact-control-technology-2nd-edition.

Autor si na začiatku roka 2021 uvedomuje možnú zmätenosť čitateľa ak sa za súčasnosť, či nedávnu minulosť označujú roky 2011, prípadne 2014. Ide však o obdobie, keď vznikal tento učebný text. Adaptívne riadenie, ako pojem, sa možno vytráca,



(a) Základá schéma adaptívneho riadenia

(b) Schéma Gain Scheduling

Obr. 1: Základá schéma adaptívneho riadenia a schéma Gain Scheduling

avšak princípy, najmä v spojení s identifikáciou systémov vo všeobecnosti, sú viac ako aktuálne. Nájdeme ich prezlečené alebo zaradené pod pojmy ako je strojové učenie (machine learning), spracovanie signálov (signal processing) alebo priebežná identifikácia systémov.

Podobne, ak sa čitateľovi zdá odporúčaná literatúra staršieho dáta, nech má na pamäti, že ide o relatívne novinky v čase poslednej rozsiahlejšej aktualizácie tohto učebného materiálu – predovšetkým kniha [1] z roku 2006. Ostatné sa dajú považovať za klasické učebnice tohto predmete a do poredia tu dávame knihu [2] najmä pre skutočnosť, že je voľne dostupná tu: https://viterbi-web.usc.edu/~ioannou/RobustAdaptiveBook95pdf/Robust_Adaptive_Control.pdf.

2 Schémy Adaptívneho riadenia

Túto časť možno chápať aj ako veľmi stručné[1] rozdelenie Adaptívneho riadenia.

2.1 Základná schéma AR

Priebeh výstupnej veličiny obsahuje informáciu o stave x systému a o parametroch systému. Preto by nejaký sofistikovaný spätnoväzbový riadiaci systém mal byť schopný vysporiadať sa so zmenami v riadenom systéme a zabezpečiť dosiahnutie požadovaného cieľa riadenia. Tieto myšlienky viedli k návrhu spätnoväzbovej štruktúry riadenia, ktorá je základom adaptívneho riadenia.

V štandardnom regulačnom obvode (v uzavretom regulačnom obvode – URO) je použitý regulátor, ktorého parametre (zosilnenia) je možné meniť (aj v počas činnosti). K uzavretému obvodu je pridaný mechanizmus pre nastavovanie zosilnení regulátora. Podľa spôsobu určenia zmeny zosilnení regulátora (v reakcii na zmenu parametrov riadeného systému a teda na zmenu v dynamike sústavy a porúch) sa odlišujú rôzne typy schém adaptívneho riadenia.

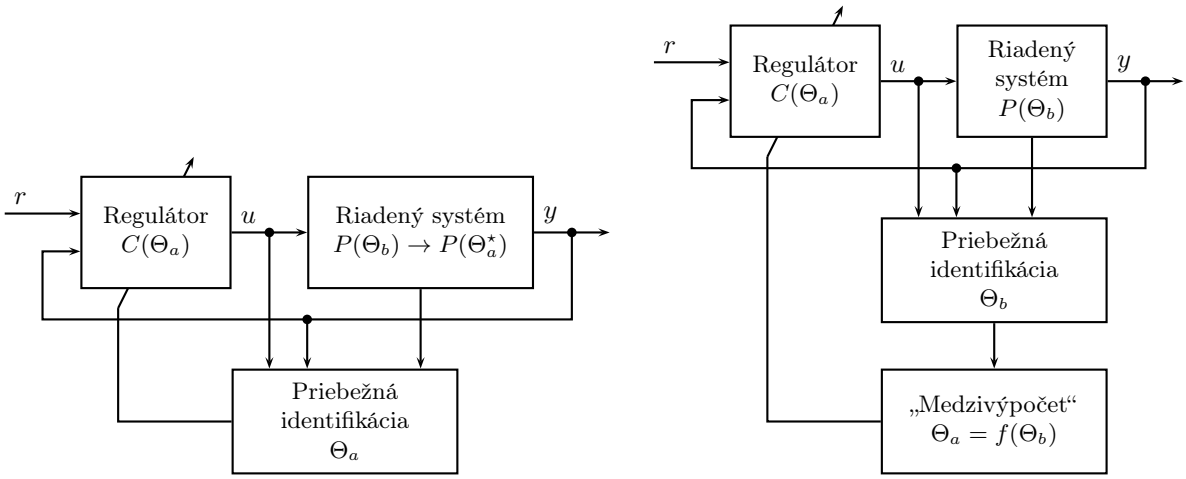
2.1.1 Robustné riadenie

Pre vysporiadanie sa so zmenami parametrov riadeného systému môže byť navrhnutý aj regulátor s konštantnými zosilneniami — týmto sa zaoberá *Robustné riadenie*. Regulátor navrhnutý metódami robustného riadenia sa však nepovažuje za adaptívny regulátor aj keď dokáže zvládnuť zmenu parametrov sústavy.

2.2 Gain Scheduling

Uvažujme, že v pracovnom bode i sú parametre modelu lietadla A_i, B_i, C_i a D_i známe. Pre daný pracovný bod teda vieme určiť regulátor, ktorý zabezpečí vyšpecifikované

[1] Všetko je tu akosi veľmi stručné...



(a) Priame adaptívne riadenie

(b) Nepriame adaptívne riadenie

Obr. 2: Principiálne schémy priameho a nepriameho adaptívneho riadenia

ciele. Zosilnenia takého regulátora označme Θ_i. Pri uvažovaní viacerych pracovných bodov máme množinu zosilnení: $\Theta_1, \ldots, \Theta_i, \ldots, \Theta_N$, kde N je počet uvažovaných pracovných bodov.

Po detegovaní pracovného bodu sa na regulátore nastavia (prepnú) príslušné zosilnenia (parametre regulátora). Tento spôsob adaptácie sa nazýva „Prepínanie parametrov regulátora“ — *Gain Scheduling* (GS).

Prechody medzi pracovnými bodmi sa riešia napríklad interpoláciou určených zosilnení alebo zvýšením počtu uvažovaných (detegovaných) pracovných bodov.

Dvomi prvkami, ktoré sú charakteristické pre GS sú *Look-up Table* (Vyhľadávacia tabuľka – tabuľka, v ktorej sa vyhľadávajú príslušné zosilnenia) a *meranie pomocných veličín*, takých ktoré sú vo vhodnom vzťahu s pracovným bodom. Na základe merania pomocných veličín sa rozhoduje (určuje), v ktorom pracovnom bode sústava operuje a podľa toho sa vyhľadajú a prepnú príslušné zosilnenia regulátora.

Výhodou GS je, že zosilnenia regulátora sa menia okamžite po detegovaní nového pracovného bodu. Výhodou je teda rýchlosť nájdenia teoreticky správnych parametrov regulátora. Časté a rýchle zmeny zosilnení regulátora však môžu viesť k nestabilite celého systému. Preto sa pridávajú obmedzenia pre frekvenciu prepínania.

Nevýhodou je, že mechanizmus prepínania a zosilnenia sú určené vopred (off-line) a teda kompenzácia prípadnej odchýlky od predpokladaného správania nie je možná. Nepredvídateľné zmeny dynamiky sústavy môžu viesť k zhoršeniu kvality riadenia alebo až k úplnému zlyhaniu. Ďalšou nevýhodou sú vysoké náklady na návrh a najmä na implementáciu, ktoré narastajú so zvyšujúcim sa počtom pracovných bodov.

2.3 Priame a nepriame AR

Vo všeobecnosti, adaptívny regulátor sa skladá z on-line estimátora (priebežná identifikácia) neznámych parametrov a zákona riadenia (algoritmus riadenia, regulátor). Zákon riadenia je motivovaný prípadom keď hodnoty parametrov sústavy sú známe, avšak konštantné zosilnenia sú nahradené časovo premenlivými.

Spôsob akým je on-line estimátor, nazývaný *Zákon adaptácie*, skombinovaný so zákonom riadenia určuje jeden z dvoch prístupov v adaptívnom riadení: Priame adaptívne riadenie a Nepriame adaptívne riadenie.

Pri nepriamom adaptívnom riadení sa priebežne identifikujú parametre uvažovaného modelu systému a následne sú tieto použité pri výpočte zosilnení zákona riadenia. Predpokladá sa pri tom, že identifikované parametre systému sú istotne ekvivalentné so skutočnými parametrami systému. Medzivýpočet potrebný pre výpočet parametrov zákona riadenia charakterizuje nepriame adaptívne riadenie.

Pri priamom adaptívnom riadení je model systému parametrizovaný z hľadiska parametrov zákona riadenia. Rovnica modelu systému je vyjadrená tak, že obsahuje

ideálne parametre zákona riadenia. Práve tieto parametre sú priebežne identifikované (pretože ideálne parametre sú neznáme) – adaptované. Výstupom priebežnej identifikácie (zákona adaptácie) sú teda priamo parametre zákona riadenia. Nie je potrebný medzivýpočet ako v prípade nepriameho adaptívneho riadenia.

Literatúra

[1] P. Ioannou and B. Fidan. *Adaptive Control Tutorial*. Society for Industrial and Applied Mathematics, USA., 2006.

[2] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, Inc, 1996.

3 Otázky a úlohy

1. Nakreslite základnú schému adaptívneho riadenia.
2. Nakreslite schému Gain Scheduling.
3. Nakreslite schému priameho adaptívneho riadenia.
4. Nakreslite schému nepriameho adaptívneho riadenia.
5. Vysvetlite rozdiel medzi priamym a nepriamym adaptívnym riadením.

O adaptívnej stabilizácii

Obsah

1 Cvičenie prvé

1. Uvažujme riadený systém v tvare

$$\dot{x}(t) = a\,x(t) + b\,u(t) \tag{1}$$

kde $x(t)$ je stavová veličina sytému, $u(t)$ je akčný zásah (výstup) regulátora. Parameter $b = 1$ a parameter a je neznáma konštanta.

- Koľkého rádu je systém (1)?
- Aká je prenosová funkcia daného dynamického systému?
- Aký je charakteristický polynóm a charakteristická rovnica dynamického systému?
- Aké sú korene charakteristického polynómu?
- Pre ktoré a je systém stabilný a pre ktoré a je nestabilný? Nájdite intervaly.
- Aké je zosilnenie sústavy? Aké sú časové konštanty?
- Ktorého reálneho systému (napríklad), je vhodným modelom takáto sústava?

2. Zostavte simulačnú schému systému (1). Zvoľte konkrétnu hodnotu parametra a z nájdeného intervalu tak aby riadený systém (1) bol stabilný. Nech začiatočný stav je $x(0) = 1$ a $u(t) = 0$. Simuláciou (pre vhodný časový úsek) ukážte, že $x = 0$ je rovnovážny stav sústavy.

3. Nech začiatočný stav sústavy $x(0) = 1$. Zvoľte konkrétnu hodnotu parametra a tak aby sústava (1) bola **nestabilná**. Spustite simuláciu a pozorujte nestabilný priebeh stavovej veličiny. Pridajte k riadenému systému regulátor daný nasledovne:

$$u = -k\,x \qquad k > |a| \tag{2}$$

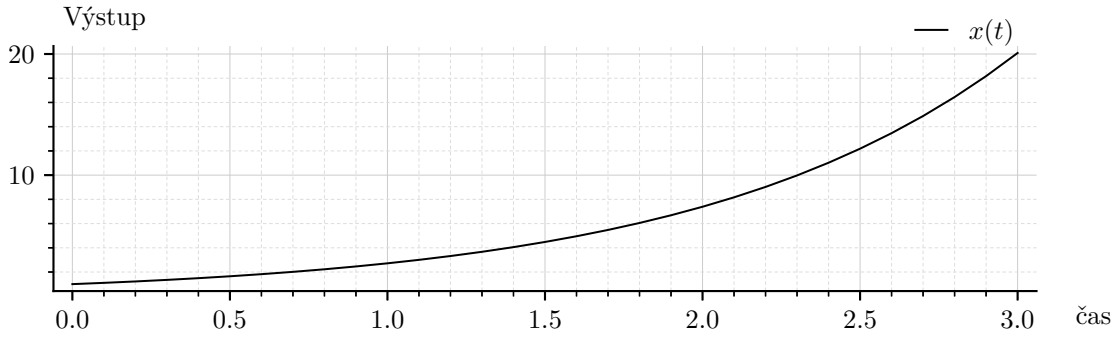
a overte, že URO je stabilný. Vysvetlite. Vyskúšajte rôzne hodnoty zosilnenia k.

4. Nech začiatočný stav sústavy $x(0) = 1$. Zvoľte konkrétnu hodnotu parametra a tak aby sústava (1) bola **nestabilná**. Pridajte k riadenému systému regulátor daný nasledovne:

$$u = -k\,x; \qquad \dot{k} = x^2 \tag{3}$$

Simuláciou vyšetrite stabilitu URO.

5. Overte, že regulátor z predchádzajúcej úlohy zabezpečí stabilizáciu URO pre rôzne hodnoty a a rôzne hodnoty začiatočného stavu $x(0)$ (vyskúšajte rôzne). Teda regulátor je adaptívny! Vysvetlite...



Obr. 1: Výsledok numerickej simulácie systému (4) pre $a = 1$ pri začiatočnom stave $x(0) = 1$.

2 Viac k téme cvičenia prvého

Zaoberáme sa systémom, ktorého vhodným modelom je diferenciálna rovnica v tvare

$$\dot{x}(t) = ax(t) + u(t) \tag{4}$$

kde $x(t)$ je výstupná veličina, $u(t)$ je vstupná veličina a konštanta a je parameter riadeného systému.

Nech je známe, že riadený systém je nestabilný. To znamená, že parameter a je kladné číslo. Nie je známa hodnota parametra a, ale je známe jeho znamienko. Keďže riadený systém je nestabilný, po vychýlení z rovnovážneho stavu (rovnovážnej polohy) sa výstupná veličina nevráti na hodnotu v rovnovážnom stave.

Úlohou je navrhnúť taký riadiaci systém, ktorý zabezpečí, že výstupná veličina sa vráti na hodnotu v rovnovážnom stave napriek tomu, že jej začiatočná hodnota (začiatočný stav) je iná ako v rovnovážnom stave. Teda začiatočný stav riadeného systému je vychýlený z rovnovážneho stavu.

Mimochodom, rovnica (4) je dynamický systém, lineárny systém a rád systému je 1 (lineárny dynamický systém 1. rádu). Z toho vyplýva, že v rovnovážnom stave je hodnota výstupu $y(t)$ nulová.

Diferenciálnu rovnicu (4) je možné (pri uvažovaní nulových začiatočných podmienok) previesť do tvaru prenosovej funkcie. Ak sa uváži, že

$$sx = ax + u \tag{5}$$

kde s je Laplaceov operátor, pričom tu plní funkciu časovej derivácie, potom možno písať

$$sx - ax = u \tag{6a}$$

$$(s - a)\, x = u \tag{6b}$$

$$\frac{x}{u} = \frac{1}{(s - a)} \tag{6c}$$

čo je prenosová funkcia zodpovedajúca systému (4). Charakteristický polynóm je $(s - a)$ a koreňom tohto polynómu je (číslo) a. Keďže je známe, že systém je nestabilný, potom pól systému, teda koreň charakteristického polynómu leží v pravej polrovine komplexnej roviny. To znamená, že $a > 0$.

Samotný riadený systém

S využitím numerickej simulácie ukážme, že systém (4) je pre $a > 0$ nestabilný. Pri tomto overení je vstupná veličina systému $u(t)$ nepodstatná. Preto je nastavená na nulovú hodnotu. Nech začiatočná hodnota stavovej veličiny $x(0) = 1$ a nech hodnota parametra $a = 1$. Simulujme 3 časové jednotky. Výsledok numerickej simulácie je na obr. 1.

Formalizácia požiadaviek na kvalitu riadenia

Úlohou je aby sa výstupná veličina vrátila na nulovú hodnotu (do rovnovážneho stavu). Ale ako rýchlo sa tam má vrátiť? Aké sú požiadavky na prechodný dej pri činnosti riadiaceho systému?

Nech sú tieto požiadavky premietnuté do akéhosi vzorového dynamického systému, ktorý má stavovú veličinu $x_m(t)$. Tento vzorový systém je

$$\dot{x}_m(t) = -a_m x_m(t) \tag{7}$$

Jeho parametrom je a_m. Ak bude $a_m > 0$, potom bude tento systém stabilný a teda vždy sa ustáli v rovnovážnom stave, to je v tomto prípade samozrejme nula. Pól systému je $-a_m$. Voľbou pólu sa volí dynamika s akou sa systém bude vracať do rovnovážneho stavu. Tento vzorový systém teda zodpovedá úlohe, ktorú máme, a navyše úplne presne špecifikuje akékoľvek iné požiadavky, ktoré nemusia priamo plynúť zo zadania úlohy.

Teoretická existencia ideálneho riadiaceho systému

Od riadiaceho systému chceme aby pre riadený systém (4) zabezpečil, že výstupná veličina riadeného systému (4) $x(t)$ sa bude správať práve tak ako veličina $x_m(t)$. V ideálnom prípade je teda odchýlka $e(t) = x(t) - x_m(t)$ nulová. Je vôbec možné zostaviť taký riadiaci systém?

Uvažujme nasledujúci predpis pre výpočet hodnoty akčného zásahu $u(t)$.

$$u(t) = -k^* x(t) \tag{8}$$

Zovšeobecnene budeme takýto predpis nazývať zákon riadenia. Predpisuje ako sa vypočíta akčný zásah. Zákon riadenia obsahuje dva prvky. Signál $x(t)$, ktorý je spätnou väzbou od riadeného systému, keďže $x(t)$ je výstupom riadeného systému, a druhým prvkom zákona riadenia je parameter k^*.

Dosaďme za $u(t)$ v riadenom systéme. Získa sa tak rovnica uzavretého regulačného obvodu (URO). V tomto prípade v tvare

$$\dot{x}(t) = ax(t) - k^* x(t) \tag{9a}$$

$$\dot{x}(t) = (a - k^*)\, x(t) \tag{9b}$$

Pripomeňme, že je žiadané aby odchýlka $e(t) = x(t) - x_m(t)$ bola nulová. Je zrejmé, že ak by platilo $(a - k^*) = -a_m$, potom

$$\dot{x}(t) = -a_m x(t) \tag{10}$$

URO a uvedený vzorový systém (7) by teda mali úplne rovnaký predpis (rovnakú rovnicu). Samozrejme, v URO vystupuje signál $x(t)$ a vo vzorovom systéme vystupuje signál $x_m(t)$. Inak sú však tieto dva systémy úplne rovnaké. To znamená, že možno písať $x(t) = x_m(t)$ a teda daná celková úloha je splnená.

Týmto sa ukázalo, že je možné zostaviť taký riadiaci systém, teda nájsť taký zákon riadenia, ktorý rieši danú úlohu.

V tomto prípade sme našli zákon riadenia (8), ku ktorému prislúcha tzv. podmienka zhody, čo v tomto prípade je

$$(a - k^*) = -a_m \tag{11}$$

Ide o podmienku zhody medzi URO a istým vzorovým systémom, ktorý sa vo všeobecnosti nazýva Referenčný model (RM). RM predpisuje ako sa má správať URO.

Pre riešenie úlohy tohto typu teda v prvom rade musí existovať *podmienka zhody* čo je, pochopiteľne, istá rovnica a táto rovnica musí byť riešiteľná. Neznámou je samozrejme parameter zákona riadenia. V tomto prípade k^*. Teoretické riešenie je

$$a - k^* = -a_m \tag{12a}$$

$$k^* = a_m + a \tag{12b}$$

Konkrétnu hodnotu parametra k^* však nevieme určiť pretože nepoznáme konkrétnu hodnotu parametra a. Ale vieme, že vôbec existuje ideálne k^*. To je veľmi, veľmi dôležité!

Zákon riadenia s premenlivým parametrom

Zákon riadenia (8), ktorý má konštantný parameter k^*, teda nevieme použiť, pretože nevieme vypočítať k^*. Zachovajme ale štruktúru zákona riadenia (aby stále existovali riešiteľné podmienky zhody) a neznámy parameter k^* nahraďme parametrom, ktorému dovolíme meniť sa v čase. Adaptovať sa. Adaptovať sa tak, aby sme splnili úlohu. Adaptívny zákon riadenia nech teda je

$$u(t) = -k(t)x(t) \tag{13}$$

kde $k(t)$ je časovo premenlivý parameter.

Parameter $k(t)$ sa vlastne môže meniť akokoľvek. Cieľom však je splnenie úlohy. To znamená, že ideálne $e(t) = x(t) - x_m(t) = 0$. To si však vyžaduje mať možnosť určiť k^*. Žiadať teda $e(t) = 0$ v tomto prípade nikam nevedie. Žiadajme ale aby $e(t) \to 0$, teda že odchýlka $e(t)$ sa asymptoticky blíži k nule. Pritom sa však nevylučuje aby mohla byť aj nulová.

Ak $e(t) \to 0$ potom vlastne $x(t) \to x_m(t)$, teda signál $x(t)$ sa hodnotou približuje k signálu $x_m(t)$. Aká je vlastne hodnota signálu $x_m(t)$? Predpisuje ju referenčný model (7). K rovnici (7) chýba explicitne napísaná začiatočná podmienka pre signál $x_m(t)$, teda $x_m(0)$. Úlohou je aby sa $x(t)$ dostalo do rovnovážneho stavu. Teda do nuly. Signál $x_m(t)$ by teda mal byť nulový. A prečo nie hneď aj od začiatku? Nech teda $x_m(0) = 0$. Potom vzhľadom na rovnicu (7) je $x_m(t) = 0$ po celý čas! Ak teda chceme $e(t) \to 0$, potom máme $x(t) \to 0$. Alebo inak povedané $e(t) = x(t) - x_m(t) = x(t) - 0$. Teda $e(t) = x(t)$.

Spôsob ako meniť (adaptovať) parameter zákona riadenia

Stále však nie je zrejmé ako adaptovať (meniť) parameter $k(t)$ tak aby sme splnili modifikovanú požiadavku $e(t) \to 0$.

URO s adaptívnym zákonom riadenia (13) je

$$\dot{x}(t) = (a - k(t))\, x(t) \qquad x(0) \neq 0 \tag{14}$$

Ak by platilo $(a - k(t)) < 0$, potom by bol URO stabilný a $x(t) \to 0$ pri akejkoľvek začiatočnej hodnote $x(0)$. A to je presne požiadavka pre splnenie úlohy. Hodnota parametra a je však neznáma, ale je zrejmé, že ak bude $k(t)$ dostatočne veľké (vzhľadom na a), potom celý výraz $(a - k(t))$ bude záporný.

Ako však získať informáciu o tom kedy je $k(t)$ dostatočne veľké? Ak nie je dostatočne veľké, potom je vo všeobecnosti odchýlka $e(t)$ nenulová! Teda pri nenulovej $e(t)$ treba $k(t)$ meniť (v čase), konkrétnejšie, treba zvyšovať jeho hodnotu. Len ak by platilo, že $e(t) = 0$, len vtedy možno prestať meniť $k(t)$.

Zmena parametra $k(t)$ v čase je jeho časová derivácia $\dot{k}(t)$.

Pre vzťah

$$\dot{k}(t) = e^2(t) \tag{15}$$

platí, že ak je $e(t)$ nenulové, tak zmena $k(t)$ je kladná, a len ak $e(t) = 0$ aj zmena $k(t)$ je nulová. Pomocou tohto predpisu pre $\dot{k}(t)$ teda dosiahneme také zmeny $k(t)$ v čase, aké sú potrebné pre dosiahnutie $x(t) \to 0$ a teda $e(t) \to 0$, čo je splnenie stanoveného cieľa!

Rovnica (15) predpisuje ako sa má meniť parameter zákona riadenia tak aby bol cieľ riadenia splnený. Vo všeobecnosti sa takýto predpis nazýva zákon adaptácie.

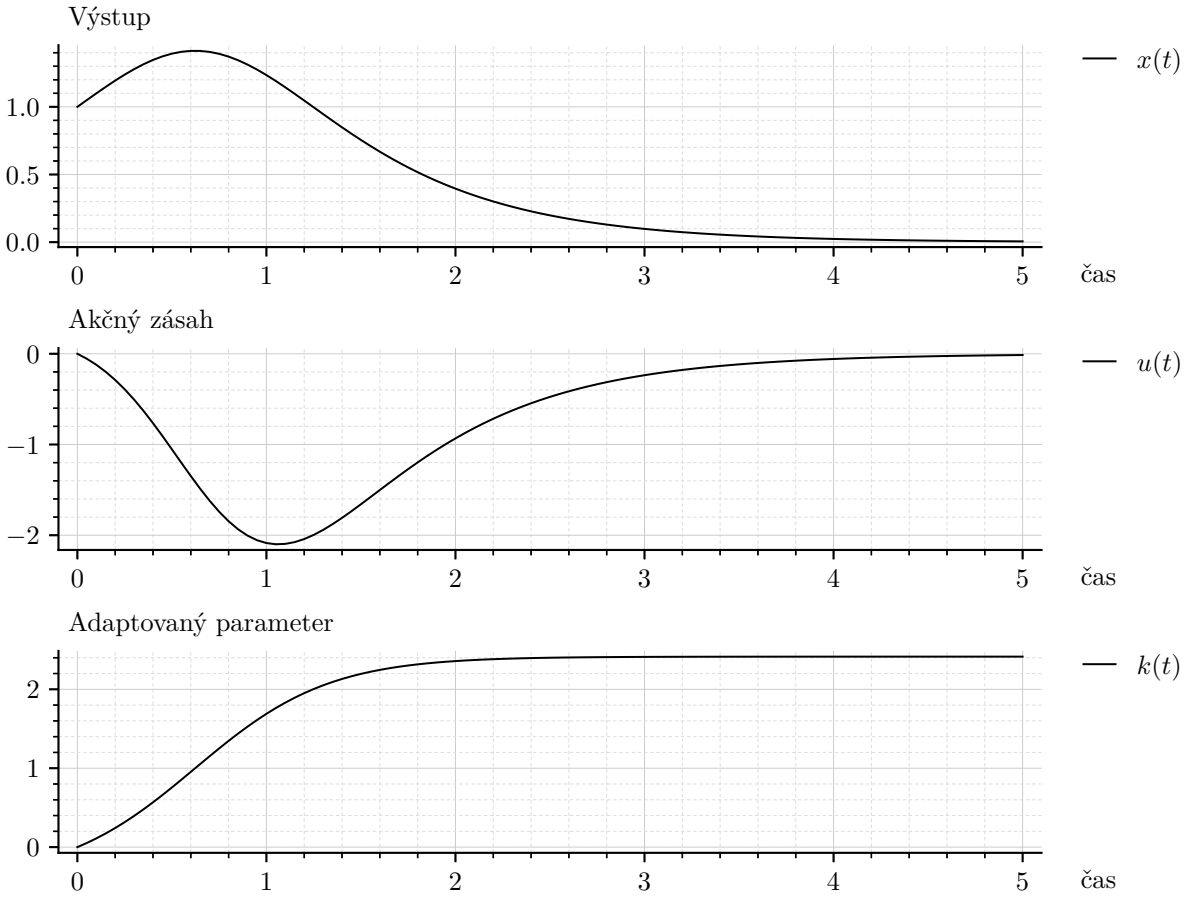
Mimochodom, keďže vieme, že v tomto prípade platí $e(t) = x(t)$, potom (15) možno písať aj v tvare

$$\dot{k}(t) = x^2(t) \tag{16}$$

ba dokonca aj

$$\dot{k}(t) = e(t)x(t) \tag{17}$$

čo by bol pre skúsenejšieho v tejto oblasti azda najvhodnejší zápis. V tomto prípade praktickejším je tvar (16), pretože si to nevyžaduje žiadny ďalší signál, len $x(t)$, ktorý je samozrejme k dispozícii.



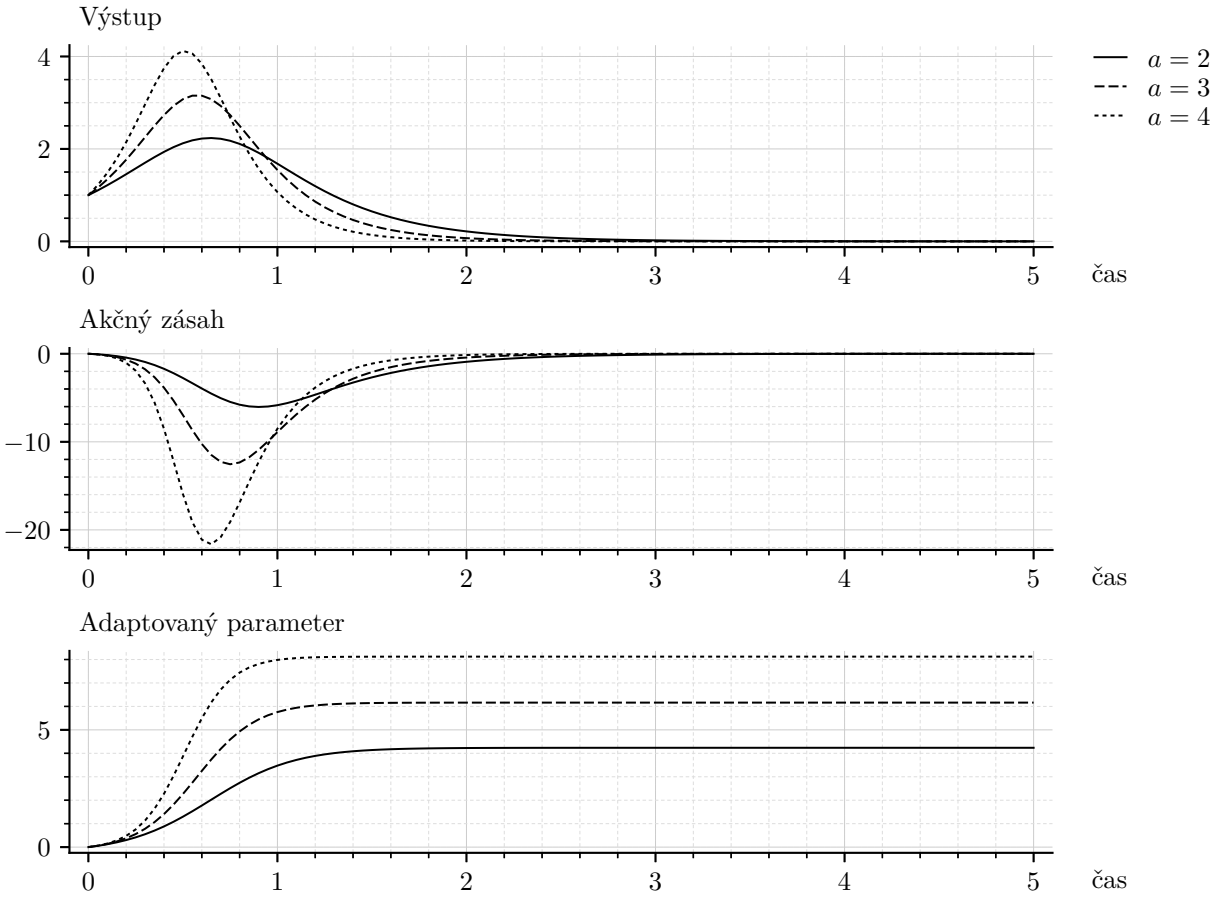
Obr. 2: Výsledok numerickej simulácie adaptívneho riadiaceho systému pre $a = 1$ pri začiatočnom stave $x(0) = 1$.

Overenie adaptívneho riadiaceho systému

Zákon adaptácie (16) teraz dopĺňa zákon riadenia (13). K uvedenému ešte pre úplnosť prislúcha aj referenčný model čo však v tomto prípade možno zanedbať. Tieto súčasti spolu tvoria adaptívny riadiaci systém, ktorý spĺňa danú úlohu.

S využitím numerickej simulácie ukážme, že uvedený adaptívny riadiaci systém stabilizuje riadený systém, t.j. pri začiatočnom stave mimo rovnovážneho stavu zabezpečí asymtotické približovanie sa výstupu riadeného systému k rovnovážnemu stavu. Na obr. 2 sú výsledky použitia tohto adaptívneho riadiaceho systému pre prípad keď je v riadenom systéme $a = 1$ a $x(0) = 1$. Je možné pozorovať vyššie uvedené predpoklady a teda, že parameter $k(t)$ sa zvyšuje až pokým neprekročí hodnotu parametra a. Vtedy sa URO stane stabilným a jeho stavová veličina sa z hodnoty, ktorú práve má, prirodzene začne vracať do rovnovážneho stavu, teda na hodnotu nula. Ďalej rýchlosť zmeny parametra $k(t)$ klesá až sa napokon ustáli. Ustáli sa na hodnote vyššej ako je hodnota parametra a.

Ten istý adaptívny riadiaci systém sa bude, samozrejme, správať kvalitatívne rovnako pre rôzne hodnoty parametra riadeného systému a (a pre rôzne začiatočné stavy riadeného systému). Na obr. 3 sú pre ilustráciu znázornené rôzne prípady hodnoty parametra a. Tým je prezentovaná schopnosť adaptácie uvedeného riadiaceho systému.



Obr. 3: Výsledok simulácie adaptívneho riadiaceho systému pre rôzne hodnoty parametra a. Začiatočný stav riadeného systému je v každom prípade $x(0) = 1$.

3 Formálny opis adaptívnej stabilizácie pre systém 1. rádu

Nasledujúci jednoduchý príklad ilustruje situáciu, v ktorej neznalosť hodnoty parametra riadeného systému znemožňuje návrh riadiaceho systému. Adaptívne riadenie rieši tento problém a umožňuje navrhnúť riadiaci systém, ktorý zabezpečí splnenie cieľa riadenia pre akúkoľvek hodnotu neznámeho parametra sústavy.

Uvažujme riadený systém v tvare

$$\dot{x} = a\,x + u \tag{18}$$

kde $x(t)$ je stavová veličina systému, $u(t)$ je akčný zásah (výstup) regulátora a parameter a je neznáma konštanta. Rovnovážny bod systému je v bode $x = 0$. Cieľom riadenia je aby stavová veličina x bola asymptoticky stabilná, teda aby po vychýlení stavu x z rovnovážneho bodu sa stav postupne približoval (asymptoticky) naspäť k rovnovážnemu bodu. To platí pre nasledujúci systém:

$$\dot{x} = -a_m\,x \tag{19}$$

kde konštanta $a_m > 0$.

Ak by bol parameter a známy, potom lineárny regulátor v tvare

$$u = -k\,x \qquad k > |a| \tag{20}$$

zabezpečí splnenie úlohy. V takomto prípade nezáleží, či samotný riadený systém (bez riadenia, $u = 0$) stabilný je alebo nie je. Poznáme absolútnu hodnotu $|a|$ a zosilnenie k zvolíme väčšie ako $|a|$. Potom je zrejmé, že

$$\dot{x} = a\,x + (-k\,x) = (a - k)\;x \tag{21}$$

je rovnica uzavretého regulačného obvodu (URO) spĺňajúceho cieľ riadenia, pretože v najhoršom (hraničnom) prípade, kedy URO je stabilný, nie však asymptoticky stabilný, ak $k = |a|$ platí $(a - |a|) \leq 0$ pre každé a.

Pre návrh regulátora (20) v podstate stačí poznať horné ohraničenie parametra a.

Na druhej strane, parameter a sa môže s časom meniť a jeho horná hranica nemusí byť známa. Pre nevhodne zvolené k potom môže nastať situácia $a > k > 0$. Vtedy URO je nestabilný. Pre neznámu hornú hranicu parametra a nie je možné navrhnúť riadiaci systém, ktorý vždy zabezpečí splnenie úlohy riadenia — stabilný URO.

Riešenie tejto úlohy metódami Adaptívneho riadenia, ktoré budú popísané podrobne neskôr, vedie na nasledovné:

Riadiaci systém v tvare

$$u = -k\,x; \qquad \dot{k} = x^2 \tag{22}$$

zabezpečí, že všetky signály v URO sú ohraničené a x s časom konverguje do nuly pričom *nezáleží* na hodnote parametra a. Požiadavkou nie je presné určenie hodnoty parametra a, ale len stabilizácia systému. Nie je požadovaná ani presná hodnota parametra k. Adaptívny riadiaci systém len zabezpečí, že k je ohraničené, teda stabilné. Pre tento konkrétny jednoduchý príklad je zrejmé, že k sa ustáli na hodnote väčšej ako a, čo však nehovorí nič o presnej hodnote parametra a ani k.

Ukážme, že riadiaci systém (22) stabilizuje stavovú – výstupnú veličinu sústavy (18).

