Teorie systémů

Michal Šerý

Automatizace

Obsah

1. Plán
2. Historie
3. Kybernetika
   - Řídicí technika
4. Teoretické základy
5. Popis systémů

Obsah

6 **Ukázka**
- Octave nebo Matlab; 1. řád
- Octave nebo Matlab; 2. řád

7 **Regulace**

8 **Dravec - kořist**
- Matematicko-Matlabovské intermezzo
- Ukázka řešení Lotka-Volterra rovnice

9 **Regulace**

10 **Unifikované signály**

Obsah

11 **Regulátory**

12 **Regulátor PID**

13 **Regulátor - pseudokód**
- ON/OFF
- P
- I
- PI
- D
- PID

14 **Příklad: Mechanický harmonický oscilátor**

15 **Přenos regulátoru**

Obsah

16 **Příklad: OCTAVE**

17 **Přechodová charakteristika**

18 **PLC**

- Charakteristika
- Dělení PLC
- Vlastnosti PLC
- Metody programování PLC
  - IL - Instruction List
  - LD - Ladder Diagram
  - ST - Structured Text
  - CFC - Continuous Function Chart
  - FBD - Function Block Diagram
  - SFC - Sequential Function Chart

19 **Blokové schéma PLC**

20 **PLC - zapojení do řídicího procesu**

Obsah

21 **Běh programu PLC**

22 **Druhy regulace**

23 **Fuzzy**

24 **Akční členy řídicího obvodu**

- Relé
- Stykač
- Relé
- Algebra blokových schémat

Co dodělat

- Maticová algebra
- Booleovské řízení
- Fuzzy řízení
- Algebra blokových schémat
- Stability systému
- Vyšetřování stability
- Akční členy
  - SSR
  - Krokové motory
  - Servopohony

"Dědkologie"

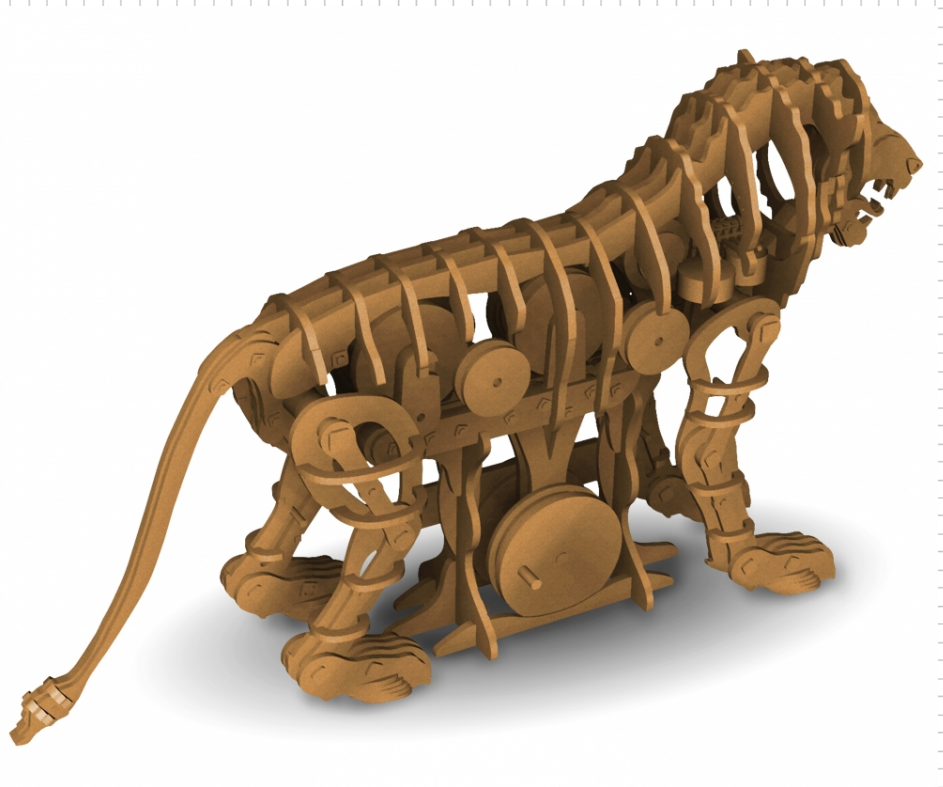
- Herón Alexandrijský - zvaný Méchanikos
  (1. století n.l. (ca 10 - 70))
  První umělé zázraky, které způsobovalo využití páry a teplého vzduchu v zařízení. Zařízení pracovala na principu teplovzdušného motoru, jehož princip popsal Herón v knize „Pneumatika“.

"Dědkologie"

Ve středověku vznikaly různá mechanická zařízení - jejichž autory byli především hodináři.
Vznikali hlavně různé mechanické hračky, orloje, zvonkohry atd. Tyto automaty již obsahovaly jednoduché programovací zařízení v podobě např. válce s kolíčky, kotouče s otvory, zářezy apod.

"Dědkologie"

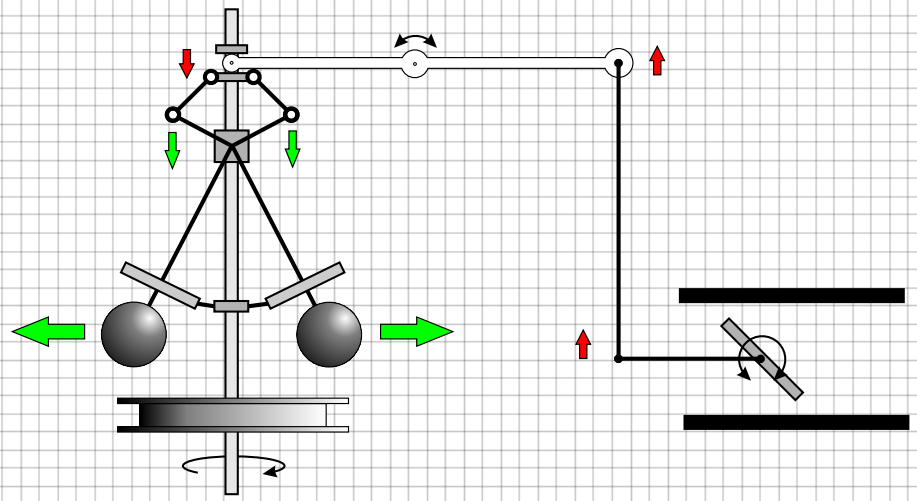
- Leonardo da Vinci (1452 - 1519)



Obrázek: Mechanický lev

"Dědkologie"

- James Watt (1736 - 1819)



Obrázek: Wattův odstředivý regulátor

"Dědkologie"

- Joseph Marie Jacquard (1752 - 1834) Jacquardův tkalcovský stav (kolem roku 1800), u kterého bylo již možno „naprogramovat“ vzor látky pomocí pásu s otvory, který procházel „čtecím zařízením“. Byl to předchůdce děrných štítků a děrné pásky.

"Dědkologie"

- Claude Elwood Shannon (1916–2001)
- Norbert Wiener (1894-1964)

Proč

- Co je to KYBERNETIKA
- Dceřinné obory
- Co může tento předmět poskytnout
  - *všeobecný přehled* o fungování zkoumaných objektů
  - Zavede základní charakteristiky a principy vztahů mezi objekty
  - Metody zkoumání
  -

Co přináší automatizace?

- Zlepšuje přesnost a jakost
- Zvyšuje rychlost výrobního procesu
- Spoří energii a materiál
- Odstraňuje subjektivní chyby člověka
- Zvyšuje produktivitu práce
- Nahrazuje člověka v místech pro něj nebezpečných a kde jeho schopnosti již nestačí
- Realizuje monotónní operace

Kybernetika

Z řeckého kybernétés ($\kappa\upsilon\beta\epsilon\rho\nu\acute{\eta}\tau\eta\varsigma$) - kormidelník. Je to vědní disciplína, která se zabývá obecnými principy řízení a přenosem a zpracováním informací ve strojích, živých organismech a společenstvích.

Kybernetika

Z řeckého kybernétés ($\kappa\upsilon\beta\epsilon\rho\nu\acute{\eta}\tau\eta\varsigma$) - kormidelník. Je to vědní disciplína, která se zabývá obecnými principy řízení a přenosem a zpracováním informací ve strojích, živých organismech a společenstvích.

Obory současní kybernetiky

- Systémová teorie: Stavové popisy, zpětné vazby, řízení, ...
- Přenos informací: Kanály, informační entropie, kryptování ...
- Umělá inteligence: Strojové vnímání a učení, robotika, ...
- Biokybernetika: Vazba člověk-stroj, náhrady orgánů ...

...

Obecné pojmy

Informace

V nejobecnějším smyslu je informace chápána jako údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Nosičem informace je nějaká fyzikální veličina - signál.

Informace

V nejobecnějším smyslu je informace chápána jako údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Nosičem informace je nějaká fyzikální veličina - signál.

Informace inženýrská

definovaná C. E. Shannonem jako „snížení neurčitosti systému“ a matematicky vyjádřená jako logaritmus pravděpodobnosti nějakého jevu (přenosu zprávy) při rovnoměrném rozložení hustoty pravděpodobnosti.

Obecné pojmy

Kauzalita

Následky nemohou předběhnout své příčiny, naopak se za svými příčinami opožďují.

Příčina $\Rightarrow$ vztah $\Rightarrow$ následek

Obecné pojmy

Řízení

Cílevědomé působení na řízený systém s tím záměrem, abychom dosáhli požadovaného stavu systému.

Řízení

- Ovládání
- Regulace

Řízení

- Ovládání
- Regulace

Ovládání

- je řízení bez zpětné vazby. Lze ho použít u jednoduchých, dobře poznaných procesů či objektů (jednoduché automatické pračky).

Obecné pojmy

Řízení

- Ovládání
- Regulace

Ovládání

- je řízení bez zpětné vazby. Lze ho použít u jednoduchých, dobře poznaných procesů či objektů (jednoduché automatické pračky).

Regulace

- je řízení se zpětnou vazbou. Umožňuje udržování určitých fyzikálních veličin na stanovených hodnotách. Při tom se v průběhu regulace zjišťují hodnoty těchto veličin a srovnávají se s hodnotami, které mají mít. Podle zjištěných odchylek, které jsou mírou přesnosti regulace, se zasahuje do regulovaného procesu tak, aby tyto odchylky byly udržovány na minimu.

Ovládání

Ovládání spouští a odstavuje jedno či více zařízení, nebo provádí sled operací v určitém pořadí.

- Místní (elektrické nářadí, kotoučová pila, soustruh, bruska)
- Dálkové (ovládání z velína, výkonové vypínače, HDO)
- Automatické (například čerpání z jímky)
- Programové (automatická pračka, automatický start zařízení)

Entropie/Informace

Příběh

C. E. Shannon zavedl vztah a přemýšlel jak svou veličinu pojmenovat. Tehdy mu John von Neuman údajně řekl:
„You should call it entropy, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it alredy has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage.“
Pokud jde o tu vaši funkci neurčitosti, doporučoval bych vám nazývat ji entropie. Ze dvou důvodů. Jednak se ve statistické fyzice definuje stejným způsobem a druhak nikdo pořádně neví, co to vlastně je. Kdoví, jestli ta vaše entropie a entropie termodynamická nejsou jedno a totéž.? „ . . . nikdo opravdu neví, co entropie skutečně je, tak v diskusi, budete mít vždy výhodu.“
Tak vznikl (dříve v termodynamice) v informační teorii pojem ENTROPIE.

Matematicky vyjádřeno

$$I = -Kn \sum_{i=1}^{s} P_i \ln P_i$$

I ... informace (celkové množství informace ve zprávě)
K ... konstanta
n ... délka zprávy
s ... počet znaků abecedy
P_i ... pravděpodobnost výskytu i-tého znaku abecedy

Informační entropie

$$H = \frac{I}{n} = -K \sum_{i=1}^{s} P_i \ln P_i$$

H … informační entropie (střední hodnota informace na jeden symbol zprávy)

Konstanta K

Například pro nejjednodušší abecedu o dvou znacích (například 0;1), pravděpodobnost výskytu znaků je stejná (P=0,5), položíme-li $H = 1$, potom

$$H = -K(0,5 \ln 0,5 + 0,5 \ln 0,5) = 1$$

$$H = -K(-\ln 2) = 1 \Rightarrow K = \frac{1}{\ln 2}$$

Potom informaci měříme v **bit**ech (z anglického **bi**nary digi**t**)

Definice systému

Zakladatelem obecné teotie systémů je považován rakouský biolog **Ludwig von Bertalanfy**.
Vychází z teorie, že každý objekt je tvořen souborem komponent a vazeb mezi nimi. Tato vazby (interakce) mají pro výsledné chování zásadní význam a díky nim celek vykazuje vlastnosti, které neplynou z analýzy vlastností jednotlivých částí.

Definice systému

Obecná teorie systémů se zabývá metodami, nástroji, principy, problémy, popisy a technikami týkajícími se systémů.
Základní pojmy: systém, systémový přístup, teorie systémů, systémové aplikace (kybernetika).
Definice pojmu systém:

- je daná množina prvků, spolu s množinou vazeb, mezi prvky navzájem a s okolím (definice užívaná v kybernetice)
- je daná množina stavů spolu s množinou přechodů mezi nimi

Dynamický systém: systém jehož výstup (stav) je závislý nejen na okamžitých hodnotách vstupů (a jejich derivací) nýbrž i na předcházejících hodnotách vstupů a stavů (hloubka paměti).

Systémový přístup

Metodický postup: Metodický postup, který upravuje pravidla daných činností tak, aby bylo dosaženo určitého cíle s minimem ztrát (maximem zisků).
Pradávná zásada českých inženýrů 4N: **Nikdy Na Nic Nezapomeň!!!**
podle typu veličin

Systémová teorie

Soubor definic, axiomů, vět, pravidel, nástrojů na vysokém stupni abstrakce, umožňující pracovat s pojmy potřebnými pro popis a řešení daného problému.
často používané pomocné discipliny:

- matematika, včetně numerického a grafického zpracování veličin
- modelování – simulace
- počítačový experiment

Definice systému

$$S = \{P; R; U; Y\}$$

- P – množina prvků
- R – množina relací
- U – množina vstupů
- Y – množina výstupů

Systém

Abstrakce

Vytvoření teoretického modelu k reálnému objektu. Dělá to pozorovatel.

Pozorovatel

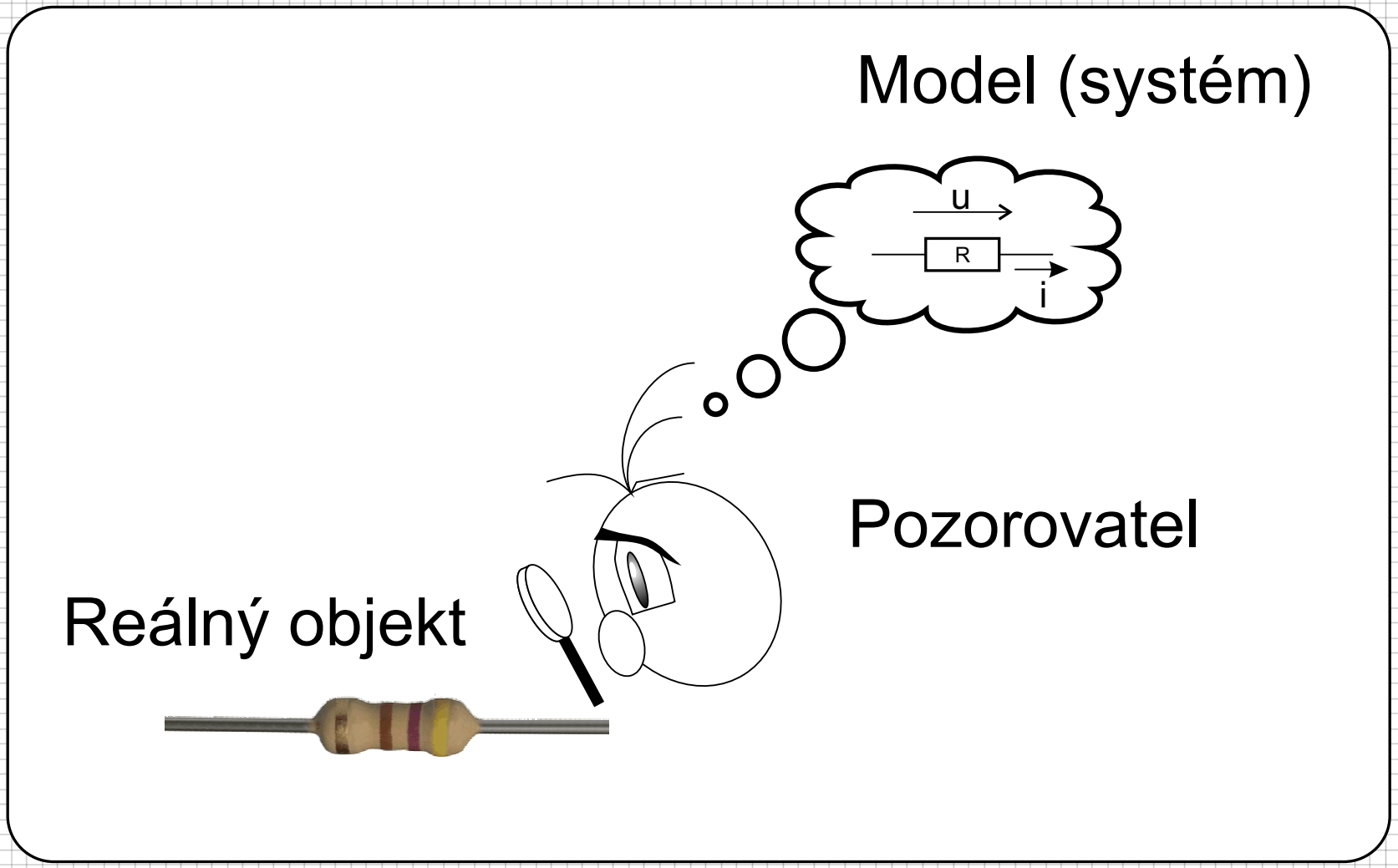
Tvůrce modelu

Určení systému

Pozorovatel definuje abstraktní systém vymezením:

- důležitých veličin reálného systému a jejich vzájemných vztahů
- hranici systému
- ostatní veličiny/vztahy tvořící okolí systému
- určíme, které veličiny jsou vstupní a výstupní, definujeme orientaci systému

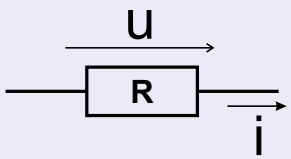
Vznik modelu



Reálný systém *S*

Má ∞ mnoho entit (napětí, tvrdost, teplota, proud, výška, odpor, . . .)
Má ∞ mnoho vztahů

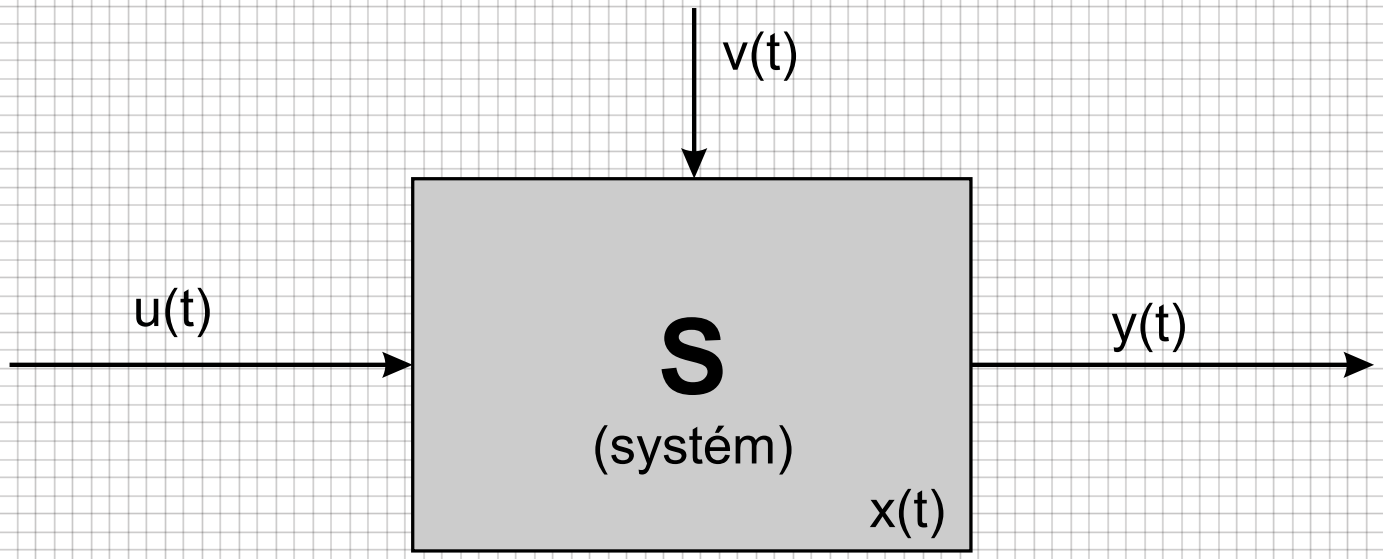
Abstraktní systém *S'*



Má konečný počet entit (napětí, proud, odpor)
Má konečný počet vztahů $U = R.I$

Abstraktní systém ***S'*** je jistým modelem (matematickým) fyzického systému ***S***. Model umožňuje *předpovídat* chování reálného systému.

Definice systému



Obrázek: Model systému

u(t) . . . vstupní veličiny
y(t) . . . výstupní veličiny
x(t) . . . stavové veličiny
v(t) . . . poruchové veličiny

Obecná teorie systémů - OTS

OTS rozlišuje systémy podle úrovně detailu jejich popisu.

- Zdrojový systém: vyjmenovává veličiny a jejich iteragující podmnožinu - např. veličiny *U, I, R*, interakce {{*U, I, R*}} (Je daná množina veličin, definovaných (uvažovaných) na určité rozlišovací úrovni)
- Datový systém: zdrojový systém + empirické hodnoty - např.
  U: 25 V, 15 V, 10 V
  R: 5 kΩ, 5 kΩ, 5 kΩ
  I: 5 mA, 3 mA, 2 mA
  (Je dán souborem variací určitých veličin v čase)
- Generativní systém: veličiny + vztahy mezi nimi $\Rightarrow$ umožňuje generovat datový systém (Je časový vztah mezi minulými, současnými a budoucími hodnotami jistých veličin)
- Strukturní systém: je určrna struktura systému, jeho podsystémy a vazby mezi nimi

Každý typ systému nese informaci navíc oproti předchozímu typu.

Základní úlohy při práci se systémy

Obvyklý úkol v technické kybernetice je nalezení vhodného řízení, tj. působení na řízený objekt tak, aby bylo dosaženo požadovaných cílů.

- Stanovení cíle úlohy a výchozích (dostupných) informací
- Volba metody popisu systému a metody řešení dané úlohy (volba modelu systému)
- Definice proměnných, určení hodnot parametrů - identifikace
- Analýza vlastností systému (obvykle včetně linearizace)
- Syntéza (za účelem změny vlastností systému)

Námitka

K čemu speciální systémová věda?

Nestačí zkoumat pouze jednotlivé komponenty samostatně?

Ne. Systém je „více“ než souhrn jeho částí. Struktura vzájemných vazeb má obrovský vliv na výsledné chování.

Námitka

K čemu speciální systémová věda?

Nestačí zkoumat pouze jednotlivé komponenty samostatně?

Ne. Systém je „více“ než souhrn jeho částí. Struktura vzájemných vazeb má obrovský vliv na výsledné chování.

Příklady: Model neuronu

Umí počítat $\phi(\sum_{ij} w_{ij} x_{ij})$ - nelineární funkce váženého součtu vstupů

$\Downarrow$

Propojení velkého množství neuronů

$\Downarrow$

Neuronová síť (umělá) - umí se „učit“

Neuron

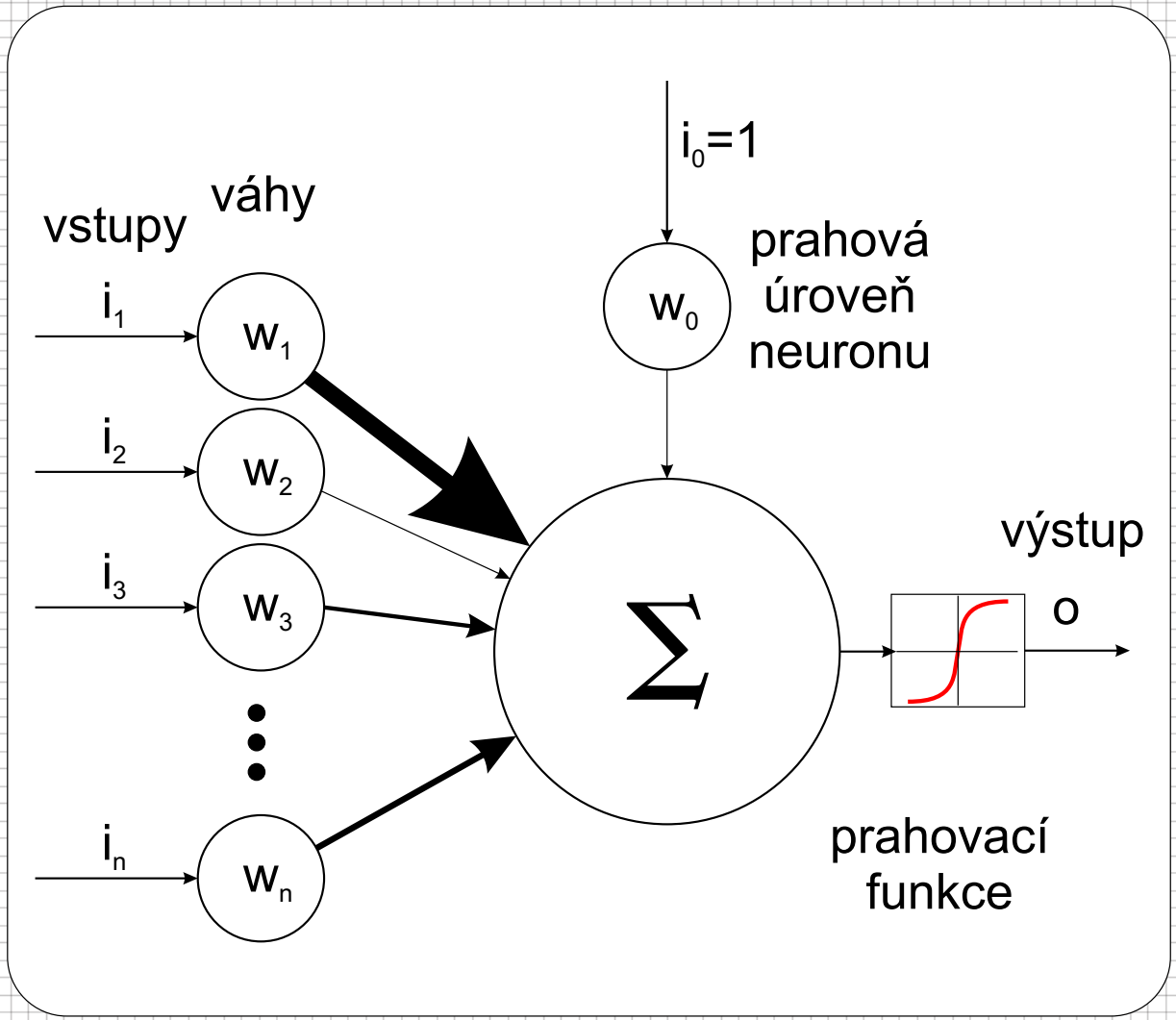
Myšlenka umělých neuronových sítí vznikla na podkladě výzkumu mozku, kde nejdříve pomocí umělých neuronových sítí se vědci snažili modelovat procesy, které probíhají v našem mozku. Posléze se neuronové sítě rozšířily do mnoha oborů technické praxe. Základním prvkem mozku je neuron, do něhož vstupuje velký počet vstupů-dendritů a vystupuje pouze jeden výstup - axon, který se ovšem může dále rozvětvovat do tzv. terminálů. Ty se připojují na dendrity pomocí synapsí. Na základě tohoto biologického popisu byl definován tzv. formální neuron, který tvoří výkonný prvek umělé neuronové sítě. Obecně neuron můžeme popsat podle tohoto vztahu:

Neuron

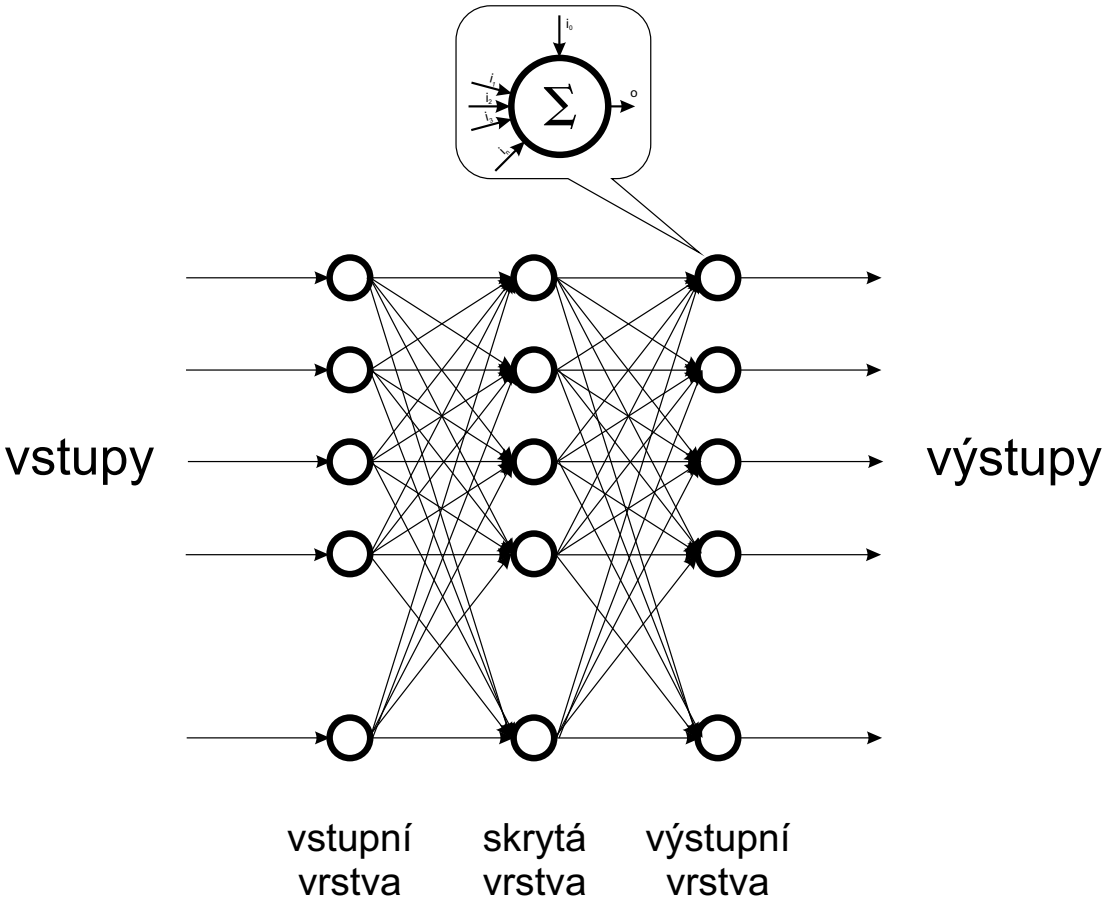
$$o = \phi(\sum_j w_j i_j + \theta)$$

o . . . výstup
i_j . . . vstupní veličiny (n)
w_j . . . váhy
θ . . . práh neuronu
ϕ . . . nelineární prahovací funkce

Neuron


$i_0=1$
vstupy
váhy
prahová úroveň neuronu
i_1
w_1
w_0
i_2
w_2
i_3
w_3
Σ
výstup
o
i_n
w_n
prahovací funkce

Neuronová síť



Dělení systémů

- Podle počtu vstupů a výstupů
  - SISO (Single Input Single Output)
  - MIMO (Multi Input Multi Output)

Dělení systémů

- Podle počtu vstupů a výstupů
  - SISO (Single Input Single Output)
  - MIMO (Multi Input Multi Output)
- Podle interakce s okolím
  - Uzavřené (volné) - neřízené (Nedochází k výměně energie s okolím)
  - Otevřené - řízené (Dochází k výměně energie s okolím)

Dělení systémů

- Podle počtu vstupů a výstupů
  - SISO (Single Input Single Output)
  - MIMO (Multi Input Multi Output)
- Podle interakce s okolím
  - Uzavřené (volné) - neřízené (Nedochází k výměně energie s okolím)
  - Otevřené - řízené (Dochází k výměně energie s okolím)
- Podle vztahu jednotlivých veličin
  - Lineární
  - Nelineární

Dělení systémů

- Podle počtu vstupů a výstupů
  - SISO (Single Input Single Output)
  - MIMO (Multi Input Multi Output)
- Podle interakce s okolím
  - Uzavřené (volné) - neřízené (Nedochází k výměně energie s okolím)
  - Otevřené - řízené (Dochází k výměně energie s okolím)
- Podle vztahu jednotlivých veličin
  - Lineární
  - Nelineární
- Podle typu veličin
  - Spojité
  - Diskrétní (v amplitudě, čase)
  - Kvazispojité (pseudospojité, hybridní) např. PWM

Dělení systémů

Dělení systémů

- Podle přítomnosti paměti
  - Statické (kombinační) - nemají paměť $\Rightarrow$ výstup závisí pouze na vstupu
  - Dynamické (sekvenční) - mají paměť $\Rightarrow$ výstup závisí na vstupu a stavu (předchozích hodnotách)
- Podle charakteru veličin
  - Deterministické
  - Stochastické (pravděpodobnostní)

Dělení systémů

- Podle přítomnosti paměti
  - Statické (kombinační) - nemají paměť $\Rightarrow$ výstup závisí pouze na vstupu
  - Dynamické (sekvenční) - mají paměť $\Rightarrow$ výstup závisí na vstupu a stavu (předchozích hodnotách)
- Podle charakteru veličin
  - Deterministické
  - Stochastické (pravděpodobnostní)
- Podle chování v čase
  - Časově invariantní (nemění se parametry v čase)
  - Časově proměnné (mění se parametry v čase např.: opotřebení)

Základní úkol

Obvyklý úkol v technické kybernetice je nalezení vhodného řízení, tj. působení na řízený objekt tak, aby bylo dosaženo požadovaných cílů.

Postup

- Rozbor úlohy a výchozích (dostupných) informací
- Volba metody popisu systému a metody řešení dané úlohy (volba modelu systému)
- Definice proměnných, určení hodnot parametrů - identifikace
- Analýza vlastností systému (obvykle včetně linearizace)
- Syntéza (za účelem změny chování systému)

Dynamický systém

Je definován množinami a zobrazeními:
Jsou dány množiny:

Dynamický systém

Je definován množinami a zobrazeními:
Jsou dány množiny:

- množina časových okamžiků $t \in T$

Dynamický systém

Je definován množinami a zobrazeními:
Jsou dány množiny:

- množina časových okamžiků $t \in T$
- množina stavů $x(t) \in X$

Dynamický systém

Je definován množinami a zobrazeními:
Jsou dány množiny:

- množina časových okamžiků $t \in T$
- množina stavů $x(t) \in X$
- množina okamžitých hodnot vstupních funkcí $u(t) \in U$

Dynamický systém

Je definován množinami a zobrazeními:
Jsou dány množiny:

- množina časových okamžiků $t \in T$
- množina stavů $x(t) \in X$
- množina okamžitých hodnot vstupních funkcí $u(t) \in U$
- množina přípustných vstupních funkcí (signálů) $\mathcal{U} = u(t) : T \to U$

Popis systémů

- Vnější popis - vyjdřuje vztah mezi vnějšími veličinami [y(t);u(t)]
  - Diferenciální rovnice
  - Přenos v Laplaceově transformaci
  - Nuly a póly systému
  - Impulzní charakteristika
  - Přechodová charakteristika
  - Frekvenční přenos
  - Frekvenční charakteristiky
  - Odezva na libovolný signál
- Vnitřní popis
  - Stavové rovnice

Diferenciální rovnice

Relaci mezi vstupem a výstupem hladkého spojitého systému můžeme obecně vyjádřit ve tvaru

$$F(y, \dot{y}, ..., y^{(n)}, u, \dot{u}, ..., u^{(m)}) = 0.$$

Pro lineární systém lze psát

$$a_n(t)y^{(n)} + ... + a_0(t)y(t) = b_m(t)u^{(m)} + ... + b_0(t)u(t).$$

Pro stacionární systém jsou koeficienty a_i a b_j = konst.
Je-li $n \geq m$ nazýváme systém ryzí nebo fyzikálně realizovatelný.