Najskôr upresníme cieľ riadenia. Cieľom je aby x konvergovalo k nule pre čas $t \to \infty$. Žiadanou hodnotou pre x je $r = 0$. Uvažujeme všeobecný (akýkoľvek) začiatočný stav x_0. Začiatok je v čase $t = 0$, teda: $x(0) = x_0$. Cieľ si vyžaduje aby URO bol stabilný. Preto nech želaný pól URO je $-a_m$, kde $a_m > 0$ sa volí podľa požiadaviek na rýchlosť regulácie. Potom referenčný (ideálny, želaný) priebeh stavu x opisuje rovnica

$$\dot{x}_m = -a_m\,x_m \tag{23}$$

kde x_m stavová veličina ideálneho URO. Je prirodzené očakávať (lepšie povedané, zvoliť si), že začiatočný stav $x_m(0) = 0$ z čoho vyplíva, že v ďalšom môžme uvažovať $x_m = 0$.

V ideálnom prípade, keď poznáme parameter a, vieme určiť pre zákon riadenia „ideálne“ k^*:

$$u = -k^*\,x; \qquad k^* = a + a_m \tag{24}$$

Vtedy sa URO presne zhoduje s ideálnym URO, teda s referenčným modelom (23)

$$\dot{x} = a\,x + (-k^*\,x) = (a - k^*)\;x = (a - a - a_m)\;x = -a_m\,x \tag{25}$$

Pretože a je neznáme, k^* nevieme vypočítať a zákon riadenia (24) nemôže byť použitý.

Žiadaná hodnota pre stavovú veličinu x je $r = 0$ a žiadame aby sa hodnota parametra k približovala k ideálnej hodnote k^* pretože len vtedy bude pól URO v bode $-a_m$. Riadený systém vyjadríme ako funkciu ideálneho parametra k^* (teda tak aby rovnica obsahovala parameter k^*). Tento krok má súvislosť s tým, že ide v podstate o identifikáciu parametra k^* — bližšie vysvetlenie neskôr. Urobíme to jednoducho pripočítaním a odpočítaním „ideálneho“ zákona riadenia (24) k rovnici riadeného systému (18), potom riadený systém má tvar

$$\dot{x} = a\,x - k^*\,x + k^*\,x + u \tag{26}$$

Pretože $a - k^* = -a_m$ máme

$$\dot{x} = -a_m\,x + k^*\,x + u \tag{27}$$

Definujme tzv. *adaptačnú odchýlku*. Ak adaptačná odchýlka nie je nulová je potrebné adaptovať riadiaci systém tak aby sa skutočný URO zhodoval so želaným URO. Adaptačná odchýlka má tvar

$$e = x - x_m \tag{28}$$

potom platí:

$$\dot{x} - \dot{x}_m = -a_m\, x + k^*\, x + u - (-a_m)\, x \tag{29a}$$
$$\dot{e} = -a_m\, (x - x_m) + k^*\, x + u \tag{29b}$$
$$\dot{e} = -a_m\, e + k^*\, x + u \tag{29c}$$

Zavedením adaptačnej odchýlky sme posunuli začiatok súradnicového systému stavového priestoru do rovnovážneho bodu riadeného systému. Inými slovami, rovnovážny stav veličiny e z rovnice (29c) je v začiatku stavového priestoru riadeného systému.

Uvažujeme použitie zákona riadenia v tvare $u = -k\, x$, po dosadení do (29c):

$$\dot{e} = -a_m\, e + k^*\, x - k\, x \tag{30}$$
$$\dot{e} = -a_m\, e - \theta\, x \tag{31}$$

kde sme zaviedli pojem *chyba nastavenia parametrov zákona riadenia*, ktorý označíme θ, a v tomto prípade $\theta = k - k^*$.

Chceme dokázať, že riadiaci systém (22) stabilizuje uzavretý regulačný obvod. Platí $\dot{\theta} = \dot{k} - \dot{k^*}$. Avšak k^* = konšt. a teda $\dot{k^*} = 0$, preto $\dot{\theta} = \dot{k} = x^2$. Vyšetrením stability systému diferenciálnych rovníc v tvare

$$\dot{e} = -a_m\, e - \theta\, x \tag{32a}$$
$$\dot{\theta} = x^2 \tag{32b}$$

dokážeme stabilitu URO.

Zvoľme kandidáta na Ljapunovovu funkciu

$$V = \frac{(e(t))^2}{2} + \frac{(\theta(t))^2}{2} \tag{33}$$

Derivácia (33) podľa času je

$$\dot{V} = \frac{1}{2}\, 2\, e(t)\, \dot{e}(t) + \frac{1}{2}\, 2\, \theta(t)\, \dot{\theta}(t) = e(t)\, \dot{e}(t) + \theta(t)\, \dot{\theta}(t) \tag{34}$$

Do (34) dosadíme za $\dot{e}$ a $\dot{\theta}$, máme

$$\dot{V} = e\, (-a_m\, e - \theta\, x) + \theta\, x^2 \tag{35a}$$
$$\dot{V} = -a_m\, e^2 - \theta\, x\, e + \theta\, x^2 \tag{35b}$$

Pretože $e = x - x_m$ a uvažujeme $x_m = 0$ a teda $e = x$, tak rovnicu (35b) možno písať v tvare

$$\dot{V} = -a_m\, x^2 - \theta\, x^2 + \theta\, x^2 \tag{36a}$$
$$\dot{V} = -a_m\, x^2 \tag{36b}$$

Pre systém (32) sme našli kladne definitnú funkciu V (Tá funkcia má skoro pre všetky x a θ kladnú hodnotu, len pre $x = 0$ a $\theta = 0$ má nulovú hodnotu. Tá nulová hodnota je práve v stacionárnom bode systému (32), ktorého stabilitu vyšetrujeme). Derivácia tejto funkcie je vo všeobecnosti záporne semidefinitná (Je rovná nule v stacionárnom – nulovom bode, a inde môže byť len záporná alebo tiež rovná nule). Teda systém je stabilný. Presný zaver o stabilite rovnovážneho stavu systému (32) v zmysle všeobecnej teórie stability podľa Ljapunova je, že rovnovážny stav je stabilný (neutrálne) podľa všetkých premenných (x a θ) a asymptoticky stabilný podľa premennej x. Prakticky to znamená, že premenná x dosiahne nulovú hodnotu po skončení prechodového deja, ale premenná θ (a teda aj k) len konečnú hodnotu. Tým sme ukázali vlastnosti, ktoré zabezpečí *adaptívny* riadiaci systém (22).

4 Otázky a úlohy

1. Vyšetrite stabilitu systému

$$\dot{e}(t) = -a_m\, e(t) + \theta(t)\, x(t)$$
$$\dot{\theta}(t) = -e(t) x(t)$$

kde $a_m > 0$.

5 Príklad realizácie niektorých úloh cvičenia prvého

Konkrétne v tejto časti sa budeme venovať najmä numerickému riešeniu diferenciálnych rovníc (ktoré vystupujú v našom cvičení prvom) s využitím, pochopiteľne, ODE solvera. Načrtneme však aj všeličo iné. Používa sa tu Python[1].

5.1 Takpovediac základný prístup

Uvažujme riadený systém v tvare

$$\dot{x}(t) = a\,x(t) + u(t) \tag{37}$$

kde $x(t)$ je stavová veličina sytému, $u(t)$ je akčný zásah (výstup) regulátora. Parameter a je neznáma konštanta.

Funkcia, ktorá realizuje diferenciálnu rovnicu riadeného systému, nech je v nasledujúcom tvare, kde sa hneď aj predpokladá, že $u(t) = 0$.

Výpis kódu 1: Súbor `ar02_file02.py`

```
30 def fcn_difRovnice_01(x, t, param_a):
31
32     a = param_a
33
34     u = 0
35
36     dotx = a*x + u
37
38     return dotx
```

Zostavme simulačnú schému pre riadený systém. Zvoľme konkrétnu hodnotu parametra a tak aby riadený systém bol stabilný. Nech začiatočný stav je $x(0) = 1$ a $u(t) = 0$. Simuláciou (pre vhodný časový úsek) ukážme, že $x = 0$ je rovnovážny stav riadeného systému.
Simulačnú schému nech realizuje nasledujúca funkcia:

Výpis kódu 2: Súbor `ar02_file02.py`

```
50 def fcn_simSch_01(t_start, t_final, T_s, param_a):
51
52     #----------------------------------------
53     t_log = np.arange(sim_t_start, sim_t_final+sim_T_s, sim_T_s).
       reshape(-1,1)
54
55     #----------------------------------------
56     x_0 = 1
57
58     #----------------------------------------
59     odeOut = odeint(fcn_difRovnice_01,
60                     x_0,
61                     t_log[:,0],
62                     args=(param_a,)
63                     )
64
65     return [t_log, odeOut]
```

Nastavme a spustime simuláciu:

Výpis kódu 3: Súbor `ar02_file02.py`

```
78 # Nastavenia simulacie
79
80 sim_t_start = 0
81 sim_t_final = 3
82 sim_T_s = 0.1
83
84 param_a = -1
85
86 # %% ------------------------------------------------------------
87
88 # Simulacia
89
```

[1] Rovnako dobre by sa to dalo zostrojiť v akomkoľvek programovacom jazyku, v ktorom máme k dispozícii ODE solver. Prípadne aj ten ODE solver by sme si naprogramovali...

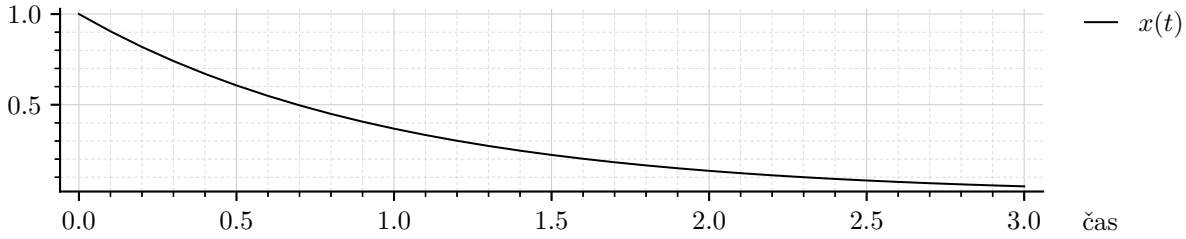
```
90  t_log, x_log, = fcn_simSch_01(sim_t_start, sim_t_final, sim_T_s,
        param_a)
91
92
```

Všetko potrebné je teraz zaznamenáné v premenných `t_log` a `x_log`. Nakreslime obrázok (pre prehľadnosť je kód kreslenia obrázku v samostatnom súbore a nie priamo tu v nasledujúcej bunke):

Výpis kódu 4: Súbor `ar02_file02.py`

```
103 # Obrazok
104
105 figName = 'figsc_ar02_fig03'
106 figNameNum = 0
107
108 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
109
```

Výstup



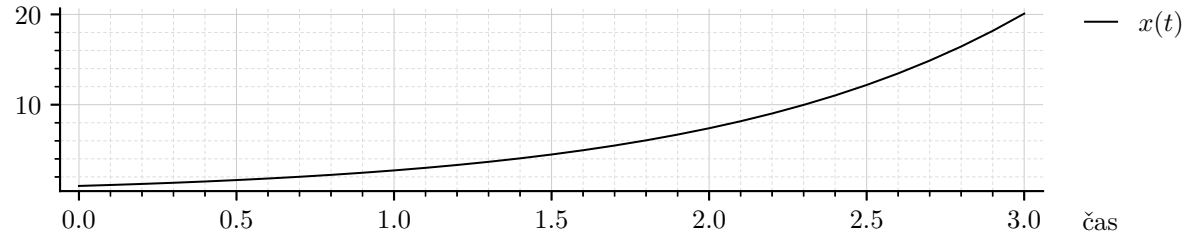
Obr. 4

V prípade, že parameter a je zvolený tak aby bol riadený systém nestabilný, výsledok je:

Výpis kódu 5: Súbor `ar02_file02.py`

```
118 param_a = 1
119
120 # Simulacia
121
122 t_log, x_log, = fcn_simSch_01(sim_t_start, sim_t_final, sim_T_s,
        param_a)
123
124 # Obrazok
125
126 figName = 'figsc_ar02_fig03'
127 figNameNum = 1
128
129 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
130
```

Výstup



Obr. 5

Nech začiatočný stav riadeného systému je $x(0) = 1$. Ponechajme konkrétnu hodnotu parametra a tak aby riadený systém bol nestabilný. Pridajme k riadenému systému riadiaci systém daný nasledovne:

$$u = -k\,x; \qquad \dot{k} = x^2 \tag{38}$$

Simuláciou vyšetrime stabilitu URO.
Funkcia, ktorá realizuje potrebné diferenciálne rovnice nech je nasledovná:

Výpis kódu 6: Súbor `ar02_file02.py`

```
164 def fcn_difRovnice_02(x, t, param_a):
165
166     x, k = x
167
168     a = param_a
169
170     dotk = x**2
171
172     u = -k*x
173
174     dotx = a*x + u
175
176     return [dotx, dotk]
177
```

Simulačná schéma, opäť realizovaná ako funkcia:

Výpis kódu 7: Súbor ar02_file02.py

```
186 def fcn_simSch_02(t_start, t_final, T_s, param_a):
187
188     #----------------------------------------
189     t_log = np.arange(sim_t_start, sim_t_final+sim_T_s, sim_T_s).
        reshape(-1,1)
190
191     #----------------------------------------
192     x_0 = [1, 0]
193
194     #----------------------------------------
195     odeOut = odeint(fcn_difRovnice_02,
196                     x_0,
197                     t_log[:,0],
198                     args=(param_a,)
199                     )
200
201     return [t_log, odeOut]
202
```

Nastavenie a spustenie simulácie, pričom pripomeňme, že parameter a je stále nastavený tak, aby riadený systém bol nestabilný.

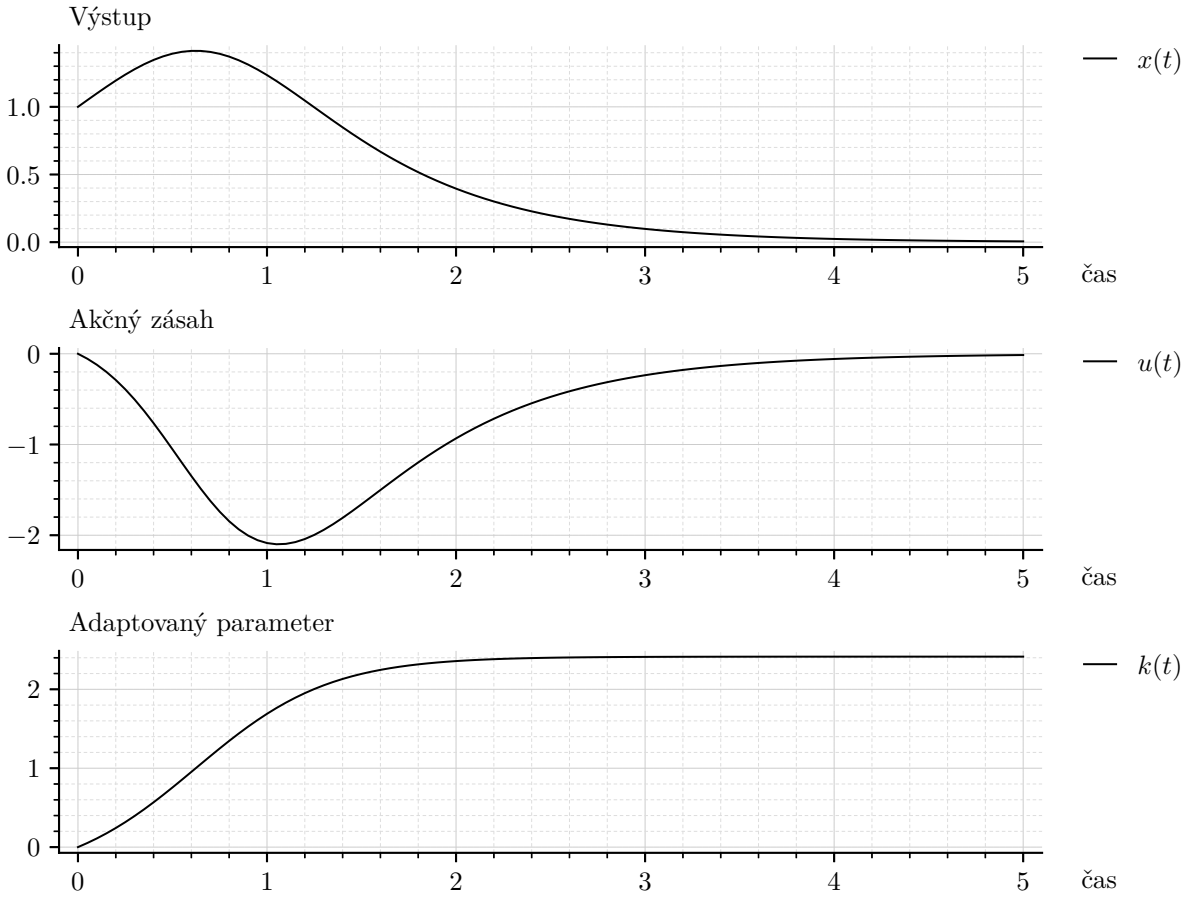
Výpis kódu 8: Súbor ar02_file02.py

```
211 # Nastavenia simulacie
212
213 sim_t_start = 0
214 sim_t_final = 5
215 sim_T_s = 0.05
216
217 # Simulacia
218
219 t_log, x_log, = fcn_simSch_02(sim_t_start, sim_t_final, sim_T_s,
        param_a)
220
```

Obrázok, ktorý zobrazuje okrem výstupnej veličiny riadeného systému aj iné veličiny potrebné pre posúdenie stability celého uzavretého regulačného obvodu.

Výpis kódu 9: Súbor ar02_file02.py

```
227 # Obrazok
228
229 figName = 'figsc_ar02_fig01'
230 figNameNum = 1
231
232 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
233
```



Obr. 6

Cieľ riadenia je splnený, čo v tomto prípade znamená, že výstupná veličina sa približuje k nulovej hodnote, a všetky signály uzavretého regulačného obvodu sú ohraničené. To znamená, že URO je stabilný.

Podrobnejšia diskusia z pohľadu Adaptívneho riadenia je nad rámec konkrétne tejto časti krátkych poznámok.

5.2 O realistickej implementácii riadiaceho systému tu uvedeného

Uvažujme riadený systém v tvare

$$\dot{x}(t) = a\,x(t) + u(t) \tag{39}$$

kde $x(t)$ je stavová veličina systému, $u(t)$ je akčný zásah (výstup) regulátora. Parameter a je neznáma konštanta.

Funkcia, ktorá realizuje diferenciálnu rovnicu riadeného systému, nech je v tvare:

Výpis kódu 10: Súbor `ar02_file03.py`

```
30 def fcn_difRovnice_01(x, t, a, u):
31
32     dotx = a*x + u
33
34     return dotx
```

Nech začiatočný stav riadeného systému je $x(0) = 1$ a nech hodnota parametra a je taká aby riadený systém bol nestabilný. Pridajme k riadenému systému riadiaci systém daný nasledovne:

$$u = -k\,x; \qquad \dot{k} = x^2 \tag{40}$$

Simulačná schéma, ktorá realizuje numerickú simuláciu riadeného systému pomocou ODE solvera, a zároveň realizuje istú implementáciu daného riadiaceho systému, je nasledovná:

Výpis kódu 11: Súbor ar02_file03.py

```
43  def fcn_simSch_03(t_start, T_s, finalIndex, param_a):
44
45      #----------------------------------------
46      # casovy vektor
47
48      t_log = np.zeros([finalIndex, 1])
49      t_log[0,:] = t_start
50
51      #----------------------------------------
52      # vektor stavu riadeneho systemu
53
54      x_0 = np.array([1])
55
56      x_log = np.zeros([finalIndex, len(x_0)])
57      x_log[0,:] = x_0
58
59      #----------------------------------------
60      # vektor adaptovaneho parametra
61
62      k_log = np.zeros([finalIndex, 1])
63
64      #----------------------------------------
65      # vektor akcneho zasahu
66
67      u_log = np.zeros([finalIndex, 1])
68
69      #----------------------------------------
70
71      timespan = np.zeros(2)
72      for idx in range(1, int(finalIndex)):
73
74          #---------------------------------------
75          # Riadeny system - simulacia (pomocou ODEsolvera)
76
77          timespan[0] = t_log[idx-1,:]
78          timespan[1] = t_log[idx-1,:] + T_s
79
80          odeOut = odeint(fcn_difRovnice_01,
81                          x_log[idx-1,:],
82                          timespan,
83                          args=(param_a, u_log[idx-1,:])
84                          )
85
86          x_log[idx,:] = odeOut[-1,:]
87          t_log[idx,:] = timespan[-1]
88
89          #---------------------------------------
90          # Riadiaci system:
91
92          # zakon adaptacie:
93          deltk = x_log[idx-1,:]*x_log[idx-1,:]
94
95          # adaptovany parameter (numericka integracia - vlastne
    sumator)
96          k_log[idx,:] = k_log[idx-1,:] +  (deltk * T_s)
97
98          # zakon riadenia:
99          u_log[idx,:] = -k_log[idx-1,:] * x_log[idx-1,:]
100
101     return [t_log, x_log, u_log, k_log]
102
```

Nastavme a spustime simuláciu:

Výpis kódu 12: Súbor ar02_file03.py

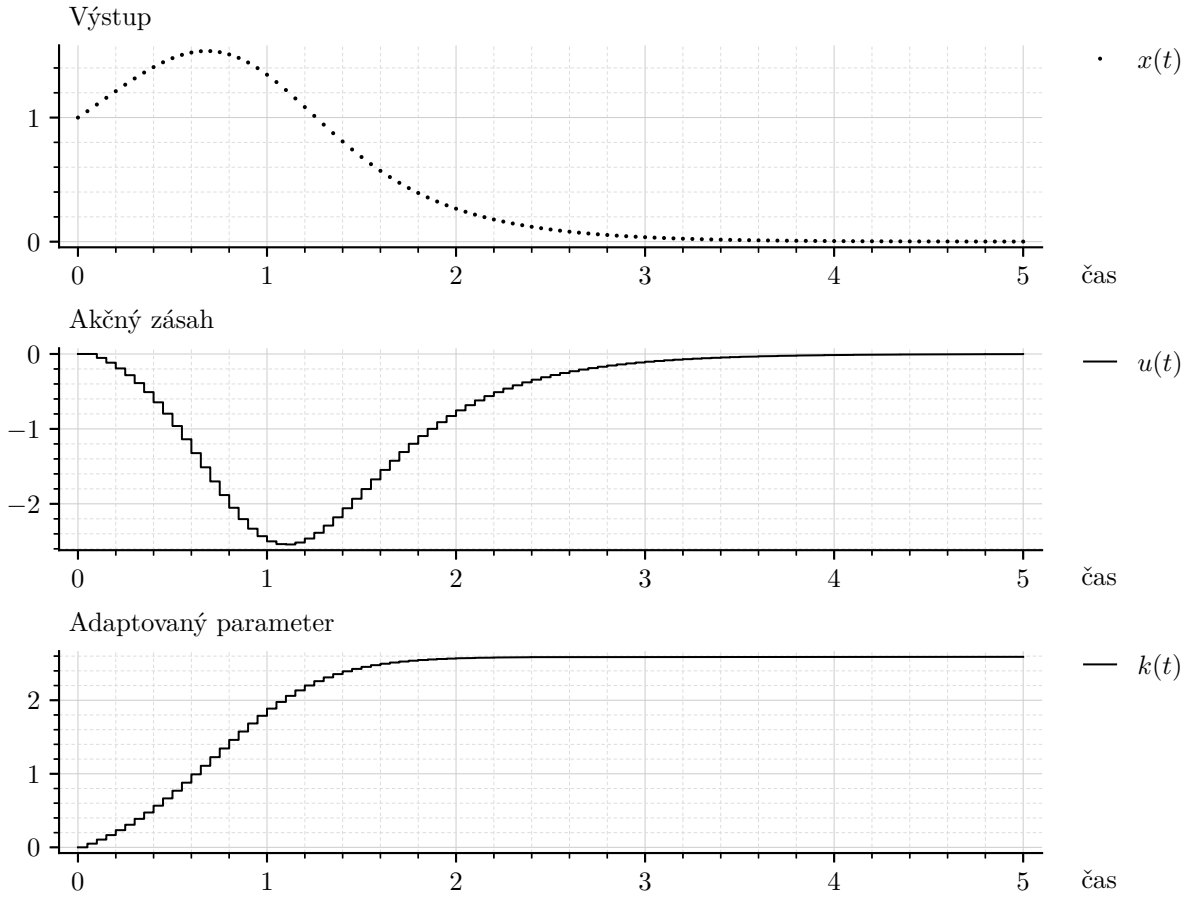
```
118 # Nastavenia simulacie
119 sim_t_start = 0
120 sim_t_final = 5
121 sim_T_s = 0.05
122 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
123
124 param_a = 1
125
126 # Simulacia
127 t_log, x_log, u_log, k_log, = fcn_simSch_03(sim_t_start,
128                                              sim_T_s,
129                                              sim_finalIndex,
130                                              param_a,
131                                              )
```

```
132
```

Obrázok, ktorý zobrazuje okrem výstupnej veličiny riadeného systému aj iné veličiny potrebné pre posúdenie stability celého uzavretého regulačného obvodu.

Výpis kódu 13: Súbor `ar02_file03.py`

```
140 figName = 'figsc_ar02_f03_f01'
141 figNameNum = 0
142
143 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
144
```



Obr. 7

V čom je tá implementácia „realistická“? To je na diskusiu, ktorú tu autor neuvádza pretože je lenivý a naničhodný.

O samonastavujúcom sa regulátore

Obsah

1 Konkrétny príklad

Princíp samonastavujúceho sa regulátora (Self-Tuning Regulator, i.e. STR) spočíva v tom, že v každej perióde vzorkovania sa identifikujú parametre modelu riadeného systému a následne, s využitím identifikovaných parametrov modelu, sa pomocou určitej metódy vypočítajú parametre regulátora.

Hneď v prvej vete sme použili pojem perióda vzorkovania. Je teda zrejmé, že celý riadiaci systém bude pracovať v diskrétnej časovej oblasti. Modelom riadeného systému bude ARX (Auto Regressive eXogenous) model. Štruktúra riadenia bude tzv. „trojzložková“. Tromi zložkami sú polynómy R, S a T. Tieto polynómy majú svoje koeficienty, ktoré sú zároveň parametrami riadiacej štruktúry, vlastne parametrami regulátora.

Skonkretizujme teraz tento všeobecný popis.

1.1 Riadený systém

Riadený systém nech predstavuje prenosová funkcia v tvare

$$G(s) = \frac{0,15}{s^2 + 0,3s + 0,2} \tag{1}$$

Ide o prenosovú funkciu druhého rádu, ktorá má v čitateli polynóm nultého stupňa, teda nemá nuly.

1.2 ARX model

Nech modelom riadeného systému je ARX (AutoRegressive eXogenous) model. Vo všeobecnosti ARX model má tvar

$$A(z^{-1})y(k) = B(z^{-1})u(k) + \xi(k) \tag{2}$$

kde

$$\begin{aligned} B(z^{-1}) &= b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} \\ A(z^{-1}) &= 1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a} \end{aligned} \tag{3}$$

a $\xi(k)$ predstavuje poruchy a šum. V ďalšom budeme uvažovať $\xi(k) = 0$. ARX model vyjadrený v tvare diferenčnej rovnice:

$$y(k) = -a_1 y(k-1) - \ldots - a_{n_a} y(k-n_a) + b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) \tag{4}$$

Nech modelom sústavy je diferenčná rovnica v tvare (4), kde hodnoty $n_a = 2$ a $n_b = 2$. Potom táto diferenčná rovnica má tvar

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) \tag{5}$$

V maticovom zápise:

$$y(k) = h^{\mathsf{T}}\Theta \tag{6}$$

kde $h^{\mathsf{T}} = \begin{bmatrix} -y(k-1) & -y(k-2) & u(k-1) & u(k-2) \end{bmatrix}$ a $\Theta = \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \end{bmatrix}^{\mathsf{T}}$. Neznámymi parametrami modelu teda sú koeficienty a_1, a_2, b_1 a b_2.

1.3 Štruktúra riadenia

Štruktúra riadenia je zrejmá zo zápisu „trojzložkového“ zákona riadenia

$$\begin{aligned} R(z^{-1})u(k) &= T(z^{-1})r(k) - S(z^{-1})y(k) \\ u(k) &= \frac{T(z^{-1})}{R(z^{-1})}r(k) - \frac{S(z^{-1})}{R(z^{-1})}y(k) \end{aligned} \tag{7}$$

kde R, S a T sú polynómy v tvare

$$\begin{aligned} R(z^{-1}) &= 1 + r_1 z^{-1} + r_2 z^{-2} + \ldots + r_{n_r} z^{-n_r} \\ S(z^{-1}) &= s_0 + s_1 z^{-1} + s_2 z^{-2} + \ldots + s_{n_s} z^{-n_s} \\ T(z^{-1}) &= t_0 + t_1 z^{-1} + t_2 z^{-2} + \ldots + t_{n_t} z^{-n_t} \end{aligned} \tag{8}$$

Ako už bolo uvedené, koeficienty polynómov sú parametrami regulátora. Počet parametrov regulátora závisí od stupňa jednotlivých polynómov. Pre tento príklad sú stupne polynómov nasledovné: $n_r = 1$, $n_s = 1$ a $n_t = 0$. Potom počet parametrov regulátora je $n_r + n_s + 1 + n_t + 1$. Teda $1 + 1 + 1 + 0 + 1 = 4$.

Zákon riadenia je možné zapísať aj v tvare diferenčnej rovnice

$$u(k) = -r_1 u(k-1) - s_0 y(k) - s_1 y(k-1) + t_0 r(k) \tag{9}$$

1.4 Výpočet parametrov regulátora

Metóda, pomocou ktorej sa v tomto prípade budú počítať parametre regulátora bude *Pole placement* – Metóda rozmiestňovania pólov uzavretého regulačného obvodu (URO). Ako už názov naznačuje, metóda spočíva v predpísaní rozmiestnenia pólov URO. Inými slovami, predpisujú sa korene charakteristického polynómu URO. Voľbou polohy pólov URO je možné zvoliť dynamické vlastnosti URO. Poloha pólov sa zadáva zvolením *želaného polynómu*, ktorý má také korene, aké si želáme. Ak napr. žiadame, aby URO mal dynamiku druhého rádu, tak želaný polynóm bude 2. stupňa.

Pripomeňme, že v tomto prípade ide o „diskrétny“ polynóm, pretože riadiaci systém pracuje v diskrétnej oblasti, t.j. model sústavy je „diskrétny“ a aj zákon riadenia je „diskrétny“.

Parametre regulátora sa vypočítajú z porovnania koeficientov pri rovnakých mocninách charakteristického polynómu URO a želaného polynómu. Charakteristický polynóm URO sa bude skladať z polynómov modelu sústavy a zo zložiek regulátora, čo sú polynómy vystupujúce v zákone riadenia. Koeficienty polynómov modelu sústavy považujeme za známe, pretože ich získame identifikáciou. Koeficienty polynómov zo zákona riadenia – parametre regulátora sú neznáme. Získame ich riešením rovnice, kde na jednej strane je charakteristický polynóm URO a na druhej strane je želaný polynóm. Takáto rovnica sa nazýva *Diofantická rovnica*.

2 Identifikácia parametrov lineárneho modelu

Uvažujeme lineárny model v tvare (6). Parametre modelu budeme určovať *rekurzívnou metódou najmenších štvorcov*. Najskôr však odvodíme tzv. Off – line odhad parametrov modelu.

2.1 Metóda najmenších štvorcov

Predpokladajme, že sme spravili N experimentov – meraní. Napr.: na vstup sústavy sme priviedli „vhodný“ signál a s nejakou periódou vzorkovania sme zaznamenávali hodnoty vstupného aj výstupného (zo sústavy) signálu. Teda máme nameraných N hodnôt vstupu, ku ktorým prislúcha N hodnôt výstupu.

V i-tom experimente sme namerali hodnotu výstupného signálu y_i. Nech model má odhadnuté „nejaké“ parametre. Potom ak privedieme na vstup modelu hodnotu u_i, čo je nameraná hodnota vstupného signálu prislúchajúca k y_i, na výstupe modelu bude odhad $\hat{y}_i$. Tento odhad bude vo všeobecnosti rozdielny od nameranej hodnoty. Namiesto „nejakých“ hodnôt parametrov modelu určme také, pre ktoré bude platiť, že suma štvorcov odchýlok vo všetkých nameraných bodoch bude minimálna.

Odchýlka v i-tom experimente je $e_i = y_i - \hat{y}_i$. Odhad výstupu v i-tom experimente je $\hat{y}_i = h_i^\mathsf{T}\Theta$, čo je rovnaký zápis ako v (6), len namiesto času k je použitý index i. Všetky odchýlky v N meraniach sú:

$$\begin{aligned} e_1 &= y_1 - h_1^\mathsf{T}\Theta \\ &\vdots \\ e_N &= y_N - h_N^\mathsf{T}\Theta \end{aligned} \tag{10}$$

čo možno zapísať aj takto:

$$e = y - \begin{bmatrix} h_1^\mathsf{T} \\ \vdots \\ h_N^\mathsf{T} \end{bmatrix} \Theta \tag{11}$$

Vektor h_1^T, za predpokladu, že začiatočný čas je $t_0 = 0$ a prvá vzorka vstupu a výstupu bola nameraná v čase $t(k=1) = 1 \cdot T_{vz}$ je $h_1^\mathsf{T} = [-y(0) \quad 0 \quad u(0) \quad 0]$. Analogicky $h_5^\mathsf{T} = h(5)^\mathsf{T} = [-y(4) \quad -y(3) \quad u(4) \quad u(3)]$.

Pre lepšiu názornosť nech je nameraných 5 vzoriek (a tiež hodnoty $y(0)$, $u(0)$).

Potom všetky odchýlky je možné zapísať takto:

$$e = y - \begin{bmatrix} -y(0) & 0 & u(0) & 0 \\ -y(1) & -y(0) & u(1) & u(0) \\ -y(2) & -y(1) & u(2) & u(1) \\ -y(3) & -y(2) & u(3) & u(2) \\ -y(4) & -y(3) & u(4) & u(3) \end{bmatrix} \Theta \tag{12}$$

Všeobecný zápis

$$e = y - H\Theta \tag{13}$$

Ako už bolo uvedené, vektor Θ, čo je vektor parametrov modelu určíme tak, aby suma štvorcov odchýlok bola minimálna. Zaveďme účelovú funkciu v tvare

$$J(\Theta) = \frac{1}{2} e^{\mathsf{T}} e = \frac{1}{2} (y - H\Theta)^{\mathsf{T}} (y - H\Theta) \tag{14}$$

Je potrebné nájsť extrém tejto účelovej funkcie, čo znamená derivovať ju podľa vektora parametrov modelu.

$$J(\Theta) = \frac{1}{2} (y - H\Theta)^{\mathsf{T}} (y - H\Theta) = \frac{1}{2} \left(y^{\mathsf{T}} y - y^{\mathsf{T}} H\Theta - \Theta^{\mathsf{T}} H^{\mathsf{T}} y + \Theta^{\mathsf{T}} H^{\mathsf{T}} H\Theta \right) \tag{15}$$

S využitím rovnosti

$$\nabla_x \left(x^{\mathsf{T}} a \right) = \nabla_x \left(a^{\mathsf{T}} x \right) = a \tag{16}$$

máme

$$\begin{aligned} \nabla_\Theta \left(y^{\mathsf{T}} y \right) &= 0 \\ \nabla_\Theta \left(y^{\mathsf{T}} H\Theta \right) &= \left(y^{\mathsf{T}} H \right)^{\mathsf{T}} = H^{\mathsf{T}} y \\ \nabla_\Theta \left(\Theta^{\mathsf{T}} H^{\mathsf{T}} y \right) &= H^{\mathsf{T}} y \\ \nabla_\Theta \left(\Theta^{\mathsf{T}} H^{\mathsf{T}} H\Theta \right) &= \left(H^{\mathsf{T}} H\Theta + \left(\Theta^{\mathsf{T}} H^{\mathsf{T}} H \right)^{\mathsf{T}} \right) = H^{\mathsf{T}} H\Theta + H^{\mathsf{T}} H\Theta \end{aligned}$$

teda

$$\begin{aligned} \nabla_\Theta (J) &= \frac{1}{2} \left(-H^{\mathsf{T}} y - H^{\mathsf{T}} y + H^{\mathsf{T}} H\Theta + H^{\mathsf{T}} H\Theta \right) \\ &= \frac{1}{2} \left(-2H^{\mathsf{T}} y + 2H^{\mathsf{T}} H\Theta \right) \\ &= -H^{\mathsf{T}} y + H^{\mathsf{T}} H\Theta \end{aligned} \tag{17}$$

V extréme je hodnota $\nabla_\Theta (J)$ nulová, teda

$$\begin{aligned} 0 &= -H^{\mathsf{T}} y + H^{\mathsf{T}} H\Theta \\ H^{\mathsf{T}} H\Theta &= H^{\mathsf{T}} y \end{aligned} \tag{18}$$

a nakoniec

$$\Theta = \left(H^{\mathsf{T}} H \right)^{-1} H^{\mathsf{T}} y \tag{19}$$

Rovnica (19) sa nazýva Gaussov vzorec.