Diferenciální rovnice LTI systému

LTI . . . lineátní stacionární (t-invariantní) systém

$$a_n \frac{d^n y(t)}{dt^n} + a_{(n-1)} \frac{d^{(n-1)} y(t)}{dt^{(n-1)}} + \cdots + a_1 \frac{dy(t)}{dt} + a_0 y(t)$$
$$= b_m \frac{d^m u(t)}{dt^m} + b_{(m-1)} \frac{d^{(m-1)} u(t)}{dt^{(m-1)}} + \cdots + b_1 \frac{du(t)}{dt} + b_0 u(t)$$

Přenos v Laplaceově transformaci

Je-li systém stacionární a předpokládáme-li nulové počáteční podmínky, lze provést Laplaceovu transformaci diferenciální rovnice systému. Potom L-obraz výstupu ku L-obrazu vstupu při nulových počátečních podmínkách nazýváme přenos systému.

$$G(s) = \frac{\mathscr{L}\{u_2(t)\}}{\mathscr{L}\{u_1(t)\}} = \frac{\mathscr{L}\{y(t)\}}{\mathscr{L}\{u(t)\}} = \frac{Y(s)}{U(s)} = \frac{b_m s^{(m)} + \ldots + b_1 s + b_0}{a_n s^{(n)} + \ldots + a_1 s + a_0}$$

Póly a nuly systému

$$G(s) = \frac{b(s)}{a(s)}$$

$$a(s) = a_n s^{(n)} + ... + a_1 s + a_0 = a_n(s - p_n)(s - p_{n-1})...(s - p_1)$$

$$b(s) = b_m s^{(m)} + ... + b_1 s + b_0 = b_m(s - n_m)(s - n_{m-1})...(s - n_1)$$

Kořeny p_i polynomu $a(s)$ se nazývají póly přenosu systému a kořeny n_i polynomu $b(s)$ se nazývají nuly přenosu systému.

Impulsní charakteristika - g(t)

Jedná se o odezvu systému na Diracův impuls $u(t) = \delta(t)$. Protože

$$\mathscr{L}\{\delta(t)\} = 1$$

a

$$Y(s) = G(s)U(s) = G(s)$$

tedy

$$g(t) = \mathscr{L}^{-1}\{G(s)\}$$

Přechodová charakteristika - h(t)

Jedná se o odezvu systému na jednotkový skok $u(t) = \mathbb{1}(t)$. Protože

$$\mathscr{L}\{\mathbb{1}(t)\} = \frac{1}{s}$$

a

$$H(s) = G(s)\frac{1}{s}$$

tedy

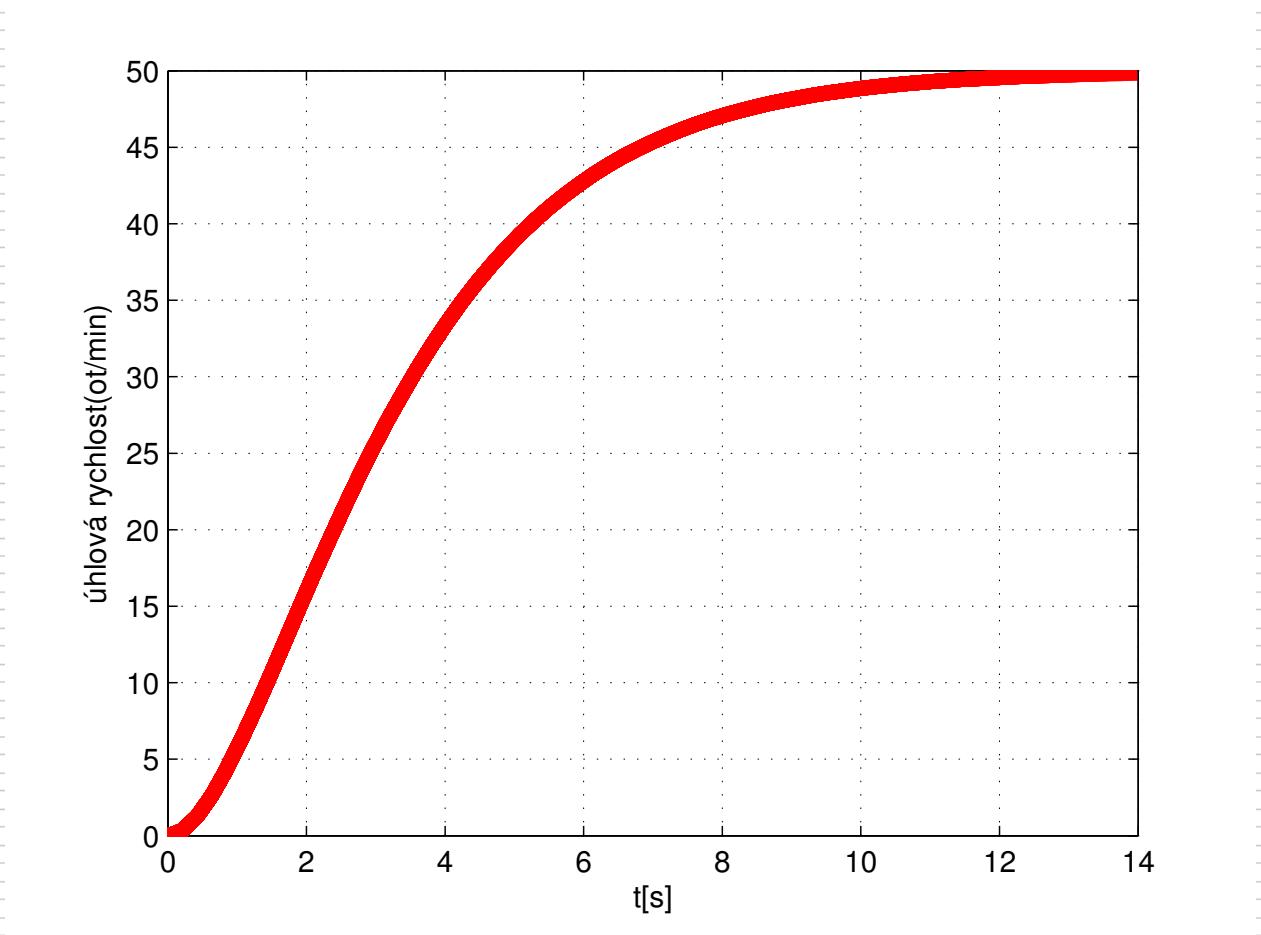
$$h(t) = \mathscr{L}^{-1}\{G(s)\frac{1}{s}\}$$

Popis systémů

Příklad: Zapneme motor se jmenovitými otáčkami 50 ot/min.

Náběh otáček v čase

Popis systémů



Frekvenční přenos

Jedná se o ustálenou odezvu lineárního stacionárního systému na harmonický vstupní signál.
Nebo Fourierův obraz výstupní veličiny ku Fourierovu obrazu vstupní veličiny.
Lze ho také získat formálně z Laplaceova přenosu v bodě $s = j\omega$.

$$F(j\omega) = G(s)|_{s=j\omega} = \frac{b_m(j\omega)^{(m)} + \ldots + b_1(j\omega) + b_0}{a_n(j\omega)^{(n)} + \ldots + a_1(j\omega) + a_0}$$

Jsou-li obě funkce $u(t)$ a $y(t)$ absolutně integrovatelné, existují jejich Fourierovy obrazy $U(j\omega)$ a $Y(j\omega)$ a potom platí

$$F(j\omega) = \frac{Y(j\omega)}{U(j\omega)}$$

Frekvenční charakteristiky

- je buď grafické zobrazení frekvenčního přenosu v komplexní rovině (Nyquistova charakteristika).

$$F(j\omega) = \Re F(\omega) + j\Im F(\omega)$$

Nebo v polárních souřadnicích

$$F(j\omega) = |F(j\omega)|e^{j \arg F(j\omega)}$$

Nejčastěji se používají v logaritmických souřadnicích $\log|F(j\omega)|$ a $\arg|F(j\omega)|$ na $\log(\omega)$ (Bodeho charakteristika), kde je možné amplitudovou a fázovou charakteristiku dobře aproximovat přímkovými asymptotami.

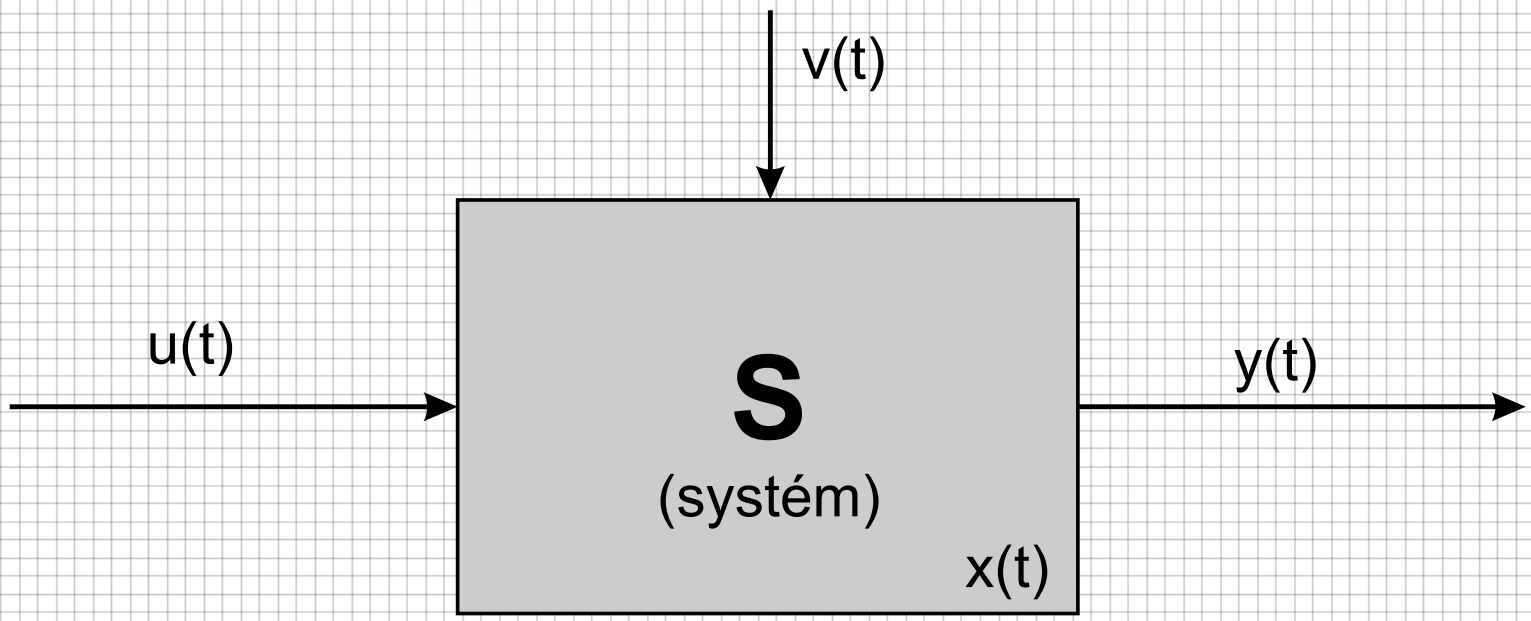
Vnitřní popis

Vyjadřuje relaci {vstup $\longrightarrow$ stav $\longrightarrow$ výstup}, popsanou přechodovou funkcí stavu $\varphi(x, u, t, \tau)$ a výstupní funkcí $g(x, u, t)$.
Pro LTI systém

$$\dot{x} = \mathbb{A}x + \mathbb{B}u$$

$$y = \mathbb{C}x + \mathbb{D}u$$

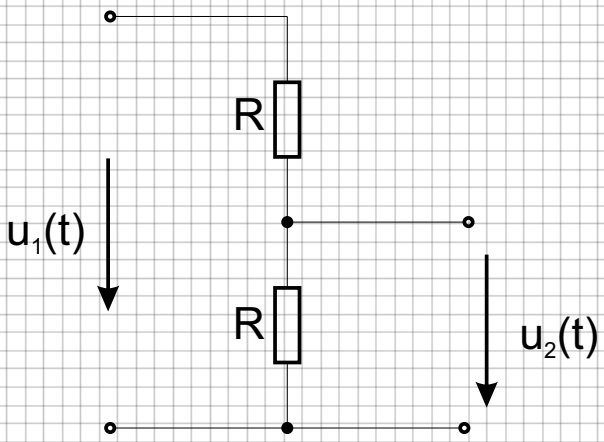
Definice systému

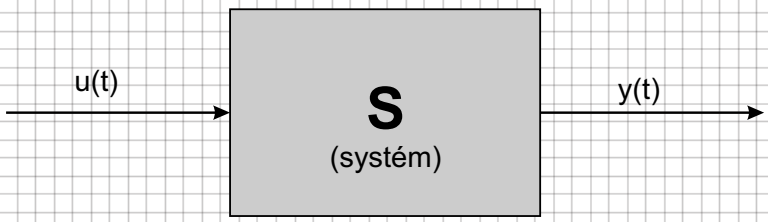


Obrázek: Model systému

u(t) . . . vstupní veličiny
y(t) . . . výstupní veličiny
x(t) . . . stavové veličiny
v(t) . . . poruchové veličiny

Příklad: Systém 0. řádu





$$u_2(t) = \frac{R}{R+R} u_1(t) = \frac{1}{2} u_1(t)$$

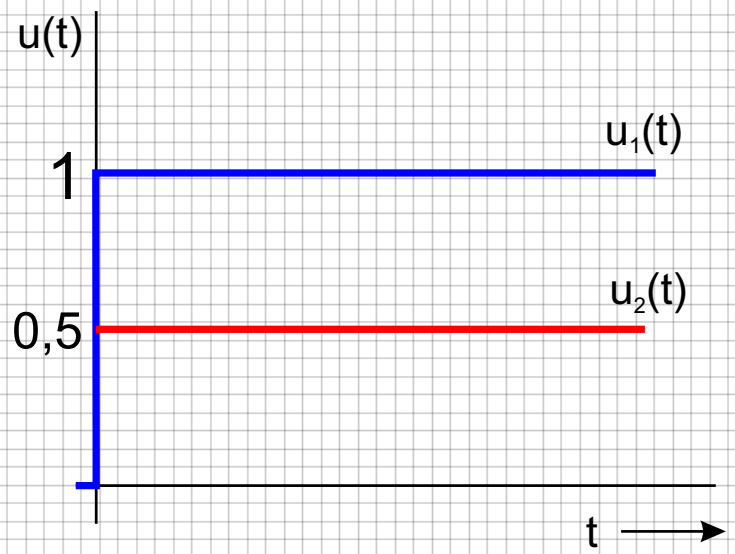
Přenos:

$$G(p) = \frac{\mathscr{L}\{u_2(t)\}}{\mathscr{L}\{u_1(t)\}} = \frac{U_2(p)}{U_1(p)} = 0,5$$

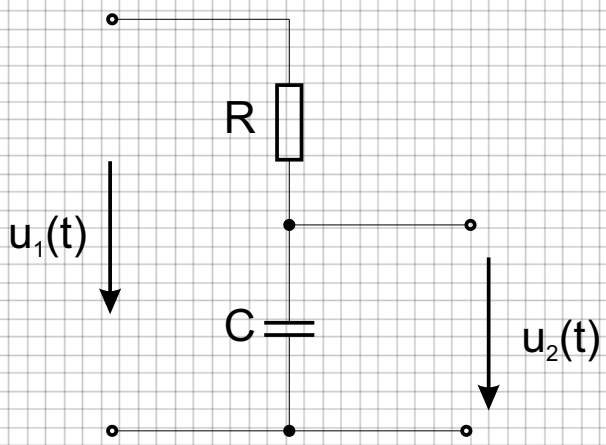
Příklad: Systém 0. řádu

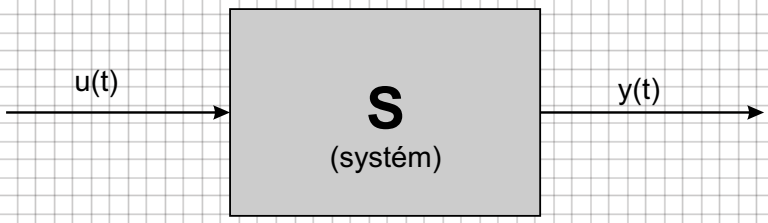
Pro jednotkový skok:

$$Y(p) = G(p)U(p) = 0,5\frac{1}{p}$$



Příklad: Systém 1. řádu





$$u_2(t) = u_C(t) = \frac{1}{C}\int i_C(t)dt \Rightarrow U_2(p) = U_C(p) = \frac{1}{pC}I(p)$$

$$U_2(p) = \frac{Z_C}{R + Z_C}U_1(p) = \frac{\frac{1}{pC}}{R + \frac{1}{pC}}U_1(p) = \frac{1}{1 + pRC}U_1(p)$$

Přenos:

$$G(p) = \frac{\mathscr{L}\{u_2(t)\}}{\mathscr{L}\{u_1(t)\}} = \frac{U_2(p)}{U_1(p)} = \frac{1}{1 + pRC}$$

Příklad: Systém 1. řádu

Pól přenosu:

$$1 + pRC = 0 \Rightarrow p_1 = -\frac{1}{RC}$$

Časová konstanta:

$$\tau_1 = -\frac{1}{p_1} = RC$$

Pro jednotkový skok:

$$Y(p) = G(p)U(p) = \frac{1}{(1+pRC)}\frac{1}{p} = \frac{1}{p(1+pRC)} = U_2(p)$$

Odezva v časové oblasti:

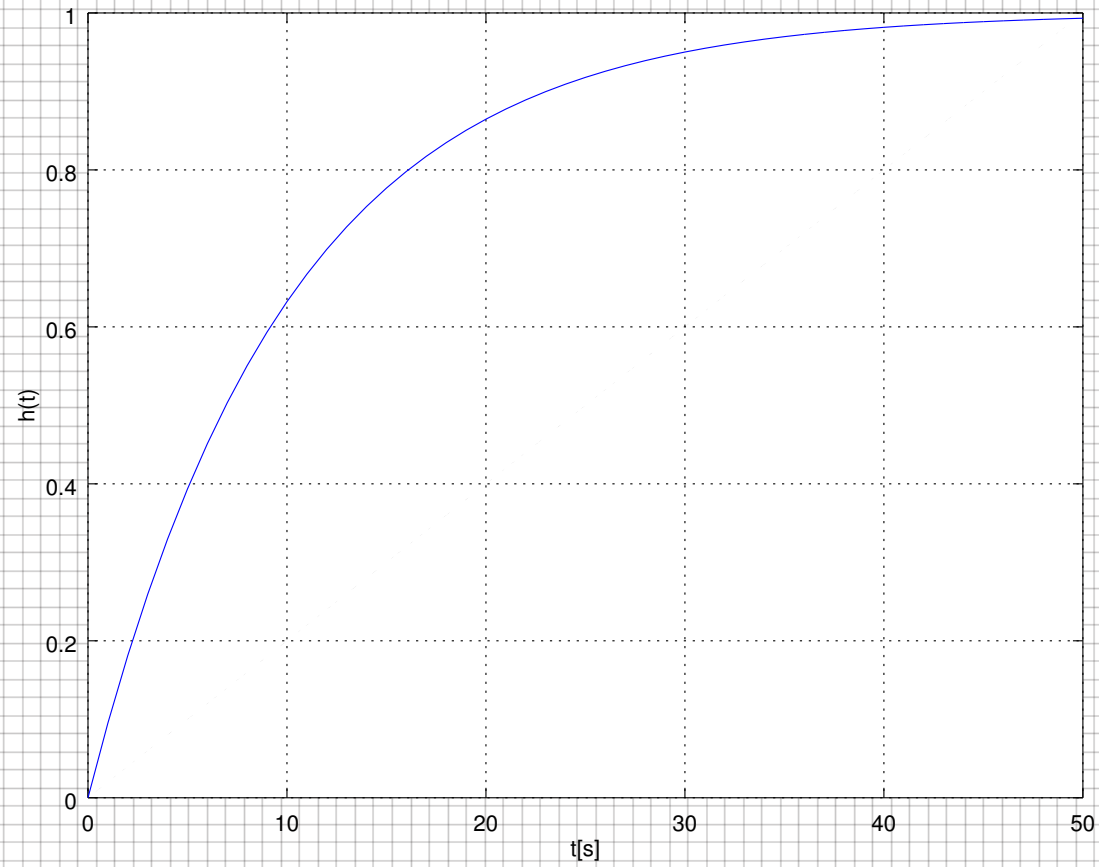
$$u_2(t) = (1 - e^{-\tau t})u_1(t)$$

Odezva

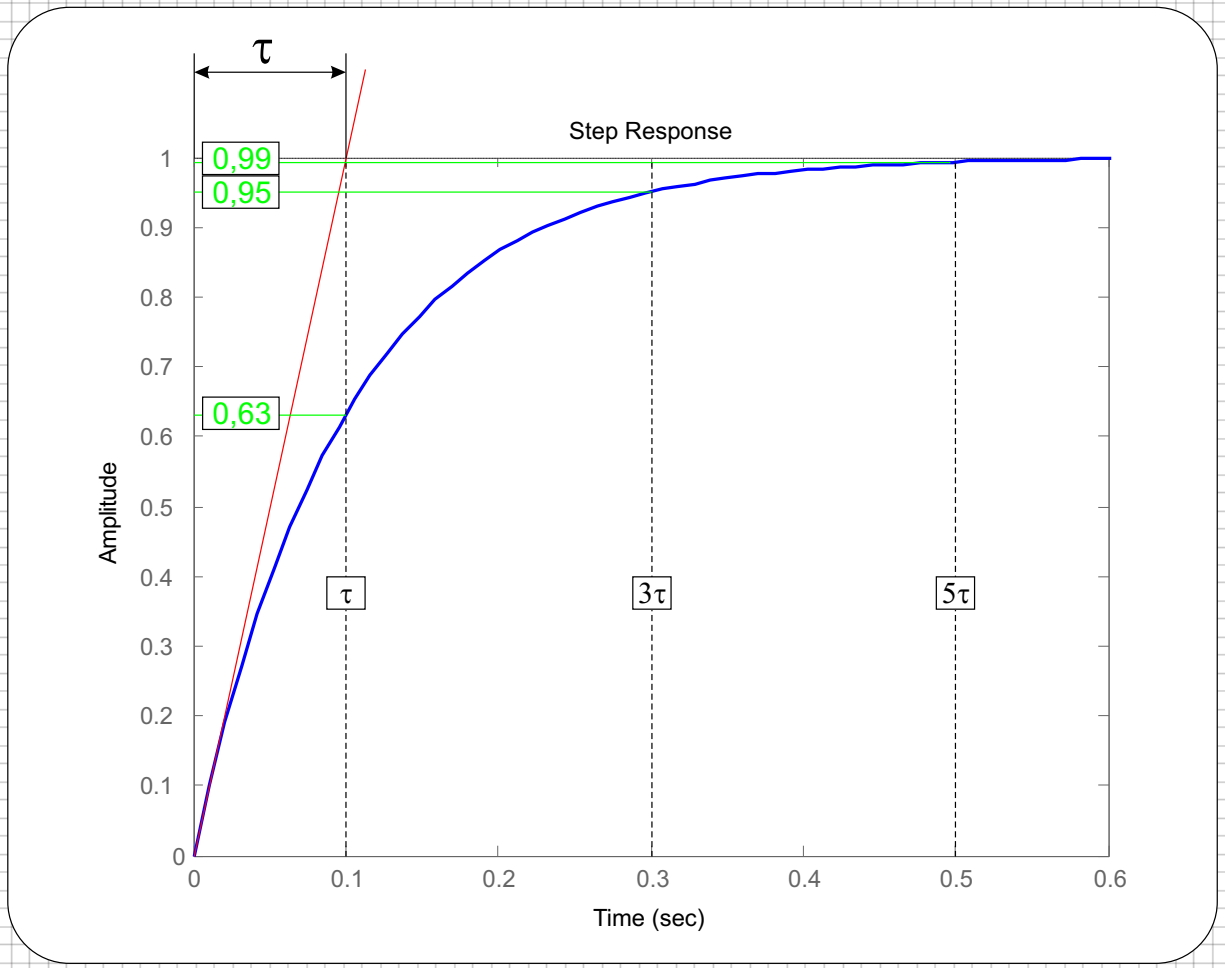
$$G(p) = \frac{\mathcal{L}\{u_2(t)\}}{\mathcal{L}\{u_1(t)\}} = \frac{U_2(p)}{U_1(p)} = \frac{1}{1+10p}$$

Matlab:

```
G=tf(1,[10 1]);
[h,time]=step(G);
plot(time,h)
grid on
xlabel('t[s]')
ylabel('h(t)')
```

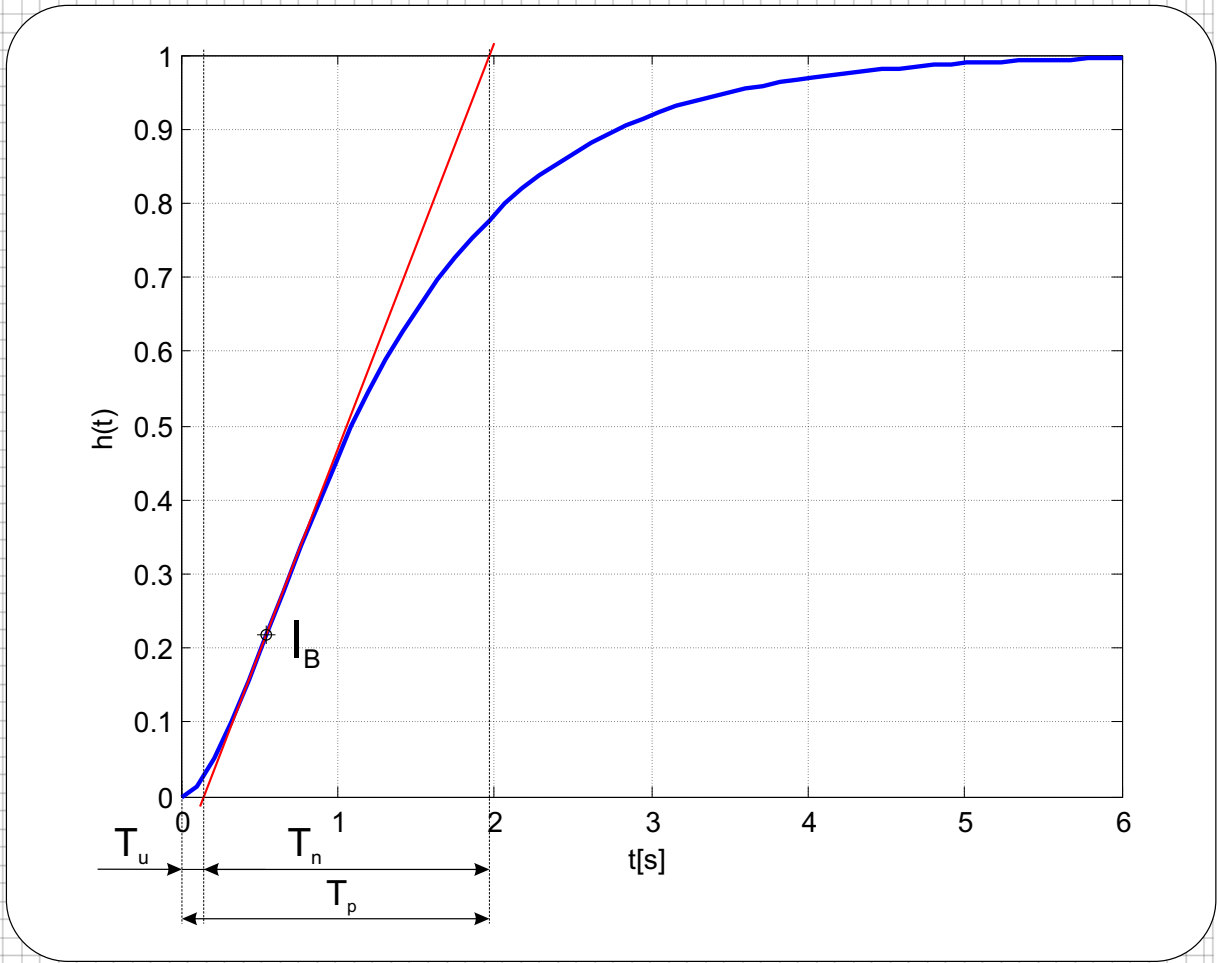


Odezva


τ
Step Response
0,99
0,95
0,63
τ
3τ
5τ
Amplitude
Time (sec)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.1
0.2
0.3
0.4
0.5
0.6

Odezva

T_n doba náběhu
T_u doba průtahu
T_p doba přechodu
I_B inflexní bod



Řízení

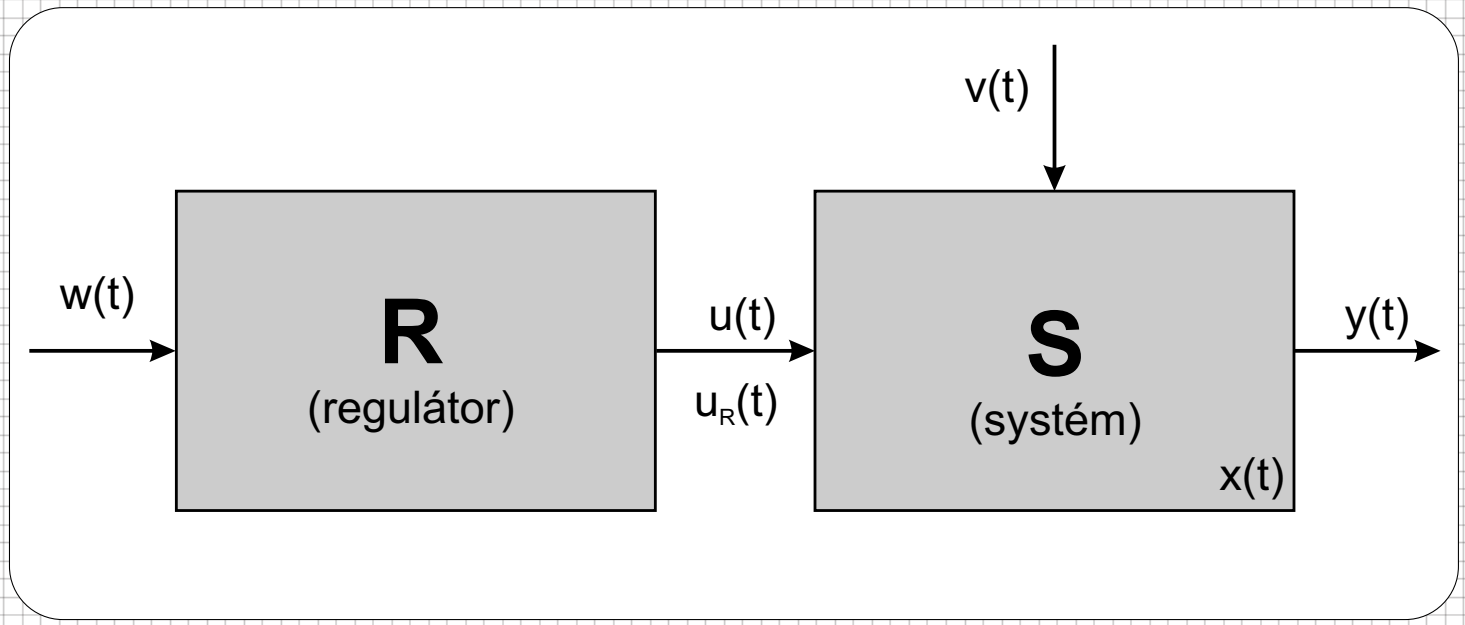
Úvahy k zamyšlení

Energie potřebná na udržení systému mimo rovnovážný stav často roste se vzdáleností od rovnovážného stavu.

(příkladem budiž Sisyfos)

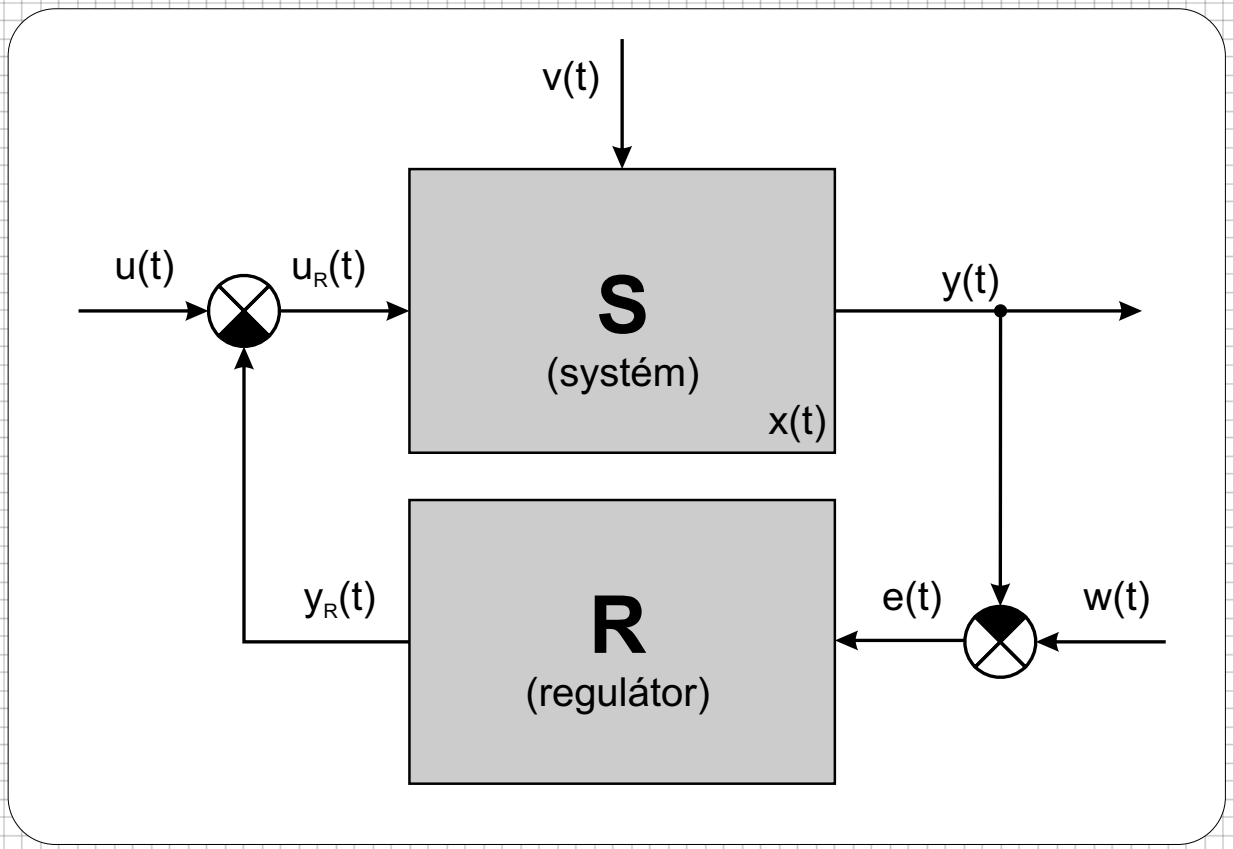
Změna stavu systému ve společnosti - musí se dělat pozvolně, jinak hrozí překmit "na druhou stranu".