Keďže $\nabla_\Theta \nabla_\Theta (J) = \left(H^{\mathsf{T}} H \right) = H^{\mathsf{T}} H$ a $H^{\mathsf{T}} H = R$ je kladne definitná, pretože H je nenulová a teda $H^{\mathsf{T}} H$ je symetrická a kladne definitná, potom nájdený extrém je minimum. Matica R sa nazýva informačná matica a $P = R^{-1}$ sa nazýva disperzná matica (alebo aj Kovariančná matica) s $(n_a + n_b)$ riadkami a $(n_a + n_b)$ stĺpcami.

Dosadením do Gaussovho vzorca získame Off–line odhad parametrov modelu.

2.2 Rekurzívna metóda najmenších štvorcov

Predpokladajme, že poznáme odhad parametrov modelu v predchádzajúcom kroku $(k-1)$ a chceme získať odhad parametrov modelu v aktuálnom kroku k. Ak poznáme odhad parametrov v kroku $(k-1)$, je zrejmé, že poznáme aj maticu $P(k-1)$.

Pre lepšiu názornosť nech situácia je takáto: Aktuálny krok je $k = 2$. V kroku $(k-1)$, teda v kroku $k = 1$ bola matica $P(k-1)$. Ak prejdeme z kroku $(k-1)$ do

kroku k, znemená to pridať nový riadok matice H. Novým riadkom je vektor $h^\mathsf{T}(k)$. Pre krok k môžme písať Gausov vzorec:

$$\Theta(k) = \left(\begin{bmatrix} H^\mathsf{T}(k-1) & h(k) \end{bmatrix} \begin{bmatrix} H(k-1) \\ h^\mathsf{T}(k) \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} H^\mathsf{T}(k-1) & h(k) \end{bmatrix} y(k) \tag{20}$$

Odkiaľ matica $P(k)$ je

$$\begin{aligned} P(k) &= \left(\begin{bmatrix} H^\mathsf{T}(k-1) & h(k) \end{bmatrix} \begin{bmatrix} H(k-1) \\ h^\mathsf{T}(k) \end{bmatrix} \right)^{-1} \\ &= \left(H^\mathsf{T}(k-1)H(k-1) + h(k)h^\mathsf{T}(k) \right)^{-1} \\ &= \left(P(k-1)^{-1} + h(k)h^\mathsf{T}(k) \right)^{-1} \end{aligned} \tag{21}$$

V rovnici (21) sa vyskytuje inverzia matice, ktorej rozmer závisí od počtu parametrov modelu. Tento počet môže byť veľký, a inverzia takto veľkej matice môže byť nerealizovateľná. Použitie Woodburryho lemmy o inverzií matíc rieši tento problém. Platí (v tomto texte nebudeme uvádzať dôkaz Woodburryho lemmy)

$$(A + BD)^{-1} = A^{-1} - A^{-1}B \left(DA^{-1}B + 1 \right)^{-1} DA^{-1} \tag{22}$$

Použitím (22) prejde (21) do tvaru

$$\begin{aligned} P(k) &= P(k-1) - P(k-1)h(k) \left(1 + h^\mathsf{T}(k)P(k-1)h(k) \right)^{-1} \cdot h^\mathsf{T}(k)P(k-1) \\ &= P(k-1) - Y(k)h^\mathsf{T}(k)P(k-1) \end{aligned} \tag{23}$$

kde

$$Y(k) = \frac{P(k-1)h(k)}{1 + h^\mathsf{T}(k)P(k-1)h(k)}$$

je tzv. „zosilnenie“. Rovnica (23) je rekurzívnym vzťahom pre výpočet novej matice P z predchádzajúcej matice P.

Odhad parametrov modelu v kroku k je

$$\Theta(k) = P(k)H^\mathsf{T}(k)y(k) = P(k) \sum_{i=1}^{k} h(i)y_i = P(k) \left(\sum_{i=1}^{k-1} h(i)y_i + h(k)y(k) \right) \tag{24}$$

Z (21) je zrejmé že platí

$$P^{-1}(k) = P(k-1) + h(k)h(k)^\mathsf{T} \tag{25a}$$

$$P^{-1}(k-1) = P^{-1}(k) - h(k)h^\mathsf{T}(k) \tag{25b}$$

Potom

$$\begin{aligned} \sum_{i=1}^{k-1} h^\mathsf{T}(i)y_i &= P^{-1}(k-1)\Theta(k-1) = \left(P^{-1}(k) - h(k)h^\mathsf{T}(k) \right) \Theta(k-1) \\ &= P^{-1}(k)\Theta(k-1) - h(k)h^\mathsf{T}(k)\Theta(k-1) \end{aligned} \tag{26}$$

čo umožní písať odhad parametrov modelu v kroku k v tvare

$$\begin{aligned} \Theta(k) &= P(k) \left(P^{-1}(k)\Theta(k-1) - h(k)h^\mathsf{T}(k)\Theta(k-1) + h(k)y(k) \right) \\ &= P(k)P^{-1}(k)\Theta(k-1) - P(k)h(k)h^\mathsf{T}(k)\Theta(k-1) + P(k)h(k)y(k) \\ &= I\Theta(k-1) + P(k)h(k) \left(y(k) - h^\mathsf{T}(k)\Theta(k-1) \right) \\ &= \Theta(k-1) + P(k)h(k)e(k) \end{aligned} \tag{27}$$

Tabuľka 1: Algoritmus rekurzívnej MNŠ

1.	Odchýlka (rezíduum)	$e(k) = y(k) - h^{\mathsf{T}}(k)\Theta(k-1)$
2.	Zosilnenie	$Y(k) = \dfrac{P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}$
3.	Kovariančná matica	$P(k) = P(k-1) - Y(k)h^{\mathsf{T}}(k)P(k-1)$
4.	Nový odhad parametrov	$\Theta(k) = \Theta(k-1) + Y(k)e(k)$

Rovnicu (27) môžeme priviesť aj do iného tvaru. Platí:

$$\begin{aligned}
\Theta(k) &= \Theta(k-1) + P(k)h(k)e(k) \\
&= \Theta(k-1) + \left(P(k-1) - Y(k)h^{\mathsf{T}}(k)P(k-1)\right)h(k)e(k) \\
&= \Theta(k-1) + \left(P(k-1)h(k) - \frac{P(k-1)h(k)h^{\mathsf{T}}(k)P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k) \\
&= \Theta(k-1) \\
&+ \left(\frac{P(k-1)h(k) + P(k-1)h(k)h^{\mathsf{T}}(k)P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right. \\
&- \left.\frac{P(k-1)h(k)h^{\mathsf{T}}(k)P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k) \\
&= \Theta(k-1) + \left(\frac{P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k) \\
&+ \left(\frac{P(k-1)h(k)h^{\mathsf{T}}(k)P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k) \\
&- \left(\frac{P(k-1)h(k)h^{\mathsf{T}}(k)P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k) \\
&= \Theta(k-1) + \left(\frac{P(k-1)h(k)}{1 + h^{\mathsf{T}}(k)P(k-1)h(k)}\right)e(k)
\end{aligned} \tag{28}$$

Potom

$$\Theta(k) = \Theta(k-1) + Y(k)e(k) \tag{29}$$

Rovnica (29) je rekurzívnym vzťahom pre výpočet nových parametrov modelu z predchádzajúcich hodnôt parametrov modelu.

2.2.1 Súhrn

Algoritmus rekurzívnej metódy najmenších štvorcov je zhrnutý v Tabuľke 1.

2.2.2 Štart algoritmu

Disperzná matica má začiatočný tvar

$$P(0) = \alpha I \tag{30}$$

kde I je jednotková matica rovnakého rozmeru ako P a α je reálne číslo napríklad z intervalu $\langle 10,\ 10^6 \rangle$.

Začiatočné hodnoty parametrov modelu môžu byť napríklad nulové, t.j. vektor Θ je nulový vektor príslušnej dĺžky:

$$\Theta(0) = 0 \tag{31}$$

Nie je to však pravidlo a začiatočné hodnoty parametrov modelu môžu byť v princípe akékoľvek. Tieto hodnoty sa spravidla využívajú v ďalších výpočtoch a tak napríklad môže vzniknúť požiadavka, že niektorý z parametrov nemôže byť nulový (napr. pre delenie nulou) a podobne. Tiež môže existovať približný, hrubý odhad týchto hľadaných (identifikovaných) parametrov a tento je potom často výhodné použiť ako začiatočný.

Tabuľka 2: Algoritmus rekurzívnej MNŠ s exponenciálnym zabúdaním

1.	Odchýlka (rezíduum)	$e(k) = y(k) - h^{\mathsf{T}}(k)\Theta(k-1)$
2.	Zosilnenie	$Y(k) = \dfrac{P(k-1)h(k)}{\lambda + h^{\mathsf{T}}(k)P(k-1)h(k)}$
3.	Kovariančná matica	$P(k) = \dfrac{1}{\lambda}\left(P(k-1) - Y(k)h^{\mathsf{T}}(k)P(k-1)\right)$
4.	Nový odhad parametrov	$\Theta(k) = \Theta(k-1) + Y(k)e(k)$

2.2.3 Modifikácia algoritmu - zabúdanie

V účelovej funkcii (14) sa nepredpokladá žiadne váhovanie jednotlivých prvkov vektora odchýlok e. Všetky prvky majú rovnakú váhu. Z pohľadu rekurzívneho algoritmu to znamená, že najstaršie odchýlky majú rovnakú váhu ako najnovšie. Často však môže byť veľmi výhodné ak by novšie vzorky, teda novšie zistené odchýlky mali vyššiu váhu ako staršie. Inými slovami výhodné by bolo zabúdať na staršie odchýlky a uvažovať len tie novšie.

V účelovej funkcii (14) je možné uvažovať váhovaciu maticu W nasledovne:

$$J(\Theta) = \frac{1}{2}e^{\mathsf{T}}We \tag{32}$$

kde matica

$$W = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_N \end{bmatrix} \tag{33}$$

umožní realizovať váhovanie jednotlivých štvorcov odchýlok.

Ak zvolíme váhovacie koeficienty ako $w_i = \lambda^{N-i}$ potom sa takého váhovanie nazýva exponenciálne zabúdanie. Číslo λ sa volí z intervalu $(0, 1\rangle$, pričom typické hodnoty sú $\lambda = 0,99$ či $\lambda = 0,95$. Potom je zrejmé, že najnovšia vzorka, najnovší štvorec odchýlky, má váhu (je prenásobený číslom) $w_N = \lambda^{N-N} = 1$. Všetky ostatné váhovacie koeficienty majú nižšiu hodnotu (hodnota exponenciálne klesá ako sa zmenšuje poradové číslo i).

Ak aplikujeme ten istý postup pre získanie rekurzívneho algoritmu MNŠ ako v prípade bez exponenciálneho zabúdania, tak verzia so zabúdaním vedie na algoritmus sumarizovaný v tabuľke 2.

3 Cvičenie druhé

1. Zrealizujme priebežnú identifikáciu parametrov ARX modelu tak ako to predpokladá konkrétny príklad v časti 1. Vyskúšajte verziu bez exponenciálneho zabúdania a s exponenciálnym zabúdaním.

3.1 Ukážka simulačnej schémy pre simuláciu daného riadeného systému

Cieľom nasledujúceho je zostaviť (univerzálnu) simulačnú schému, do ktorej je možné následne doplniť v podstate akýkoľvek riadiaci systém.

Simulačná schéma v tomto prípade realizuje len simuláciu samotného riadeného systému. Vstupný signál riadeného systému je tu zvolený (daný vopred), nie je generovaný riadiacim systémom.

Riadený systém nech predstavuje prenosová funkcia v tvare

$$G(s) = \frac{0,15}{s^2 + 0,3s + 0,2} \tag{34}$$

Ide o prenosovú funkciu druhého rádu, ktorá má v čitateli polynóm nultého stupňa, teda nemá nuly.

Parametre riadeného systému vo forme vstupného vektora matice dynamiky potom sú:

Výpis kódu 1: Súbor ar03_pr01.py

```
27 A = np.array([[0, 1], [-0.2, -0.3]])
28 b = np.array([[0], [0.15]])
```

Funkcia, ktorá realizuje diferenciálne rovnice riadeného systému, nech je nasledovná:

Výpis kódu 2: Súbor ar03_pr01.py

```
34 def fcn_difRovnice(x, t, u):
35
36     dotx = np.dot(A,x) + np.dot(b,u)
37
38     return dotx
```

Simulačnú schému nech realizuje nasledujúca funkcia:

Výpis kódu 3: Súbor ar03_pr01.py

```
47 def fcn_simSch_01_zaklad(t_start, T_s, finalIndex, sig_u_ext):
48
49     #----------------------------------------
50     t_log = np.zeros([finalIndex, 1])
51     t_log[0,:] = t_start
52
53     #----------------------------------------
54     x_0 = np.array([0, 0])
55
56     x_log = np.zeros([finalIndex, len(x_0)])
57     x_log[0,:] = x_0
58
59     #----------------------------------------
60
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         u = sig_u_ext[idx-1,:]
70
71         odeOut = odeint(fcn_difRovnice,
72                         x_log[idx-1,:],
73                         timespan,
74                         args=(u,)
75                         )
76
77         x_log[idx,:] = odeOut[-1,:]
78         t_log[idx,:] = timespan[-1]
79
80     return [t_log, x_log]
```

Nastavenia potrebné pre samotnú simuláciu a vygenerovanie signálov, ktoré sa používajú pri simulácii (ktoré sú dopredu známe - dané):

Výpis kódu 4: Súbor ar03_pr01.py

```
89 # Nastavenia simulacie
90
91 sim_t_start = 0
92 sim_t_final = 200
93 sim_T_s = 0.1
94 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
```

Pre simuláciu je potrebné vytvoriť vstupný signál $u(t)$ pre riadený systém. Nech je nasledovný:

Výpis kódu 5: Súbor ar03_pr01.py

```
103 tab_delt_u = np.array([
104                         [0, 0],
105                         [1, 1],
```

```
                        [50, 0],
                        [100, -1],
                        [150, 0],
                        ])


sig_delt_u = np.zeros([sim_finalIndex, 1])
for idx in range(sig_delt_u.shape[0]):
    lastValue = tab_delt_u[:,1][tab_delt_u[:,0]<=idx*sim_T_s ][-1]
    sig_delt_u[idx] = lastValue


sig_u_ext = sig_delt_u
```

Vstupný signál $u(t)$ je zobrazený na obr. 1.

Spustenie simulácie:

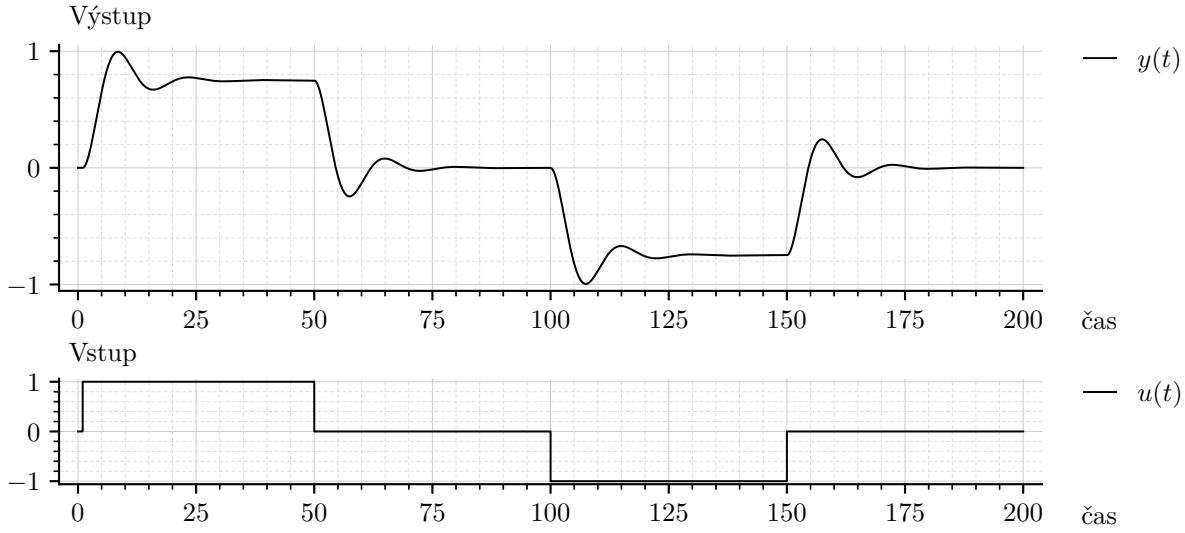
Výpis kódu 6: Súbor `ar03_pr01.py`

```
# Spustenie simulacie

t_log, x_log = fcn_simSch_01_zaklad(
                    sim_t_start,
                    sim_T_s,
                    sim_finalIndex,
                    sig_u_ext,
                    )
```

Nakreslenie obrázka (pre prehľadnosť je kód v samostatnom súbore):

Výpis kódu 7: Súbor `ar03_pr01.py`

```
# Obrazok

figName = 'figsc_ar03_fig01'
figNameNum = 0

exec(open('./misc/' + figName + '.py', encoding='utf-8').read())

```



Obr. 1

3.2 Doplnenie algoritmu RMNŠ do simulačnej schémy

V predchádzajúcom bola vytvorená simulačná schéma pre simuláciu uvažovaného riadeného systému. Tu je cieľom doplniť do simulačnej schémy algoritmus RMNŠ.

Pre úplnosť, ARX (AutoRegressive eXogenous) model, ktorý je identifikovaný (klasickou RMNŠ), je vo všeobecnosti nasledovný:

$$A(z^{-1})y(k) = B(z^{-1})u(k) + \xi(k) \tag{35}$$

kde

$$B(z^{-1}) = b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b}$$
$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a} \tag{36}$$

a $\xi(k)$ predstavuje poruchy a šum. V ďalšom budeme uvažovať $\xi(k) = 0$. ARX model vyjadrený v tvare diferenčnej rovnice:

$$y(k) = -a_1 y(k-1) - \ldots - a_{n_a} y(k-n_a) + b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) \tag{37}$$

Nech modelom riadeného systému je diferenčná rovnica v uvedenom tvare, kde hodnoty $n_a = 2$ a $n_b = 2$. Potom táto diferenčná rovnica má tvar

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) \tag{38}$$

V maticovom zápise:

$$y(k) = h^{\mathsf{T}} \Theta \tag{39}$$

kde $h^{\mathsf{T}} = \begin{bmatrix} -y(k-1) & -y(k-2) & u(k-1) & u(k-2) \end{bmatrix}$ a $\Theta = \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \end{bmatrix}^{\mathsf{T}}$. Neznámymi parametrami modelu teda sú koeficienty a_1, a_2, b_1 a b_2.

Takpovediac diferenciálne rovnice riadeného systému ostavajú samozrejme rovnaké ako sme ich realizovali v predchádzajúcom, teda aj tu („na začiatku skriptu“) platí výpis kódu 1 a 2.

Simulačnú schému nech realizuje nasledujúca funkcia:

Výpis kódu 8: Súbor `ar03_pr02.py`

```
47 def fcn_simSch_02_lenRMNS(t_start, T_s, finalIndex, sig_u_ext):
48
49     #----------------------------------------
50     t_log = np.zeros([finalIndex, 1])
51     t_log[0,:] = t_start
52
53     #----------------------------------------
54     x_0 = np.array([0, 0])
55
56     x_log = np.zeros([finalIndex, len(x_0)])
57     x_log[0,:] = x_0
58
59     #----------------------------------------
60
61     u_log = np.zeros([finalIndex, 1])
62
63     #----------------------------------------
64
65     RMNS_theta_0 = np.array([[ 0.001],
66                              [ 0.001],
67                              [ 0.001],
68                              [ 0.001]])
69
70     RMNS_theta_log = np.zeros([finalIndex, len(RMNS_theta_0)])
71     RMNS_theta_log[0,:] = RMNS_theta_0.reshape(1,-1)
72
73     RMNS_P_0 = np.identity(4) * 10**6
74
75     RMNS_P_log = np.zeros([finalIndex, RMNS_P_0.size])
76     RMNS_P_log[0,:] = RMNS_P_0.reshape(1,-1)
77
78     #----
79
80     RMNS_y_predict_log = np.zeros([finalIndex, 1])
81
82     #----------------------------------------
83     timespan = np.zeros(2)
84     for idx in range(1, int(finalIndex)):
85
86         timespan[0] = t_log[idx-1,:]
87         timespan[1] = t_log[idx-1,:] + T_s
88
89         odeOut = odeint(fcn_difRovnice,
90                         x_log[idx-1,:],
```

```
                        timespan,
                        args=(u_log[idx-1,:],)
                        )

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

        #---------------------
        # ALGORITMUS RMNS
        y_k = x_log[idx,0]

        h_k = np.array([[-x_log[idx-1,0]],
                        [-x_log[idx-2,0]], # pozor na to idx-2 !!!
                        [u_log[idx-1,0]],
                        [u_log[idx-2,0]], # pozor na to idx-2 !!!
                        ])

        theta_km1 = RMNS_theta_log[idx-1,:].reshape(4,-1)
        P_km1 = RMNS_P_log[idx-1,:].reshape(4,4)

        #----------
        lambdaKoef = 1.0

        e_k = y_k - np.matmul(h_k.T, theta_km1)
        Y_k =  np.matmul(P_km1, h_k) / (lambdaKoef + np.matmul(np.
    matmul(h_k.T, P_km1), h_k))
        P_k = (1/lambdaKoef) * (P_km1 - np.matmul(np.matmul(Y_k, h_k.
    T), P_km1))
        theta_k = theta_km1 + Y_k * e_k

        #----

        RMNS_theta_log[idx,:] = theta_k.reshape(1,-1)
        RMNS_P_log[idx,:] = P_k.reshape(1,-1)

        #----------

        RMNS_y_predict_log[idx,:] = np.matmul(h_k.T, theta_km1)

        #---------------------
        u_log[idx,:] = sig_u_ext[idx-1,:]

    return [t_log, x_log, RMNS_y_predict_log, RMNS_theta_log]
```

V uvedenej simulačnej schéme je implementovaný RMNŠ algoritmus, ktorého výstupom je vektor parametrov `theta_k` a následne je tiež vypočítaná (v každom cykle) jednokroková predikcia výstupného signálu zapisovaná do vektora `RMNS_y_predict_log`.

Všimnime si tiež napríklad, že faktor zabúdania λ (premenná `lambdaKoef`) je nastavený na hodnotu $\lambda = 1$, teda algoritmus nevyužíva zabúdanie.

Nastavenia potrebné pre samotnú simuláciu a vygenerovanie signálov, ktoré sa používajú pri simulácii (ktoré sú dopredu známe - dané):

Výpis kódu 9: Súbor `ar03_pr02.py`

```
# Nastavenia simulacie

sim_t_start = 0
sim_t_final = 55
sim_T_s = 0.25
sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
```

Pre simuláciu je potrebné vytvoriť vstupný signál $u(t)$ pre riadený systém. Nech je nasledovný:

Výpis kódu 10: Súbor `ar03_pr02.py`

```
# Preddefinovane signaly

period_time = 40
period_tab = np.array([
                       [0, 1],
                       [10, 0],
                       [20, -1],
                       [30, 0],
                       ])

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

```

```
166 for period in range(int(sim_t_final/period_time) + 1):
167
168
169     for idx in range( int((period*period_time)/sim_T_s), int((period*
        period_time + period_time)/sim_T_s)):
170
171         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
        period_time))<=idx*sim_T_s ][-1]
172         try:
173             sig_vysl[idx] = lastValue
174         except:
175             break
176
177 sig_u_ext = sig_vysl
```

Spustenie simulácie:

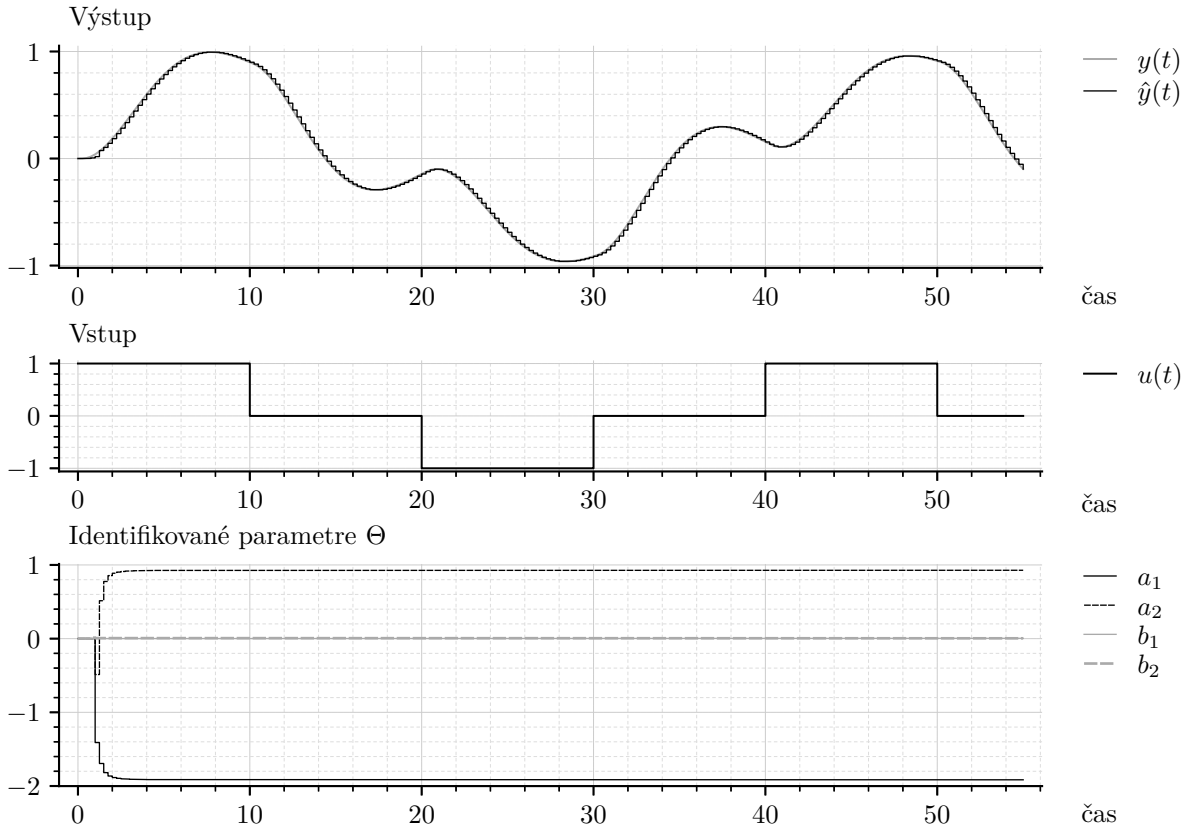
Výpis kódu 11: Súbor `ar03_pr02.py`

```
191 # Spustenie simulacie
192
193 t_log, x_log, RMNS_y_predict_log, RMNS_theta_log =
        fcn_simSch_02_lenRMNS(
194     sim_t_start,
195     sim_T_s,
196     sim_finalIndex,
197     sig_u_ext,
198     )
```

Nakreslenie obrázku (pre prehľadnosť je kód v samostatnom súbore):

Výpis kódu 12: Súbor `ar03_pr02.py`

```
206 # Obrazok
207
208 figName = 'figsc_ar03_fig02'
209 figNameNum = 0
210
211 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
212
```



Obr. 2

Pre zaujímavosť, priebežne identifikované parametre Θ sú zapisované do poľa `RMNS_theta_log`. Posledný riadok v tomto poli je:

Výpis kódu 13: Súbor ar03_pr02.py

```
220 print(RMNS_theta_log[-1,:])
```

```
[-1.91569536  0.92772642  0.00456786  0.00445568]
```

3.3 Algoritmus RMNŠ pri zašumených dátach

V predchádzajúcom bola vytvorená simulačná schéma pre simuláciu uvažovaného riadeného systému a bol do nej doplnený algoritmus RMNŠ.

Výstupná veličina samotného simulovaného riadeného systému je, pochopiteľne, bez šumu. Tu je cieľom preskúmať ako je RMNŠ schopný vysporiadať sa s prítomnosťou šumu v dátach výstupnej veličiny.

Aj tu platí, že diferenciálne rovnice riadeného systému ostávajú samozrejme rovnaké ako sme ich realizovali v predchádzajúcom, teda aj tu („na začiatku skriptu“) platí výpis kódu 1 a 2.

Simulačnú schému nech realizuje nasledujúca funkcia:

Výpis kódu 14: Súbor ar03_pr03.py

```
47 # Simulacna schema:
48
49 def fcn_simSch_02_lenRMNS_noise(t_start, T_s, finalIndex, sig_u_ext,
       lambdaKoef):
50
51     #----------------------------------------
52     t_log = np.zeros([finalIndex, 1])
53     t_log[0,:] = t_start
54
55     #----------------------------------------
56     x_0 = np.array([0, 0])
57
58     x_log = np.zeros([finalIndex, len(x_0)])
59     x_log[0,:] = x_0
60
61
62     y_log_noise = np.zeros([finalIndex, 1])
63     y_log_noise[0,0] = x_log[0,0]
64
65     #----------------------------------------
66
67     u_log = np.zeros([finalIndex, 1])
68
69     #----------------------------------------
70
71     RMNS_theta_0 = np.array([[ 0.001],
72                              [ 0.001],
73                              [ 0.001],
74                              [ 0.001]])
75
76
77     RMNS_theta_log = np.zeros([finalIndex, len(RMNS_theta_0)])
78     RMNS_theta_log[0,:] = RMNS_theta_0.reshape(1,-1)
79
80     RMNS_P_0 = np.identity(4) * 10**2
81
82     RMNS_P_log = np.zeros([finalIndex, RMNS_P_0.size])
83     RMNS_P_log[0,:] = RMNS_P_0.reshape(1,-1)
84
85     RMNS_y_predict_log = np.zeros([finalIndex, 1])
86
87     #----------------------------------------
88     timespan = np.zeros(2)
89     for idx in range(1, int(finalIndex)):
90
91         timespan[0] = t_log[idx-1,:]
92         timespan[1] = t_log[idx-1,:] + T_s
93
94         odeOut = odeint(fcn_difRovnice,
95                         x_log[idx-1,:],
96                         timespan,
97                         args=(u_log[idx-1,:],)
98                         )
99
```

```
        x_log[idx,:] = odeOut[-1,:]
        t_log[idx,:] = timespan[-1]

        # Tu sa umelo pridava sum k vystupnej velicine riadeneho systemu

        y_log_noise[idx,0] = x_log[idx,0] + np.random.normal(0, 0.1, size=1)

        #---------------------
        # ALGORITMUS RMNS

        # Pri RMNS sa vyuziva zasumena vystupna velicina

        y_k = y_log_noise[idx,0]

        h_k = np.array([[-y_log_noise[idx-1,0]],
                        [-y_log_noise[idx-2,0]], # pozor na idx-2 !!!
                        [u_log[idx-1,0]],
                        [u_log[idx-2,0]], # pozor na idx-2 !!!
                        ])

        theta_km1 = RMNS_theta_log[idx-1,:].reshape(4,-1)
        P_km1 = RMNS_P_log[idx-1,:].reshape(4,4)

        #----------
        e_k = y_k - np.matmul(h_k.T, theta_km1)
        Y_k =  np.matmul(P_km1, h_k) / (lambdaKoef + np.matmul(np.matmul(h_k.T, P_km1), h_k))
        P_k = (1/lambdaKoef) * (P_km1 - np.matmul(np.matmul(Y_k, h_k.T), P_km1))
        theta_k = theta_km1 + Y_k * e_k

        #----------
        RMNS_theta_log[idx,:] = theta_k.reshape(1,-1)
        RMNS_P_log[idx,:] = P_k.reshape(1,-1)

        RMNS_y_predict_log[idx,:] = np.matmul(h_k.T, theta_km1)

        #---------------------
        u_log[idx,:] = sig_u_ext[idx-1,:]

    return [t_log, x_log, RMNS_y_predict_log, RMNS_theta_log, y_log_noise]

```

V uvedenej simulačnej schéme je implementovaný RMNŠ algoritmus, ktorého výstupom je vektor parametrov `theta_k` a následne je tiež vypočítaná (v každom cykle) jednokroková predikcia výstupného signálu zapisovaná do vektora `RMNS_y_predict_log`.

Nastavenia potrebné pre samotnú simuláciu a vygenerovanie signálov, ktoré sa používajú pri simulácii (ktoré sú dopredu známe - dané):

Výpis kódu 15: Súbor `ar03_pr03.py`

```
# Nastavenia simulacie

sim_t_start = 0
sim_t_final = 250
sim_T_s = 0.1
sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
```

Pre simuláciu je potrebné vytvoriť vstupný signál $u(t)$ pre riadený systém. Nech je nasledovný:

Výpis kódu 16: Súbor `ar03_pr03.py`

```
# Preddefinovane signaly

period_time = 200
period_tab = np.array([
                       [0, 1],
                       [80, 0],
                       [120, -1],
                       [180, 0],
                       ])

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

for period in range(int(sim_t_final/period_time) + 1):
```

```
175
176     for idx in range( int((period*period_time)/sim_T_s), int((period*
        period_time + period_time)/sim_T_s)):
177
178         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
        period_time))<=idx*sim_T_s ][-1]
179         try:
180             sig_vysl[idx] = lastValue
181         except:
182             break
183
184 sig_u_ext = sig_vysl
```

3.3.1 Bez zabúdania

Ďalším nastavením, špeciálne dôležitým v tomto príklade je koeficient zabúdania λ:

Výpis kódu 17: Súbor `ar03_pr03.py`

```
191 sim_lambdaKoef = 1.0
```

Pamätajme teda, že faktor zabúdania λ (premenná `lambdaKoef`) je tu nastavený na hodnotu $\lambda = 1$, teda algoritmus nevyužíva zabúdanie.
Spustenie simulácie:

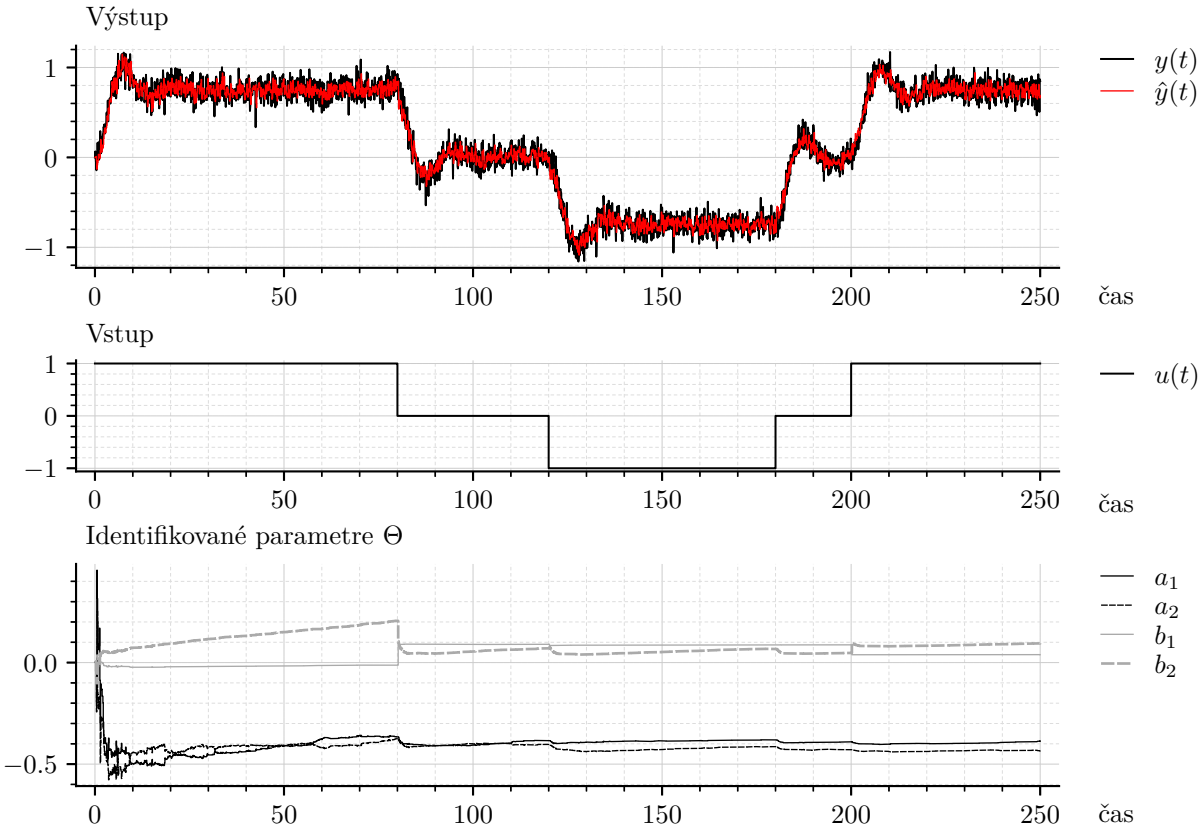
Výpis kódu 18: Súbor `ar03_pr03.py`

```
203 # Spustenie simulacie
204
205 t_log, x_log, RMNS_y_predict_log, RMNS_theta_log, y_log_noise =
        fcn_simSch_02_lenRMNS_noise(
206     sim_t_start,
207     sim_T_s,
208     sim_finalIndex,
209     sig_u_ext,
210     sim_lambdaKoef
211     )
```

Nakreslenie obrázka (pre prehľadnosť je kód v samostatnom súbore):

Výpis kódu 19: Súbor `ar03_pr03.py`

```
219 # Obrazok
220
221 figName = 'figsc_ar03_fig03'
222 figNameNum = 0
223
224 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
225
```



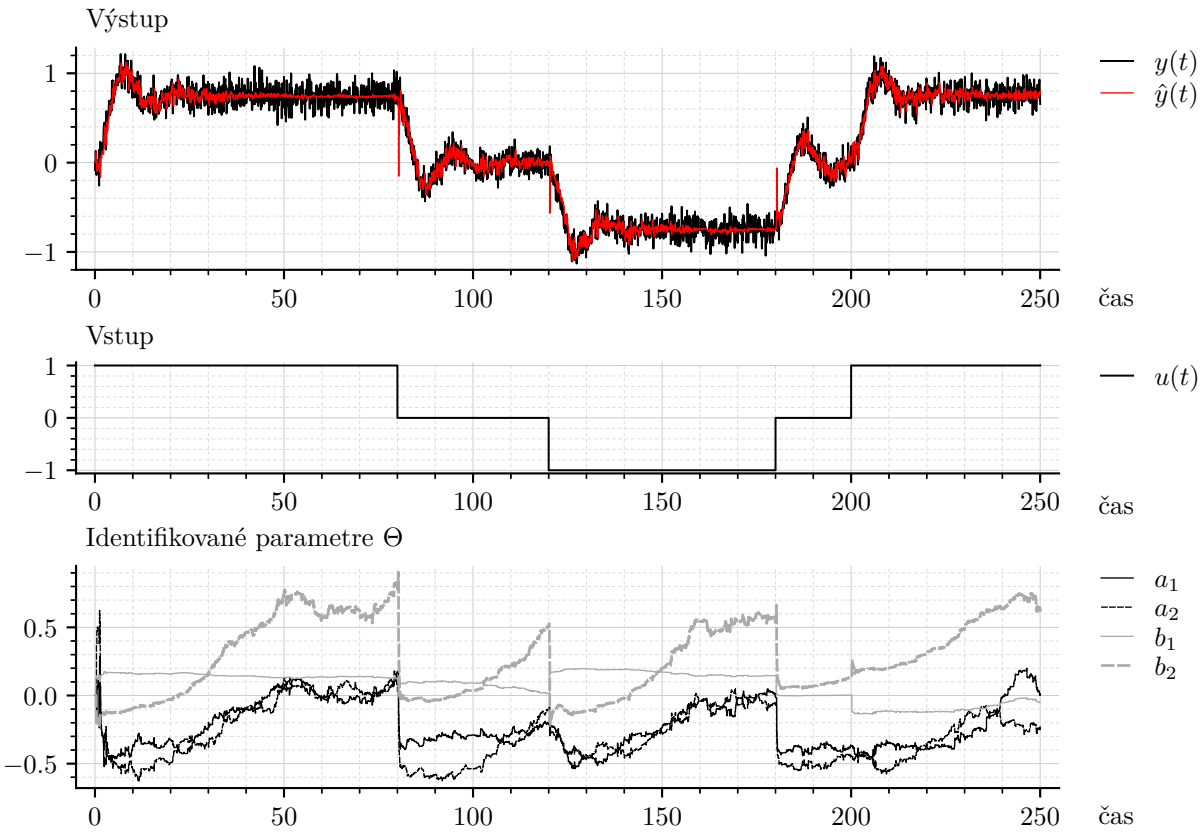
Obr. 3

3.3.2 So zabúdaním

Iná simulácia nech je s nasledovným koeficientom zabúdania:

Výpis kódu 20: Súbor `ar03_pr03.py`

```
240 sim_lambdaKoef = 0.987
241
242 # Spustenie simulacie
243
244 t_log, x_log, RMNS_y_predict_log, RMNS_theta_log, y_log_noise =
        fcn_simSch_02_lenRMNS_noise(
245     sim_t_start,
246     sim_T_s,
247     sim_finalIndex,
248     sig_u_ext,
249     sim_lambdaKoef
250     )
251
252 # Obrazok
253
254 figName = 'figsc_ar03_fig03'
255 figNameNum = 1
256
257 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
```

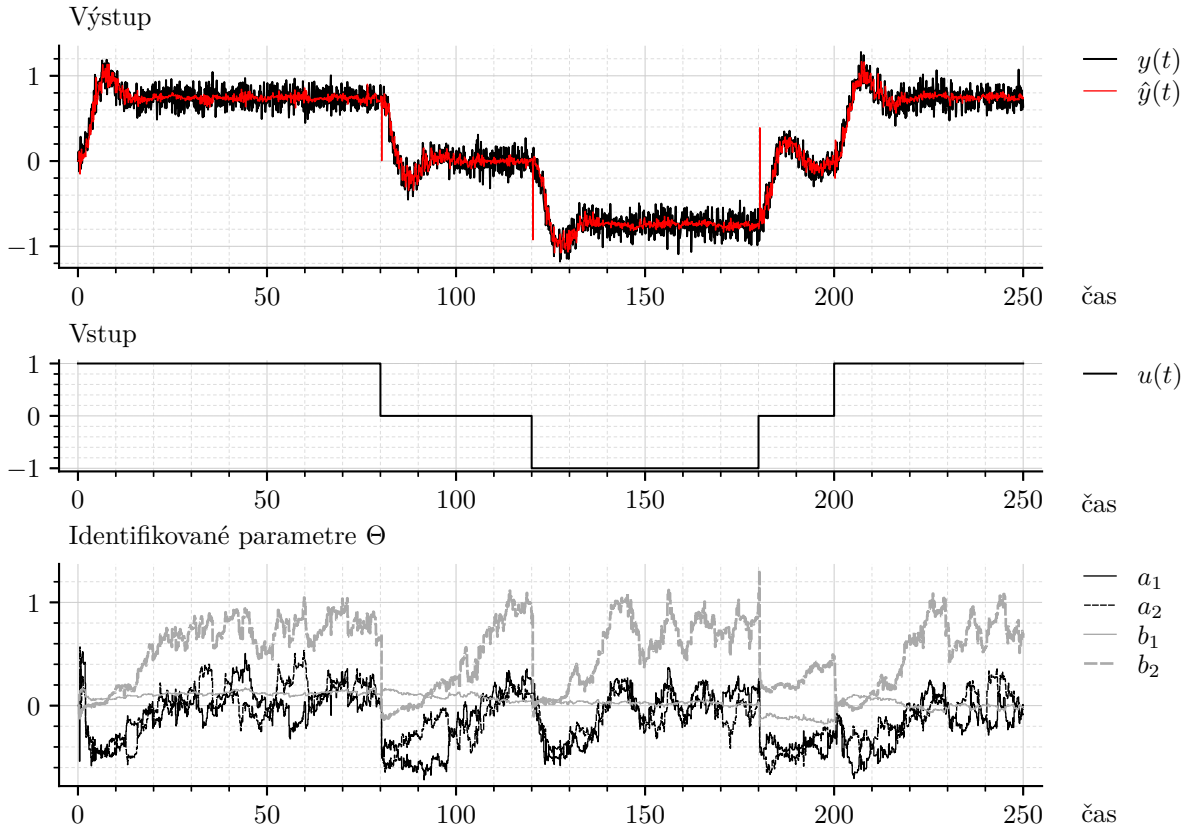


Obr. 4

Extrémnou voľbou koeficientu zabúdania pre tento prípad by bolo:

Výpis kódu 21: Súbor `ar03_pr03.py`

```
280 sim_lambdaKoef = 0.957
281
282 # Spustenie simulacie
283
284 t_log, x_log, RMNS_y_predict_log, RMNS_theta_log, y_log_noise =
        fcn_simSch_02_lenRMNS_noise(
285     sim_t_start,
286     sim_T_s,
287     sim_finalIndex,
288     sig_u_ext,
289     sim_lambdaKoef
290     )
291
292 # Obrazok
293
294 figName = 'figsc_ar03_fig03'
295 figNameNum = 2
296
297 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
```

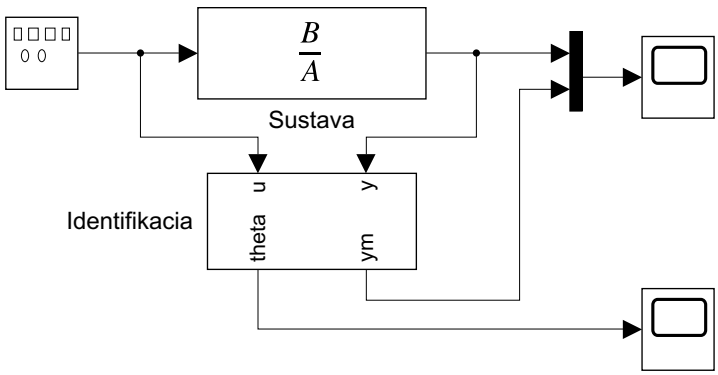


Obr. 5

3.4 Iná ukážka simulačnej schémy pre simuláciu RMNŠ

V tejto časti sa použije Simulink pre riešenie úloh ako v predchádzajúcom.

Majme sústavu, lepšie povedané dynamický systém (ktorý neskôr budeme riadiť), a k tomu istý blok, ktorého funkciou je realizovať priebežnú identifikáciu predmetného systému. Schematicky znázornené:



Obr. 6

Blok Identifikácia slúži na priebežnú identifikáciu a teda jeho hlavným výstupom sú parametre Θ a tiež sa uvažuje výstupná veličina modelu $\hat{y}$ (na obr. označená ako `ym`).

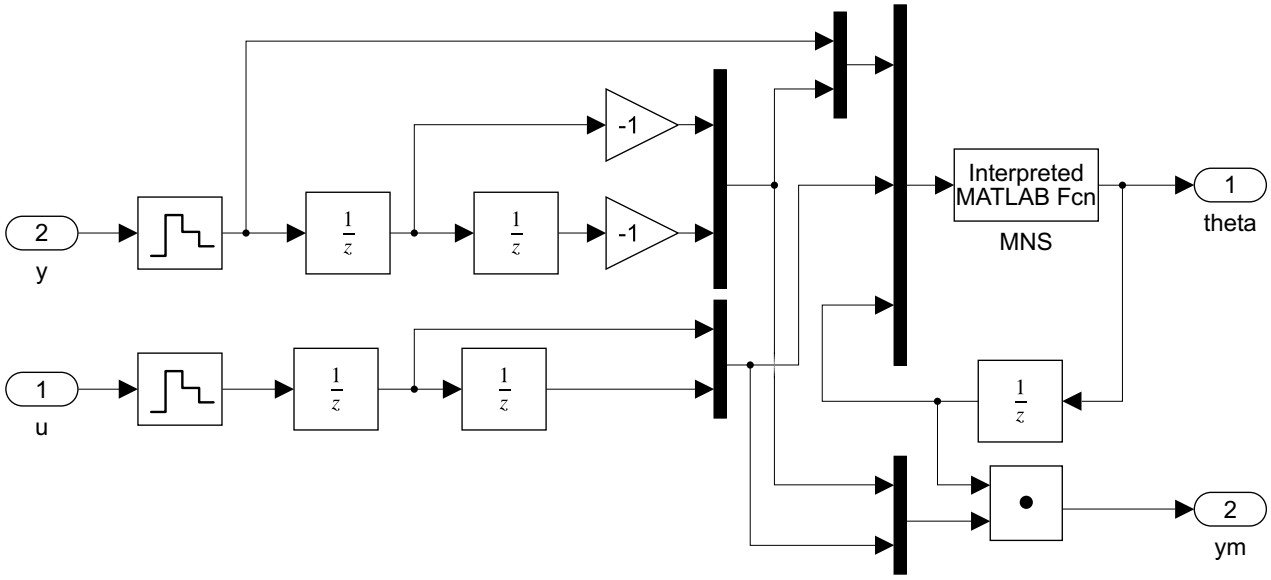
Pred spustením schémy (v Simulinku) je samozrejme potrebné inicializovať parametre riadeného systému (sústavy) a ako sa ukáže aj iné veci (v tomto prípade). Je to realizované v skripte:

Výpis kódu 22: Súbor `ar03_spustima_RMNS.m`

```
1 clear all;
2 clc
```

```
3
4   global P
5
6   % Perioda vzorkovania
7   Tvz = 0.1;
8
9   % Identifikovana sustava
10  B = [0 0.15];
11  A = [1 0.3 0.2];
12
13  % Prepis do diskretneho tvaru (len tak pre zaujimavost...)
14  Gs = tf(B,A);
15  Gz = c2d(Gs,Tvz);
16  [Bz,Az] = tfdata(Gz,'v')
17
18  % Startovacia matica P
19  P = 10^6 * eye(4,4) ;
```

Samotný blok Identifikácia je realizovaný nasledovne:



Obr. 7

Obsahuje vzorkovanie signálov (zero order hold) a oneskorovanie signálov (v zmysle z^{-1}). Tým je realizované získavanie tzv. signálneho vektora a podobne. Ďalej je súčasťou bloku funkcia, ktorá realizuje samotný algoritmus RMNŠ. Kód funkcie:

Výpis kódu 23: Súbor `MNS.m`

```
1   function odhadTheta = MNS(vst)
2   global P
3
4   P_n = P;
5
6   theta = vst(6:9);
7
8   h_n1 = [vst(2:5)];
9   y_n1 = vst(1);
10
11  e_n1 = y_n1 - h_n1' * theta;
12  Y_n1 = (P_n*h_n1)/(1+h_n1'*P_n*h_n1);
13  P_n1 = P_n - Y_n1*h_n1'*P_n;
14  odhadTheta = theta + Y_n1*e_n1;
15
16  P = P_n1;
```

4 Metóda rozmiestňovania pólov

Pripomeňme, že zákon riadenia (štruktúra riadenia) samonastavujúceho sa regulátora v uvažovanom konkrétnom príklade je v tvare

$$R(z^{-1})u(k) = T(z^{-1})r(k) - S(z^{-1})y(k) \tag{40a}$$

$$u(k) = \frac{T(z^{-1})}{R(z^{-1})}r(k) - \frac{S(z^{-1})}{R(z^{-1})}y(k) \tag{40b}$$

kde R, S a T sú polynómy v tvare

$$R(z^{-1}) = 1 + r_1 z^{-1} + r_2 z^{-2} + \ldots + r_{n_r} z^{-n_r} \tag{41a}$$

$$S(z^{-1}) = s_0 + s_1 z^{-1} + s_2 z^{-2} + \ldots + s_{n_s} z^{-n_s} \tag{41b}$$

$$T(z^{-1}) = t_0 + t_1 z^{-1} + t_2 z^{-2} + \ldots + t_{n_t} z^{-n_t} \tag{41c}$$

Ako už bolo uvedené, koeficienty polynómov sú parametrami regulátora. Počet parametrov regulátora závisí od stupňa jednotlivých polynómov. Pre uvažovaný konkrétny príklad sú stupne polynómov nasledovné: $n_r = 1$, $n_s = 1$ a $n_t = 0$. Potom počet parametrov regulátora je $n_r + n_s + 1 + n_t + 1$. Teda $1+1+1+0+1 = 4$. Zákon riadenia je možné zapísať aj v tvare diferenčnej rovnice:

$$u(k) = -r_1 u(k-1) - s_0 y(k) - s_1 y(k-1) + t_0 r(k) \tag{42}$$

Uvažujme zákon riadenia v tvare (40), ktorého parametre budeme počítať pomocou metódy rozmiestňovania pólov. Najprv odvodíme rovnicu uzavretého obvodu (URO).

4.1 Rovnica URO

Model riadeného systému je

$$A(z^{-1})y(k) = B(z^{-1})u(k) \tag{43}$$

Dosadením (7) do (43) máme

$$A(z^{-1})y(k) = B(z^{-1})\left(\frac{T(z^{-1})}{R(z^{-1})}r(k) - \frac{S(z^{-1})}{R(z^{-1})}y(k)\right) \tag{44}$$

Úpravou

$$Ay(k) = \frac{BT}{R}r(k) - \frac{BS}{R}y(k) \tag{45a}$$

$$RAy(k) = BTr(k) - BSy(k) \tag{45b}$$

$$(RA + BS)\,y(k) = BTr(k) \tag{45c}$$

$$y(k) = \frac{BT}{(AR + BS)}r(k) \tag{45d}$$

$$\frac{y(k)}{r(k)} = \frac{BT}{(AR + BS)} \tag{45e}$$

Charakteristický polynóm uzavretého regulačného obvodu je:

$$A(z^{-1})R(z^{-1}) + B(z^{-1})S(z^{-1}) \tag{46}$$

Nech želaný polynóm je

$$P(z^{-1}) = 1 + p_1 z^{-1} + p_2 z^{-2} + \ldots + p_{n_p} z^{-n_p} \tag{47}$$

potom diofantická rovnica, z ktorej sa vypočítajú koeficienty polynómov R a S je

$$A(z^{-1})R(z^{-1}) + B(z^{-1})S(z^{-1}) = P(z^{-1}) \tag{48}$$

V tomto prípade máme

$$A = 1 + a_1 z^{-1} + a_2 z^{-2} \tag{49a}$$
$$B = b_1 z^{-1} + b_2 z^{-2} \tag{49b}$$
$$R = 1 + r_1 z^{-1} \tag{49c}$$
$$S = s_0 + s_1 z^{-1} \tag{49d}$$

a nech želaný polynóm pre tento prípad je

$$P = 1 + p_1 z^{-1} + p_2 z^{-2} \tag{50}$$

Diofantická rovnica pre tento prípad

$$\begin{aligned} &\left(1 + a_1 z^{-1} + a_2 z^{-2}\right)\left(1 + r_1 z^{-1}\right) + \left(b_1 z^{-1} + b_2 z^{-2}\right)\left(s_0 + s_1 z^{-1}\right) \\ &= 1 + p_1 z^{-1} + p_2 z^{-2} \end{aligned} \tag{51}$$

Roznásobením

$$\begin{aligned} &1 + r_1 z^{-1} + a_1 z^{-1} + a_1 r_1 z^{-2} + a_2 z^{-2} + a_2 r_1 z^{-3} \\ &+ b_1 s_0 z^{-1} + b_1 s_1 z^{-2} + b_2 s_0 z^{-2} + b_2 s_1 z^{-3} \\ &= 1 + p_1 z^{-1} + p_2 z^{-2} \end{aligned} \tag{52}$$

Na ľavej strane ponecháme členy, v ktorých sa nachádzajú neznáme koeficienty polynómov zo zákona adaptácie a ostatné členy presunieme na pravú stranu

$$\begin{aligned} &r_1 z^{-1} + a_1 r_1 z^{-2} + a_2 r_1 z^{-3} + b_1 s_0 z^{-1} + b_1 s_1 z^{-2} + b_2 s_0 z^{-2} + b_2 s_1 z^{-3} \\ &= 1 + p_1 z^{-1} + p_2 z^{-2} - 1 - a_1 z^{-1} - a_2 z^{-2} \end{aligned} \tag{53}$$

Po úprave

$$\begin{aligned} &\left(r_1 + b_1 s_0\right) z^{-1} + \left(a_1 r_1 + b_1 s_1 + b_2 s_0\right) z^{-2} + \left(a_2 r_1 + b_2 s_1\right) z^{-3} \\ &= \left(p_1 - a_1\right) z^{-1} + \left(p_2 - a_2\right) z^{-2} \end{aligned} \tag{54}$$

Porovnaním koeficientov pri rovnakých mocninách získame rovnice

$$r_1 + b_1 s_0 = p_1 - a_1 \tag{55a}$$
$$a_1 r_1 + b_2 s_0 + b_1 s_1 = p_2 - a_2 \tag{55b}$$
$$a_2 r_1 + b_2 s_1 = 0 \tag{55c}$$

V maticovom zápise:

$$\begin{bmatrix} 1 & b_1 & 0 \\ a_1 & b_2 & b_1 \\ a_2 & 0 & b_2 \end{bmatrix} \begin{bmatrix} r_1 \\ s_0 \\ s_1 \end{bmatrix} = \begin{bmatrix} p_1 - a_1 \\ p_2 - a_2 \\ 0 \end{bmatrix} \tag{56}$$

Maticový zápis vyplývajúci z diofantickej rovnice v prípade, keď stupne polynómov R, S a P sú všeobecné, je v tvare

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_1 & 0 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & \cdots & 0 & b_2 & b_1 & 0 & \cdots & 0 \\ a_2 & a_1 & 1 & \cdots & 0 & b_3 & b_2 & b_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n_a} & \vdots & \vdots & \cdots & 1 & b_{n_b} & \vdots & \vdots & \cdots & b_1 \\ 0 & \ddots & \vdots & \cdots & a_1 & 0 & \ddots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \cdots & \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n_a} & 0 & 0 & 0 & \cdots & b_{n_b} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_2 \\ \vdots \\ r_{n_r} \\ s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{n_s} \end{bmatrix} = \begin{bmatrix} p_1 - a_1 \\ p_2 - a_2 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{57}$$

Takáto sústava rovníc bude mať riešenie ak

$$n_r = n_b - 1 \tag{58a}$$
$$n_s = n_a - 1 \tag{58b}$$

4.2 Polynóm T

Zatiaľ sme vypočítali koeficienty polynómov R a S. Otázkou ostáva, ako určiť polynóm T. Je potrebné určiť jeho stupeň a vypočítať koeficienty. V úvode sme „zvolili“, že stupeň polynómu T je $n_t = 0$. Teda jediným koeficientom bude t_0. Ukážme teraz, že jednou z možností, ako určiť polynóm T, je žiadať nulovú regulačnú odchýlku v ustálenom stave. Dostaneme tak polynóm T práve nultého stupňa a aj výpočet koeficientu t_0.

Keďže charakteristická rovnica URO je rovnaká ako želaný polynóm P, je možné písať rovnicu uzavretého obvodu v tvare

$$y(k) = \frac{BT}{P} r(k) \tag{59}$$

Aby platilo

$$y(\infty) = r(\infty) \tag{60}$$

musí byť

$$BT = P \tag{61a}$$

$$T = \frac{P}{B} \tag{61b}$$

A keďže „donekonečna“ je v diskrétnej doméne „dojednotky“, teda $z = 1$, potom

$$T = \frac{P(1)}{B(1)} \tag{62}$$

čo v tomto prípade znamená

$$T(z^{-1}) = \frac{1 + p_1 + p_2}{b_1 + b_2} = t_0 \tag{63}$$

4.2.1 Alternatívy spôsob určenia polynómu T

Alternatíva 1

Ďalší spôsob ako určiť koeficienty polynómu T je nasledovný. Nájdeme obraz referenčného signálu tak, že jeho časovú formu pretransformujeme pomocou Z-transformácie. Totiž v mnohých prípadoch je možné dopredu určiť časový priebeh referenčného signálu a navyše sa referenčný signál skladá zo skokov, rámp a podobných, pomocou Z-transformácie, ľahko transformovateľných signálov. Obraz referenčného signálu je

$$\{r(t)\}_q = \frac{F\left(q^{-1}\right)}{G\left(q^{-1}\right)} \tag{64}$$

Tento obraz použijeme vo vzťahu pre regulačnú odchýlku:

$$e = r - y = r - \frac{BT}{P} r = \frac{F}{G} - \frac{BT}{P}\frac{F}{G} = \frac{F(P - BT)}{GP} = \frac{FN}{P} \tag{65}$$

kde sme označili

$$\frac{P - BT}{G} = N \tag{66}$$

Z tohto označenia môžeme písať diofantickú rovnicu, ktorá doplní (48), a vznikne tak sústava.

$$GN + BT = P \tag{67}$$

Z tejto rovnice sa dá určiť aj polynóm vyššieho ako nultého stupňa.

Alternatíva 2

Ďalší spôsob ako určiť koeficienty polynómu T je takýto: ak bude polynóm T obrátenou hodnotou polynómu B, teda $T = 1/B$, zaistí sa tak nie len nulová regulačná odchýlka v ustálenom stave, ale aj to, že polynóm B nebude mať žiadny vplyv na dynamiku URO. Dynamika URO bude daná len želaným charakteristickým polynómom P, takto:

$$y(t) = \frac{1}{P} r(t) \tag{68}$$

Zákon riadenia potom môžeme uvažovať v tvare

$$Ru(t) = \frac{1}{B} r(t) - Sy(t) \quad \Rightarrow \quad r = BRu + BSy \tag{69}$$

Ale ak $B = q^{-D}\tilde{B}$, tak aby sme mohli napísať predchádzajúcu rovnicu musíme dať q^{-D} na druhú stranu k r. Teda:

$$rq^{D} = \tilde{B}Ru + \tilde{B}Sy \tag{70}$$

Ak potom vyjadríme akčný zásah, je zrejmé, že bude závisieť od budúcich hodnôt referenčného signálu $u(t) = r(t + D) - \ldots$ Toto však nemusí byť prekážkou, pretože v mnohých prípadoch je referenčný signál dopredu známy.

4.3 Súhrn pre tento prípad

Zhrňme výpočty potrebné pre určenie parametrov regulátora pre tento prípad.

Hodnoty parametrov modelu a_1, a_2, b_1 a b_2 sú po identifikácii (v danej perióde vzorkovania) známe a prirodzene sú známe aj hodnoty koeficientov želaného polynómu P. Hodnoty parametrov regulátora získame riešením

$$\begin{bmatrix} 1 & b_1 & 0 \\ a_1 & b_2 & b_1 \\ a_2 & 0 & b_2 \end{bmatrix} \begin{bmatrix} r_1 \\ s_0 \\ s_1 \end{bmatrix} = \begin{bmatrix} p_1 - a_1 \\ p_2 - a_2 \\ 0 \end{bmatrix} \tag{71}$$

a

$$t_0 = \frac{1 + p_1 + p_2}{b_1 + b_2} \tag{72}$$

4.4 Rýchlostný algoritmus metódy rozmiestňovania pólov

Táto subsekcia je tu len pre ilustráciu širších obzorov týkajúcich sa oblasti používania metódy rozmiesňovania pólov v prípadoch podobných tomuto.

Keďže ide o rýchlostný algoritmus, je potrebné vyjadriť prírastok akčnej veličiny:

$$\Delta u(t) = u(t) - u(t-1) = (1 - q^{-1})u(t) \tag{73}$$

$$u(t) = \frac{\Delta u(t)}{(1 - q^{-1})} \tag{74}$$

Riadený systém a jeho ARX model (s nulovým šumom):

$$Ay(t) = Bu(t) \tag{75}$$

do ktorého dosadíme $u(t)$ a upravíme...

$$Ay(t) = B\frac{\Delta u(t)}{(1 - q^{-1})} \tag{76a}$$

$$(1 - q^{-1})Ay(t) = B\Delta u(t) \tag{76b}$$

Zákon riadenia uvažujeme v tvare:

$$\Delta u(t) = \frac{S}{R}(r(t) - y(t)) \tag{77}$$

Rovnica URO potom bude mať tvar

$$(1-q^{-1})Ay(t) = B\Delta u(t) \tag{78a}$$
$$(1-q^{-1})Ay(t) = B\frac{S}{R}(r(t)-y(t)) \tag{78b}$$
$$(1-q^{-1})ARy(t) = BS(r(t)-y(t)) \tag{78c}$$
$$(1-q^{-1})ARy(t) = BSr(t) - BSy(t) \tag{78d}$$
$$((1-q^{-1})AR + BS)y(t) = BSr(t) \tag{78e}$$
$$y(t) = \frac{BS}{(1-q^{-1})AR + BS}r(t) \tag{78f}$$

Takže charakteristický polynóm je

$$P = \left(1-q^{-1}\right)AR + BS \tag{79}$$

Ale veď potom

$$BS = P - \left(1-q^{-1}\right)AR \tag{80}$$

a tak rovnica URO:

$$y(t) = \frac{P-(1-q^{-1})AR}{P}r(t) \tag{81a}$$
$$y(t) = \left(\frac{P}{P} - \frac{(1-q^{-1})AR}{P}\right)r(t) \tag{81b}$$
$$y(t) = r(t) - \frac{(1-q^{-1})AR}{P}r(t) \tag{81c}$$

Ak sa teraz opýtame aká bude regulačná odchýlka v ustálenom stave t.j. $y(\infty)$, $r(\infty)$ a čo je najdôležitejšie $q = 1$ potom:

$$\begin{aligned} y(\infty) &= r(\infty) - \frac{(1-1)AR}{P}r(\infty) \\ y(\infty) &= r(\infty) \end{aligned} \tag{82}$$

Teda regulačná odchýlka v ustálenom stave bude nulová (pretože sa zhodujú hodnoty referenčného signálu a výstupnej veličiny).

5 Cvičenie tretie

1. Zrealizujte (v simulačnom prostredí) samonastavujúci sa regulátor tak ako to predpokladá uvažovaný konkrétny príklad.

   Nech želaným charakteristickým polynómom (pre uvažovaný konkrétny príklad) je $P(z^{-1}) = 1 + p_1 z^{-1} + p_2 z^{-2}$ pričom $p_1 = -1,6$ a $p_2 = 0,64$, teda dvojnásobný koreň $z_{1,2} = 0,8$.

   Alternatívne, nech v želanom charakteristickom je $p_1 = -0,8$ a $p_2 = 0,16$, teda dvojnásobný koreň $z_{1,2} = 0,4$.

   Pozn: pre uvažovaný príklad sa odporúča perióda vzorkovania $T_{vz} = 0,1$ [čas].

5.1 Konkrétny príklad samonastavujúceho sa regulátora

V predchádzajúcom bola prezentovaná simulačná schéma, v ktorej bol implementovaný algoritmus RMNŠ.

Tu je cieľom doplniť do simulačnej schémy výpočet, ktorý na základe priebežne identifikovaných parametrov riadeného systému vypočíta hodnoty parametrov daného zákona riadenia a následne vypočíta samotný akčný zásah.

Výpočet parametrov zákona riadenia využíva matódu rozmistňovania pólov URO a je dplnený výpočtom pre zabezpečenie nulovej trvalej regulačnej odchýlky.

Nech želaným charakteristickým polynómom (pre uvažovaný konkrétny príklad) je $P(z^{-1}) = 1 + p_1 z^{-1} + p_2 z^{-2}$ pričom $p_1 = -1,6$ a $p_2 = 0,64$, teda dvojnásobný koreň $z_{1,2} = 0,8$.

Pozn: pre uvažovaný príklad sa odporúča perióda vzorkovania $T_{vz} = 0,1$ [čas].

V tomto konkrétnom príklade uvažovaný zákon riadenia je možné zapísať v tvare diferenčnej rovnice:

$$u(k) = -r_1 u(k-1) - s_0 y(k) - s_1 y(k-1) + t_0 r(k) \tag{83}$$

Hodnoty parametrov modelu a_1, a_2, b_1 a b_2 sú po identifikácii (v danej perióde vzorkovania) známe a prirodzene sú známe aj hodnoty koeficientov želaného polynómu P. Hodnoty parametrov regulátora získame riešením

$$\begin{bmatrix} 1 & b_1 & 0 \\ a_1 & b_2 & b_1 \\ a_2 & 0 & b_2 \end{bmatrix} \begin{bmatrix} r_1 \\ s_0 \\ s_1 \end{bmatrix} = \begin{bmatrix} p_1 - a_1 \\ p_2 - a_2 \\ 0 \end{bmatrix} \tag{84}$$

a

$$t_0 = \frac{1 + p_1 + p_2}{b_1 + b_2} \tag{85}$$

Aj tu platí, že diferenciálne rovnice riadeného systému ostávajú samozrejme rovnaké ako sme ich realizovali v predchádzajúcom, teda aj tu („na začiatku skriptu“) platí výpis kódu 1 a 2.

Ďalej nech simulačnú schému realizuje nasledujúca funkcia:

Výpis kódu 24: Súbor `ar03_pr04.py`

```
47 def fcn_simSch_05_STR(t_start, T_s, finalIndex, sig_r_ext):
48
49     #----------------------------------------
50     t_log = np.zeros([finalIndex, 1])
51     t_log[0,:] = t_start
52
53     #----------------------------------------
54     x_0 = np.array([0, 0])
55
56     x_log = np.zeros([finalIndex, len(x_0)])
57     x_log[0,:] = x_0
58     #----------------------------------------
59
60     RMNS_theta_0 = np.array([[ -1.5],
61                              [ 0.5],
62                              [ -2e-5],
63                              [ 1.5e-3]])
64
65     RMNS_theta_log = np.zeros([finalIndex, len(RMNS_theta_0)])
66     RMNS_theta_log[0,:] = RMNS_theta_0.reshape(1,-1)
67
68     RMNS_P_0 = np.diag([10*2, 10**2, 10**5, 10**5])
69
70     RMNS_P_log = np.zeros([finalIndex, RMNS_P_0.size])
71     RMNS_P_log[0,:] = RMNS_P_0.reshape(1,-1)
72
73     RMNS_y_predict_log = np.zeros([finalIndex, 1])
74
75     #----------------------------------------
76
77     u_log = np.zeros([finalIndex, 1])
78         # v tomto bode by sa dalo vypocitat u_0, ale tu na to kasleme
79
80     #----------------------------------------
81     timespan = np.zeros(2)
82     for idx in range(1, int(finalIndex)):
83
84         timespan[0] = t_log[idx-1,:]
85         timespan[1] = t_log[idx-1,:] + T_s
86
87         odeOut = odeint(fcn_difRovnice,
88                         x_log[idx-1,:],
89                         timespan,
90                         args=(u_log[idx-1,:],)
```

```
                        )

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

        #--------------------
        # ALGORITMUS RMNS
        y_k = x_log[idx,0]

        h_k = np.array([[-x_log[idx-1,0]],
                        [-x_log[idx-2,0]],  # pozor na idx-2 !!!
                        [u_log[idx-1,0]],
                        [u_log[idx-2,0]],
                        ])

        theta_km1 = RMNS_theta_log[idx-1,:].reshape(4,-1)
        P_km1 = RMNS_P_log[idx-1,:].reshape(4,4)

        #----------
        lambdaKoef = 0.95

        e_k = y_k - np.matmul(h_k.T, theta_km1)

        Y_k =  np.matmul(P_km1, h_k) / (lambdaKoef + np.matmul(np.
    matmul(h_k.T, P_km1), h_k))

        P_k = (1/lambdaKoef) * (P_km1 - np.matmul(np.matmul(Y_k, h_k.
    T), P_km1))
        theta_k = theta_km1 + Y_k * e_k

        #----------

        RMNS_theta_log[idx,:] = theta_k.reshape(1,-1)
        RMNS_P_log[idx,:] = P_k.reshape(1,-1)

        RMNS_y_predict_log[idx,:] = np.matmul(h_k.T, theta_km1)

        #--------------------
        # Vypocty pre parametre zakona riadenia a akcny zasah

        # koeficienty zelaneho polynomu:

        par_p1 = -1.6
        par_p2 = 0.64

        # parametre riadeneho systemu

        par_a1 = RMNS_theta_log[idx-1,0]
        par_a2 = RMNS_theta_log[idx-1,1]
        par_b1 = RMNS_theta_log[idx-1,2]
        par_b2 = RMNS_theta_log[idx-1,3]

        # Parametre RST regulatora

        matrix_A = np.array([[     1, par_b1,      0],
                             [par_a1, par_b2, par_b1],
                             [par_a2,      0, par_b2],
                             ])

        matrix_b = np.array([[par_p1 - par_a1],
                             [par_p2 - par_a2],
                             [0],
                             ])

        par_r1, par_s0, par_s1, = np.linalg.solve(matrix_A, matrix_b)

        par_t0 = (1 + par_p1 + par_p2)/(par_b1 + par_b2)

        # vypocita sa akcny zasah u(k)

        par_RST = np.array([par_r1, par_s0, par_s1, par_t0])
        vekt_omega = np.array([-u_log[idx-1,:], -x_log[idx,0], -x_log
    [idx-1,0], sig_r_ext[idx,0]])

        u_log[idx,:] = np.dot(par_RST, vekt_omega)

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

    return [t_log, x_log, u_log, RMNS_y_predict_log, RMNS_theta_log]
```

V uvedenej simulačnej schéme je implementovaný RMNŠ algoritmus rovnako ako

v predchádzajúcom...

Všimnime si však, že faktor zabúdania λ (premenná `lambdaKoef`) je nastavený na hodnotu $\lambda = 0,95$.