Regulační obvod



Obrázek: Blokové schéma ovládání

Regulační obvod

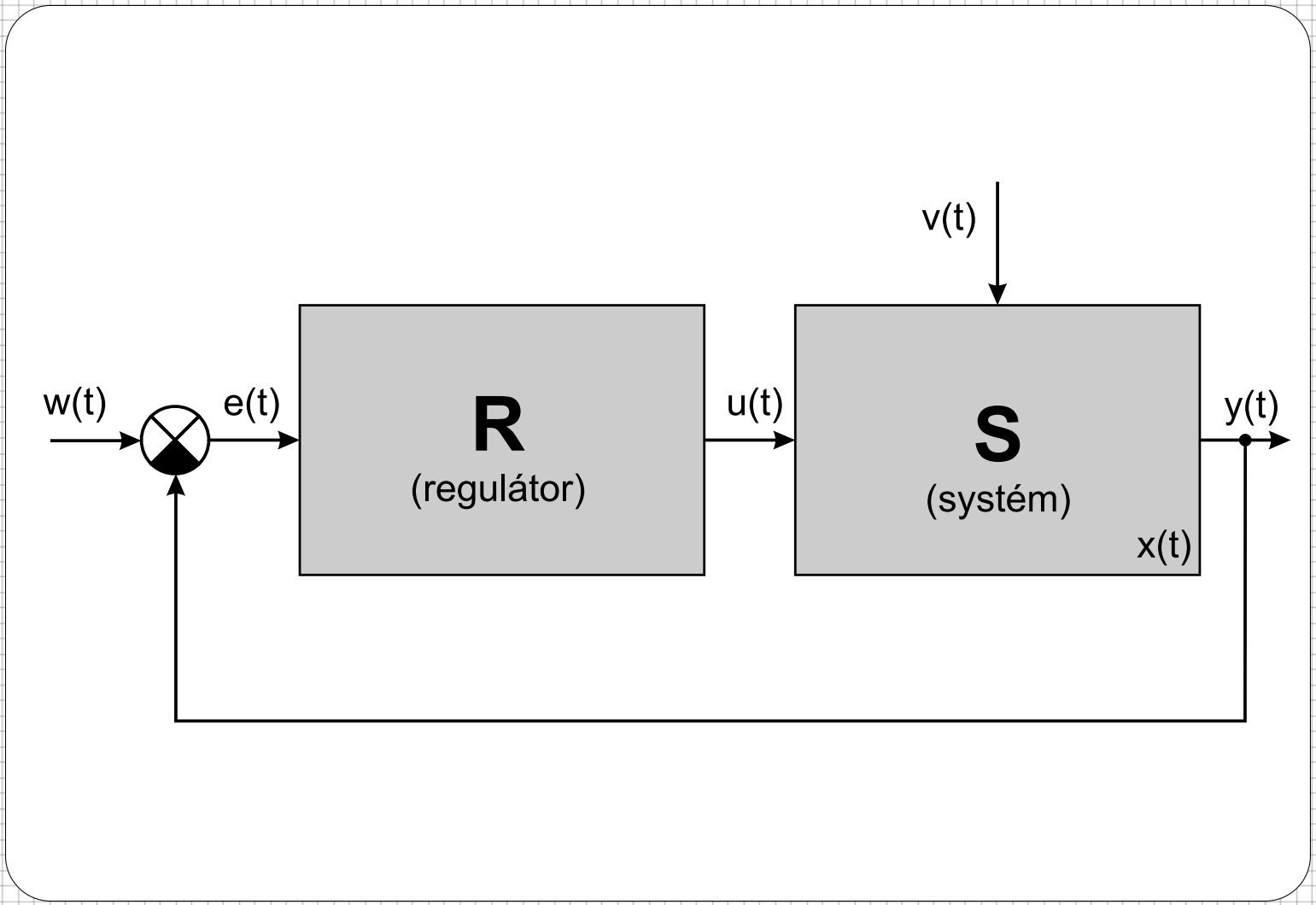


$$e(t) = w(t) - y(t)$$

$$u_R(t) = u(t) - y_R(t)$$

Obrázek: Blokové schéma regulace

Regulační obvod - jiná varianta



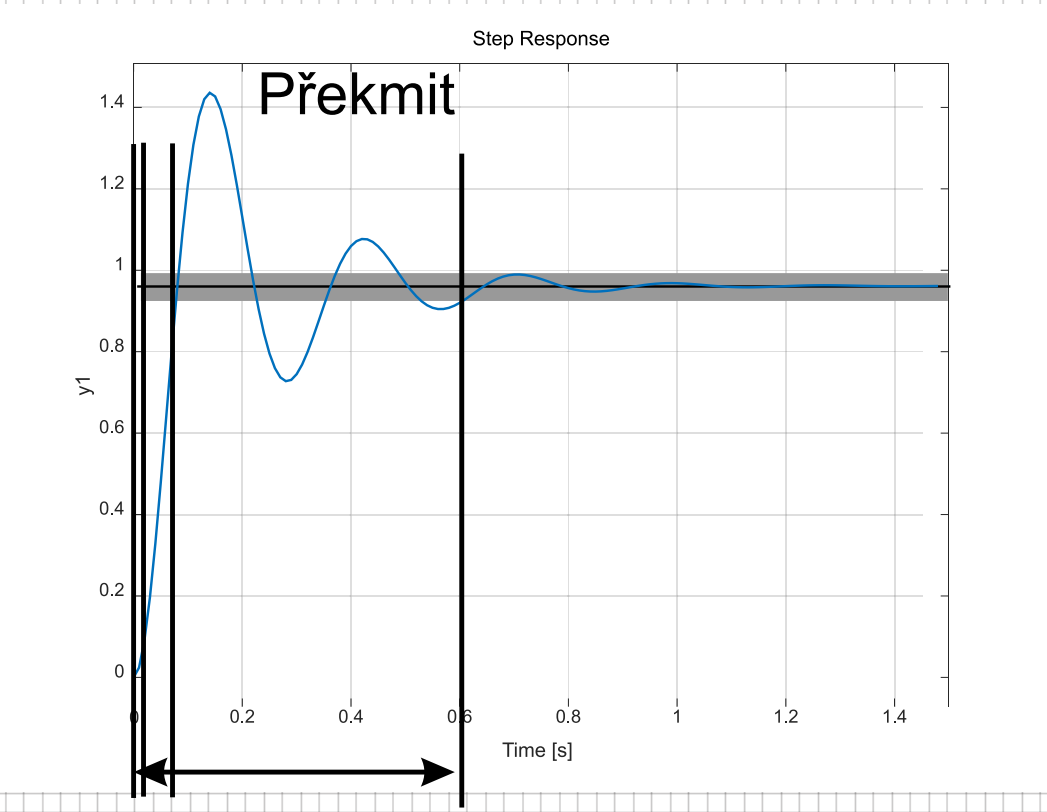
Obrázek: Blokové schéma regulace

Vsuvka: slovník pojmů AN - CZ

OL - open loop	OS - otevřená smyčka
CL - close loop	US - uzavřená smyčka
CL RESPONSE	CL odezva
RISE TIME	Doba náběhu
OVERSHOOT	Překmit
SETTLING TIME	Doba ustálení
S-S ERROR	Odchylka ustáleného stavu

Přechodová charakteristika

Doba náběhu

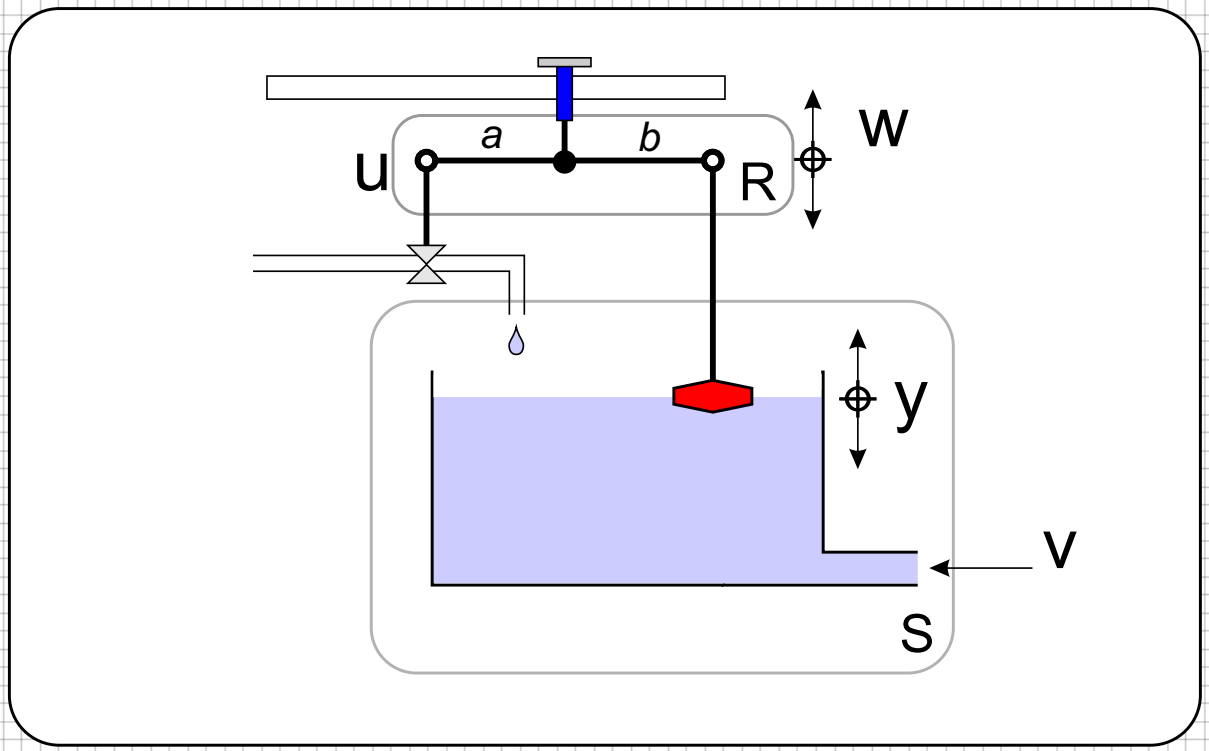


Pásmo necitlivosti
Error band

Doba ustálení

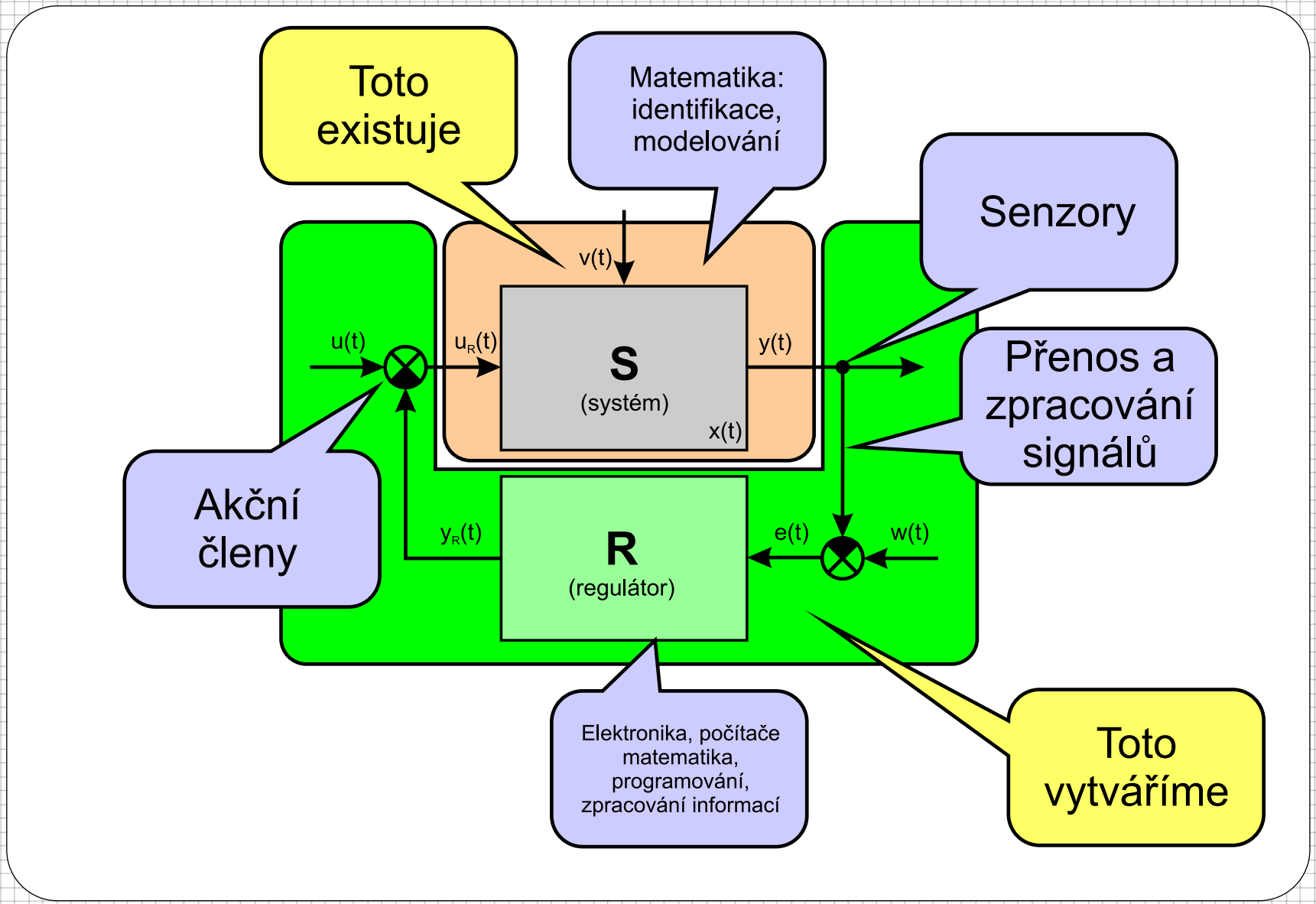
Obrázek: Význam některých pojmů

Příklad regulačního obvodu



Obrázek: Splachovadlo

Rozbor problematiky řídicí úlohy



Popis problému

$A(t)$ ... velikost populace kořisti v čase t
$B(t)$ ... velikost populace dravců v čase t
Pokud nejsou predátoři předpokládejme, že prostředí umožňuje exponenciální růst populace kořisti (králíci Fibbonaci, Austrálie) tzn.: $\frac{dA}{dt} = rA(t)$.
Pokud není kořist předpokládejme, že populace dravců klesá také exponenciálně tzn.: $\frac{dB}{dt} = -\delta B(t)$.
Interakce mezi predátorem a kořistí je dána vztahem AB a koeficientem „úspěšnosti“, který na populaci predátorů má pozitivní vliv a označíme si ho β a na populaci kořisti má vliv negativní a označíme si ho jako α.

Lotka-Volterra rovnice

$$\frac{dA}{dt} = rA(t) - \alpha AB$$

$$\frac{dB}{dt} = -\delta B(t) + \beta AB$$

Diferenciální rovnice

$$\frac{dA}{dt} = rA(t)$$

odpovídá

$$\frac{dx}{dt} = 2tx(t)$$

například při počáteční podmínce:

$$x(0) = 1$$

Zápis diferenciální rovnice v OCTAVE

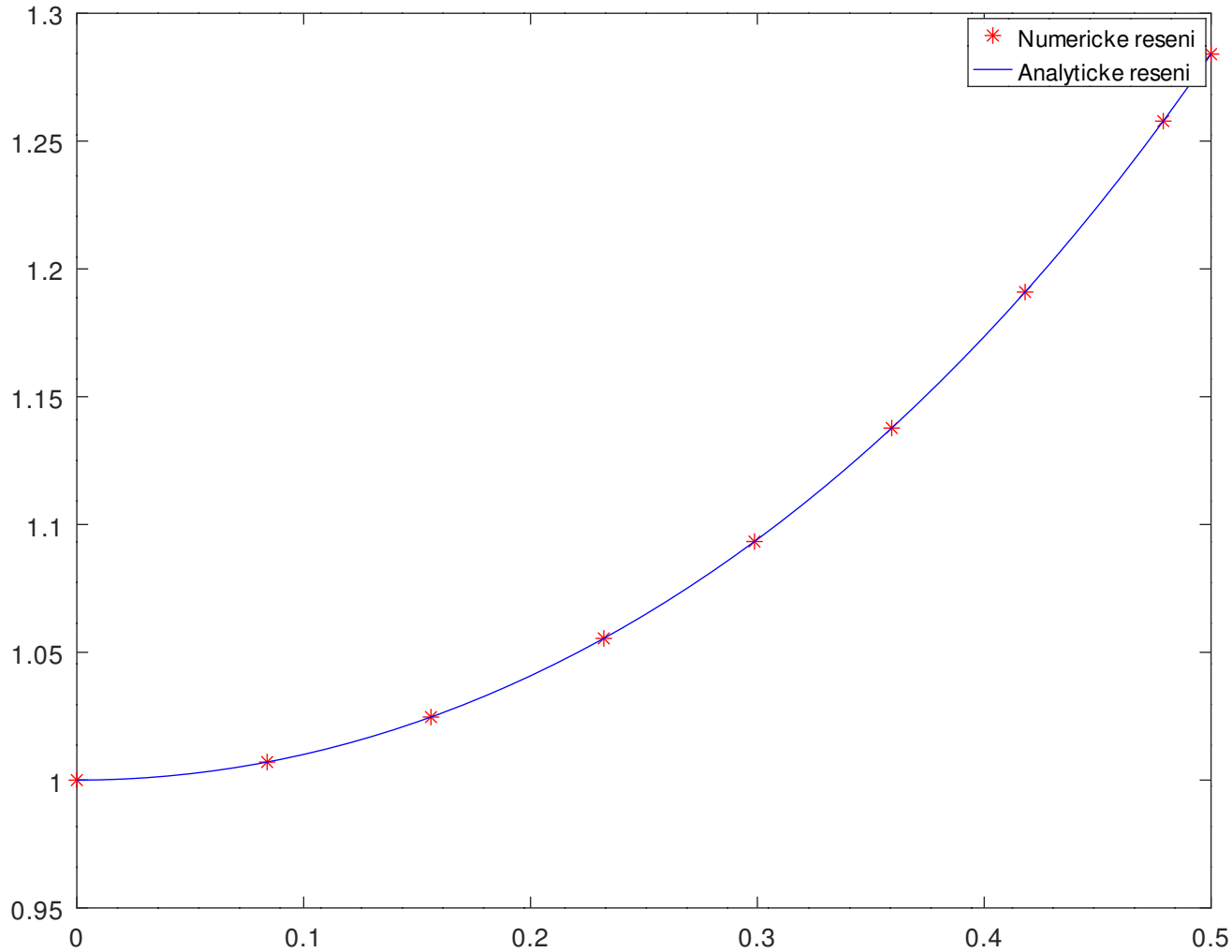
Soubor `dfdt.m`

```
1 function deriv_val = dfdt(t,x)
2 %  Diferenciální rovnice
3 %      dx/dt = 2*t*x;
4 deriv_val = 2*t*x;
```

Řešení jednoduché dif. rovnice

```
1  %  Jednoduchý příklad řešení v OCTAVE
2  %    dx/dt = 2*x*t,  x(0) = 1
3  %  analytické řešení x(t) = exp(t^2).
4
5  %  Řešitel dif. rovnic ode45 vygeneruje vektor t a x hodnot:
6  %
7  %  [t,x]=ode45(@function_file_name, [start_time, end_time], initial_value)
8  %  pozn.: "@" Musí být pře názvem souboru řešené funkce (a bez ".m")
9  options=odeset('AbsTol',1.e-12,'RelTol',1.e-9,'InitialStep',2,'MaxStep',2);
10 [t,x] = ode45(@dfdt,[0,0.5],1,options);
11 %  graf numerické aproximace (červené body '*'):
12 plot(t,x,'r*')
13 hold on %  fixace obrázku
14 %  Analytické řešení - výpočet
15 tvals = [0:0.01:0.5];
16 true_solution = exp(tvals.^2);
17 %  graf anal. řešení modrý
18 plot(tvals,true_solution,'b');
19 %  Legenda
20 legend('Numerické řešení','Analytické řešení')
```

Porovnání analytického a numerického řešení dif. rovnice



Lotka-Volterra rovnice

$$\frac{dA}{dt} = rA(t) - \alpha AB$$

$$\frac{dB}{dt} = -\delta B(t) + \beta AB$$

možno upravit do vektorového tvaru:

$$\mathbf{x} = \begin{bmatrix} A \\ B \end{bmatrix} \qquad \text{nebo} \qquad \mathbf{x}(t) = \begin{bmatrix} A(t) \\ B(t) \end{bmatrix}$$

Lotka-Volterra rovnice

$$\begin{bmatrix} \frac{dA}{dt} \\ \frac{dB}{dt} \end{bmatrix} = \begin{bmatrix} rA(t) - \alpha AB \\ -\delta B(t) + \beta AB \end{bmatrix}$$

nebo:

$$\frac{d\mathbf{x}}{dt} = \begin{bmatrix} rx_1(t) - \alpha x_1 x_2 \\ -\delta x_2(t) + \beta x_1 x_2 \end{bmatrix}$$

Zvolené podmínky

$$\frac{dA}{dt} = rA(t) - \alpha AB$$

$$\frac{dB}{dt} = -\delta B(t) + \beta AB$$

koeficienty:

$$r = 10 \qquad \alpha = 1.2$$

$$\delta = 4 \qquad \beta = 0.9$$

počáteční podmínky:

$$A(0) = 4 \qquad B(0) = 0.5$$

Definice diferenciální rovnice D-K v jazyce Matlab (OCTAVE)