Tiež je potrebné všimnúť si, že štartovacie hodnoty RMNŠ algoritmu sú:

```
RMNS_theta_0 = np.array([[-1.5], [0.5], [-2e-5], [1.5e-3]])
```

a

```
RMNS_P_0 = np.diag([10*2, 10**2, 10**5, 10**5])
```

Nastavenia potrebné pre samotnú simuláciu a vygenerovanie signálov, ktoré sa používajú pri simulácii (ktoré sú dopredu známe - dané):

Výpis kódu 25: Súbor `ar03_pr04.py`

```
175 # Nastavenia simulacie
176
177 sim_t_start = 0
178 sim_t_final = 38
179 sim_T_s = 0.1
180 sim_finalIndex = int(((sim_t_final - sim_t_start)/sim_T_s) + 1)
181
182
183 # Preddefinovane signaly
184
185 period_time = 40
186 period_tab = np.array([
187                        [0, 1],
188                        [10, 0],
189                        [20, -1],
190                        [30, 0],
191                        ])
192
193 sig_vysl = np.zeros([sim_finalIndex, 1])
194
195 for period in range(int(sim_t_final/period_time) + 1):
196
197     for idx in range( int((period*period_time)/sim_T_s), int((period*
        period_time + period_time)/sim_T_s)):
198
199         lastValue = period_tab[:,1][(period_tab[:,0] + (period*
        period_time))<=idx*sim_T_s ][-1]
200         try:
201             sig_vysl[idx] = lastValue
202         except:
203             break
204
205 sig_r_ext = sig_vysl
206
```

Spustenie simulácie:

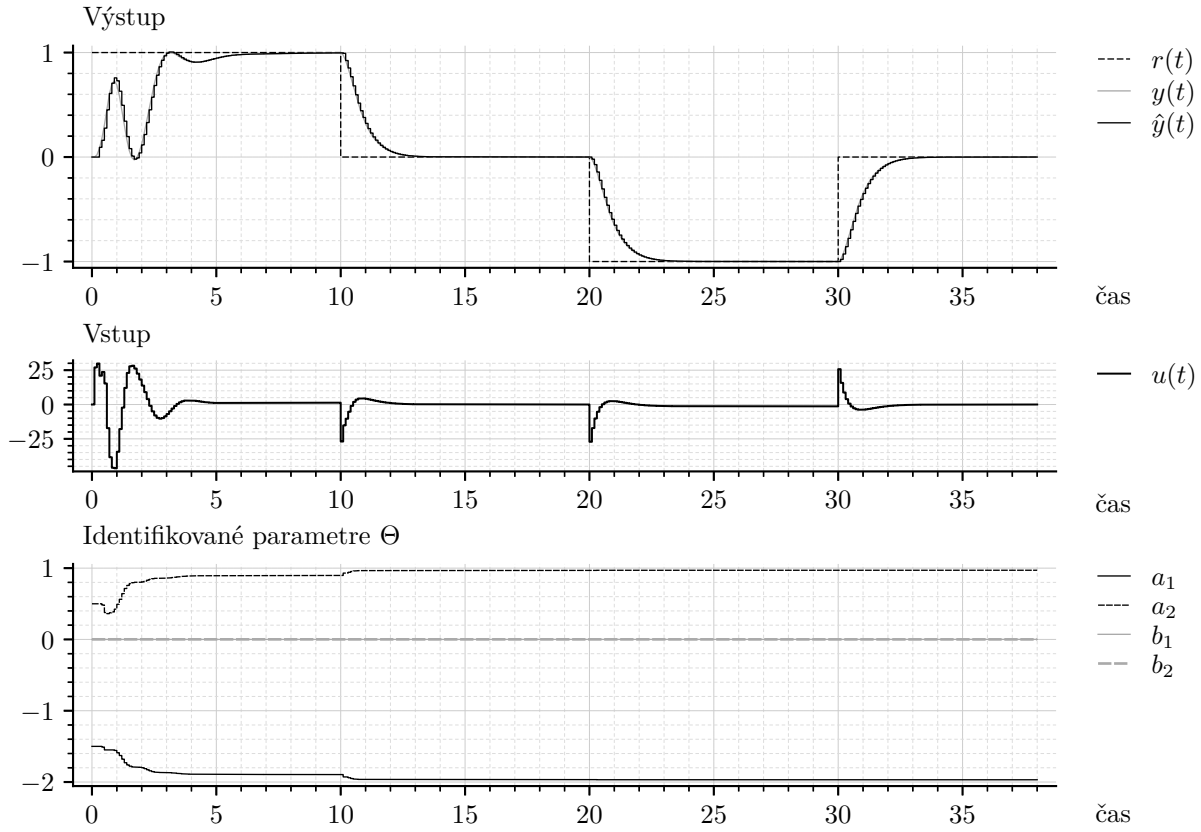
Výpis kódu 26: Súbor `ar03_pr04.py`

```
223 # Spustenie simulacie
224
225 t_log, x_log, u_log, RMNS_y_predict_log, RMNS_theta_log =
        fcn_simSch_05_STR(
226     sim_t_start,
227     sim_T_s,
228     sim_finalIndex,
229     sig_r_ext,
230     )
```

Nakreslenie obrázka:

Výpis kódu 27: Súbor `ar03_pr04.py`

```
238 # Obrazok
239
240 figName = 'figsc_ar03_fig04'
241 figNameNum = 0
242
243 exec(open('./misc/' + figName + '.py', encoding='utf-8').read())
244
```

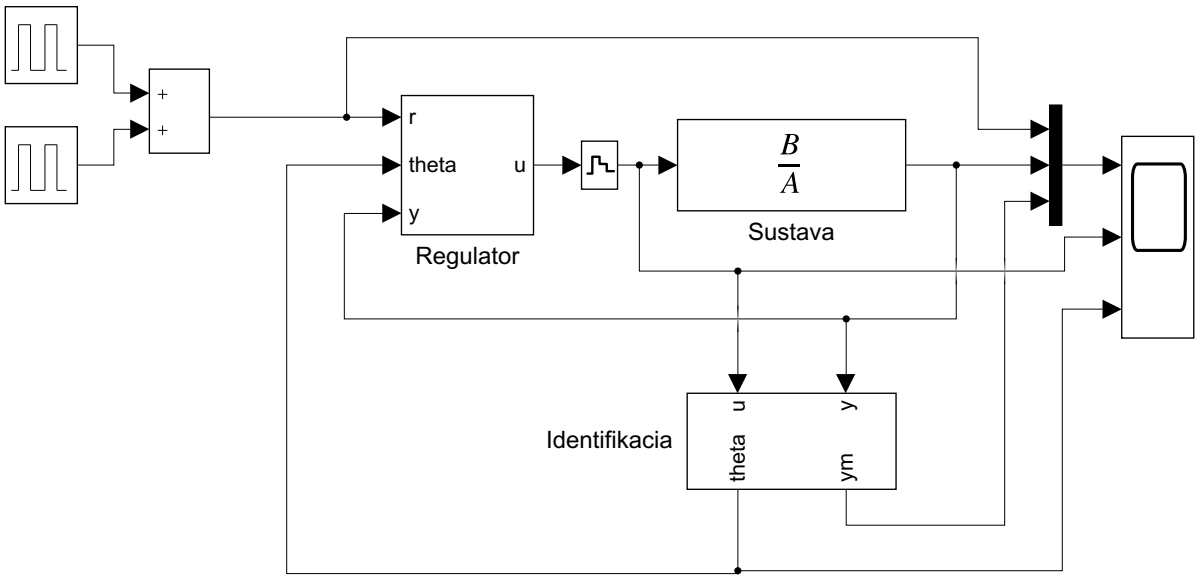


Obr. 8

5.2 Simulácia v Simulinku

V tejto časti sa použije Simulink pre riešenie úloh ako v predchádzajúcom.

Máme riadený systém, ku ktorému sme v predchádzajúcom vyrobili blok pre priebežnú identifikáciu. Teraz pridajme blok, ktorý bude realizovať výpočet akčného zásahu. Schematicky znázornené:



Obr. 9

Pred spustením schémy (v Simulinku) je samozrejme potrebné inicializovať parametre riadeného systému (sústavy) a ako sa ukáže aj iné veci (v tomto prípade). Je to realizované v skripte:

Výpis kódu 28: Súbor `ar03_spustima_STR.m`

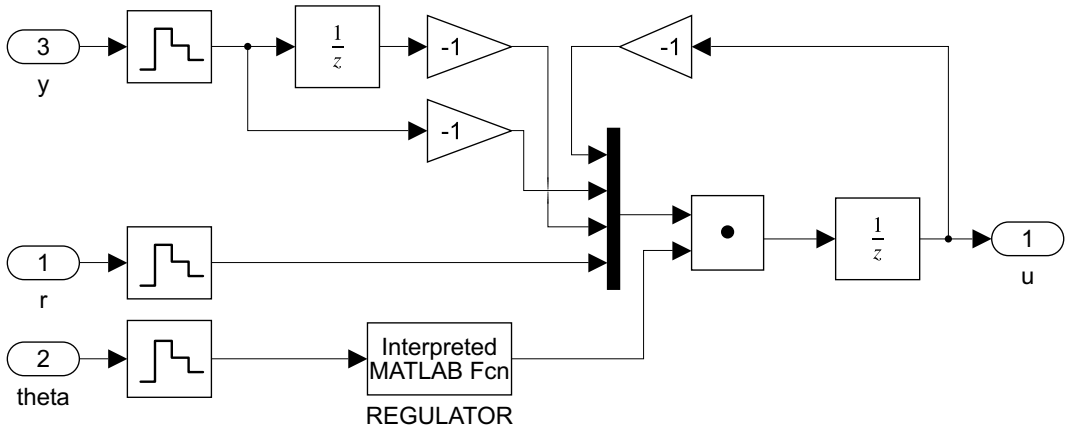
```
clear all;
clc

global P ZP lambdaKoef

% Perioda vzorkovania
Tvz = 0.1;

% Identifikovana sustava
B = [0 0.15];
A = [1 0.3 0.2];

% Zelany polynom
ZP = conv([1 -0.8],[1 -0.8]);

lambdaKoef = 0.95

% Startovacia matica P
P = diag([20, 10^2, 10^5, 10^5]) ;
```

Blok Identifikácia je realizovaný ako na obr. 7 a používa funkciu:

Výpis kódu 29: Súbor `MNSvRST.m`

```
function odhadTheta = MNSvRST(vst)
global P lambdaKoef

P_n = P;

theta = vst(6:9);

h_n1 = [vst(2:5)];
y_n1 = vst(1);

e_n1 = y_n1 - h_n1' * theta;
Y_n1 = (P_n*h_n1)/(lambdaKoef + h_n1'*P_n*h_n1);
P_n1 = (1/lambdaKoef) * (P_n - Y_n1*h_n1'*P_n);
odhadTheta = theta + Y_n1*e_n1;

P = P_n1;
```

Blok Regulátor je realizovaný nasledovne:



Obr. 10

Funkcia, ktorú používa blok Regulátor je nasledovná:

Výpis kódu 30: Súbor `REGULATOR.m`

```
function vyst = REGULATOR(theta)

global ZP

a1 = theta(1);
a2 = theta(2);
b1 = theta(3);
b2 = theta(4);

MATICA = [1 b1 0; a1 b2 b1; a2 0 b2];
```

```
PRAVASTRANA = [ZP(2) - a1; ZP(3) - a2; 0];
RS = MATICA\PRAVASTRANA;

T = (1 + ZP(2) + ZP(3))/(b1 + b2);

vyst = [RS' T]';
```

6 Otázky a úlohy

1. Stručne vysvetlite princíp rekurzívnej metódy najmenších štvorcov.
2. Napíšte Gaussov vzorec a podrobne vysvetlite jednotlivé prvky
3. Vyjadrite ARX model v tvare diskrétnej prenosovej funkcie alebo v tvare diferenčnej rovnice.
4. Odvoďte Gaussov vzorec a ukážte, že nájdený extrém je minimum.
5. Aké (ktoré) prvky obsahuje signálny vektor pri priebežnej identifikácii metódou najmenších štvorcov? (Čo tvorí prvky signálneho vektora pri priebežnej identifikácii metódou najmenších štvorcov?)
6. Modelom riadeného systému je ARX model. Zákon riadenia má tvar $R(z^{-1})u(k) = T(z^{-1})r(k) - S(z^{-1})y(k)$. Nájdite charakteristický polynóm URO.
7. Modelom riadeného systému je ARX model. Zákon riadenia má tvar $u(k) = \Delta u(k)/(1 - z^{-1})$, kde
$$\Delta u(k) = \frac{S(z^{-1})}{R(z^{-1})}(r(k) - y(k))$$
Nájdite charakteristický polynóm URO.
8. Stručne vysvetlite výpočet parametrov regulátora metódou pole-placement.
9. Vysvetlite podstatu metód návrhu polynómu $T(z^{-1})$ pri STR.

MRAC gradientný

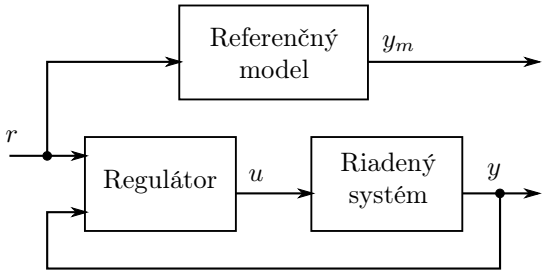
Obsah

V prvom rade poznámka k nadpisu: MRAC je skratka z *Model Reference Adaptive Control*, čo znamená Adaptívne riadenie s referenčným modelom. Ide o istú schému priameho adaptívneho riadenia, ktorá pre návrh zákona adaptácie využíva myšlienku o gradiente istej účelovej funkcie, ktorej optimom je vlastne splnenie cieľa riadenia. Slangovo povedané: MRAC („mrak“) gradientný.

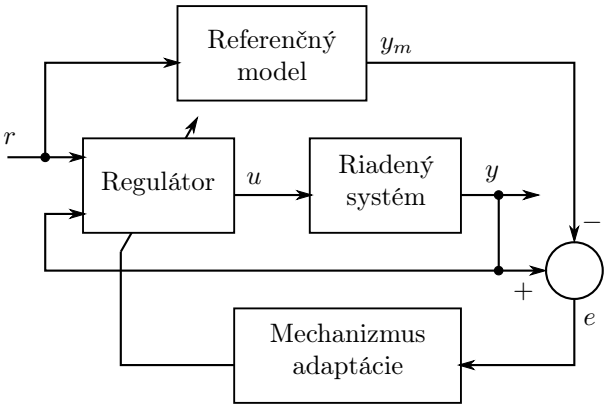
1 Riadenie s referenčným modelom

Pri riadení s referenčným modelom (*model reference control – MRC*) sú žiadané vlastnosti uzavretého regulačného obvodu opísané referenčným modelom, čo je jednoducho lineárny, časovo invariantný systém s prenosovou funkciou $W_m(s)$, ktorého vstupom je referenčná (žiadaná) veličina (hodnota) r. Do referenčného modelu sa premietnu požiadavky na výsledný regulačný obvod – URO. Výstup referenčného modelu y_m sa potom správa práve tak, ako to žiadame od výstupnej (riadenej) veličiny systému. URO je chápaný ako celok, ktorý vznikne pripojením zákona riadenia (regulátora) k riadenému systému. Vstupom URO je referenčná veličina r a výstupom je výstupná veličina systému y. Podobným spôsobom sa predpisujú požiadavky pri návrhoch napr. servo-systémov.

Zákon riadenia je zostavený tak, že prenosová funkcia URO má rovnakú štruktúru (tvar) ako referenčný model. Tým je daná štruktúra (tvar) zákona riadenia. Nie že by štruktúra bola daná jednoznačne, jednoducho, zákon riadenia musí byť taký, že



Obr. 1: Riadenie s referenčným modelom – principiálna schéma



Obr. 2: Adaptívne riadenie s referenčným modelom – principiálna schéma

umožní zhodu URO a referenčného modelu. Otázkou ostáva nastavenie parametrov, ktoré zákon riadenia obsahuje.

V predchádzajúcom príklade, v ktorom sme sa venovali adaptívnej stabilizácii, má zákon riadenia tvar $u = -k\,x$. Parameter zákona riadenia je k. Pri lineárnom regulátore, teda keď je jednoznačné, že parameter k je konštanta, nemení sa v čase, je k určené jednoduchou podmienkou, ktorá však vyžaduje znalosť parametra sústavy. V prípade riadiaceho systému, kde sa pripúšťa, že k sa môže (a má!) meniť v čase (adaptovať sa) je k v každom čase určené predpisom v tvare diferenciálnej rovnice.

1.1 MRAC

Na princípe riadenia s referenčným modelom je založená široká trieda metód adaptívneho riadenia nazývaná *Adaptívne riadenie s referenčným modelom* čo je prekladom z angličtiny: *Model Reference Adaptive Control – MRAC*. Niekedy sa zvykne takýto systém riadenia skratkou MRAS – Model Reference Adaptive System.

Pri adaptívnom riadení s referenčným modelom obsahuje riadiaci systém bežnú spätnoväzbovú slučku, v ktorej sú riadený systém a regulátor. Ako už bolo uvedené ďalšia spätnoväzbovú slučka v systéme mení parametre regulátora. Bežná spätnoväzbová slučka sa niekedy nazýva aj vnútorná slučka a spätnoväzbová slučka pre nastavovanie parametrov regulátora sa nazýva vonkajšia slučka. V tomto prípade je spätnou väzbou vo vonkajšej slučke rozdiel medzi výstupom riadeného systému a referenčného modelu, ktorý sa nazýva *adaptačná odchýlka*, označuje sa e.

Mechanizmus, ktorý adaptuje parametre regulátora (zákona riadenia) môže byť v adaptívnom riadení s referenčným modelom získaný dvomi spôsobmi. Použitím gradientnej metódy alebo použitím Lyapunovovej teórie stability. Oba prípady sú predmetom ďalších častí, pričom prvý uvedený je označený ako MRAC – gradientntný.

2 MIT algoritmus adaptácie: MRAC gradientný

Takzvané MIT pravidlo (MIT rule) je pôvodný mechanizmus adaptácie používaný v adaptívnom riadení s referenčným modelom. Názov vyplýva zo skutočnosti, že bol vyvinutý na MIT (Massachusetts Institute of Technology). Základnú myšlienku vyjadríme v nasledujúcom príklade.

Uvažujme, pre riadenie systému s výstupom y je použitý regulátor s jedným nastaviteľným parametrom Θ. Želané správanie uzavretého regulačného obvodu je špecifikované pomocou referenčného modelu, ktorého výstup je veličina y_m. Nech $e = y - y_m$ je adaptačná odchýlka. Jednou z možností ako postupovať pri nastavovaní parametra Θ je meniť ho tak aby sa účelová funkcia v tvare

$$J(\Theta) = \frac{1}{2}e^2 \tag{1}$$

minimalizovala. Pre zníženie hodnoty funkcie J, je rozumné meniť parameter Θ proti smeru derivácie J podľa Θ (gradientu funkcie), teda zmenu parametrov možno vyjadriť v tvare

$$\frac{\mathrm{d}\Theta}{\mathrm{d}t} = -\alpha \frac{\partial J}{\partial \Theta} = -\alpha e \frac{\partial e}{\partial \Theta} \tag{2}$$

kde $\alpha > 0$ je voliteľný parameter pre nastavenie veľkosti zmeny adaptovaného parametra a teda rýchlosti adaptácie, nazýva sa aj *adaptačné zosilnenie*. Toto je princíp slávneho MIT algoritmu adaptácie parametrov regulátora.

Tento algoritmus možno použiť aj keď regulátor obsahuje viac ako jeden parameter. Potom Θ nie je skalár ale vektor a výraz $\frac{\partial J}{\partial \Theta}$ je skutočne gradientom.

Tiež je možné algoritmus modifikovať použitím inej účelovej funkcie, pričom princíp ostáva zachovaný.

Parciálna derivácia $\frac{\partial e}{\partial \Theta}$ sa nazýva *citlivostná funkcia*, hovorí o tom ako veľmi je adaptačná odchýlka e ovplyvnená zmenou parametrov regulátora. Túto funkciu je možné vyjadriť pri predpoklade, že zmeny parametrov regulátora sú o veľa pomalšie ako zmeny všetkých ostatných veličín v systéme. Potom parametre regulátora môžme považovať za nezávislé od času. V ďalšom sa ukáže, že citlivostné funkcie často (nie vždy ako ukazuje nasledujúca časť) obsahujú parametre sústavy a teda neznáme parametre. Preto ich nie je možné priamo použiť a je potrebné nájsť ich vhodnú aproximáciu, takú, ktorá neobsahuje neznáme parametre.

2.1 Príklad: Adaptácia dopredného zosilnenia

Uvažujme riadený systém daný prenosovou funkciou v tvare

$$\frac{y(s)}{u(s)} = \frac{k}{s+1} \tag{3}$$

kde k je neznámy parameter, ale znamienko parametra k je známe. Úlohou je nájsť dopredný regulátor, ktorý spolu s prenosovou funkciou bude tvoriť systém špecifikovaný referenčným modelom. Referenčný model je definovaný v tvare prenosovej funkcie

$$\frac{y_m(s)}{r(s)} = \frac{k_m}{s+1} \tag{4}$$

Uvažujme zákon riadenia v tvare

$$u = \Theta r \tag{5}$$

kde u je akčný zásah (vstup riadeného systému) a r je žiadaná hodnota. Tento zákon riadenia umožní, že prenosová funkcia zo žiadanej hodnoty r na výstupnú veličinu y je v tvare

$$\frac{y(s)}{r(s)} = \frac{k\Theta}{s+1} \tag{6}$$

Prenosová funkcia (6) sa zhoduje s prenosovou funkciou referenčného modelu (4) ak

$$\Theta^\star = \frac{k_m}{k} \tag{7}$$

kde parameter regulátora je označený symbolom * pretože je to ideálna hodnota, pri ktorej je cieľ riadenia splnený. Táto hodnota však nie je známa, pretože k nie je známe. Veľmi dôležité však je, že sme tým ukázali existenciu takej hodnoty. Ak by ani teoreticky neexistovala ideálna hodnota parametra regulátora, ktorú adaptujeme, samotná adaptácia by nemala zmysel. Rovnica (7) sa nazýva podmienka zhody a pri návrhu adaptívneho riadenia je vždy dôležité ukázať, že podmienky zhody existujú a že majú riešenie.

Pre návrh zákona adaptácie teraz použijeme MIT algoritmus. Adaptačná odchýlka je

$$e = y - y_m \tag{8a}$$

$$e = \frac{k\Theta}{s+1} r - y_m \tag{8b}$$

Pretože Θ považujeme za nezávislú od času, môžme písať

$$\frac{\partial e}{\partial \Theta} = \frac{k}{s+1} r \tag{9}$$

čo je citlivostná funkcia potrebná, ako už vieme, v zákone adaptácie podľa MIT algoritmu. Obsahuje však neznáme k. Ak poznáme znamienko k môže byť toto zosilnenie absorbované do adaptačného zosilnenia α. Hodnota α je ľubovolná, preto nie je potrebné poznať presnú hodnotu k, len jeho znamienko, aby bolo možné správne zvoliť znamienko konštanty α a zabezpečiť záporné výsledné znamienko v zákone adaptácie. Predpokladali sme, že znamienko k je známe. Preto je možné citlivostnú funkciu použiť v zákone adaptácie tak ako je, okrem zosilnenia k. Nie je potrebná žiadna aproximácia ako v iných prípadoch, napríklad v príklade v nasledujúcej časti.

Zákon adaptácie potom je

$$s\Theta = -\alpha e \frac{\partial e}{\partial \Theta} \tag{10}$$

kde s predstavuje operátor derivácie podľa času. Po dosadení (9):

$$s\Theta = -\alpha k e \frac{1}{s+1} r = -\alpha e y_m \tag{11}$$

Keďže máme predpis pre zmenu adaptovaného parametra, zákon adaptácie, stačí už len integrovať výstup zákona adaptácie a získame tak signál adaptovaného parametra zákona riadenia.

Všimnime si, že v tomto bode na základe uvedeného nemôžme urobiť žiadne závery o stabilite celého adaptívneho systému.

Prípad, keď je potrebné aproximovať citlivostnú funkciu, pretože obsahuje viac neznámych parametrov riadeného systému je opísaný, spolu s ďalšími detailmi, v nasledujúcej časti.

2.2 Príklad: Statický systém 1. rádu

Nech riadený systém je daný prenosovou funkciou v tvare

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

teda statický systém 1. rádu.

Vo všeobecnosti sú parametre a_0 a b_0 neznáme, alebo sa menia v čase (tak, že je možné uplatniť adaptívne riadenie v rozsahu tohto kurzu). Pre potreby numerickej simulácie nech sa použijú hodnoty $a_0 = 0,55$ a $b_0 = 1,0$.

Nech zákon riadenia je v tvare

$$u = \Theta_1 y + \Theta_2 r \tag{13}$$

2.2.1 Návrh adaptívneho riadiaceho systému

Navrhnime adaptívny riadiaci systém s využitím gradientného algoritmu adaptácie (MRAC — gradientný). Pri tom nech referenčný model je v tvare

$$\frac{y_m(s)}{r(s)} = \frac{b_m}{s + a_m} \tag{14}$$

kde $a_m = 1,0$ a $b_m = 1,0$.

V prvom rade, je vôbec možné, aby sa, v zmysle riadenia s referenčným modelom, zhodoval uzavretý regulačný obvod (URO) s referenčným modelom? Zostavme URO:

$$y = \frac{b_0}{(s+a_0)} u \tag{15a}$$

$$y = \frac{b_0}{(s+a_0)} \left(\Theta_1 y + \Theta_2 r\right) \tag{15b}$$

$$y = \frac{b_0\Theta_1 y}{(s+a_0)} + \frac{b_0\Theta_2 r}{(s+a_0)} \tag{15c}$$

$$(s+a_0)\, y = b_0\Theta_1 y + b_0\Theta_2 r \tag{15d}$$

$$(s+a_0)\, y - b_0\Theta_1 y = b_0\Theta_2 r \tag{15e}$$

$$(s+a_0-b_0\Theta_1)\, y = b_0\Theta_2 r \tag{15f}$$

$$y = \frac{b_0\Theta_2}{(s+a_0-b_0\Theta_1)} r \tag{15g}$$

$$\frac{y}{r} = \frac{b_0\Theta_2}{(s+a_0-b_0\Theta_1)} \tag{15h}$$

Je potrebné zistiť, kedy, a či vôbec, sa bude prenosová funkcia (15h) zhodovať s referenčným modelom (14). Je očividné, že ak by boli parametre zákona riadenia Θ_1 a Θ_2 také, že

$$a_0 - b_0\Theta_1^\star = a_m \tag{16a}$$

$$b_0\Theta_2^\star = b_m \tag{16b}$$

teda

$$\Theta_1^\star = \frac{-a_m + a_0}{b_0} \tag{17a}$$

$$\Theta_2^\star = \frac{b_m}{b_0} \tag{17b}$$

potom by sa URO a RM zhodovali.

Rovnice (17) sú podmienkami zhody. Nie len, že existujú, ale sú aj riešiteľné. To znamená, že má význam pokúšať sa adaptovať zákon riadenia s daným cieľom, pretože je možné teoreticky dosiahnuť zhodu medzi URO a referenčným modelom.

Neadaptívny zákon riadenia (len tak mimochodom)

Využiť podmienky zhody (17) by sme mohli, ak by sme poznali parametre riadeného systému a_0 a b_0. My ich poznáme, keďže sme si ich vyššie uviedli pre potreby numerickej simulácie. Preto sa na chvíľu nevenujme adaptívnemu zákonu riadenia a otestujme neadaptívny, teda taký, ktorého parametre sú dané podmienkami zhody (17). Po dosadení čísiel platí

$$\Theta_1^\star = 1 \tag{18a}$$

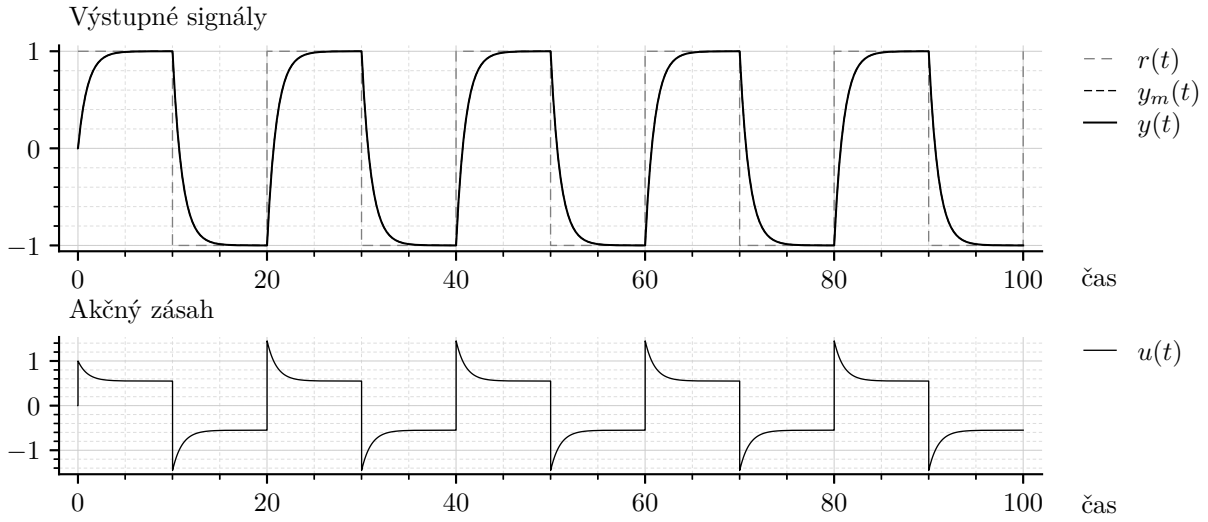
$$\Theta_2^\star = -0,45 \tag{18b}$$

S využitím týchto parametrov zákona riadenia sa URO zhoduje s referenčným modelom, čo ilustruje obr. 3.

Späť k prípadu, keď navrhujeme adaptívny riadiaci systém s využitím gradientného prístupu, inými slovami, s využitím MIT pravidla. Pripomeňme, že zákon adaptácie má v tomto prípade vo všeobecnosti tvar

$$\dot{\Theta} = -\alpha e \frac{\partial e}{\partial \Theta} \tag{19}$$

V tomto prípade však máme dva parametre zákona riadenia a teda v tomto prípade je Θ vektorom, $\Theta^\mathsf{T} = \begin{bmatrix}\Theta_1 & \Theta_2\end{bmatrix}$. Z toho vyplíva, že aj citlivostná funkcia $\frac{\partial e}{\partial \Theta}$ je má dva prvky (je vektorom). V každom prípade, pre nájdenie citlivostnej funckie (citlivostných funkcií) je potrebné vyjadriť adaptačnú odchýlku e tak, aby obsahovala



Obr. 3: Výsledok s použitím podmienok zhody (17).

parametre zákona riadenia Θ. Platí $e = y - y_m$. Ak sa za y dosadí výraz, ktorý opisuje URO, potom

$$e = y - y_m \tag{20a}$$
$$e = \frac{b_0\Theta_2}{(s + a_0 - b_0\Theta_1)} r - y_m \tag{20b}$$
$$e = b_0\Theta_2 \, (s + a_0 - b_0\Theta_1)^{-1} \, r - y_m \tag{20c}$$

Tento výraz potom možno derivovať podľa parametrov zákona riadenia Θ_1 a Θ_2.

Nájdime prvú citlivostnú funkciu $\frac{\partial e}{\partial \Theta_1}$.

$$\frac{\partial e}{\partial \Theta_1} = b_0\Theta_2 \, (-1) \, (s + a_0 - b_0\Theta_1)^{-2} \, (-1) b_0 r \tag{21a}$$
$$\frac{\partial e}{\partial \Theta_1} = b_0 \, (s + a_0 - b_0\Theta_1)^{-1} \left(\frac{b_0\Theta_2}{(s + a_0 - b_0\Theta_1)} r \right) \tag{21b}$$
$$\frac{\partial e}{\partial \Theta_1} = \frac{b_0}{(s + a_0 - b_0\Theta_1)} \, y \tag{21c}$$

Ďalej nájdime druhú citlivostnú funkciu $\frac{\partial e}{\partial \Theta_2}$.

$$\frac{\partial e}{\partial \Theta_2} = b_0 \, (s + a_0 - b_0\Theta_1)^{-1} \, r \tag{22a}$$
$$\frac{\partial e}{\partial \Theta_2} = \frac{b_0}{(s + a_0 - b_0\Theta_1)} \, r \tag{22b}$$

To bolo ľahké. Akokoľvek, nájdené citlivostné fukcie nevieme realizovať, teda nevieme ich použiť v zákone adaptácie. Pretože obsahujú neznáme parametre riadeného systému, parametre a_0 a b_0. Aproximujme citlivostné funkcie. Ak α v zákone adaptácie je ľubovolné číslo, potom aj $\alpha\, b_0$ je ľubovolné číslo. Teda hodnotu b_0 stačí nahradiť len príslušným znamienkom (potrebujeme poznať znamienko parametra b_0). Ďalej, polynóm $(s + a_0 - b_0\Theta_1)$ je charakteristickým polynómom URO. To znamená, že ak by sa URO zhodoval s referenčným modelom, potom by pre tento charakteristický polynóm platilo $(s + a_0 - b_0\Theta_1^\star)$. To ale znamená, že by sa zhodoval s charaketristickým polynómom referenčného modelu. Ak predpokladáme, že URO bude aspoň blízko referenčnému modelu počas celej doby, potom je charakteristický polynóm referenčného modelu vhodnou aproximáciou neznámeho polynómu v citlivostných funkciách. Z uvedeného plynie, že aproximácie citlivostných

funkcií by mohli byť:

$$\frac{\partial e}{\partial \Theta_1} \approx \frac{1}{(s+a_m)}\, y \tag{23a}$$

$$\frac{\partial e}{\partial \Theta_2} \approx \frac{1}{(s+a_m)}\, r \tag{23b}$$

S použitím týchto aproximácií citlivostných funkcií je teraz možné zostaviť zákony adaptácie podľa MIT pravidla, teda:

$$\dot{\Theta} = -\alpha e \frac{\partial e}{\partial \Theta} \tag{24a}$$

$$\begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix} = -\alpha e \begin{bmatrix} \frac{\partial e}{\partial \Theta_1} \\ \frac{\partial e}{\partial \Theta_2} \end{bmatrix} \tag{24b}$$

$$\begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix} = -\alpha e \begin{bmatrix} \frac{1}{(s+a_m)}\, y \\ \frac{1}{(s+a_m)}\, r \end{bmatrix} \tag{24c}$$

Alebo samostatne zapísané:

$$\dot{\Theta}_1 = -\alpha e \left(\frac{1}{(s+a_m)}\, y \right) \tag{25a}$$

$$\dot{\Theta}_2 = -\alpha e \left(\frac{1}{(s+a_m)}\, r \right) \tag{25b}$$

2.2.2 Numerické simulácie

Neadaptívny zákon riadenia (len tak mimochodom)

Vráťme sa na moment k prípadu, keď sme sa zaoberali neadaptívnym prípadom. Pre všeobecnú úplnosť, takto vyzerá simulačná schéma, ktorej výsledkom je obrázok 3.