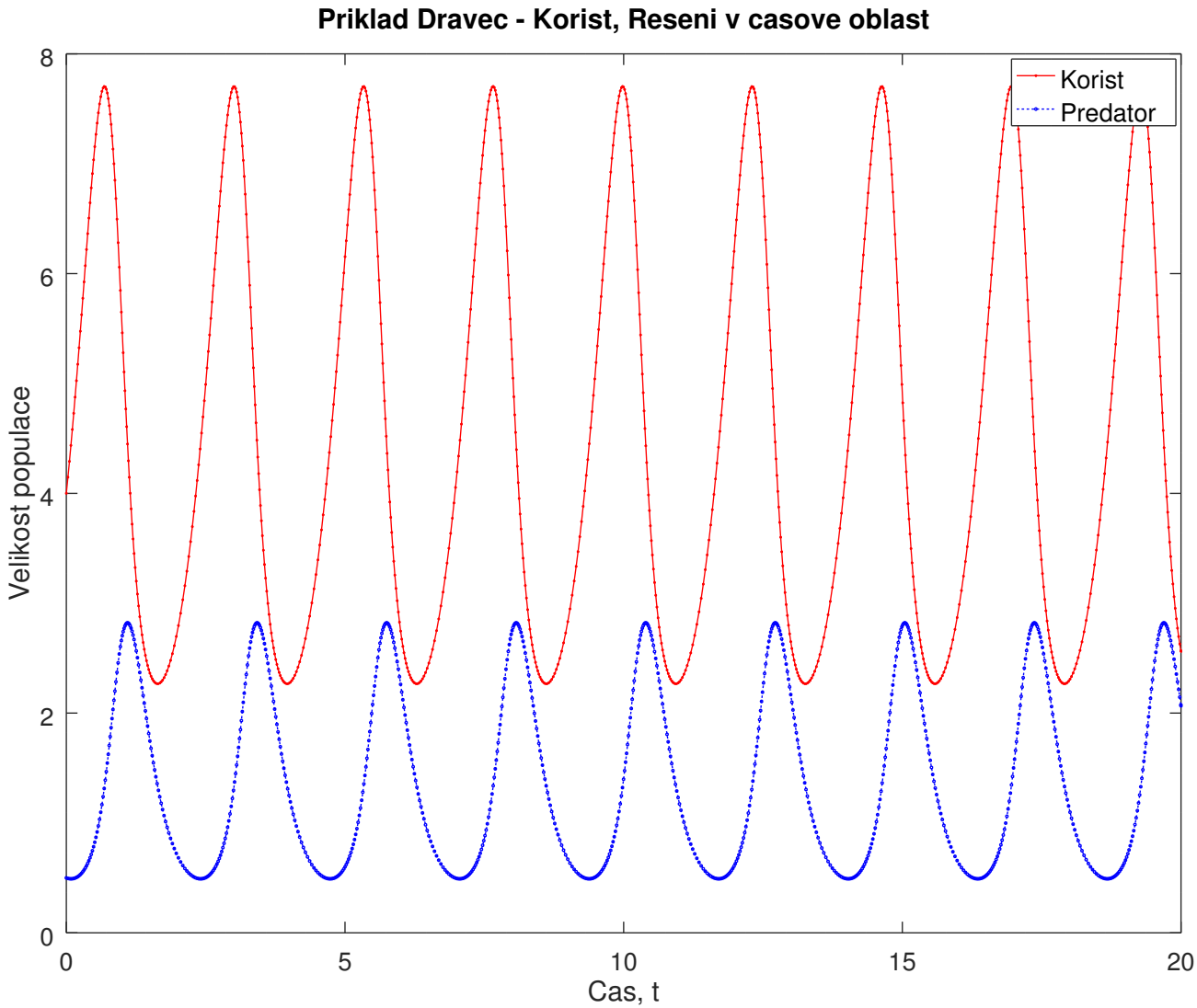
```
function deriv_vals = pred_prey_odes(t,x)
%  Vypocet dif. rovnice:
%      dA/dt = myA A + myB A*B
%      dB/dt = myC B + myD A*B
myA=2;
myB=-1.5;
myC=-4;
myD=0.9;

deriv_vals = zeros(size(x));
%  Calculate dA/dt, the first value in the deriv_vals vector.  Remember A =
%  x(1) and B = x(2).
deriv_vals(1) = myA*x(1) + myB*x(1).*x(2);
%  Calculate dB/dt, the second value in the deriv_vals vector.  Remember A =
%  x(1) and B = x(2).
deriv_vals(2) = myC*x(2) + myD*x(1).*x(2);
```

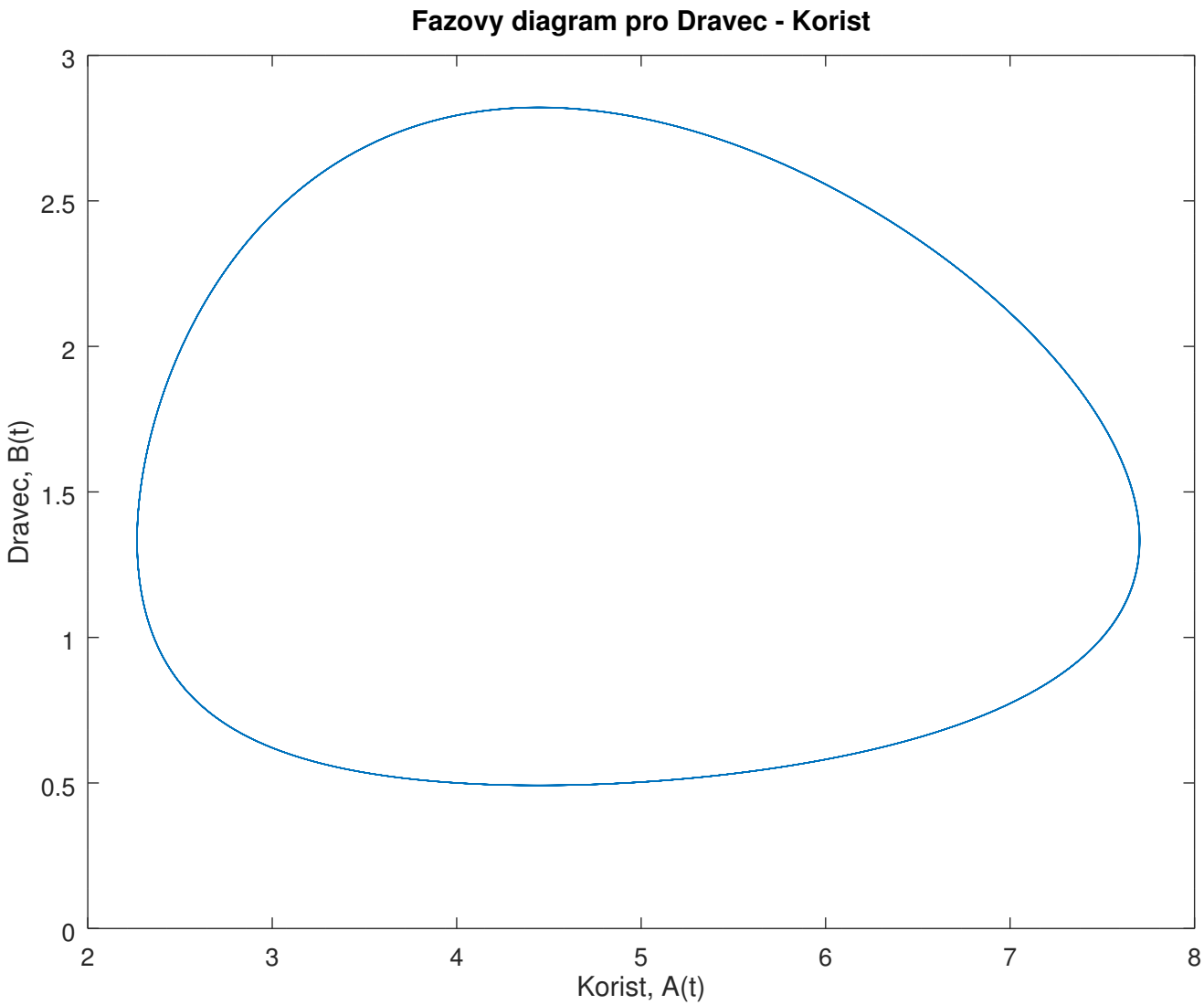
Definice diferenciální rovnice D-K v jazyce Matlab (OCTAVE)

```
%  Define initial and final times
t0 = 0;
tf = 20;
%  Define initial values vector with A(0) = 4, and B(0) = 0.5
init_vals = [4; 0.5];
%  Use ode45
options=odeset('AbsTol',1.e-12,'RelTol',1.e-9,'InitialStep',2,'MaxStep',2);
[t,x] = ode45(@pred_prey_odes,[t0,tf],init_vals,options);
%  Peel off our A(t) and B(t) values for easier reading and plotting:
A = x(:,1);  %  A is the first row of the x matrix, collecing all columns.
B = x(:,2);  %  B is the second row of the x matrix, collecting all columns
%  Plot A(t) values in red with *'s, B(t) values in blue with circles.
plot(t,A,'r*-',"markersize", 1,t,B,'bo:',"markersize", 1)
xlabel('Cas, t')
ylabel('Velikost populace')
title('Priklad Dravec - Korist, Reseni v casove oblast')
legend('Korist','Predator')
%  Plot the phase plane, A-values versus B-values
figure %  This opens a new window for an additional plot
plot(A,B)
xlabel('Korist, A(t)')
ylabel('Dravec, B(t)')
title('Fazovy diagram pro Dravec - Korist')
```

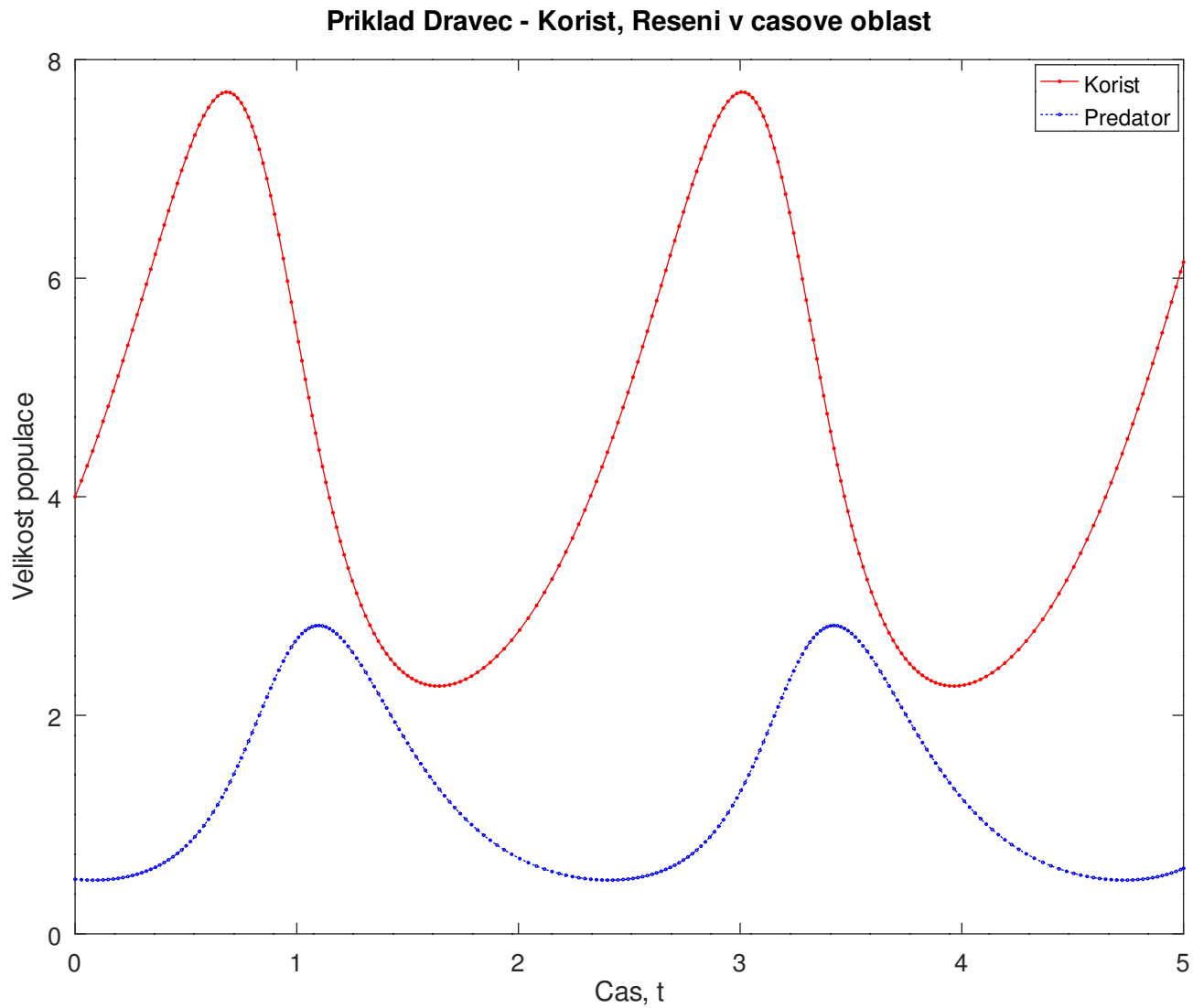
Grafické výstupy - časová oblast



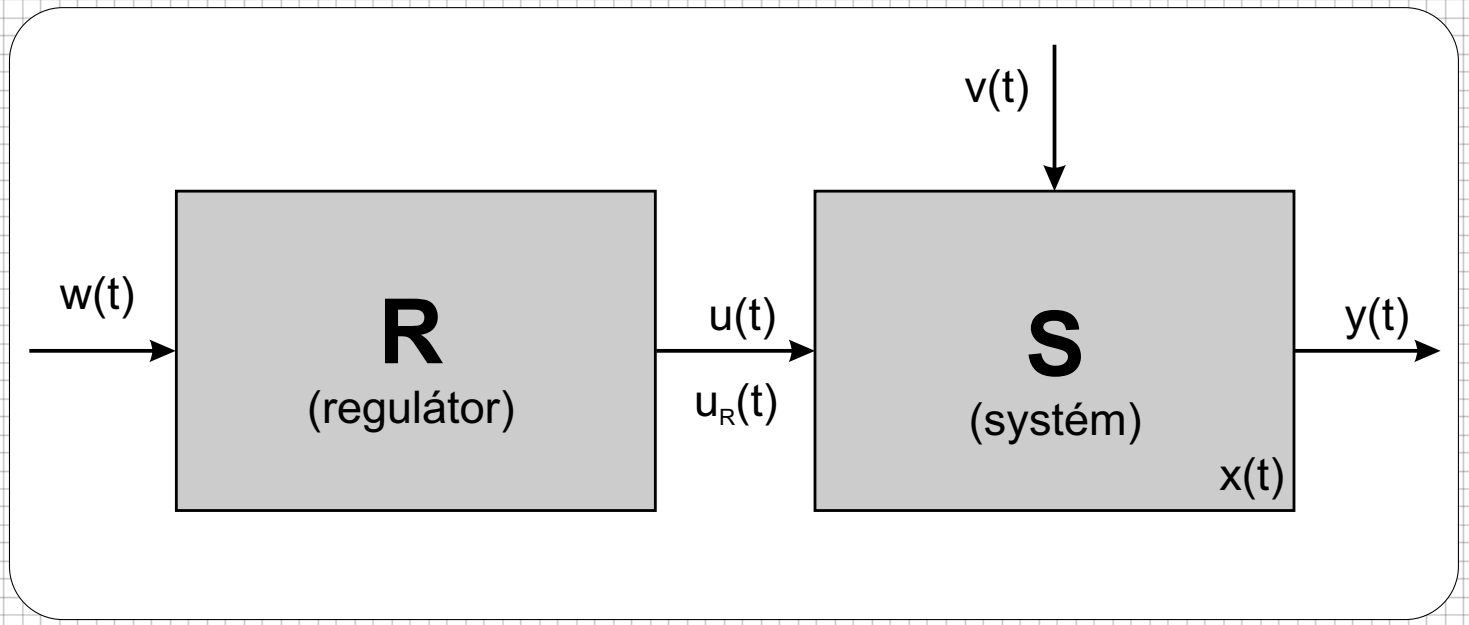
Grafické výstupy - fázová oblast



Grafické výstupy - časová oblast (detail)