Výpis kódu 1: Súbor `ar04_ss1r_nonadapt.py`

```
43  def fcn_simSch1(t_start, T_s, finalIndex, sig_r_ext):
44
45      A_m = np.array([[-1]])
46      b_m = np.array([[1]])
47
48      A = np.array([[-0.55]])
49      b = np.array([[1]])
50      c = np.array([[1]])
51
52      ThetaStar = np.array([[b_m[-1, 0]],[A_m[-1, 0] - A[-1, 0]]])
53
54      #----------------------------------------
55      t_log = np.zeros([finalIndex, 1])
56      t_log[0,:] = t_start
57
58      #----------------------------------------
59      x_m_0 = np.array([0])
60
61      x_m_log = np.zeros([finalIndex, len(x_m_0)])
62      x_m_log[0,:] = x_m_0
63
64      #-------------------------
65      x_0 = np.array([0])
66
67      x_log = np.zeros([finalIndex, len(x_0)])
68      x_log[0,:] = x_0
69
70      y_log = np.zeros([finalIndex, 1])
71      y_log[0,:] = np.dot(c, x_0)
72
73      #-------------------------
74      u_log = np.zeros([finalIndex, 1])
75      u_log[0,:] = 0
76
77      #----------------------------------------
78      timespan = np.zeros(2)
79      for idx in range(1, int(finalIndex)):
80
```

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

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

        # ------------
        odeOut = odeint(fcn_LTIS,
                        x_m_log[idx-1,:],
                        timespan,
                        args=(A_m, b_m, sig_r_ext[idx-1,:])
                        )

        x_m_log[idx,:] = odeOut[-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, x_log[idx,:])

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

        omega = np.array([sig_r_ext[idx-1,:], y_log[idx-1,:]])

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

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

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

```

Uvedenú simulačnú schému uvádzame bez dodatočného komentára. Niektoré použité prvky sa čitateľovi objasnia až vtedy ak sa oboznámi s ďalšími nasledujúcimi časťami učebného textu.

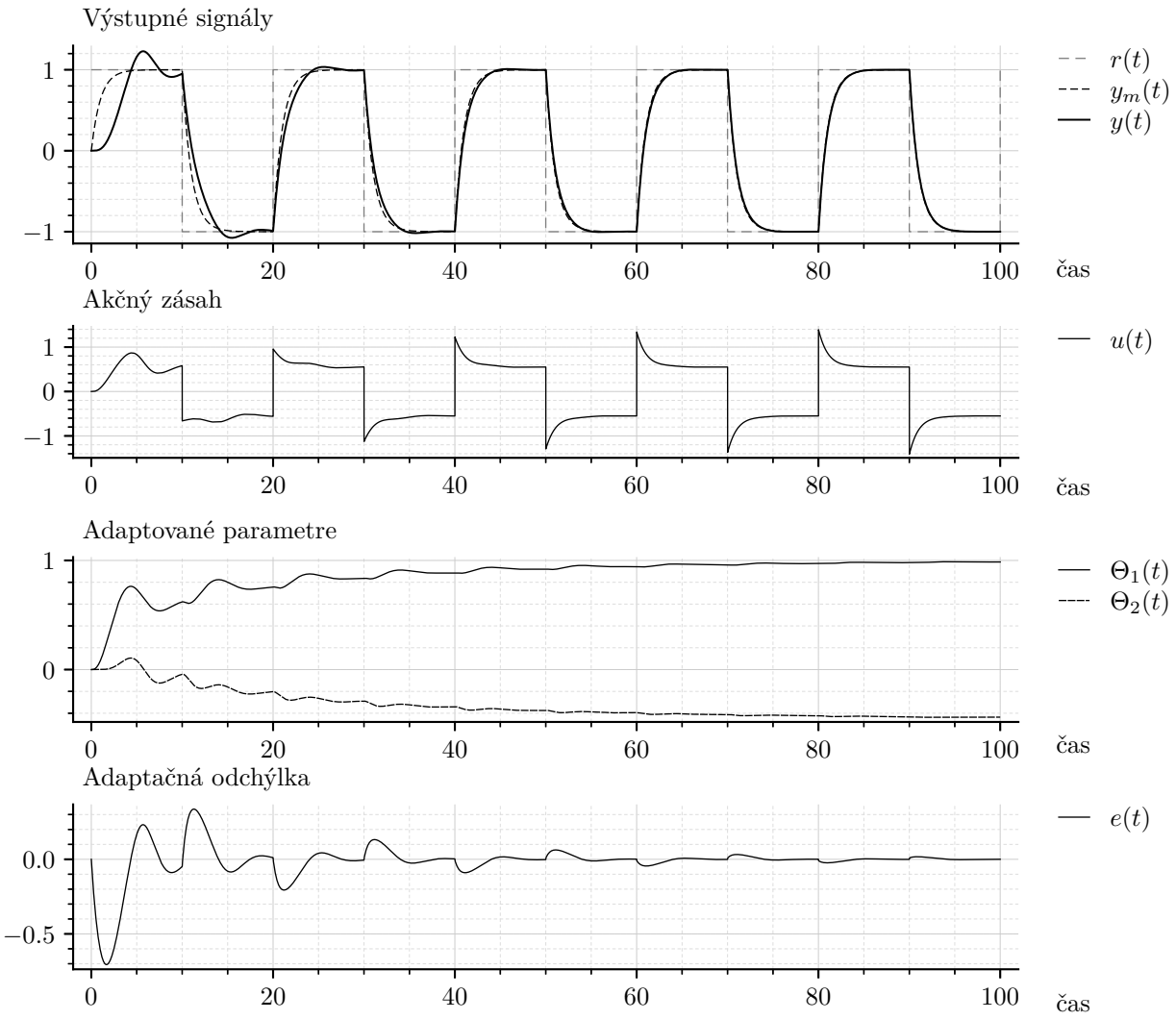
Adaptívny riadiaci systém

Zvoľme $\alpha = 0,5$ a nech $\Theta_1(0) = 0$ a $\Theta_2(0) = 0$. Výsledky numerickej simulácie sú na obr. 4.

Simulačná schéma je v tomto prípade implementovaná nasledovne:

Výpis kódu 2: Súbor `ar04_ss1r_adapt.py`

```
def fcn_simSch2(t_start, T_s, finalIndex, sig_r_ext):

    A_m = np.array([[-1]])
    b_m = np.array([[1]])

    A = np.array([[-0.55]])
    b = np.array([[1]])
    c = np.array([[1]])

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

    #----------------------------------------
    x_m_0 = np.array([0])

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

    #------------------------
    x_0 = np.array([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, x_0)

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

```

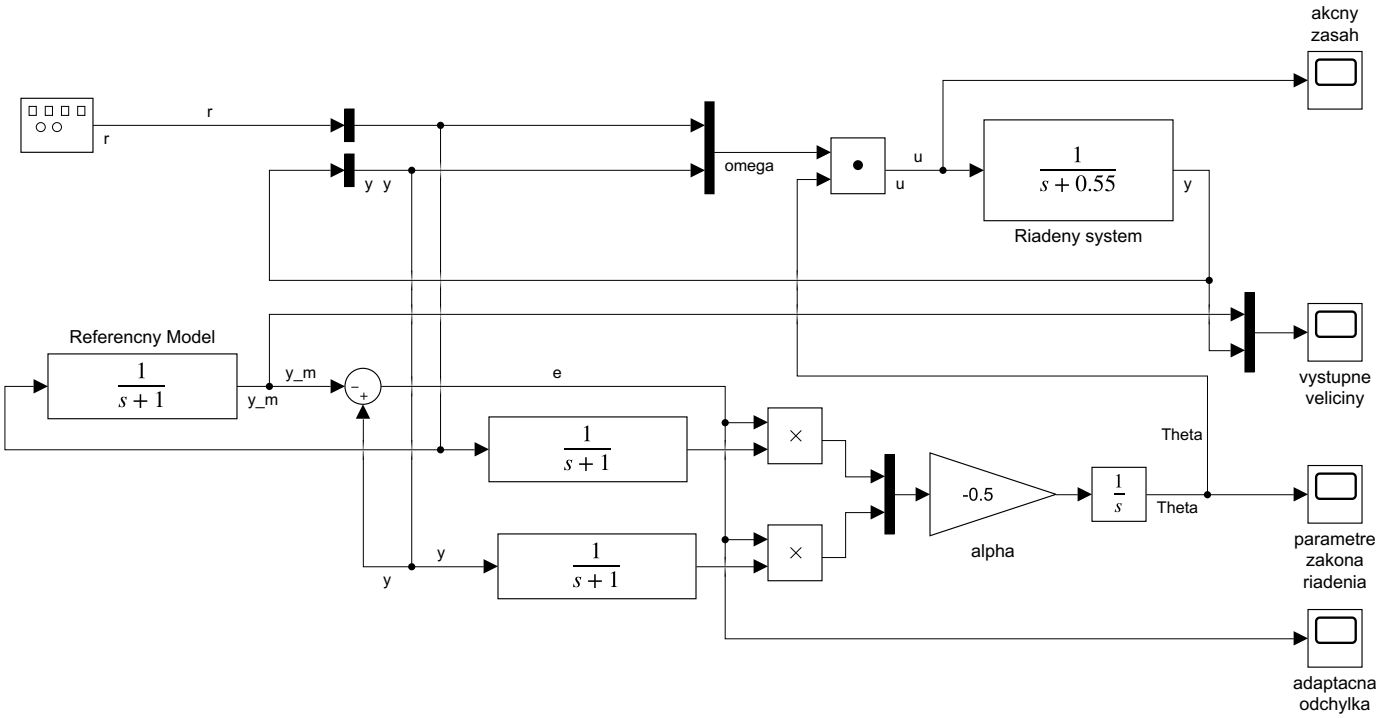


Obr. 4: k časti 2.2.2

```
75      xf1_log = np.zeros([finalIndex, 1])
76      xf2_log = np.zeros([finalIndex, 1])
77
78      Theta_log = np.zeros([finalIndex, 2])
79
80      #----------------------------------------
81      timespan = np.zeros(2)
82      for idx in range(1, int(finalIndex)):
83
84          timespan[0] = t_log[idx-1,:]
85          timespan[1] = t_log[idx-1,:] + T_s
86
87          t_log[idx,:] = timespan[-1]
88
89          # ------------
90          # Referencny model realizovany pomocou riadneho ode solevera
91          # Takyto ode solver nemusi byt vzdy dostupny z pohladu
    implemetnacie riadiaceho systemu
92
93          odeOut = odeint(fcn_LTIS,
94                          x_m_log[idx-1,:],
95                          timespan,
96                          args=(A_m, b_m, sig_r_ext[idx-1,:])
97                          )
98
99          x_m_log[idx,:] = odeOut[-1,:]
100
101         # ------------
102         odeOut = odeint(fcn_LTIS,
103                         x_log[idx-1,:],
```

```
                        timespan,
                        args=(A, b, u_log[idx-1,:])
                        )

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

        # --------------------------
        alpha = 0.5

        omega = np.array([sig_r_ext[idx-1,:], y_log[idx-1,:]])
        adaptErr = y_log[idx-1, 0] - x_m_log[idx-1, 0]

        # Tu je numericka integracia realizovana jednoducho sumatorom - treba teda dbat na krok integrovania - teda tu to, co volame periodou vzorkovania
        dxf1 = np.matmul(A_m, xf1_log[idx-1,:]) + np.matmul(b_m, [omega[0,0]])
        xf1_log[idx,:] = xf1_log[idx-1,:] + dxf1 * T_s
        dTheta_1 = -alpha * adaptErr * xf1_log[idx-1,:]

        dxf2 = np.matmul(A_m, xf2_log[idx-1,:]) + np.matmul(b_m, [omega[1,0]])
        xf2_log[idx,:] = xf2_log[idx-1,:] + dxf2 * T_s
        dTheta_2 = -alpha * adaptErr * xf2_log[idx-1,:]

        Theta_log[idx,:] = np.array([
            Theta_log[idx-1, 0] + dTheta_1 * T_s,
            Theta_log[idx-1, 1] + dTheta_2 * T_s,
        ]).T


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

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

```

Ak by sme takúto simulačnú schému chceli zostaviť v Simulinku, mohla by vyzerať nasledovne:



Obr. 5

2.3 Príklad: Systém 2. rádu s astatizmom

Uvažujme riadený systém daný prenosovou funkciou v tvare

$$\frac{y(s)}{u(s)} = \frac{b_0}{s^2 + a_1 s} \tag{26}$$

kde $y(s)$ je obraz výstupného signálu, $u(s)$ je obraz vstupného signálu a a_1, b_0 sú reálne konštanty – neznáme parametre sústavy. V časovej oblasti je modelom sústavy diferenciálna rovnica v tvare

$$\ddot{y}(t) + a_1\dot{y}(t) = b_0 u(t) \tag{27}$$

Ide o sústavu druhého rádu s astatizmom. Preto je vhodné použiť pre jej riadenie PD (proporcionálno-derivačný) zákon riadenia v tvare

$$u(s) = \Theta_1 \left(r(s) - y(s)\right) - \Theta_2 s y(s) \tag{28}$$

kde r je žiadaná hodnota. Zákon riadenia (28) vznikne úpravou štandardného PD regulátora v tvare

$$u(s) = \left(\Theta_1 + \Theta_2 s\right) e_r(s) \tag{29}$$

kde $e_r = r - y$ je regulačná odchýlka, pričom sa predpokladá, že $r(t) =$ konšt. a teda $\dot{r}(t) = 0$. V časovej oblasti možno napísať štandardný PD regulátor (29) v tvare

$$u(t) = \Theta_1 \left(r(t) - y(t)\right) + \Theta_2 \left(\dot{r}(t) - \dot{y}(t)\right) \tag{30}$$

a upravený PD zákon riadenia (28) má v časovej oblasti tvar

$$u(t) = \Theta_1 \left(r(t) - y(t)\right) - \Theta_2 \dot{y}(t) \tag{31}$$

Dosadením (28) do (26) získame prenosovú funkciu uzavretého regulačného obvodu (URO) v tvare

$$\frac{y(s)}{r(s)} = \frac{b_0\Theta_1}{s^2 + \left(a_1 + b_0\Theta_2\right)s + b_0\Theta_1} \tag{32}$$

Referenčný model nech je definovaný takto

$$\frac{y_m(s)}{r(s)} = \frac{b_{0m}}{s^2 + a_{1m}s + a_{0m}} \tag{33}$$

kde $b_{0m} = a_{0m}$ a a_{1m} sú konštanty. Je zrejmé, že ideálne parametre regulátora sú

$$\Theta_1^\star = \frac{a_{0m}}{b_0} \tag{34}$$

$$\Theta_2^\star = \frac{a_{1m} - a_1}{b_0} \tag{35}$$

Pri ideálnych parametroch je adaptačná odchýlka e nulová

$$e = y - y_m \tag{36}$$

Definujme účelovú funkciu vektora parametrov $\Theta = \begin{bmatrix}\Theta_1 & \Theta_2\end{bmatrix}^T$ v tvare

$$J(\Theta) = \frac{1}{2}e^2(\Theta, t) \tag{37}$$

Pri ideálnych parametroch $\Theta^\star$ je adaptačná odchýlka e nulová a účelová funkcia $J(\Theta)$ nadobúda munimum. Preto navrhnime zákon adaptácie parametrov Θ tak aby sme sa pri ich zmene (adaptácii) pohybovali proti smeru gradientu (vzhľadom na parametre Θ) kvadratickej účelovej funkcie a teda zmenšovali hodnotu účelovej funkcie pretože sa tak približujeme k jej extrému – minimu. Potom aj adaptačná odchýlka e sa bude zmenšovať a výstupná veličina y bude sledovať priebeh veličiny y_m, čo je cieľom riadenia. Zákon adaptácie nech má tvar

$$\dot{\Theta} = -\alpha\frac{\partial J}{\partial \Theta} \tag{38}$$

kde $\frac{\partial J}{\partial \Theta}$ je gradient J vzhľadom na parametre Θ a určuje kladný smer, preto je použité znamienko mínus, čím dostávame smer „proti gradientu“ a α je ľubovolná kladná konštanta, ktorá umožňuje nastaviť „krok“ pohybu, presnejšie rýchlosť pohybu proti

smeru gradientu. Parameter α sa v adaptívnom riadení nazýva *rýchlosť adaptácie* alebo aj *adaptačné zosilnenie*.

Vyjadrime $\frac{\partial J}{\partial \Theta}$ v tvare

$$\frac{\partial J}{\partial \Theta} = \frac{\partial}{\partial \Theta}\left(\frac{1}{2}e^2(\Theta,t)\right) = \frac{1}{2}2e(\Theta,t)\frac{\partial e(\Theta,t)}{\partial \Theta} = e\frac{\partial e}{\partial \Theta} \tag{39}$$

potom zákon adaptácie je v tvare

$$\dot{\Theta} = -\alpha e\frac{\partial e}{\partial \Theta} \tag{40}$$

Rovnicu (36) možno písať v tvare

$$\begin{aligned} e &= \frac{b_0\Theta_1}{s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1} r - y_m \\ &= b_0\Theta_1 \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1} r - y_m \end{aligned} \tag{41}$$

Parciálna derivácia rovnice (41) podľa prvého parametra Θ_1 je

$$\begin{aligned} \frac{\partial e}{\partial \Theta_1} &= \Big((b_0) \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1} \\ &\quad - (b_0\Theta_1) \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-2} b_0 \Big) r \\ &= \Big(b_0 \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1} \\ &\quad \cdot \left(1 - b_0\Theta_1 \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1}\right)\Big) r \\ &= b_0 \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1} (r - y) \end{aligned} \tag{42}$$

a parciálna derivácia rovnice (41) podľa druhého parametra Θ_2 je

$$\begin{aligned} \frac{\partial e}{\partial \Theta_2} &= \left(b_0\Theta_1(-1) \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-2} (b_0 s) \right) r \\ &= -\left(b_0 s\right) \left(s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1\right)^{-1} y \end{aligned} \tag{43}$$

Citlivostné funkcie (42) a (43) obsahujú neznáme parametre sústavy a tiež nateraz neznáme parametre regulátora a preto ich nie je možné použiť. Všimnime si, že ak by mali parametre regulátora práve ideálnu hodnotu, teda $\Theta_1 = \Theta_1^\star$ a $\Theta_2 = \Theta_2^\star$ potom platí

$$s^2 + (a_1 + b_0\Theta_2)\, s + b_0\Theta_1 = s^2 + a_{1m}s + a_{0m} \tag{44}$$

A ďalej, ak poznáme znamienko konštanty b_0 môže byť toto zosilnenie absorbované do adaptačného zosilnenia α. Hodnota α je ľubovolná, preto nie je potrebné poznať presnú hodnotu b_0, len jeho znamienko, aby bolo možné správne zvoliť znamienko konštanty α a zabezpečiť záporné výsledné znamienko v zákone adaptácie. Uvážením uvedeného môžeme citlivostné funkcie aproximovať nasledovne

$$\frac{\partial e}{\partial \Theta_1} = \frac{1}{(s^2 + a_{1m}s + a_{0m})}\,(r - y) \tag{45}$$

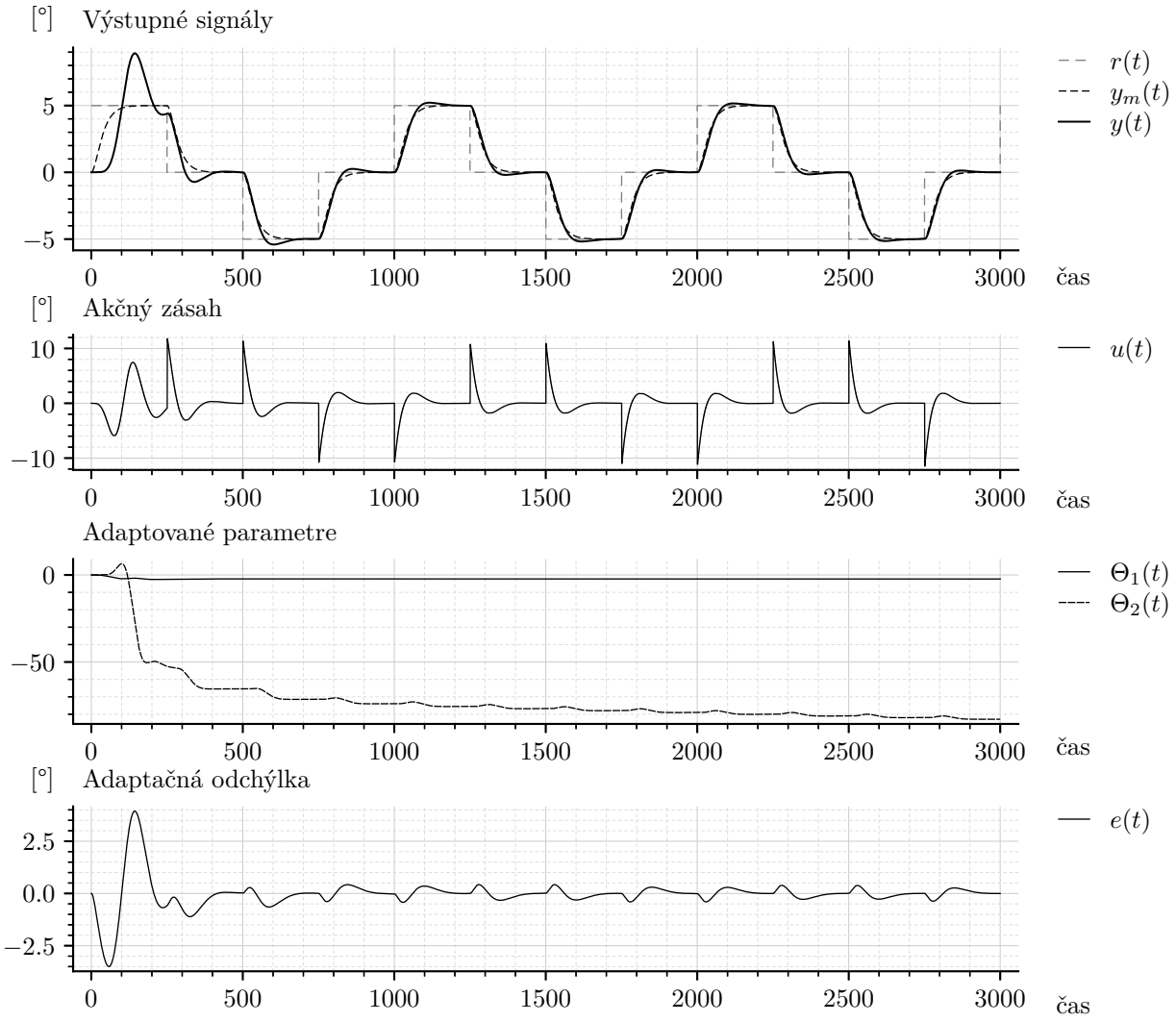
$$\frac{\partial e}{\partial \Theta_2} = \frac{-s}{(s^2 + a_{1m}s + a_{0m})}\, y \tag{46}$$

Zákony adaptácie pre jednotlivé parametre sú potom v tvare

$$\Theta_1 s = -\alpha_1 \left(\frac{1}{(s^2 + a_{1m}s + a_{0m})}\,(r - y) \right) e \tag{47}$$

$$\Theta_2 s = -\alpha_2 \left(\frac{-s}{(s^2 + a_{1m}s + a_{0m})}\, y \right) e \tag{48}$$

kde sme zaviedli samostatné adaptačné zosilnenia α_1 a α_2 pre oba zákony adaptácie, čo umožní ich lepšie naladenie.



Obr. 6: Simulácia pri $v = 5$ [m/s], viď text v časti 2.3.1

2.3.1 Numerické simulácie

Pri návrhu autopilota pre kormidlovanie nákladnej lode sa používa zjednodušený model lode tzv. Nomotov (K. Nomoto – vedec, ktorý sa zaoberal návrhom autopilota pre lode) model, ktorý má tvar prenosovej funkcie:

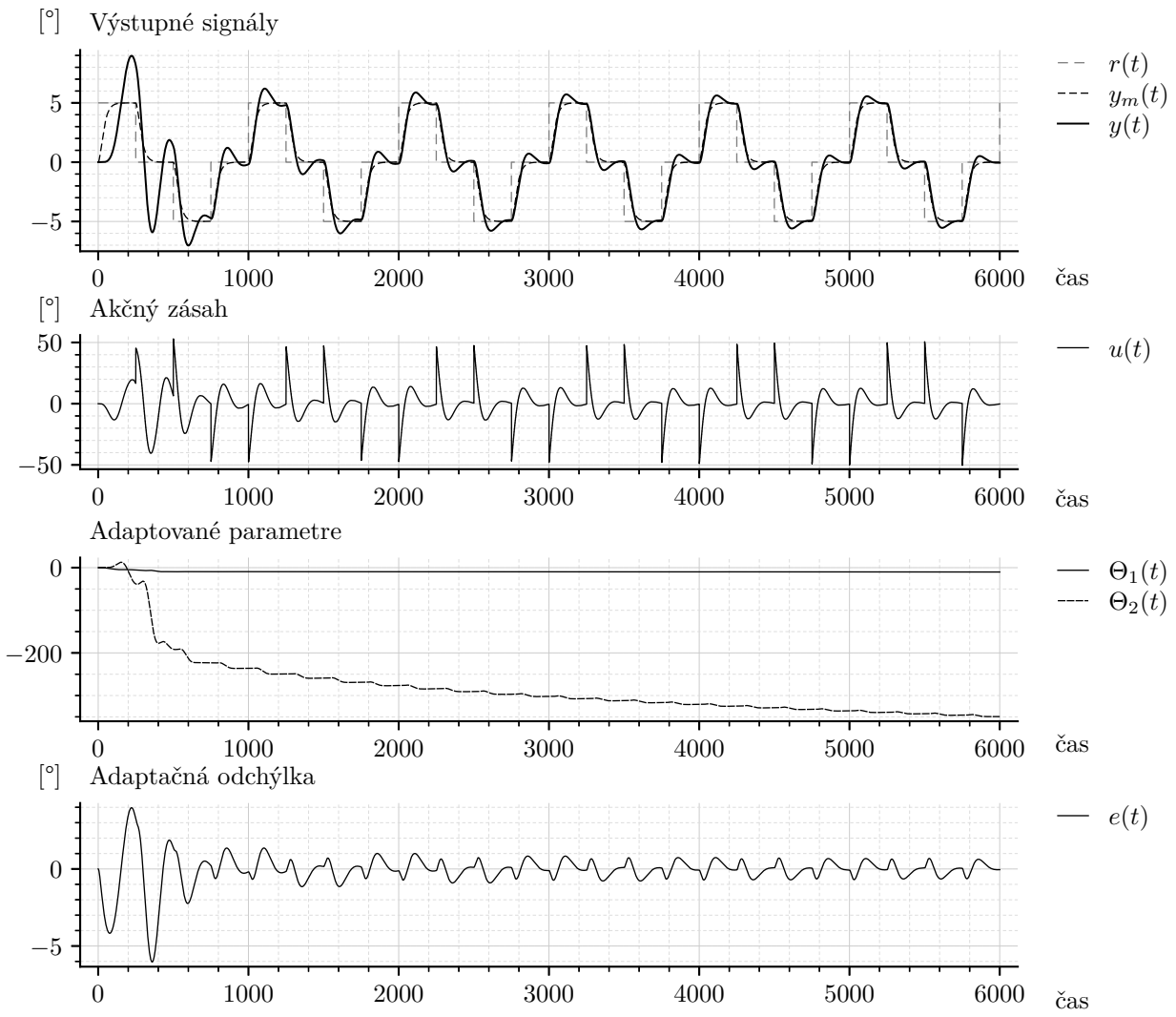
$$\varphi(s) = \frac{\frac{K}{\tau_1}}{s^2 + \frac{1}{\tau_1}s}\,\delta(s) \tag{49}$$

kde $\varphi(s)$ je uhol natočenia lode v radiánoch (azimut, kurz lode), δ je uhol vychýlenia kormidla (riadiaca plocha väčšinou v zadnej časti lode ponorená vo vode) v radiánoch. Parametre v prenosovej funkcii (49) sú definované nasledovne

$$K = K_0 \frac{v}{L} \tag{50}$$

$$\tau_1 = \tau_{10} \frac{L}{v} \tag{51}$$

kde v je rýchlosť lode v smere danom uhlom $\varphi(s)$ v metroch za sekundu, L je dĺžka lode v metroch a K_0, τ_{10} sú konštanty závislé na veľkom množstve faktorov (typ lode atď.) Uvažujme nákladnú loď danú parametrami v Tabuľke 1.



Obr. 7: Simulácia pri $v = 2$ [m/s], viď text v časti 2.3.1

Tabuľka 1: Parametre lode

Parameter	Hodnota
L	161 m
K_0	$-3,86$
τ_{10}	$5,66$
v	5 m s^{-1}

Požiadavky na dynamiku kormidlovania nákladnej lode nech sú definované referenčným modelom v tvare prenosovej funkcie:

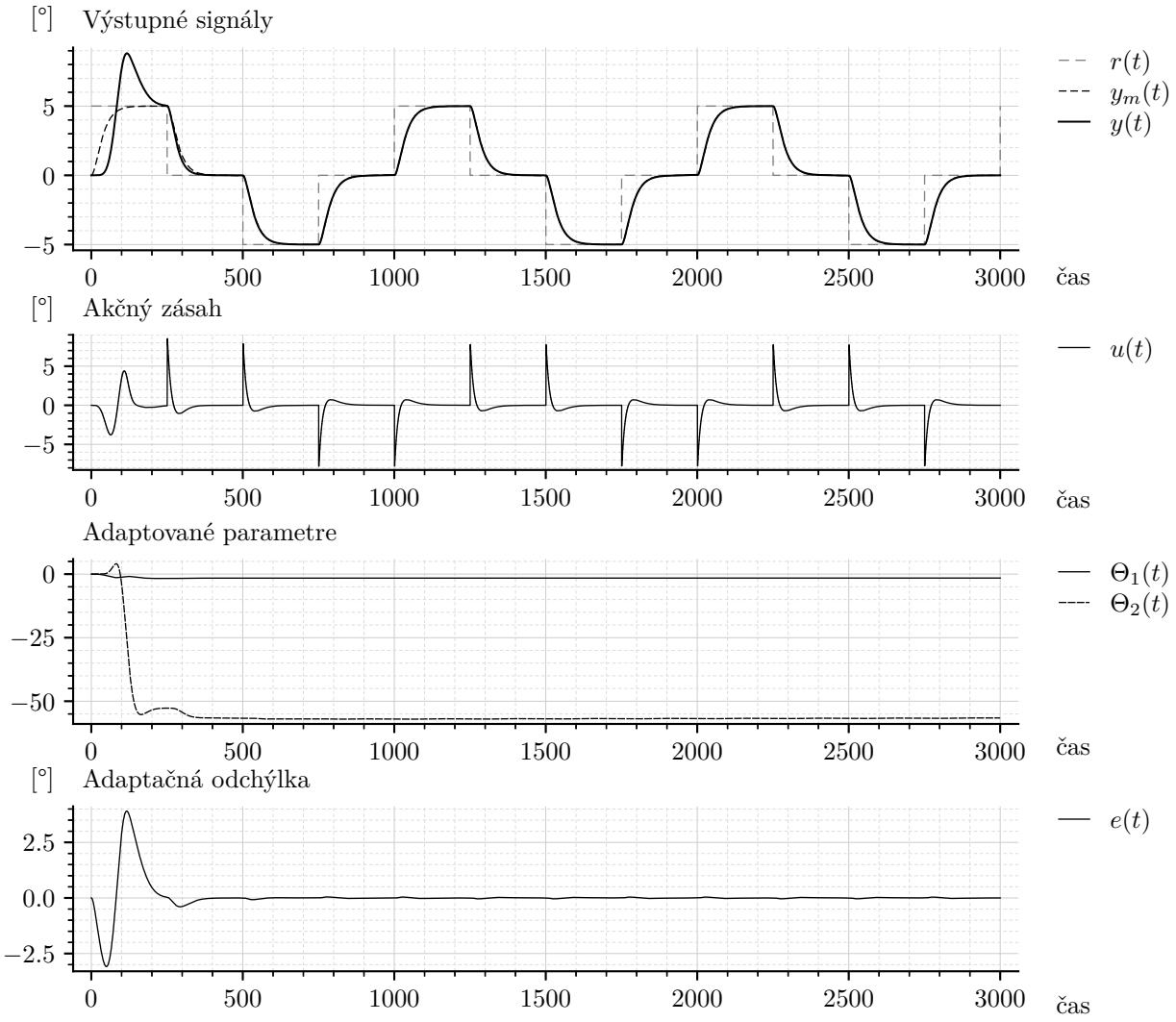
$$\frac{y_m(s)}{r(s)} = \frac{0,0025}{s^2 + 0,1s + 0,0025} \tag{52}$$

kde r je referenčný kurz (rozkaz kapitána) a y_m je požadovaná reakcia lode (priebeh zmeny kurzu).

Je zrejmé, že uvedený opis riadeného systému (lode) a požiadavky na riadiaci systém dané referenčným modelom sa zhodujú so všeobecným zápisom so začiatku tohto príkladu – opis návrhu adaptívneho riadiaceho systému je teda v časti 2.3.

Simulácia 1

Zrealizujme akési „vzorové výsledky“, ktoré získame pri uvažovaní rýchlosti lode $v = 5$ [m/s]. Tieto výsledky sú uvedené na obr. 6. Pri simulácii boli použité (voliteľné) hodnoty $\alpha_1 = 0,025$ a $\alpha_2 = 25$.



Obr. 8: Simulácia pri $v = 8$ [m/s], viď text v časti 2.3.1

Simulácia 2

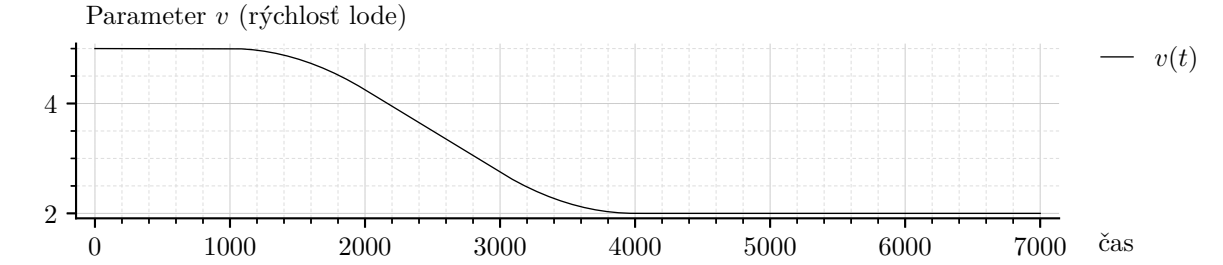
Zmeňme simulovanú rýchlosť lode na $v = 2$ [m/s], teda znížime rýchlosť. Pre tento prípad sú výsledky na obr. 7.

Simulácia 3

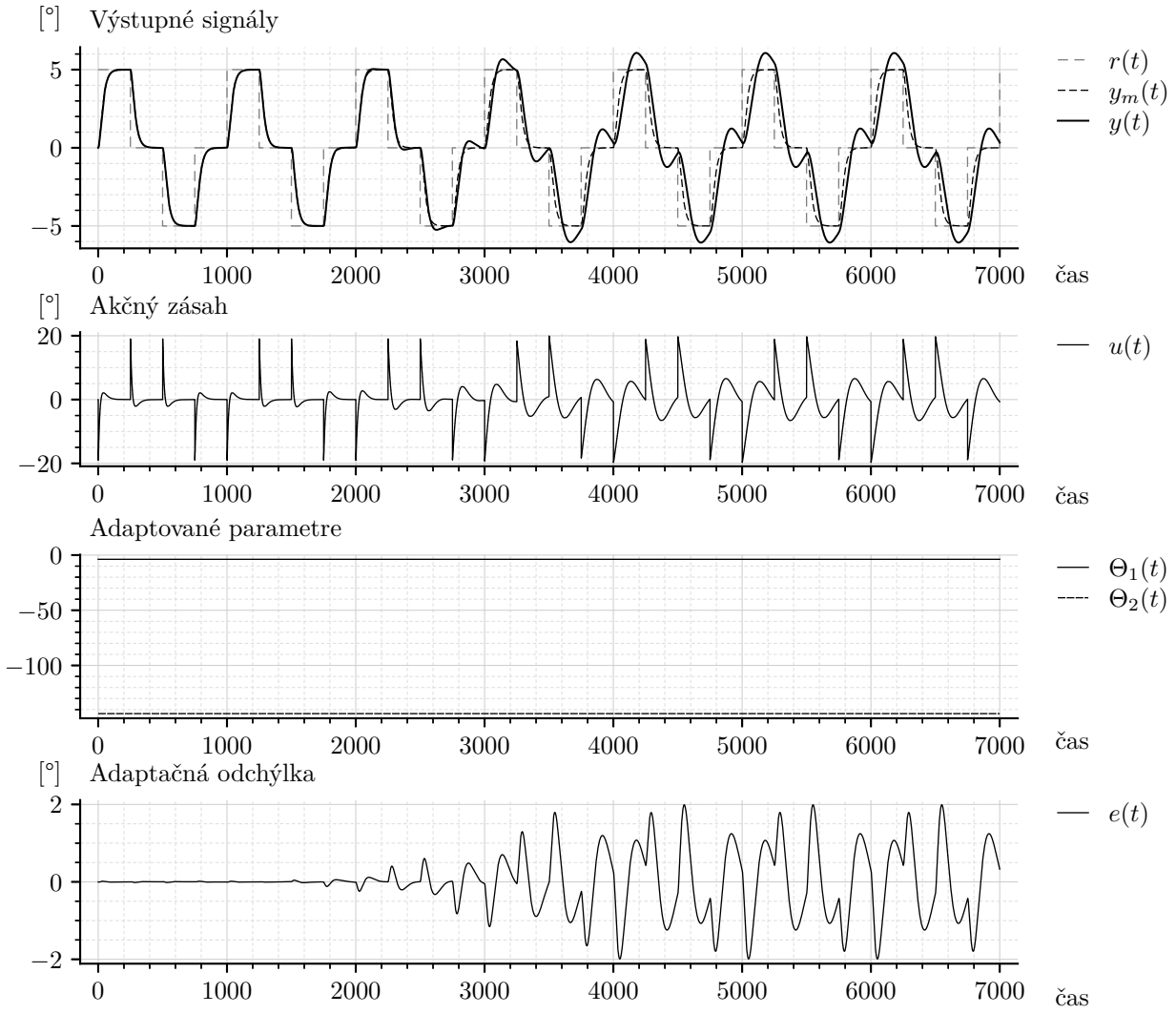
Ak zvýšime rýchlosť lode na $v = 8$ [m/s], tak sa dosiahnu výsledky ako na obr. 8.

Simulácia 4

V predchádzajúcom sme síce skúšali rôzne rýchlosti lode v [m/s], avšak počas celej simulácie bola rýchlosť v [m/s] konštantná. Uvažujme prípad, keď sa bude rýchlosť v [m/s] v čase meniť. Táto časová zmena je zobrazená na obr. 9.



Obr. 9



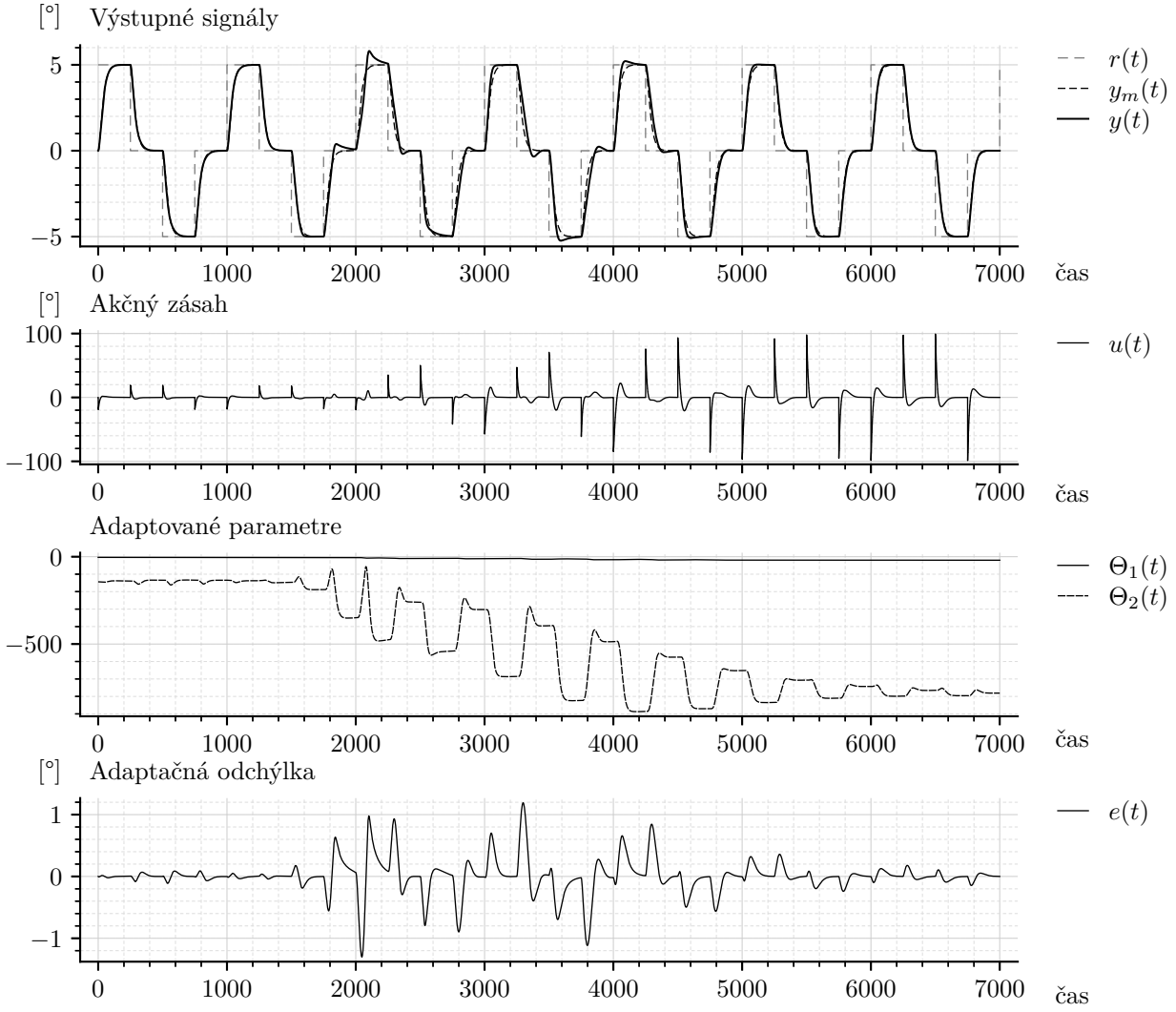
Obr. 10: Simulácia pri v [m/s] podľa obr. 9 pričom parametre zákona riadenia sa neadaptujú.

Rýchlosť sa postupne zmení z hodnoty 5 [m/s] na hodnotu 2 [m/s]. Ak by sme „nastavili“ parametre zákona riadenia tak aby bol cieľ riadenia splnený pri rýchlosti 5 [m/s] a potom ich už nezmenili (neadaptovali), potom by výsledok vyzeral ako na obr. 10.

Nech sú začiatočné parametre zákona riadenia také aby bol cieľ riadenia splnený pri rýchlosti 5 [m/s] ale v tomto prípade uvažujme aj zákon adaptácie – teda parametre zákona riadenia sa môžu adaptovať. Potom výsledok môže vyzerať ako na obr. 11 (nech to pritom ilustruje vhodné nastavenie celkového adaptívneho riadiaceho systému).

3 Otázky a úlohy

1. Aká je úloha referenčného modelu v riadení s referenčným modelom?
2. Ktorý signál je vstupom referenčného modelu?
3. Nakreslite principiálnu schému Adaptívneho riadenia s referenčným modelom.
4. Čo znamená skratka MRAC?
5. V krátkosti vysvetlite mechanizmus adaptácie parametrov regulátora, ktorý využíva MIT algoritmus adaptácie (MIT rule).



Obr. 11: Simulácia pri v [m/s] podľa obr. 9 pričom parametre zákona riadenia sa adaptujú.

6. Model riadeného systému je zadaný v tvare prenosovej funkcie:

$$\frac{y(s)}{u(s)} = \frac{b_0}{s}$$

kde y je výstup, u je vstup, $b_0 > 0$ je neznámy parameter systému. Cieľom riadenia je aby výstup y sledoval výstup referenčného modelu y_m, ktorý je daný prenosovou funkciou

$$\frac{y_m(s)}{r(s)} = \frac{b_m}{s + a_m}$$

kde r je referenčný signál, $a_m = b_m > 0$ sú známe konštanty. Uvažujte použitie zákona riadenia v tvare

$$u = \Theta(r - y)$$

kde Θ je parameter zákona riadenia, ktorý je potrebné adaptovať.

Navrhnite zákon adaptácie použitím gradientnej metódy.

7. Podľa Vášho názoru, akú najväčšiu výhodu a nevýhodu má MIT mechanizmus adaptácie využívajúci gradientnú metódu.

8. Navrhnite adaptívny riadiaci systém s využitím gradientného algoritmu adaptácie (MRAC — gradientný). Stručne komentujte postup návrhu.

Riadený systém: $\frac{y(s)}{u(s)} = \frac{k}{s+1}$, kde $k > 0$. Referenčný model: $\frac{y_m(s)}{r(s)} = \frac{k_m}{s+1}$. Zákon riadenia: $u = \Theta\, r$.

9. Navrhnite adaptívny riadiaci systém s využitím gradientného algoritmu adaptácie (MRAC — gradientný). Stručne komentujte postup návrhu.

   Riadený systém: $\frac{y(s)}{u(s)} = \frac{k}{s}$, kde $k > 0$. Referenčný model: $\frac{y_m(s)}{r(s)} = \frac{c_m}{s+c_m}$. Zákon riadenia: $u = \Theta\,(r - y)$.

10. Navrhnite adaptívny riadiaci systém s využitím gradientného algoritmu adaptácie (MRAC — gradientný). Stručne komentujte postup návrhu.

    Riadený systém: $\frac{y(s)}{u(s)} = \frac{b}{s+a}$, kde $b > 0$. Referenčný model: $\frac{y_m(s)}{r(s)} = \frac{b_m}{s+a_m}$. Zákon riadenia: $u = \Theta_1\, y + \Theta_2\, r$.

Doplnkový text

Obsah

Cieľom tohto textu je najmä uviesť vybrané výsledky Lyapunovovej teórie stability tak ako sú používané v ďalších častiach predmetu. K tomu je však potrebné pripomenúť niektoré pojmy z teórie systémov. Zároveň táto časť slúži na zoznámenie sa s označovaním a zapisovaním, aké je používané v rámci textov k predmetu. Všetky uvedené pojmy a princípy budú ilustrované pomocou názorných príkladov, kde skúmaným systémom bude kyvadlo.

1 Kyvadlo

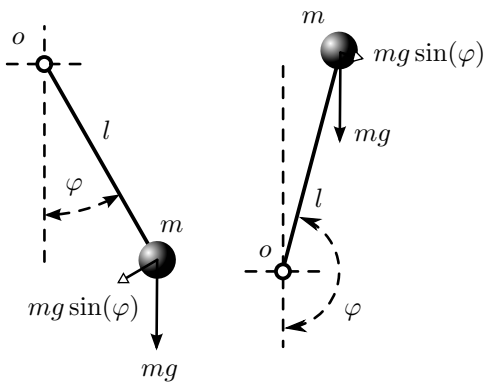
Uvažujme kyvadlo, ktorého kmity sú tlmené viskóznym trením s koeficientom β [kg m^2 s^{-1}]. Kyvadlo je na Obr. 1, kde hmotný bod s hmotnosťou m [kg] pripevnený na ramene so zanedbateľnou hmotnosťou a dĺžkou l [m] kmitá, o označuje os otáčania kolmú na rovinu, v ktorej kyvadlo kmitá, uhol medzi zvislicou a ramenom kyvadla je označený φ [rad] a gravitačné zrýchlenie g [m s^{-2}].

Pohybová rovnica opisujúca dynamiku rotačného pohybu kyvadla je v tvare

$$ml^2\ddot{\varphi} = -\beta\dot{\varphi} - mgl\sin\varphi + u \tag{1a}$$

$$ml^2\ddot{\varphi} + \beta\dot{\varphi} + mgl\sin\varphi = u \tag{1b}$$

kde u [kg m^2 s^{-2}] je externý moment sily pôsobiaci na rameno kyvadla, $\dot{\varphi}$ [rad s^{-1}] je uhlová rýchlosť a $\ddot{\varphi}$ [rad s^{-2}] je uhlové zrýchlenie ramena kyvadla.



Obr. 1: Kyvadlo

2 Niekoľko (dosť veľa) úvodných pojmov

2.1 Modelovanie systému

Rovnica (1) je modelom uvažovaného dynamického systému. Model je matematická reprezentácia v tomto prípade fyzikálneho systému. Model umožňuje uvažovať o systéme a predpovedať ako sa bude systém správať. Uvedený model opisuje vstupno-výstupné správanie sa dynamického systému, kde vstupom je externý moment sily u a výstupom je uhol φ, avšak budeme pracovať aj opisom systému v „stavovom priestore“.

Stav systému je súbor premenných (súbor veličín), ktoré sumarizujú minulosť systému pre potreby predpovede budúcnosti systému. Pre fyzikálny systém je stav zložený z premenných potrebných pre výpočet zmeny hmotnosti, hybnosti a energie. Kľúčovou otázkou pri vytváraní modelu je ako presne má byť táto zmena popísaná.

Stavové premenné tvoria vektor $x \in \mathbb{R}^n$, ktorý sa nazýva *stavový vektor*. Vstupy, pomocou ktorých je systém riadený, tvoria vektor vstupov $u \in \mathbb{R}^p$ a merateľné výstupy systému tvoria vektor výstupov $y \in \mathbb{R}^q$. V tomto prípade máme $p = q = 1$. Dynamický systém potom možno reprezentovať rovnicami v tvare

$$\dot{x} = f(x, u) \tag{2a}$$
$$y = h(x, u) \tag{2b}$$

kde $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ a $h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^q$ sú hladké funkcie. Model v takomto tvare nazývame *model v stavovom priestore*.

Rozmer stavového vektora sa nazýva *rád systému*. Systém (2) sa nazýva *časovo-invariantný* pretože funkcie f a h nie sú priamo závislé na čase t. Pri časovo-variantných systémoch sú. Model pozostáva z dvoch funkcií: funkcia f určuje rýchlosť zmeny stavového vektora ako funkciu stavu x a vstupu u, a funkcia h určuje merateľné výstupy ako funkciu stavu x a vstupu u.

Systém sa nazýva lineárny ak sú funkcie f a h lineárne vzhľadom na x a u. Lineárny model v stavovom priestore má tvar

$$\dot{x} = Ax + Bu \tag{3a}$$
$$y = Cx + Du \tag{3b}$$

Tabuľka 1: Parametre kyvadla

Parameter	Hodnota	Jednotky
m	1	kg
l	1	m
g	$9,81$	m s^{-2}
β	$2 \cdot 0,5 \cdot \sqrt{g/l}$	kg m^2 s^{-1}

kde A, B, C a D sú konštantné matice. Takýto systém sa nazýva lineárny a časovo-invariantný, v skratke LTI z anglického linear and time-invariant. Matica A sa nazýva dynamická matica, matica B sa nazýva vstupná matica, matica C sa nazýva výstupná matica a matica D sa nazýva priamy člen. Drvivá väčšina systémov nemá priamy člen, čo znamená, že vstup nemá priamy vplyv na výstup.

Iná forma lineárnych diferenciálnych rovníc, ktorá je zovšeobecnením avšak linearizovanej dynamickej rovnice kyvadla (o linearizácii neskôr), má tvar

$$\frac{\mathrm{d}^n y}{\mathrm{d}t^n} + a_{n-1}\frac{\mathrm{d}^{(n-1)} y}{\mathrm{d}t^{(n-1)}} + \cdots + a_0 y = b_0 u \tag{4}$$

kde t je nezávisle premenná (čas), $y(t)$ je závisle premenná (výstup) a $u(t)$ je vstup. Zápis $\frac{\mathrm{d}^n y}{\mathrm{d}t^n}$ značí n-tú deriváciu y podľa času t (namiesto n bodiek). Hovoríme, že rovnica (4) je diferenciálna rovnica n-tého rádu, ktorá modeluje dynamiku systému n-tého rádu. Tento model môže byť konvertovaný na model v stavovom priestore napríklad definovaním stavového vektora v tvare

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y \\ \frac{\mathrm{d}y}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}^{(n-1)} y}{\mathrm{d}t^{(n-1)}} \end{bmatrix} \tag{5}$$

potom model v stavovom priestore možno zapísať v tvare

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ -a_{n-1}x_n - \cdots - a_0 x_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_0 u \end{bmatrix} \tag{6a}$$

$$y = x_1 \tag{6b}$$

čo po vhodnej definícii matíc A, B, C a D má tvar (3).

Ešte všeobecnejší systém získame ak výstup bude lineárnou kombináciou všetkých stavových veličín (predpokladáme, že výstup nezávisí priamo od vstupu), teda

$$y = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \tag{7}$$

Potom model v stavovom priestore je

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ b_0 \end{bmatrix} u \tag{8a}$$

$$y = \begin{bmatrix} c_1 & c_2 & \dots & c_{n-2} & c_{n-1} & c_n \end{bmatrix} x \tag{8b}$$

Vráťme sa späť k nelineárnym dynamický systémom. Model kyvadla (1b) je nelineárna diferenciálna rovnica. Rovnicu (1b) upravíme na tvar

$$\ddot{\varphi} = -\frac{\beta}{ml^2}\dot{\varphi} - \frac{g}{l}\sin(\varphi) + \frac{1}{ml^2}u \tag{9}$$

Stavom kyvadla sú dve veličiny: uhol natočenia ramena kyvadla φ a uhlová rýchlosť ramena kyvadla $\dot{\varphi}$. Stavový vektor má preto dva prvky $x^{\mathsf{T}} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$, kde $x_1 = \varphi$ a $x_2 = \dot{\varphi}$. Model kyvadla v stavovom priestore je v tvare

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{\beta}{ml^2}x_2 - \frac{g}{l}\sin(x_1) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u \tag{10a}$$

$$\varphi = x_1 \tag{10b}$$

Toto je nelineárny časovo-invariantný systém druhého rádu.

Mimochodom, miestami v ďalšom texte budeme uvažovať tento systém, avšak bez vstupu, inými slovami externý moment sily je nulový, $u = 0$. Potom

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{\beta}{ml^2}x_2 - \frac{g}{l}\sin(x_1)\end{bmatrix} \tag{11a}$$

$$\varphi = x_1 \tag{11b}$$

Toto je autonómny nelineárny časovo-invariantný systém druhého rádu. Jeho správanie závisí len od začiatočného stavu na začiatku uvažovaného času.

2.2 Numerické riešenie diferenciálnych rovníc (a ODE solver)

Majme rovnicu opisujúcu dynamiku rotačného pohybu kyvadla. Rovnica je v tvare

$$ml^2\ddot{\varphi}(t) + \beta\dot{\varphi}(t) + mgl\sin\varphi(t) = u(t) \tag{12}$$

Samozrejme, ide o diferenciálnu rovnicu. Presnejšie o obyčajnú diferenciálnu rovnicu 2. rádu, ktorá je nehomogénna (vyskytuje sa v nej „externý signál“ (vstup)). Cieľom je nájsť numerické riešenie tejto rovnice pre dané začiatočné podmienky a pre prípadné dané vstupy (vstupné signály).

Analytické riešenie diferenciálnej rovnice je v podstate nejaká funkcia času (a prípadne iných veličín). Numerické riešenie je postuppnosť hodnôt, číselných hodnôt, ktoré pri daných predpokladoch vyhovujú diferenciálnej rovnici. Je postupnosť hodnôt - vektor hodnôt, ku ktorému prislúcha časový vektor určujúci časovú postupnosť hodnôt numerického riešenia. Tu sme vynechali pár miliónov detailov, ale snáď sa dá vytušiť, čo sa tu myslí pod numerickým riešením.

2.2.1 ODE solver

Pre hľadanie numerického riešenia využijeme ODE solver. ODE je skratka pre obyčajné diferenciálne rovnice (ordinary differential equation).

Úlohou ODE solvera je nájsť numerické riešenie na základe rovnice (diferenciálnej), ktorú je možné vo všeobecnosti zapísať v tvare

$$\dot{x}(t) = f\left(t, x(t), \ldots\right) \tag{13}$$

kde f je funkcia, ktorej argumenty sú čas t, prirodzene, samotný výstupný (hľadaný, neznámy) signál $x(t)$ a prípadne iné ďalšie parametre či veličiny - napríklad externý vstup. Uvedená rovnica doslova predpisuje aká je časová zmena signálu $x(t)$. Časová zmena signálu, inými slovami časová derivácia (derivácia podľa času) je označená ako $\dot{x}(t)$.

Ak teda do funkcie f dosadíme hodnoty argumentov (čas, signál $x(t)$, a prípadne iné), získame hodnotu časovej zmeny $\dot{x}(t)$. Na základe informácie o $\dot{x}(t)$, ktorá zodpovedá aktuálnemu (dosadenému) signálu $x(t)$, môžeme určiť hodnotu $x(t)$ o nejaký čas neskôr. Túto novú hodnotu $x(t)$ možno opäť dosadiť do funkcie f a následne nájsť ďalšiu ešte ďalej v čase - atď. ODE solver využíva práve tento jednoduchý princíp pre postupné hľadanie hodnôt (numerických hodnôt) signálu $x(t)$.

Vo všeobecnosti sa uvedený princíp nazýva numerická integrácia. ODE solver teda numericky integruje. Je množstvo metód pre numerickú integráciu, ktoré sa líšia spôsobom riešenia problémov súvisiacich so samotným procesom numerickej integrácie (voľba (optimalizácia) časového kroku integrácie, zohľadnenie matematických vlastností daného typu diferenciálnych rovníc a iné). ODE solvre sa môžu líšiť aj samotnou implementáciou niektorej z metód numerickej integrácie. Podrobnejší opis ODE solvera je nad rámec tohto textu.

2.2.2 Rovnica vyššieho rádu ako sústava rovníc 1. rádu

ODE solver z princípu pracuje s diferenciálnou rovnicou prvého rádu, prípadne so sústavou diferenciálnych rovníc 1. rádu. Napríklad sústavu dvoch rovníc prvého rádu

je možné vo všeobecnosti zapísať v tvare

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = F\left(t, \begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}, \dots\right) \tag{14}$$

V našom prípade hľadáme riešenie pre rovnicu druhého rádu. Každú diferenciálnu rovnicu vyššieho rádu je možné zapísať ako sústavu rovníc prvého rádu. V takejto novej sústave rovníc, vo všeobecnosti, vznikli nové veličiny (signály), ktoré sa vo všeobecnosti môžu líšiť od pôvodných veličín (signálov) v pôvodnej rovnici vyššieho rádu.

Nové veličiny vystupujúce v sústave rovníc sa v teórii systémov súhrnne označujú ako stav systému (stavové veličiny systému). Ak poznáme aktuálny stav systému potom spravidla vieme určiť predchádzajúce aj budúce stavy (vo všeobecnosti).

Napr. v rovnici kyvadla vystupujú veličiny (signály) $\ddot{\varphi}(t)$, $\dot{\varphi}(t)$ a $\varphi(t)$. Je zrejmé (možno nie nad slnko jasné), že ako stav systému je možné zvoliť veličiny $\varphi(t)$ a $\dot{\varphi}(t)$, teda polohu a uhlovú rýchlosť kyvadla. Ak poznáme tieto, poznáme celú históriu a budúcnosť pohybu kyvadla.

Môže existovať viac možností voľby stavových veličín. Pri lineárnych systémoch je možností nekonečne veľa (nekonečne veľa stavových priestorov). Z praktického hľadiska však majú význam len niektoré voľby - napr. pri pohybových systémoch, akým je kyvadlo, sú to prirodzene polohy, rýchlosť, zrýchlenie, trh atď., v závislosti od rádu systému.

Jednou z možností ako previesť rovnicu vyššieho rádu na sústavu rovníc prvého rádu je nasledovný postup. V tomto prípade je zhodou okolností výsledkom aj prakticky využiteľný stavový priestor (stavové veličiny $\varphi(t)$ a $\dot{\varphi}(t)$). Nech

$$x_1(t) = \varphi(t) \tag{15}$$

potom

$$\dot{x}_1(t) = \dot{\varphi}(t) \tag{16}$$

Ďalej nech

$$\dot{x}_1(t) = \dot{\varphi}(t) = x_2(t) \tag{17}$$

a to znamená, že

$$\dot{x}_2(t) = \ddot{\varphi}(t) \tag{18}$$

Tým sme získali veličiny $x_1(t) = \varphi(t)$ a $x_2(t) = \dot{\varphi}(t)$. Je možné zostaviť stavový vektor $x = \begin{bmatrix}x_1(t) & x_2(t)\end{bmatrix}^\mathsf{T}$ a teda $\dot{x} = \begin{bmatrix}\dot{x}_1(t) & \dot{x}_2(t)\end{bmatrix}^\mathsf{T}$.

Cieľom je konkretizovať funkciu F v rovnici

$$\dot{x} = F(t, x, \dots) \tag{19}$$

čo je kompaktný zápis sústavy

$$\dot{x}_1(t) = F_1\left(t, x_1(t), x_2(t), \dots\right) \tag{20}$$

$$\dot{x}_2(t) = F_2\left(t, x_1(t), x_2(t), \dots\right) \tag{21}$$

Prvú rovnicu v tomto prípade máme:

$$\dot{x}_1(t) = x_2(t) \tag{22}$$

Druhá rovnica vyplynie z postrehu, že pôvodnú rovnicu druhého rádu možno zapísať v tvare

$$\begin{aligned}\ddot{\varphi}(t) &= -\frac{\beta}{ml^2}\dot{\varphi}(t) - \frac{g}{l}\sin\left(\varphi(t)\right) + \frac{1}{ml^2}u(t)\\ &= -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right) + \frac{1}{ml^2}u(t)\end{aligned} \tag{23}$$

kde sú využité novo zavedené stavové veličiny $x_1(t)$ a $x_2(t)$. Je zrejmé, že druhá rovnica sústavy je

$$\dot{x}_2(t) = -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right) + \frac{1}{ml^2}u(t) \tag{24}$$

a teda rovnice kyvadla v stavovom priestore sú

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}x_2(t)\\ -\frac{\beta}{ml^2}x_2(t) - \frac{g}{l}\sin\left(x_1(t)\right)\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix} u(t) \tag{25}$$

čím je funkcia F jasne stanovená (skonkretizovaná) a sústava spĺňa požiadavky pre využitie v ODE solveri.

2.2.3 Používanie ODE solvera

ODE solver ako funkcia v programe môže mať napríklad nasledujúce vstupy (argumenty) a výstupy:

```
x = odesolver(fcnF, init, timeVect)
```

kde `x` je, samozrejme, hľadané numerické riešenie. Prvým argumentom je funkcia s názvom `fcnF`, ktorá implementuje sústavu diferenciálnych rovníc v zmysle predchádzajúceho textu. `init` označuje začiatočné hodnoty stavových veličín. `timeVect` označuje časové okamihy (vzorky), v ktorých hľadáme hodnoty numerického riešenia.

MATLAB

MATLAB obsahuje hneď niekoľko ODE solverov. Tu budeme používať `ode45`.

Vytvorme funkciu, ktorá realizuje sústavu diferenciálnych rovníc (25), avšak, uvažujme, že vstupný signál $u(t)$ je nulový. Teda neuvažujme vstupný signál vôbec. Ešte inými slovami, externý moment sily je nulový, $u(t) = 0$ a preto potom možno písať

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}x_2\\ -\frac{\beta}{ml^2}x_2 - \frac{g}{l}\sin(x_1)\end{bmatrix} \tag{26}$$

Toto je autonómny nelineárny časovo-invariantný systém druhého rádu. Jeho správanie závisí len od začiatočného stavu na začiatku uvažovaného času.

Funkcia, ktorá realizuje uvedenú sústavu, môže byť nasledovná:

Celý súbor PravaStr.m

```
function dotx = PravaStr(t,x)

global m l g beta

dotx1 =  x(2);
dotx2 = - (beta/m*l^2)*x(2)  -  (g/l)*sin(x(1));

dotx = [dotx1; dotx2];

end
```

Vytvorme „hlavný skript“, v ktorom všetko potrebné nastavíme a v ktorom budeme volať ODE solver. Ako prvé nech su globálne premenné (v tomto prípade parametre kyvadla):

Časť súbora hlSkript.m

```
global m l g beta

m = 1; %kg
l = 1; %m
g = 9.81; %m/s^2
beta = 2*0.5*sqrt(g/l); %kgm^2/s
```

Definujme časový vektor, ktorý určí pre aké časové okamihy ODE solver vráti numerické riešenie:

Časť súbora hlSkript.m

```
timeVect = 0:0.1:5;
```

Zavolajme ODE solver, pričom ostáva zvoliť začiatočné podmienky - začiatočný stav kyvadla. Nech začiatočný stav je $x_1(0) = 0.25$ [rad] a $x_2(0) = 0$ [rad/s].

Časť súbora hlSkript.m

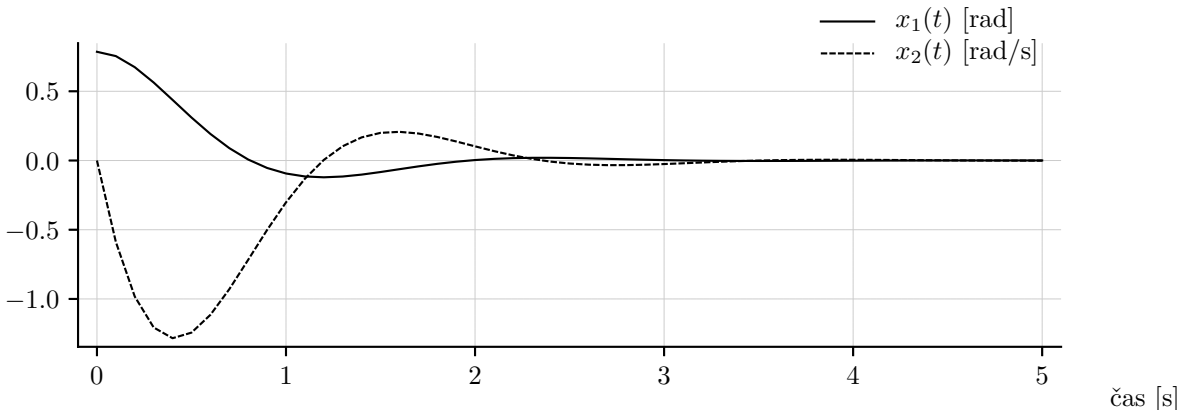
```
8 [t,x] = ode45(@(t,x) PravaStr(t,x), timeVect, [pi/4; 0]);
```

Premenná `x` teraz obsahuje dva stĺpce - prvý stĺpec je prvá stavová veličina a druhý stĺpec je druhá stavová veličina. Pre nakreslenie vypočítaného riešenia:

Časť súbora hlSkript.m

```
9  figure(1)
10 plot(t,x)
```

Výsledné numerické riešenie je graficky znázornené na obr. 2.

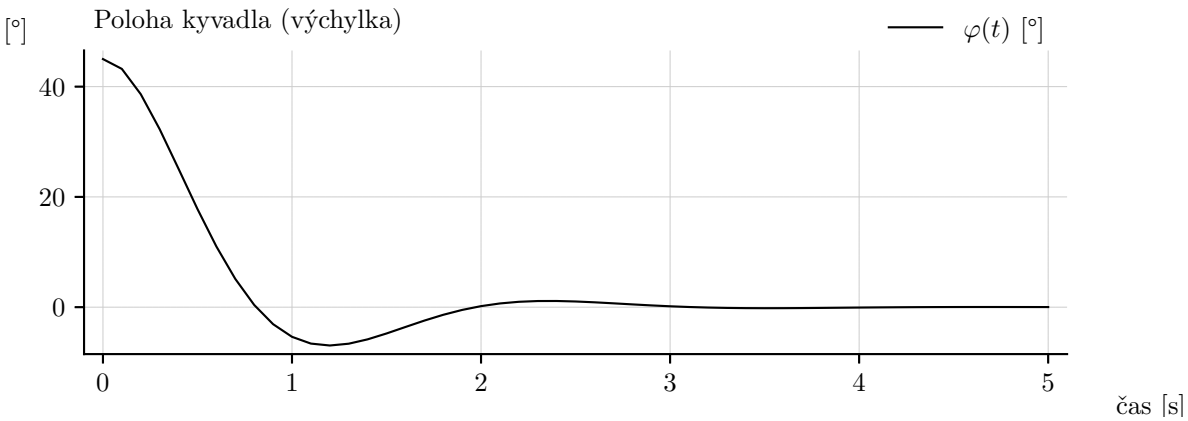


Obr. 2: Grafické zobrazenie numerického riešenia

Na obr. 2 ide však len o akési základné zobrazenie. Zmysluplnejšie by napríklad mohlo byť, ak by sme do grafu nakreslili len priebeh polohy (výchylky) kyvadla samostatne a navyše nie v radiánoch ale v stupňoch – viď obr. 3. Pre takýto obrázok možno do hl. skriptu pridať:

Časť súbora hlSkript.m

```
11 figure(2)
12 plot(t,x(:,1)*180/pi)
```



Obr. 3: Grafické zobrazenie priebehu polohy kyvadla

Python

Pre informáciu, nasledovne by vyzeralo hľadanie numerického riešenia v rámci jazyka Python.

Knižnica SciPy, presnejšie scipy.integrate obsahuje ODEsolver s názvom `odeint`. Vytvorme skript využívajúci tento ODE solver:

Skript v Python-e

```python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

m = 1.0
l = 1.0
g = 9.81
beta = 2 * 0.5 * np.sqrt(g/l)

def fcn_rovniceKyvadla(x, t, u):
    x_1, x_2 = x
    dotx_1 = x_2
    dotx_2 = -(beta/m*l**2) * x_2 - (g/l) * np.sin(x_1) + (1.0/m*l
    **2) * u
    return [dotx_1, dotx_2]


timeVect = np.arange(0, 5.1, 0.1)

u = 0

x = odeint(fcn_rovniceKyvadla,
           [np.pi/4, 0],   # zaciatocne podmienky
           timeVect,
           args=(u,),
           )

plt.figure(1)
plt.plot(timeVect, x)
plt.xlabel(u'cas [s]')
plt.legend(['$x_1(t)$ [rad]', '$x_2(t)$ [rad/s]'])

plt.figure(2)
plt.plot(timeVect, x[:,0]*180/np.pi)
plt.xlabel(u'cas [s]')
plt.ylabel(u'$x_1(t)$ [stupne]')
```

Pozornému čitateľovi iste neuniklo, že uvedený skript v Pythone obsahuje funkciu, ktorá realizuje sústavu diferenciálnych rovníc (25), ale v tomto prípade zahŕňa aj vstupnú veličinu $u(t)$. Táto je potom v tomto prípade nastavená na nulovú hodnotu.

2.3 Vektorové pole a fázový portrét

Kvalitatívne správanie sa nelineárneho dynamického systému je dôležité pre porozumenie kľúčovým konceptom Lyapunovovej teórie stability systémov. Pre analýzu je dôležitá istá trieda systémov nazývaná planárne dynamické systémy. Tieto systémy majú dve stavové veličiny $x \in \mathbb{R}^2$, čo umožňuje znázorniť stavový priestor v rovine so súradnicovým systémom (x_1, x_2). Navyše výsledky kvalitatívnej analýzy platia vo všeobecnosti a môžu byť použité aj pri systémoch vyššieho rádu. Preto sú tieto systémy dôležité z hľadiska analýzy. Do tejto triedy systémov patrí aj model kyvadla.

Výhodným spôsobom ako porozumieť správaniu dynamického systému so stavom $x \in \mathbb{R}^2$ je nakresliť *fázový portrét systému*. Začneme zavedením konceptu *vektorového poľa*. Pre systém obyčajných diferenciálnych rovníc zapísaných kompaktne vo vektorovej rovnici (ako rovnica (11a)) v tvare

$$\dot{x} = F(x) \tag{27}$$

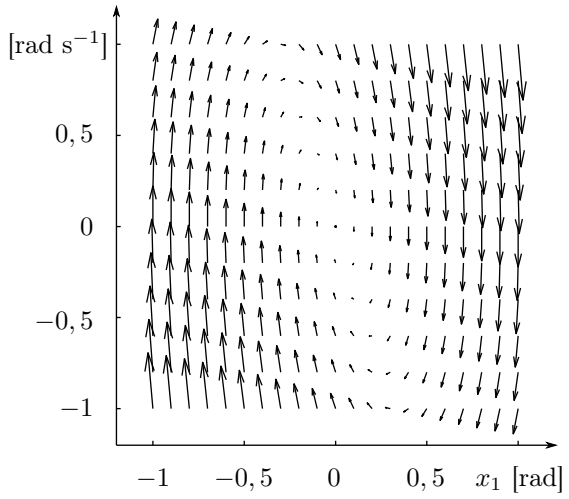
pravá strana rovnice definuje v každom $x \in \mathbb{R}^n$ rýchlosť $F(x) \in \mathbb{R}^n$. Táto rýchlosť hovorí o tom ako sa x mení a môže byť reprezentovaná vektorom.

Pri planárnom dynamickom systéme, každý stav zodpovedá bodu v rovine a $F(x)$ je vektor rýchlosti reprezentujúci veľkosť a smer zmeny (rýchlosti) daného stavu. Tieto vektory môžme vykresliť na mriežke bodov v rovine a získať tak vizuálny obraz dynamiky systému, tak ako na Obr. 4. Pre vykreslenie tohto vektorového poľa boli použité parametre kyvadla uvedené v Tabuľke 1 a tieto parametre budú používané aj v ďalšom.

Vektorové pole na obr. 4 bolo vygenerované v Matlabe použitím nasledujúceho kódu:

Kód pre vygenerovanie obr. 4

```
m = 1; %kg
```



Obr. 4: Vektorové pole znázorňujúce dynamiku kyvadla (obrázok vytvorený v MATLABe, viď text)