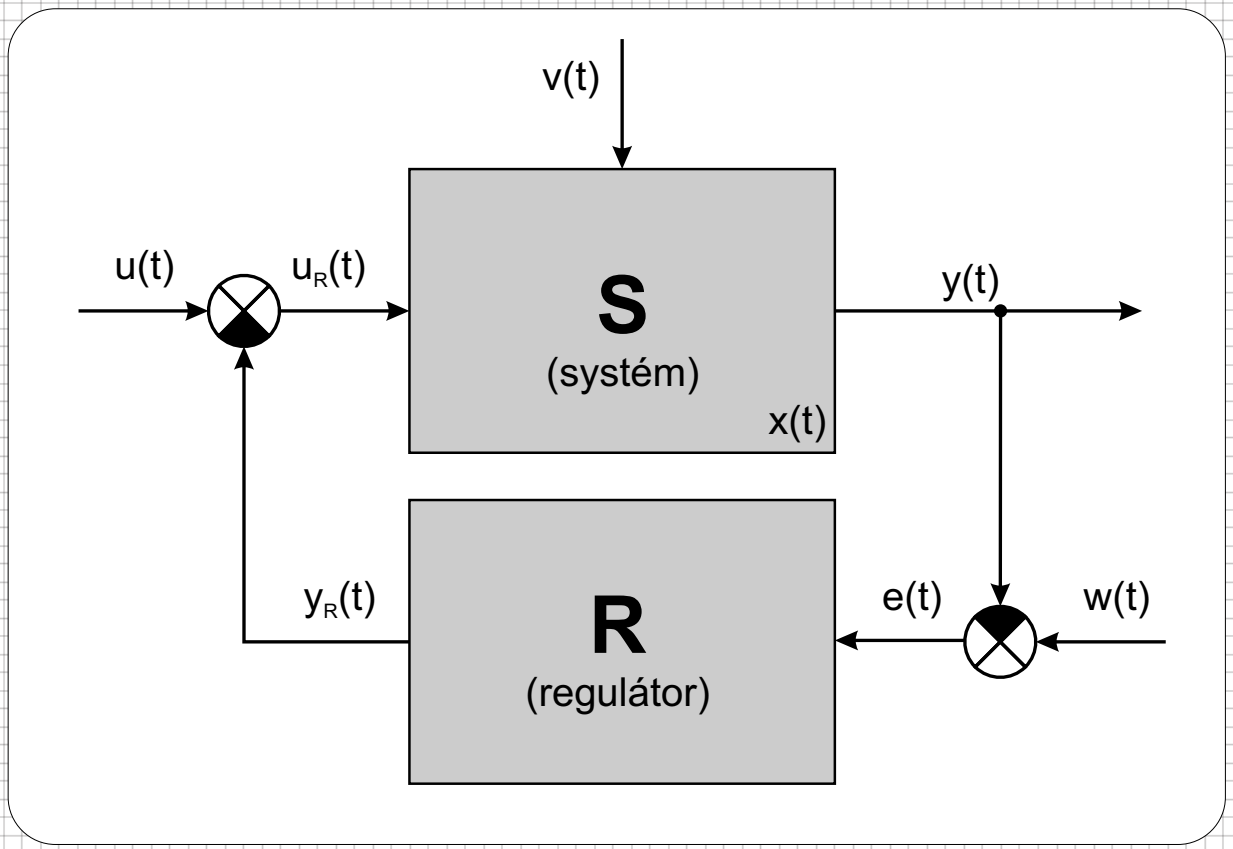


Regulační obvod



Obrázek: Blokové schéma ovládání

Regulační obvod

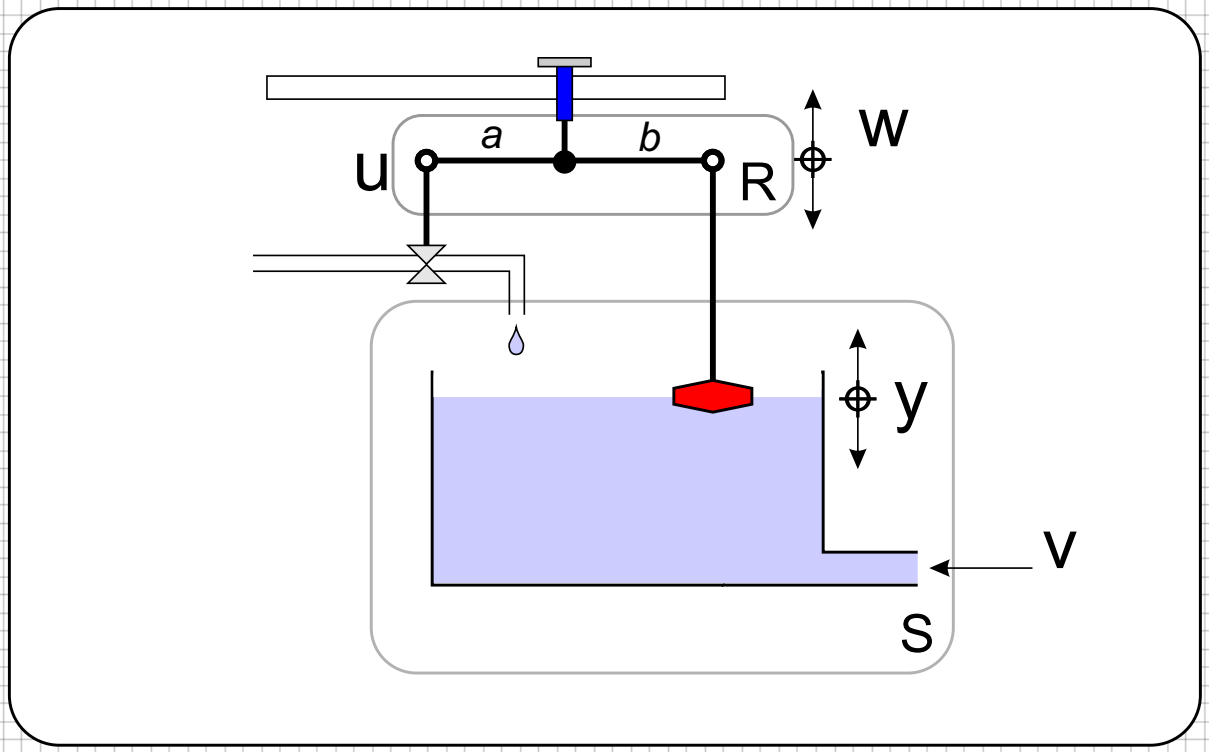


$$e(t) = w(t) - y(t)$$

$$u_R(t) = u(t) - y_R(t)$$

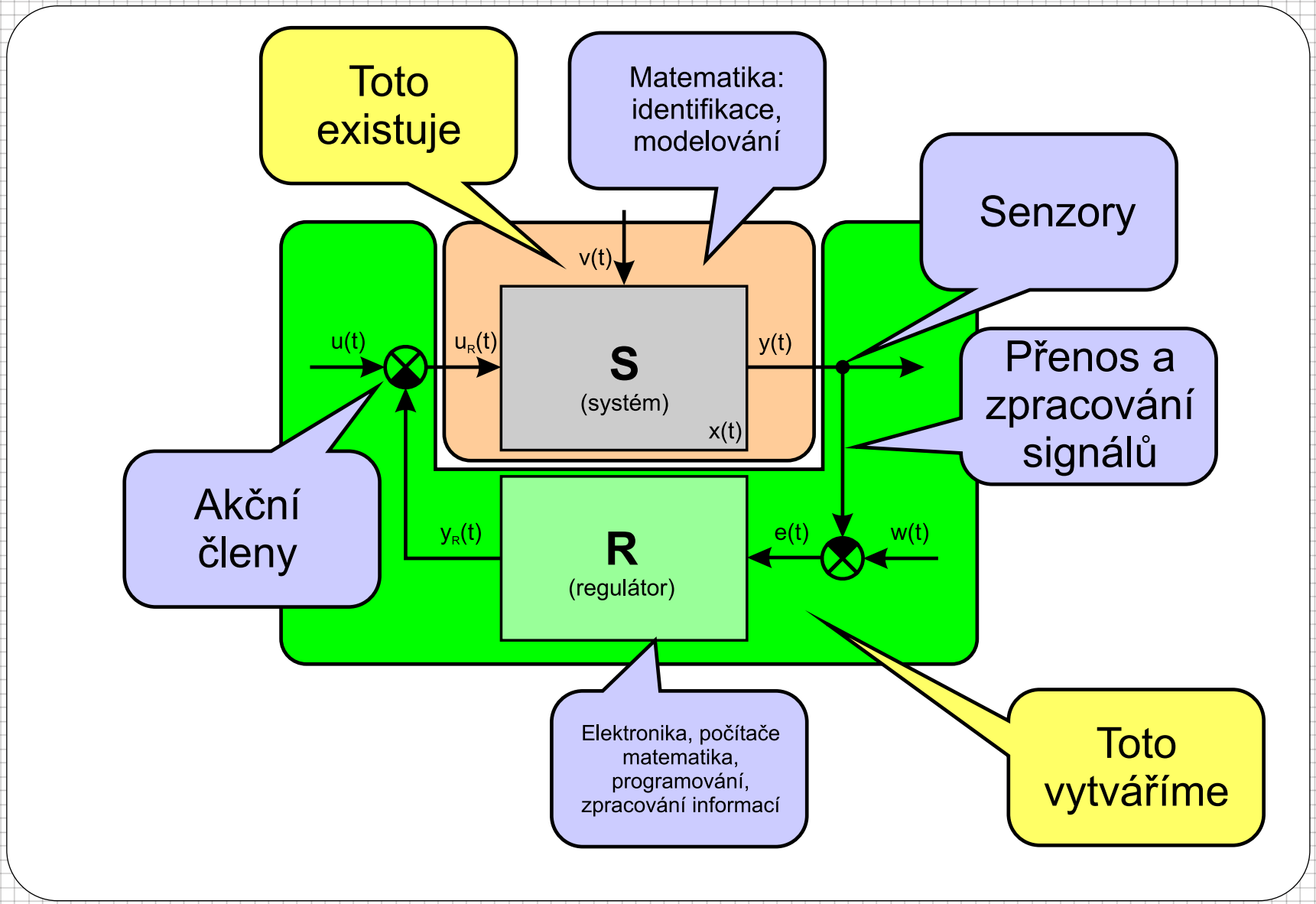
Obrázek: Blokové schéma regulace

Příklad regulačního obvodu



Obrázek: Splachovadlo

Rozbor problematiky řídicí úlohy



Unifikované signály

Napěťové

0÷1 V, -1÷+1 V, 0÷5 V, -5÷+5 V, 0÷10 V, -10÷+10 V

Proudové

0÷20 mA, 4÷20 mA v proudové smyčce.

Logické

0 a 5 V, 0 a 12 V, 0 a 24 V, 12 a -12 V,

Regulátory

Typy regulátorů

- dvojpolohový
- třípolohový
- spojitý
- diskrétní

Regulátory spojité (kvazispojité)

Složky regulátoru

- P roporcionální (P)
- I ntegrační (S umační)
- D erivační(D iferenciální)

Možné kombinace:
P; I; PI; PID; PD

Regulátor PID - rovnice

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

po Laplaceově transformaci dostaneme rovnici:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

- K_p . . . proporcionální zesílení
- K_i . . . integrační zesílení
- K_d . . . derivační zesílení

Rovnice (přenos) regulátoru PID

$$F_R(p) = \frac{Y_R(p)}{E(p)} = r_0 + r_d p + \frac{r_i}{p} = K_R(1 + T_D p + \frac{1}{T_i p})$$

PID - zajímavosti

Vznik PID regulátoru se datuje k roku 1890.
PID regulátory byly následně vyvinuty k automatickému řízení lodí.
Jeden z prvních příkladů regulátoru typu PID vyvinul Elmer Sperry v roce 1911, zatímco první teoretický rozbor PID regulátoru publikoval rusko-americký inženýr Nicolas Minorsky.
PID regulátory jsou široce průmyslově využívány. Odhaduje se, že až 95 % regulačních obvodů v průmyslu je realizováno PID regulátory.

Zajímavost

Souhrn průzkumů provedených v několika průmyslových odvětvích během devadesátých let uvedený v *Yu, Ch. (1999), Autotuning of PID Controllers – Relay Feedback Approach, Springer Verlag* zmiňuje mimo jiné následující skutečnosti:
Z asi 2000 sledovaných regulačních obvodů v papírenském průmyslu jen 20 % pracovalo tak dobře, že automatická regulace vedla k menšímu kolísání regulované veličiny než ruční řízení.
Kvalita regulace v 30 % obvodů byla velmi bídná v důsledku špatného nastavení regulátorů a v dalších 30 % kvůli špatným vlastnostem regulačních ventilů.
V oblasti řízení chemických procesů nebylo 30 % smyček raději vůbec provozováno v automatickém režimu a bylo řízeno pouze ručně a ve 20 % byly v důsledku bezradnosti uživatelů, jak regulátor nastavit, ponechány původní hodnoty konstant, s nimiž byl regulátor dodán z výrobního závodu. V asi 30 % se opět objevily problémy v důsledku špatných vlastností regulačních ventilů.

Obecné tipy pro návrh PID regulátoru

Když navrhujete PID regulátor pro daný systém, postupujte podle níže uvedených kroků, abyste získali požadovanou odpověď.

Získejte odpověď s otevřenou smyčkou a zjistěte, co je třeba zlepšit.
Přidejte proporcionální složku (zesílení) řízení ke zlepšení doby náběhu.
Chcete-li omezit překmit, přidejte derivační složku.
Přidejte integrační složku, abyste snížili chybu v ustáleném stavu.

Upravujte každý ze zisků K_p, K_i a K_d, dokud nezískáte celkovou požadovanou odpověď.

Charakteristika složek: P

Zvýšením proporcionálního zesílení K_p dosáhujeme toho, že při stejné úrovni regulační odchylky dostaneme větší řídicí signál.
Skutečnost, že regulátor bude silněji „působit“ při dané úrovni regulační odchylky má za následek, že systém uzavřené smyčky reaguje rychleji, ale také s větším překmitem.
Dalším efektem zvyšování K_p je, že má tendenci snižovat, ale ne eliminovat, regulační odchylku v ustáleném stavu.

Charakteristika složek: D

Přidání derivační složky regulátoru (K_d) dává regulátoru schopnost „předpovídat (anticipate)“ regulační odchylku. S jednoduchým proporcionálním řízením, je-li K_p konstantní, je jediným způsobem, jakým se vliv regulátoru zvýší, je případ, kdy se odchylka zvětší. Při derivační složce řízení může být řídicí signál větší, pokud se chyba začne snižovat, i když je velikost chyby stále relativně malá. Toto „předvídání“ má tendenci přidat do systému tlumení, čímž se snižuje překmit.
Přidání derivační složky však nemá žádný vliv na ustálený stav chyby.

Charakteristika složek: I

Přidání integrční složky do regulátoru (K_i) má tendenci pomoci snížit ustálený hodnotu odchylky. Pokud dojde ke stabilní regulační odchylce, integrátor se sumuje a sumuje, čímž zvyšuje řídící signál a tím snižuje regulační odchylku. Nevýhodou integrční složky je však to, že může způsobit zpomalení dodezvy (případě oscilace), protože když signál chyby změní znaménko, může chvíli trvat, než se integrátor vynuluje.

Charakteristika složek: shrnutí

Obecné účinky každého parametru regulátoru (K_p, K_d, K_i) na systém uzavřené smyčky jsou shrnuty v následující tabulce. Všimněte si, že tyto pokyny platí v mnoha případech, ale ne vždy. Pokud skutečně chcete znát efekt vyladění jednotlivých zisků, budete muset provést více analýz nebo bude muset provést testování na aktuálním systému.

A konečně, mějte na paměti, že nepotřebujete implementovat všechny tři regulátory (proporcionální, derivační a integrální) do jediného systému, pokud to není nutné. Pokud například PI regulátor splňuje dané požadavky (například výše uvedený příklad), nemusíte v systému implementovat derivační složku. Udržujte regulátor co nejjednodušší.

Obecné tipy pro návrh PID regulátoru

S-S ERROR (Steady state error) - odchylka ustáleného stavu

$$\lim_{t\to\infty} e(t) = e_{ss} = \lim_{s\to 0} \frac{sU(s)}{1+G(s)}$$

CL odezva	K_p	K_i	K_d
Doba náběhu	$\searrow$	$\searrow$	$\leftrightarrow$
Překmit	$\nearrow$	$\nearrow$	$\searrow$
Doba ustálení	$\leftrightarrow$	$\nearrow$	$\searrow$
Odchylka ustáleného stavu	$\searrow$	$\searrow$	0

Regulátor ON/OFF

```
--Pseudocode--
motor = full speed;
while (poloha menší než očekávaná)
{}
motor = stop;
while (poloha větší nebo rovna než očekávaná)
{}
----------------
```

Regulátor P

```
--Pseudocode--
error = (cílová hodnota) - (měřená hodnota);
Kp = 0.5;
speed = Kp * error;
---------------
```

```
--Pseudocode--
Kp = 0.5;
while (condition)
{
error = (cílová hodnota) - (měřená hodnota);
speed = Kp * error;
}
---------------
```

Regulátor I

```
integral = integral + error*dT;
```

Regulátor PI

```
--Pseudocode--
Kp = 0.5;
Ki = 0.2;
while (condition)
{
error = (target value) - (sensor reading);
integral = integral + error;
speed = Kp*error + Ki*integral;
}
---------------
```

Regulátor PI

Dva potencionální problémy:
1. Co udělat když error=0?

```
--Pseudocode--
Kp = 0.5;
Ki = 0.2;
while (condition)
{
error = (target value) - (sensor reading);
integral = integral + error;
if (error is 0) { integral = 0;}
speed = Kp*error + Ki*integral;
}
```

Regulátor PI

Dva potencionální problémy:
2. Co udělat když error je příliš velký?
1. řešení

```
if (integral is greater than or equal to the maximum
value)
{
integral = maximum value;
}
```

```
2. řešení
if ( abs(error) is greater than useful for the
integral)
{
disable the integral (set the integral to 0); }
```

Regulátor PI

```
Kp = 0.5;
Ki = 0.2;
while (condition)
{
error = (target value) - (sensor reading);
integral = integral + error;
if (error = 0){ integral = 0; }
if ( abs(error) > 40){ integral = 0; }
speed = Kp*error + Ki*integral;
}
```

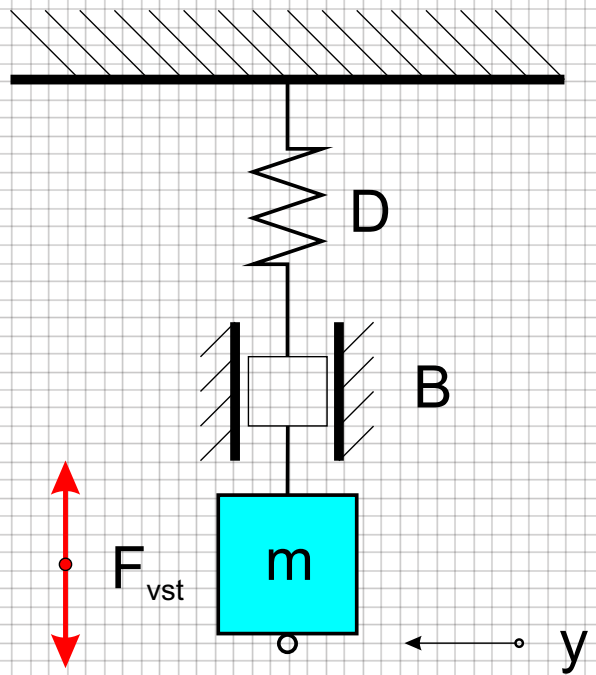
Regulátor D

derivative = ((current error) – (previous error)) / dT

Regulátor PID

```
Kp = 0.5;
Ki = 0.2;
Kd = 0.1;
while (condition)
{
error = (target value) - (sensor reading);
integral = integral + error;
if (error = 0){ integral = 0;}
if ( abs(error) > 40) { integral =0;}
derivative = error - previous error;
previous error = error;
speed = Kp*error + Ki*integral + Kd*derivative;
}
```