```
l = 1; %m
g = 9.81; %m/s^2
beta = 2*0.5*sqrt(g/l); %kgm^2/s
[x1, x2] = meshgrid(-1:.1:1, -1:.2:1);
x1dot = x2;
x2dot = -(beta/m*l^2).*x2 - (g/l).*sin(x1);
quiver(x1,x2,x1dot,x2dot,1.5);
axis([-1.2 1.2 -1.2 1.2])
axis equal
```

Body, v ktorých je vektor rýchlosti nulový sú obzvlášť zaujímavé, pretože definujú stacionárne body systému: ak je autonómny systém v takom stave na začiatku, ostane v tom stave po celý čas.

Fázový portrét (nazývaný aj Fázový diagram) pozostáva z „prúdnic“ nakreslených podľa vektorového poľa. Inými slovami, pre istú množinu začiatočných stavov vykreslíme riešenia diferenciálnej rovnice v rovine a smer pohybu v stavovom priestore vyznačíme šípkou. To zodpovedá sledovaniu „šípky vektorového poľa“ v každom bode stavového priestoru a nakresleniu výslednej trajektórie. Po vykreslení niekoľkých trajektórii pre rôzne začiatočné stavy získame fázový portrét ako na Obr. 5.

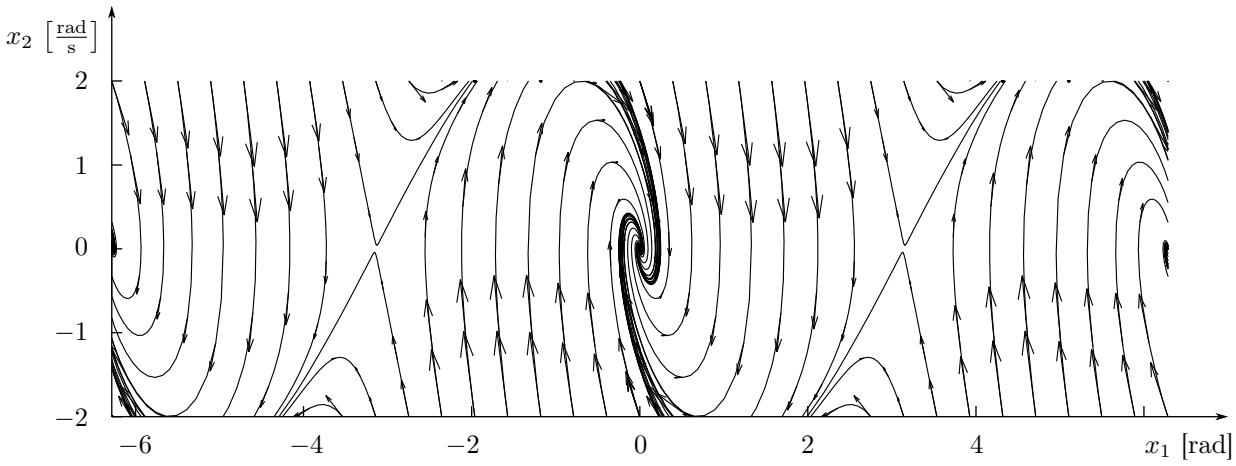
Zdrojový kód pre MATLAB pre získanie tohto obrázku je nasledovný:

Kód pre vygenerovanie obr. 5

```
global m l g beta
m = 1; %kg
l = 1; %m
g = 9.81; %m/s^2
beta = 2*0.5*sqrt(g/l); %kgm^2/s

for uhlovarychlost = -2:4:2
  for uhol = -360:22.5:360
    [t,x]=ode45(@PravaStr,[0 5],[uhol*pi/180 uhlovarychlost]);
    hold on
    stav = x;
    x = x(1:5:end-70,:);
    x1dot = x(:,2);
    x2dot=-(beta/m*l^2)*x(:,2)-(g/l)*sin(x(:,1));
    quiver(x(:,1),x(:,2),x1dot,x2dot,0.5,'k')
    plot(stav(:,1),stav(:,2),'k');
    hold off
  end
end

axis equal
axis([-2*pi 2*pi -2 2])
```

kde funkcia `PravaStr` je

```
function dotx = PravaStr(t,x)
  global m l g beta
```



Obr. 5: Fázový portrét kyvadla (obrázok vytvorený v Matlabe, viď text pre zdrojový kód)

```
3     dotx(1)=x(2);
4     dotx(2)=-(beta/m*l^2)*x(2)-(g/l)*sin(x(1));
5     dotx=dotx';
6   end
```

Fázový portrét je nástroj, ktorý umožňuje posudzovať celkovú dynamiku systému pomocou vykreslenia niekoľkých riešení v stavovom priestore (rovine) systému. Napríklad je možné vidieť, či sa všetky trajektórie s narastajúcim časom približujú k jednému bodu alebo či ide o komplikovanejšie správanie systému. Fázový portrét však nehovorí o veľkosti rýchlosti zmeny stavu (avšak toto môže byť odvodené z dĺžky vektorov vo vektorovom poli systému).

Ekvilibrium dynamického systému je bod v stavovom priestore, ktorý reprezentuje rovnovážne podmienky pre dynamiku systému. Ide o stacionárny bod, v ktorom je vektor rýchlosti trajektórie systému nulový, ako už bolo uvedené.

Hovoríme, že stav x_e je ekvilibrium dynamického systému

$$\dot{x} = F(x)$$

ak $F(x_e) = 0$. Ak má autonómny systém začiatočnú podmienku $x(0) = x_e$, potom ostane v tomto stave a riešenie má tvar $x(t) = x_e$ po celý čas $t > 0$, kde sme uvažovali začiatočný čas $t_0 = 0$.

Stacionárne body (ekvilibriá) patria medzi najdôležitejšiu vlastnosť dynamického systému, pretože definujú stavy s nemennými pracovnými podmienkami systému. Systém môže mať nula, jeden alebo viac stacionárnych bodov.

Stacionárne body uvažovaného kyvadla sú

$$x_e = \begin{bmatrix} \pm n\pi \\ 0 \end{bmatrix} \tag{28}$$

kde $n = 0, 1, 2, \ldots$. Pre párne n sú to stavy keď kyvadlo visí smerom dole a pre nepárne n je kyvadlo v inverznej polohe. Fázový portrét na Obr. 5 je nakreslený pre $-2\pi \leq x_1 \leq 2\pi$, teda na obrázku je päť stacionárnych bodov.

3 Stabilita

Pripomeňme, že sa zaoberáme autonómnym systémom (homogénnou diferenciálnou rovnicou) v tvare

$$\dot{x} = F(x) \tag{29}$$

a tiež pripomeňme, čo rozumieme pod pojmom riešenie systému, alebo skrátene riešenie. Hovoríme, že $x(t)$ je *riešenie* diferenciálnej rovnice (29) na časovom intervale od $t_0 \in \mathbb{R}$ do $t_f \in \mathbb{R}$ ak

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = F\big(x(t)\big) \qquad \text{pre } t_0 < t < t_f \tag{30}$$

Daná diferenciálna rovnica môže mať mnoho riešení, najčastejšie nás však zaujíma úloha so zadaným začiatočným stavom, inými slovami so zadanými začiatočnými podmienkami, kedy $x(t)$ je predpísané v začiatočnom čase t_0 a úlohou je nájsť riešenie vyhovoujúce pre celý budúci čas $t > t_0$. Vtedy $x(t)$ je riešenie diferenciálnej rovnice (29) so začiatočným stavom $x_0 \in \mathbb{R}^n$ v čase t_0 ak

$$x(t_0) = x_0 \quad \text{a} \quad \frac{\mathrm{d}x(t)}{\mathrm{d}t} = F\big(x(t)\big) \quad \text{pre } t_0 < t < t_f \tag{31}$$

Najčastejšie sa stretávame s diferenciálnymi rovnicami, pre ktoré existuje jedinečné riešenie, navyše pre celý čas $t > t_0$ čo znamená že $t_f = \infty$. Častým je tiež, že funkcia F je nezávislá od času, preto môžme uvažovať $t_0 = 0$.

Stabilita riešenia určuje či iné riešenia v blízkosti skúmaného riešenia ostávajú v jeho blízkosti, približujú sa k nemu alebo sa od neho vzďaľujú. Uvedieme niekoľko neformálnych a formálnych definícií stability:
Nech $x(t;a)$ je riešenie diferenciálnej rovnice so začiatočným stavom a. Toto riešenie je stabilné ak iné riešenia, ktoré začínajú v blízkosti a zostávajú v blízkosti $x(t;a)$. Formálne, hovoríme, že riešenie $x(t;a)$ je stabilné ak pre všetky $\epsilon > 0$ existuje $\delta > 0$ taká, že

$$\|b - a\| < \delta \;\Rightarrow\; \|x(t;b) - x(t;a)\| < \epsilon, \;\forall\, t > 0 \tag{32}$$

Všimnime si, že to neznamená, že $x(t;b)$ sa približuje k $x(t;a)$, len ostáva v jeho blízkom okolí. Navyše hodnota δ môže závisieť od ϵ, teda napríklad ak chceme ostať blízko nejakého riešenia potom musíme začať veľmi blízko tohto riešenia. Takto definovaná stabilita sa nazýva *stabilita v zmysle Lyapunova.*

Ilustrujeme uvedenú podmienku (32) na riešení diferenciálnej rovnice kyvadla (11a). Začiatočný čas zvoľme $t_0 = 0$ [s], konečný čas zvoľme $t_f = 1,4$ [s], začiatočnú polohu kyvadla zvoľme $\varphi = 45°$ a začiatočná rýchlosť kyvadla nech je nulová. Začiatočný stav v stavovom priestore je $a = \begin{bmatrix}0,7854 & 0\end{bmatrix}^\mathsf{T}$. Týmto začiatočným podmienkam prislúcha riešenie $x(t;a)$, ktoré je znázornené v stavovom priestore na Obr. 6a, kde je vyznačený aj začiatočný stav a. Nebudeme skúmať všetky $\epsilon > 0$, preskúmame len jedno. Napríklad pre $\epsilon = 0,15$ hľadajme $\delta > 0$, ktorá spĺňa podmienku (32). Taká δ existuje, pretože pre riešenie $x(t;b)$, ktoré začína v stave $b = \begin{bmatrix}0,8727 & 0\end{bmatrix}^\mathsf{T}$ platí, že $\|x(t;b) - x(t;a)\| < \epsilon$, čo je zrejmé z Obr. 6a a aj z Obr. 6b, kde je navyše predĺžený čas riešenia až do nekonečna. Potom sme našli napríklad $\delta = 0,1$ pretože platí

$$\|b - a\| = \sqrt{(0,8721 - 0,7854)^2 + (0 - 0)^2} = 0,0873 < 0,1 \tag{33}$$

čo je tiež zrejmé najmä z Obr. 6b. Týmto sme nezistili nič o stabilite riešenia $x(t;a)$, pretože sme neoverili, či je podmienka (32) splnená pre všetky $\epsilon > 0$.

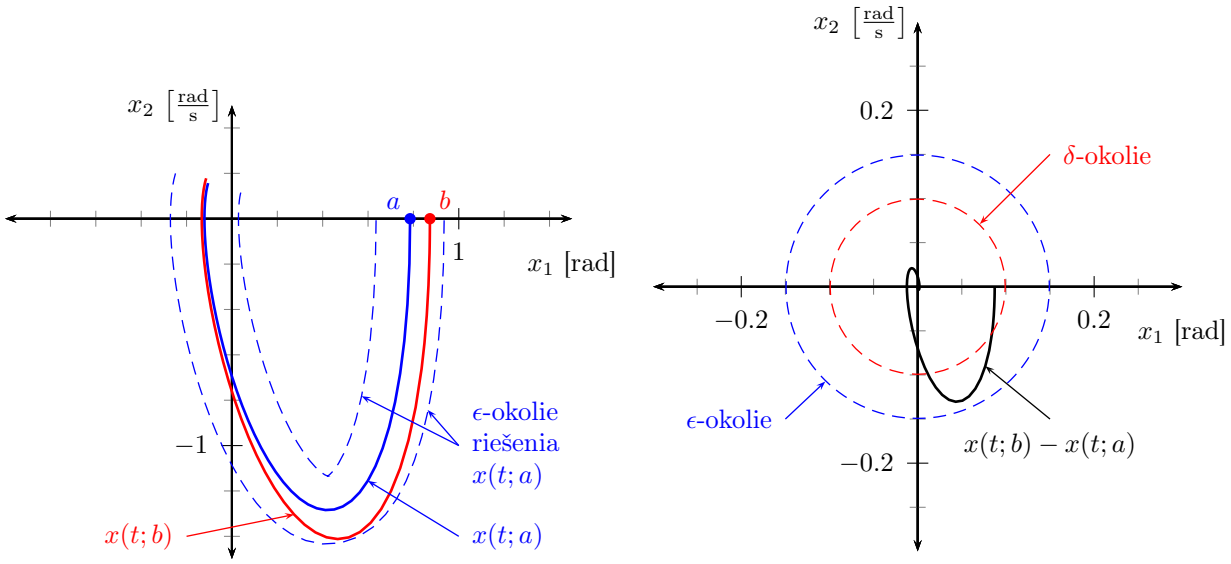
Ak je riešenie stabilné v zmysle Lyapunova, ale trajektórie okolitých riešení k nemu nekonvergujú, hovoríme, že riešenie je *neutrálne stabilné.*

Riešenie $x(t;a)$ je *asymptoticky stabilné* ak je stabilné v zmysle Lyapunova a zároveň $x(t;b) \to x(t;a)$ s rastúcim časom $t \to \infty$ pri začiatočnom stave b, ktorý je dostatočne blízko stavu a.

Veľmi dôležitým špeciálnym prípadom je ak pre skúmané riešenie platí $x(t;a) = x_e$. Potom nehovoríme o stabilite riešenia ale o *stabilite stacionárneho bodu.* Príkladom asymptoticky stabilného stacionárneho bodu sú body

$$x_{e_{-2}} = \begin{bmatrix}-2\pi \\ 0\end{bmatrix}, \quad x_{e_0} = \begin{bmatrix}0 \\ 0\end{bmatrix}, \quad \text{a} \quad x_{e_2} = \begin{bmatrix}2\pi \\ 0\end{bmatrix}$$

na Obr. 5, vidíme, že ak začíname blízko asymptoticky stabilného stacionárneho bodu, s narastajúcim časom sa k nemu približujeme.



(a) Trajektórie riešení systému v stavovom priestore pre rôzne začiatočné stavy a a b v časovom intervale $t_0 < t < t_f$

(b) Trajektória rozdielu dvoch riešení daných začiatočnými stavmi a a b v časovom intervale $t_0 < t < \infty$

Obr. 6: Ilustračný príklad k definícii stability riešenia systému

Riešenie $x(t;a)$ je *nestabilné* ak nie je stabilné. Konkrétnejšie, hovoríme, že riešenie $x(t;a)$ je nestabilné ak pre akékoľvek dané $\epsilon > 0$ neexistuje $\delta > 0$ taká, že ak $\|b-a\| < \delta$ potom $\|x(t;b) - x(t;a)\| < \epsilon$, $\forall t > 0$. Príkladom nestabilného stacionárneho bodu sú body

$$x_{e_{-1}} = \begin{bmatrix} -\pi \\ 0 \end{bmatrix}, \quad \text{a} \quad x_{e_1} = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$$

na Obr. 5.

Predchádzajúce definície nezohľadňujú oblasť, na ktorej môžu byť použité. Presnejšie je definovať riešenie ako *lokálne stabilné* (alebo *lokálne asymptoticky stabilné*) ak je stabilné pre všetky začiatočné stavy $x \in B_r(a)$, kde $B_r(a) = \{x : \|x-a\| < r\}$ je oblasť s polomerom $r > 0$ okolo bodu a. Riešenie je *globálne stabilné* ak je stabilné pre všetky $r > 0$.

3.1 Stabilita lineárnych systémov

Lineárny dynamický systém má tvar

$$\dot{x} = Ax, \quad x(0) = x_0 \tag{34}$$

kde $A \in \mathbb{R}^{n\times n}$ je štvorcová matica. Začiatok stavového priestoru je vždy stacionárnym bodom lineárneho systému a stabilita tohto stacionárneho bodu môže byť určená pomocou vlastných čísel matice A.

Vlastné čísla $\lambda(A)$ sú korene *charakteristického polynómu* systému $\det(sI - A)$, kde $s \in \mathbb{C}$ je komplexná premenná a I je jednotková matica. Konkrétne vlastné číslo (i-te vlastné číslo) označujeme λ_i, pričom $\lambda_i \in \lambda(A)$.

Pre lineárny systém stabilita stacionárneho bodu (ako veľmi dôležitého špeciálneho prípadu spomedzi všetkých riešení) závisí len od matice A, čo znamená, že stabilita je vlastnosť systému. Pre lineárny systém preto hovoríme o stabilite systému namiesto o stabilite konkrétneho riešenia alebo ekvilibria.

Stabilitu lineárneho systému možno zhrnúť do jednej vety:
Systém

$$\dot{x} = Ax$$

je asymptoticky stabilný vtedy a len vtedy keď reálne časti všetkých vlastných čísel matice A sú záporné a systém je nestabilný keď aspoň jedno vlastné číslo matice A má kladnú reálnu časť.

4 Linearizácia a jej použitie pri analýze stability

Výhodnou vlastnosťou diferenciálnych rovníc je, že je často možné určiť lokálnu stabilitu stacionárneho bodu pomocou aproximácie nelineárneho systému lineárnym systémom.

Uvažujme nelineárny systém

$$\dot{x} = F(x) \tag{35}$$

ktorý má ekvilibrium v bode x_e. Zaujíma nás stabilita tohto stacionárneho bodu. Aproximujme (linearzujme) nelineárnu funkciu $F(x)$ v okolí bodu x_e pomocou prvých dvoch členov Taylorvho radu

$$F(x) \approx F(x_e) + \left.\frac{\partial F}{\partial x}\right|_{x_e} (x - x_e) \tag{36}$$

Platí $F(x_e) = 0$, a zavedieme nový stavový vektor $z = x - x_e$. To znamená, že $x = z + x_e$, potom $\dot{x} = \dot{z} + \dot{x}_e$, avšak x_e sa s časom nemení a preto platí $\dot{x} = \dot{z}$. Lineárna aproximácia pôvodného nelineárneho systému v okolí bodu x_e má potom tvar

$$\dot{z} = Az \tag{37}$$

kde

$$A = \left.\frac{\partial F}{\partial x}\right|_{x_e} \tag{38}$$

V prípade kyvadla je nelineárny model systému v tvare (11a) a teda

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix};\ F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) - \frac{\beta}{ml^2}x_2 \end{bmatrix} \tag{39}$$

Linearizujme nelineárny model systému (11a) v okolí rovnovážneho stavu $x_e = \begin{bmatrix} 0 & 0 \end{bmatrix}^\mathsf{T}$. Kľúčovým je výpočet matice A podľa (38). V tomto prípade máme

$$\frac{\partial F}{\partial x} = \begin{bmatrix} \frac{\partial}{\partial x_1}(F_1) & \frac{\partial}{\partial x_2}(F_1) \\ \frac{\partial}{\partial x_1}(F_2) & \frac{\partial}{\partial x_2}(F_2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos(x_1) & -\frac{\beta}{ml^2} \end{bmatrix} \tag{40}$$

Po dosadení hodnôt stacionárneho bodu za $x_1 = 0$ (a $x_2 = 0$) do (40) máme

$$A = \left.\frac{\partial F}{\partial x}\right|_{x_e} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{\beta}{ml^2} \end{bmatrix} \tag{41}$$

a vzľadom na fakt, že v tomto prípade $x_e = 0$ je linearizovaný model systému v tvare

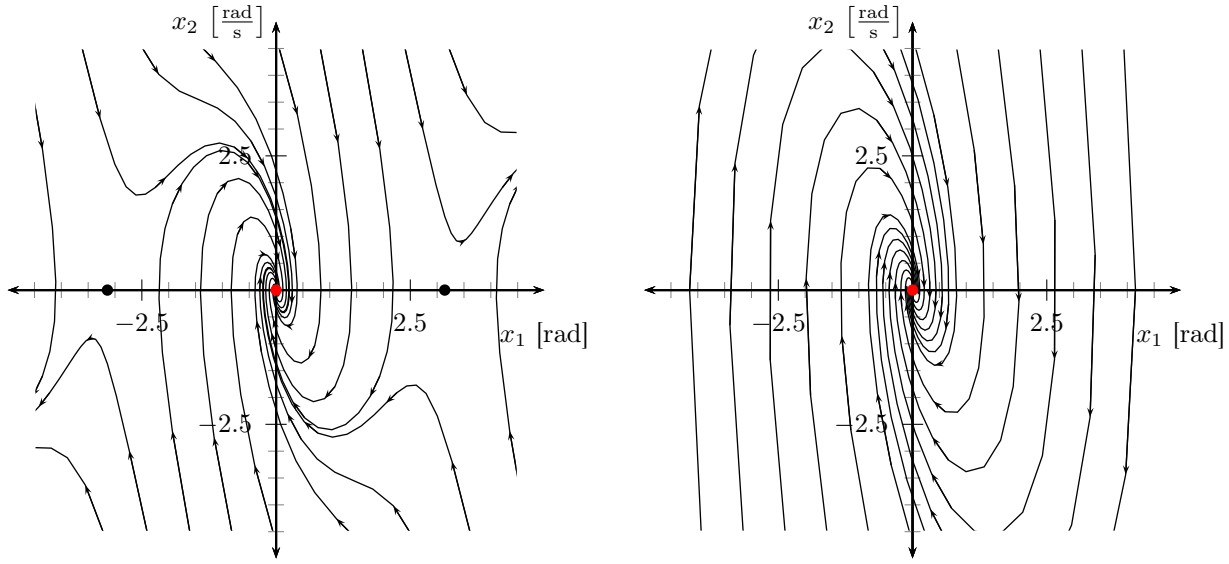
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{\beta}{ml^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{42}$$

Lokálna stabilita stacionárneho bodu nelineárneho systému teraz môže byť určená pomocou vlastných čísel matice A.

Mimochodom výsledný lineárny model (42) je rovnaký, ako keby sme uvažovali, len malé výchylky kyvadla (malé hodnoty uhla φ), pri ktorý dostatočne presne platí, že $\sin(\varphi) = \varphi$ (angl. Small-angle approximation).

Skutočnosť, že lineárny model môže byť použitý pre opis správania nelineárneho systému v okolí rovnovážneho stavu je veľmi výhodná. Je to možné využiť aj pre návrh spätnoväzbového regulátora, ktorý udržiava stav nelineárneho systému v okolí rovnovážneho stavu. Pritom samotný regulátor je navrhnutý pre lineárnu aproximáciu systému. Keďže stav systému je regulátorom udržiavaný blízko stacionárneho bodu, tak aj lineárna aproximácia použitá pri stabilizácii je dostatočne presná.

Na Obr. 7 je porovnanie fázových portrétov nelineárneho modelu kyvadla a linearizovaného modelu kyvadla v okolí začiatku stavového priestoru. Všimnime si, že v okolí stacionárneho bodu (začiatku súradnicového systému) sú tieto fázové portréty takmer identické.



(a) Fázový portrét nelineárneho modelu kyvadla

(b) Fázový portrét lineárneho modelu kyvadla

Obr. 7: Porovnanie fázových portrétov nelineárneho systému a jeho linearizovanej aproximácie

5 Analýza stability pomocou Lyapunovových funkcií

Vráťme sa opäť k nelineárnym systémom v tvare

$$\dot{x} = F(x)$$

Máme definované kedy je riešenie nelineárneho dynamického systému stabilné. Teraz hľadáme odpoveď na otázku ako dokázať, že dané riešenie je stabilné, asymptoticky stabilné alebo nestabilné. Pri fyzikálnych systémoch môžme vyvodiť závery o stabilite podľa toho či systém na danej trajektórii stráca energiu, inými slovami či hodnota energie klesá (disipatívny systém), alebo nie. Zovšeobecnenie tejto techniky pre ľubovolný dynamickým systém je založené na použití Lyapunovových funkcií na miesto energie.

V tejto časti sa venujeme použitiu Lyapunovových funkcií pri určovaní stability riešenia dynamického systému. Konkrétne nás zaujíma stabilita stacionárnych bodov. Je výhodné uvažovať, že vyšetrovaný stacionárny bod je v začiatku stavového priestoru, teda $x_e = 0$. Ak nie je, vždy je možné prepísať rovnice systému zavedením nových (posunutých) stavových veličín $z = x - x_e$.

5.1 Lyapunovove funkcie

Lyapunovova funkcia $V : \mathbb{R}^n \to \mathbb{R}$ sa používa (podobne ako energia) pri vyšetrovaní stability (riešenia) systému. Zjednodušene povedané, ak nájdeme nezápornú funkciu, ktorá je klesajúca pozdĺž trajektórie systému, môžeme usúdiť, že bod, v ktorom táto funkcia dosiahne minimum je stabilným (lokálne) stacionárnym bodom systému.

Pre formálny opis uvedeného je potrebných niekoľko definícií: Hovoríme, že spojitá funkcia V je *kladne definitná* ak $V(x) > 0$ pre všetky $x \neq 0$ a $V(0) = 0$, a podobne, funkcia je *záporne definitná* ak $V(x) < 0$ pre všetky $x \neq 0$ a $V(0) = 0$. Hovoríme, že funkcia V je *kladne semidefinitná* ak $V(x) \geq 0$ pre všetky x, teda $V(x)$ môže byť nulová aj v inom bode ako $x = 0$, podobne, funkcia je *záporne semidefinitná* ak $V(x) \leq 0$ pre všetky x.

Teorém, ktorý charakterizuje stabilitu stacionárneho bodu dynamického systému, ktorý je v začiatku stavového priestoru, je nasledovný:

Nech V je nezáporná skalárna funkcia na $\mathbb{R}^n$ a nech $\dot{V}$ reprezentuje časovú deriváciu

funkcie V pozdĺž trajektórie dynamického systému $\dot{x} = F(x)$:

$$\dot{V} = \frac{\partial V}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial V}{\partial x}F(x).$$

Nech B_r je okolie začiatku súradnicového systému s polomerom r. Ak existuje $r > 0$ taký, že V je kladne definitná funkcia a $\dot{V}$ je záporne semidefinitná funkcia pre všetky $x \in B_r$, potom $x = 0$ je lokálne stabilný stacionárny bod v zmysle Lyapunova. Ak V je kladne definitná a $\dot{V}$ je záporne definitná pre všetky $x \in B_r$, potom $x = 0$ je lokálne asymtoticky stabilný bod.

Ak V spĺňa práve uvedené podmienky, potom hovoríme, že V je (lokálna) *Lyapunovova funkcia* systému.

5.2 Analýza stability

Výsledky uvedené vyššie majú pre planárne systémy svoju geometrickú reprezentáciu. Pre kladne definitnú V môžme definovať vrstevnice ako body v stavovom priestore, ktoré spĺňajú $V(x) = c$, kde $c > 0$. Tam, kde platí $\dot{V} < 0$ vektorové pole ukazuje vždy smerom k „nižšej“ vrstevnici, od vrstevnice c_2 k c_1, kde $c_2 > c_1$. To znamená, že trajektória smeruje k menším a menším hodnotám fukcie V a ak $\dot{V}$ je záporne definitná, potom sa x približuje k nule. Aj toto ilustruje nasledujúci príklad určenia stability stacionárneho bodu kyvadla.

Lyapunovove funkcie často nie je jednoduché nájsť, a tiež nie sú unikátne. V mnohých prípadoch je výhodné pri hľadaní vhodného kandidáta na Lyapunovovu funkciu začať funkciou, ktorá vyjadruje celkovú energiu systému, najmä pri fyzikálnych systémoch. Tak je to aj v nasledujúcom príklade.

Uvažujme nelineárny model kyvadla, podľa (11a), v tvare

$$\dot{x}_1 = x_2 \tag{43a}$$

$$\dot{x}_2 = -\frac{g}{l}\sin(x_1) - \frac{\beta}{ml^2}x_2 \tag{43b}$$

Úlohou je vyšetriť stabilitu stacionárneho bodu $x_e = \begin{bmatrix}0 & 0\end{bmatrix}^\mathsf{T}$. V tomto prípade teda nie je potrebné posúvať súradnicový systém, pretože vyšetrovaný stacionárny bod je v začiatku.

Ako kandidáta na Lapunovovu funkciu použijeme celkovú (mechanickú) energiu systému. Nulovú potenciálnu energiu nech má kyvadlo pre $\varphi = x_1 = 0$. Potenciálna energia je $U = mgh$, kde $h = 2l - l - l\cos(x_1)$, teda

$$U = mgl - mgl\cos(x_1) \tag{44}$$

Pri rotačnom pohybe, vykonávanom kyvadlom, je (rotačná) kinetická energia

$$K = \frac{1}{2}ml^2x_2^2 \tag{45}$$

Celková energia kyvadla je súčtom kinetickej a potenciálnej energie.

Kandidát na Lyapunovovu funkciu systému je potom v tvare

$$V = \frac{1}{2}ml^2x_2^2 + mgl - mgl\cos(x_1) \tag{46}$$

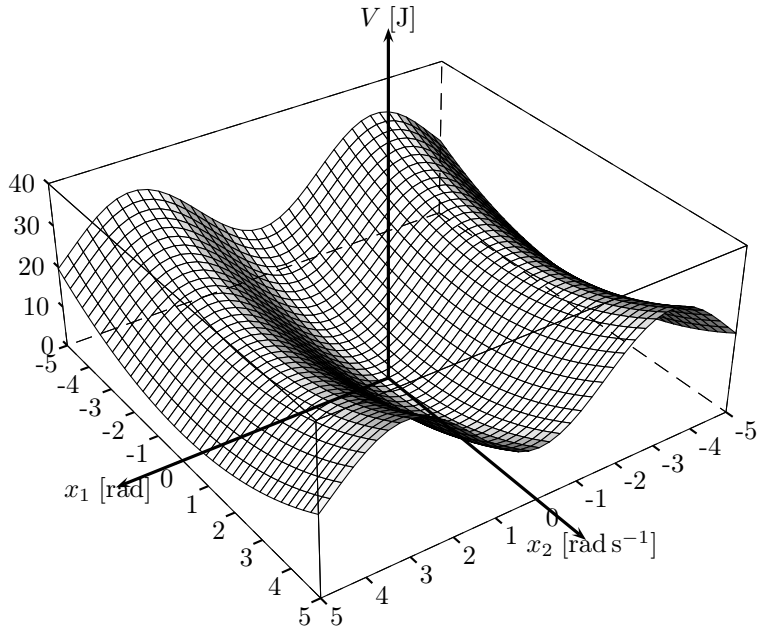
Funkcia (46) je znázornená na Obr. 8. Táto funkcia je nezáporná pre všetky x_1 a x_2, avšak nie je kladne definitná pre všetky x_1 a x_2, pretože bod $x_e = \begin{bmatrix}0 & 0\end{bmatrix}^\mathsf{T}$ nie je jediný, kde funkcia nadobúda nulu. Ak uvažujeme len okolie začiatku B_r s polomerom $0 < r < \pi$, potom funkcia V je kladne definitná, pretože vtedy $-\pi < x_1 < \pi$ a len pre $x_e = \begin{bmatrix}0 & 0\end{bmatrix}^\mathsf{T}$ máme $V(x_e) = 0$.

Časová derivácia funkcie pozdĺž trajektórie je

$$\dot{V} = ml^2x_2\left(-\frac{g}{l}\sin(x_1) - \frac{\beta}{ml^2}x_2\right) + mgl\sin(x_1)x_2 \tag{47a}$$

$$\dot{V} = -mglx_2\sin(x_1) - \beta x_2^2 + mgl\sin(x_1)x_2 \tag{47b}$$

$$\dot{V} = -\beta x_2^2 \tag{47c}$$



Obr. 8: Znázornenie celkovej mechanickej energie kyvadla V ako funkcie stavu kyvadla (46)

Funkcia $\dot{V}$ je záporne definitná pre všetky x_2 a x_1, a teda aj pre tie, ktoré patria do uvažovaného okolia B_r. Preto môžme hovoriť, že na uvažovanom okolí stacionárneho bodu existuje Lyapunovova funkcia systému a preto je tento systém lokálne stabilný.

Ak chceme určiť stabilitu inému stacionárnumu bodu, napr. $x_e = \begin{bmatrix}\pi & 0\end{bmatrix}^\mathsf{T}$, je potrebné najskôr posunúť tento bod do začiatku stavového priestoru. Zavedieme nové posunuté stavové veličiny:

$$z = x - x_e = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \pi \\ 0 \end{bmatrix} \tag{48}$$

potom platí

$$x_1 = z_1 + \pi \tag{49a}$$
$$x_2 = z_2 \tag{49b}$$

Dosadením (49) do pôvodného systému (43) máme nový posunutý systém, ktorý má vyšetrovaný bod v začiatku stavového priestoru v tvare

$$\dot{z}_1 = z_2 \tag{50a}$$
$$\begin{aligned} \dot{z}_2 &= -\frac{g}{l}\sin(z_1 + \pi) - \frac{\beta}{ml^2} z_2 = \\ &= \frac{g}{l}\sin(z_1) - \frac{\beta}{ml^2} z_2 \end{aligned} \tag{50b}$$

Mimochodom, (50) je model inverzného kyvadla.

Môžme vyskúšať zvoliť toho istého kandidáta na Lyapunovovu funkciu ako (46) s dosadením (49), potom

$$V = \frac{1}{2} ml^2 z_2^2 + mgl - mgl\cos(z_1 + \pi) \tag{51a}$$
$$V = \frac{1}{2} ml^2 z_2^2 + mgl + mgl\cos(z_1) \tag{51b}$$

Okamžite vidíme, že táto funkcia nemôže byť kandidátom na Lyapunovovu funkciu, pretože v nulovom bode (začiatku) nemá nulovú hodnotu $V(0) \neq 0$ a teda nie je kladne definitná.

Aby sme dokázali, že daný stacionárny bod je *nestabilný* (v tomto prípade vieme, že je nestabilný), musíme nájsť takú funkciu $V(x)$, ktorá je v okolí rovnovážneho bodu kladne definitná a jej časová derivácia pozdĺž trajektórie systému je *tiež kladne definitná funkcia.*
. . . nájsť takú funkciu pre inverzné kyvadlo vôbec nie je jednoduché. Užitočnejším v tomto prípade je linearizovať systém v okolí daného stacionárneho bodu a určiť stabilitu tohto lineárneho systému pomocou vlastných čísiel dynamickej matice alebo tiež pomocou Lyapunovovej funkcie. O druhej možnosti hovorí nasledujúca časť.

5.3 Lyapunovova rovnica

Ako sme už uviedli, vo všeobecnosti nie je jednoduché nájsť vhodného kandidáta na Lypunovovu funkciu. Pre lineárny dynamický systém v tvare

$$\dot{x} = Ax \tag{52}$$

je však možné skonštruovať Lyapunovovu funkciu systematickým spôsobom, nasledovne:
Uvažujme kvadratickú formu ako kandidáta na Lyapunovovu funkciu

$$V(x) = x^\mathsf{T} P x \tag{53}$$

kde $P \in \mathbb{R}^{n\times n}$ je symetrická matica, teda platí $P = P^\mathsf{T}$. Podmienka aby V bola kladne definitná je v tomto prípade ekvivalentná podmienke, že P je *kladne definitná matica*: $x^\mathsf{T} P x > 0$, $\forall\, x \neq 0$, čo zapisujeme ako $P > 0$. Platí, že ak je matica P symetrická, potom je matica P kladne definitná vtedy a len vtedy keď všetky vlastné čísla tejto matice sú reálne a kladné.

Časová derivácia uvažovaného kandidáta na Lyapunovovu funkciu pozdĺž trajektórie systému v tomto prípade je

$$\dot{V} = \frac{\partial V}{\partial x}\dot{x} = x^\mathsf{T}\left(A^\mathsf{T} P + PA\right) x = -x^\mathsf{T} Q x \tag{54}$$

Požiadavka aby $\dot{V}$ bola záporne definitná znamená, že matica Q musí byť kladne definitná. Preto, pre nájdenie Lyapunovovej funkcie pre lineárny systém je postačujúce zvoliť maticu $Q > 0$ a vyriešiť tzv. *Lyapunovovu rovnicu*:

$$A^\mathsf{T} P + PA = -Q \tag{55}$$

čo je lineárna algebraická rovnica, ktorú možno riešiť pomocou lineárnej algebry. Dá sa ukázať, že táto rovnica má riešenie vždy ak vlastné čísla matice A sú v ľavej polrovine komlexnej roviny (ich reálne časti sú záporné) a navyše nájdená P je kladne definitná ak Q je kladne definitná. Preto je vždy možné nájsť kvadratickú Lyapunovovu funkciu pre stabilný lineárny systém.

6 Otázky a úlohy

1. Je zadaný systém v tvare:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -x_1 - x_2$$

   Načrtnite (zhruba) fázový portrét systému alebo aspoň vektorové pole systému.

2. Linearizujte model kyvadla v okolí stacionárneho bodu $x_e = \begin{bmatrix}\pi & 0\end{bmatrix}^\mathsf{T}$. Bonus: Aký typ stacionárneho bodu je uvedený bod? Svoju odpoveď zdôvodnite.

3. Pomocou kandidáta na Lyapunovovu funkciu vyšetrite stabilitu systému

$$\dot{x}_1(t) = x_2(t)$$
$$\dot{x}_2(t) = -x_1(t) + 2x_2(t)$$

4. Pomocou kandidáta na Lyapunovovu funkciu vyšetrite stabilitu systému.

$$\dot{x}_1 = -k_1\, x_1 + x_2 \qquad k_1 > 0$$
$$\dot{x}_2 = -k_2\, x_1 \qquad k_2 > 0$$