Příklad: Mechanický harmonický oscilátor



D . . . konstanta pružiny
y . . . poloha
B . . . tlumení
m . . . hmotnost
F_{vst} . . . vstupní síla

$$F_D = Dy$$

$$F_B = Bv = B\frac{dy}{dt} = B\dot{y}$$

$$F_m = ma = m\frac{d^2y}{dt^2} = m\ddot{y}$$

Rovnováha sil

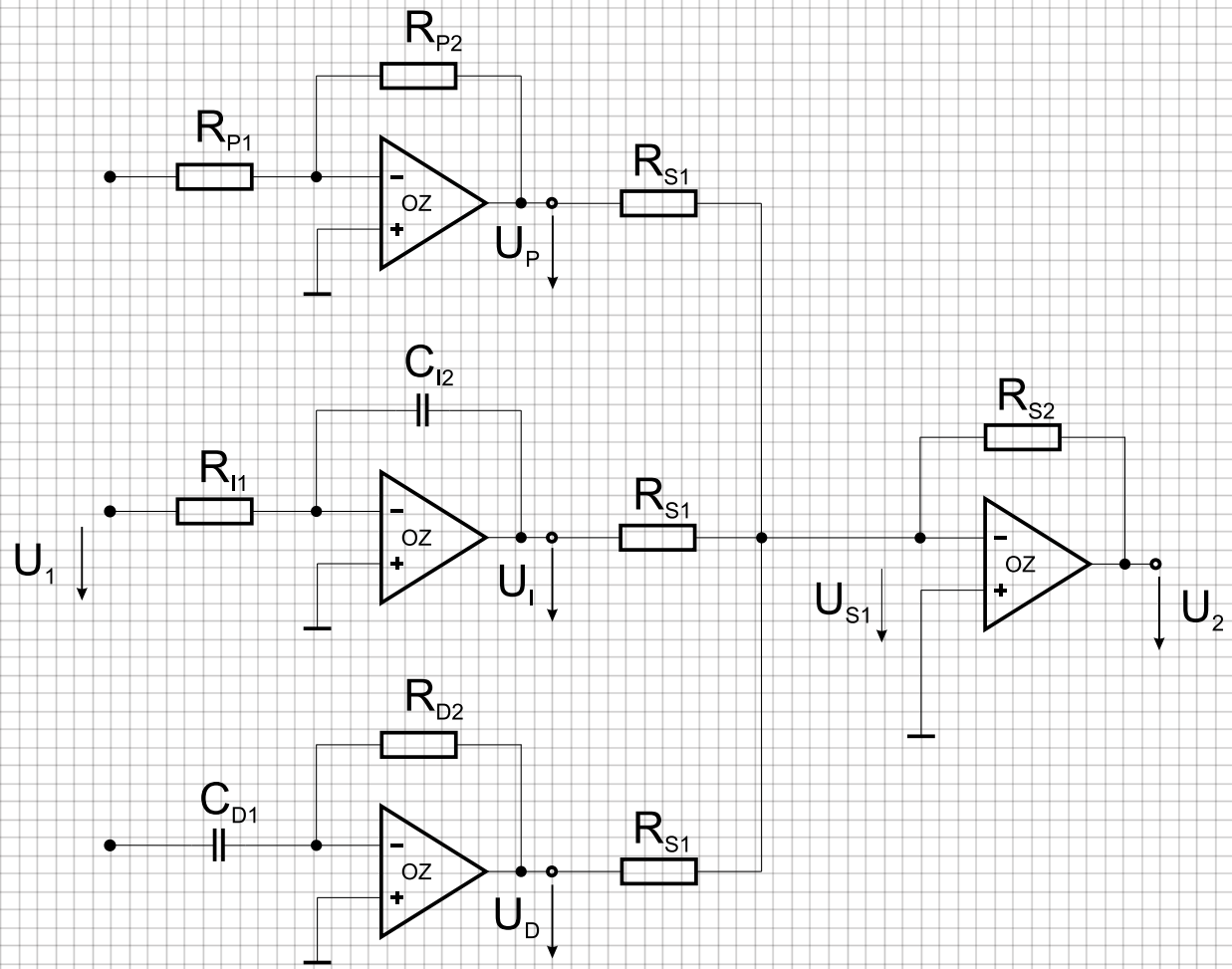
$$F_m + F_B + F_D = F_{vst}$$

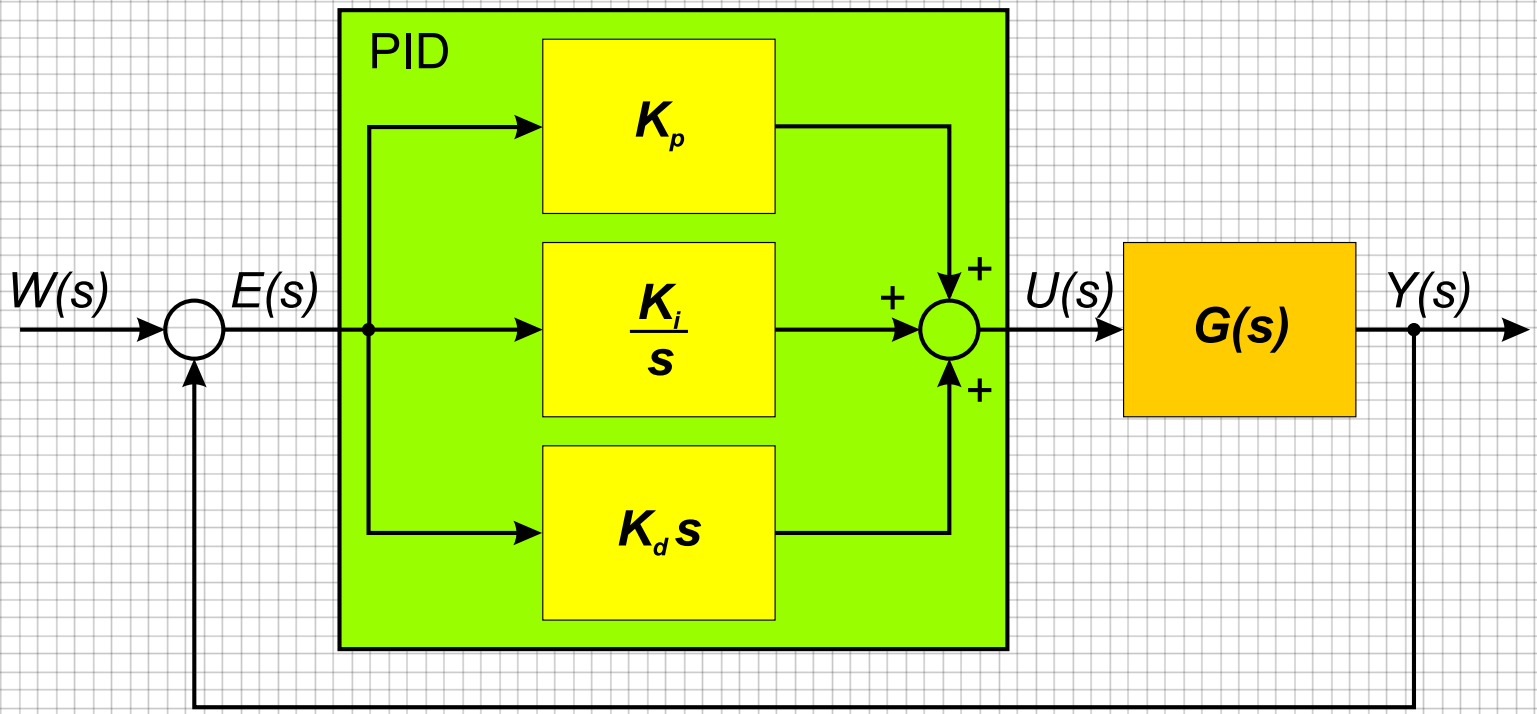
Příklad: Mechanický harmonický oscilátor

$$m\frac{d^2y}{dt^2} + B\frac{dy}{dt} + Dy = m\ddot{y}(t) + B\dot{y}(t) + Dy(t) = F_{vst}(t)$$

$$ms^2Y(s) + bsY(s) + kY(s) = F(s) \qquad \Rightarrow$$

$$G(s) = \frac{Y(s)}{F(s)} = \frac{1}{ms^2 + bs + k}$$


R_P2
R_P1
R_S1
OZ
U_P
C_I2
R_I1
R_S1
U_1
OZ
U_I
R_S2
U_S1
OZ
U_2
R_D2
C_D1
R_S1
OZ
U_D



$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

Přenos regulátoru

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

Vzhledem k problémům při rychlé změně regulační odchylky se omezuje vliv derivační složky

$$u(t) = K_d \dot{y}(t)$$

omezením frekvenčního pásma (τ_1): $\dot{y}(t) = (e(t) - y(t))/\tau_1$.
Obvykle se pokládá

$$\tau_1 = \frac{K_d}{N K_p} \qquad N \in < 10, 20 >$$

.

Přenos regulátoru

Potom Laplaceův obraz je

$$U(s) = K_d s Y(s) = \frac{K_d s E(s)}{(\tau_1 s + 1)}$$

potom modifikovaný přenos PID regulátoru je:

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + \frac{K_d s}{\tau_1 s + 1}$$

$$\frac{U(s)}{E(s)} = \frac{(K_p \tau_1 + K_d)s^2 + (K_p + K_i)s + K_i}{s(\tau_1 s + 1)}$$

Octave 1

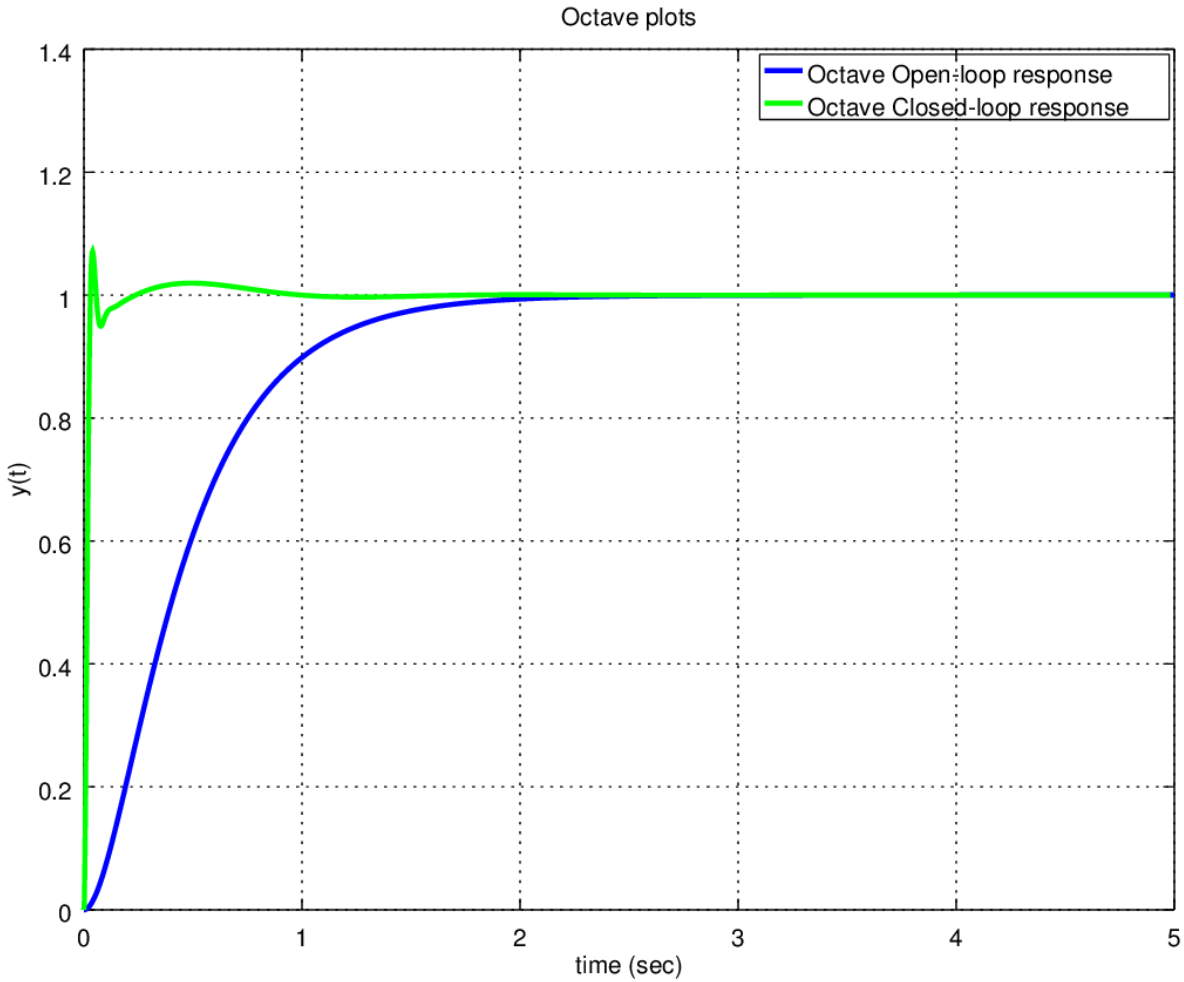
```
1
2 tic; # Measures performance using stopwatch time
3 # System parameters M = 1 kg, b = 10 N.s/m, k = 20 N/m, F(s) = 1
4 M = 1; b = 10; k = 20;
5 # System in Transfer function form:
6 num = 1;
7 den = [M b k];
8 # Modified PID controller
9 Kp = 290;
10 ti = 0.75;
11 t1 = 0.5/100;
12 N=15;
13 Ki = Kp/ti;
14 Kd = t1*N*Kp;
15 # Numerator controller
16 numc = [Kp*t1+Kd Kp+Ki*t1 Ki]/t1;
17 # Denominator controller
18 denc = [t1 1 0]/t1;
19 # Numerator Feedback system
20 num_feedback = conv(num, numc);
21 # Denominator Feedback system
22 den_feedback = conv(den, denc);
23 # Feedback system
24 den_closed_loop = den_feedback+[0 0 num_feedback];
```

A Listings Demo 1

Matlab

```
# Feedback system
den_closed_loop = den_feedback+[0 0 num_feedback];
# Output of Feedback system
num_y = num_feedback;
# Output of Open-loop system (scaled by 20 for it to be settled to 1)
sys_open_loop = tf(20*num, den);
# Output of Closed-loop system
sys_y = tf(num_y, den_closed_loop);
t = (0:.002:5)';
y_open_loop = step(sys_open_loop, t);
# y_open_loop = step(sys_open_loop, 1, 5, length(t));
y_closed_loop = step(sys_y, t);
# y_closed_loop = step(sys_y, 1, 5, length(t));
figure(1);
plot(t, y_open_loop, t, y_closed_loop);
grid on;
xlabel('time (sec)'); ylabel('y(t)');
# title('Matlab plots');
title('Octave plots');
# legend('Matlab Open-loop response', 'Matlab Closed-loop response');
legend('Octave Open-loop response', 'Octave Closed-loop response');
toc; # Measures performance using stopwatch time
```

Odezva systému na jednotkový skok - přechodová charakteristika



Obrázek: Systém

PLC (Programable Logic Controller)

PLC (Programable Logic Controller)

PLC je číslicový elektronický systém vyvinutý pro realizaci řízení v průmyslovém prostředí.

PLC zajišťuje tyto základní funkce:

- logické řízení (DI, DO, RE)
- „analogové“ řízení (AI, AO, PWM, . . .)
- časovou návaznost úkonů
- interakci s obsluhou
- komunikaci s okolím (nadřízené a podřízení ŘS, . . .)

Možná kritéria dělení PLC:

- podle výkonu
- podle aplikační oblasti
- podle druhu vstupů a výstupů
- podle počtu vstupů a výstupů
- podle provedení
  - kompaktní
  - modulární

Vlastnosti PLC:

- odolnost
- spolehlivost
- modularita
- výhodná cena HW
- cena SW (speciální jazyky)
- flexibilita
- nenáročnost na obsluhu
- škálovatelnost

Metody programování PLC

Jazyky používané pro programování PLC:

- IL - Instruction List (posloupnost instrukcí)
- LD - Ladder Diagram (kontaktní plán / liniové či reléové schéma)
- SFC - Sequential Function Chart (vývojové schéma)
- FBD - Function Block Diagram (schéma funkčních bloků)
- CFC - Continuous Function Chart (volně propojované bloky)
- ST - Structured Text (vyšší programovací jazyk - obdoba Pascalu)

http://automatizace.hw.cz/programovaci-rezimy-pro-plc-dle-iec-611313-codesys

IL - Instruction List

Podobá se hodně asembleru.

Výhody

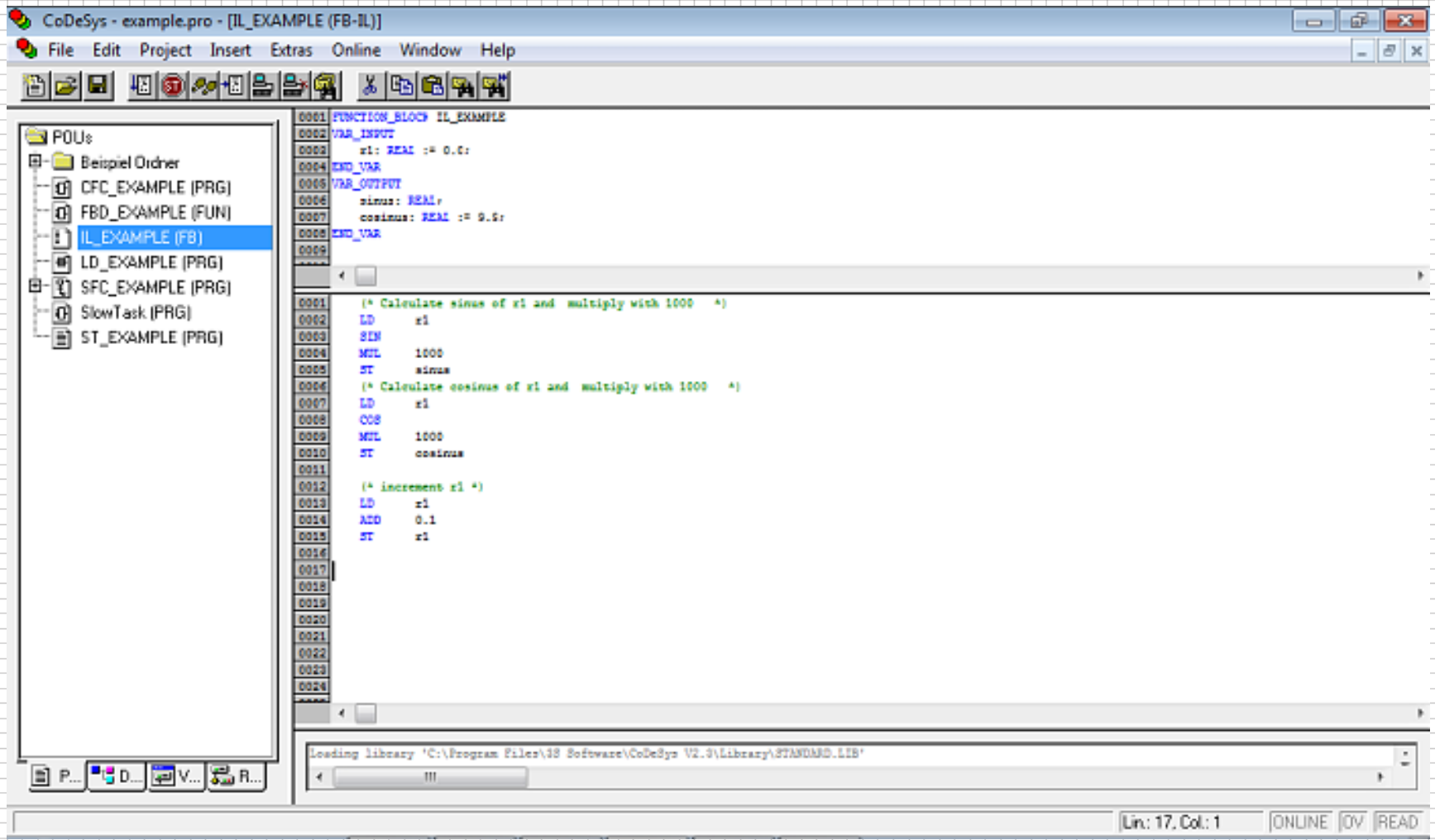
+ Přesná definice chování programu
+ Paměťově i na rychlost zpracování úsporný program

Nevýhody

- Nutnost znát nebo se alespoň dobře orientovat v příkazech a registrech
- Nutno znát hodně instrukcí
- Specifický pro jednotlivé PLC
- Mnoho psaní
- Horší přehlednost programu a orientace v něm

Vhodný pro: psaní krátkých vysoce optimalizovaných částí programu

IL - jazyk mnemokódů



Obrázek: IL

LD - Ladder Diagram

Vychází z grafické podoby projektování releových obvodů

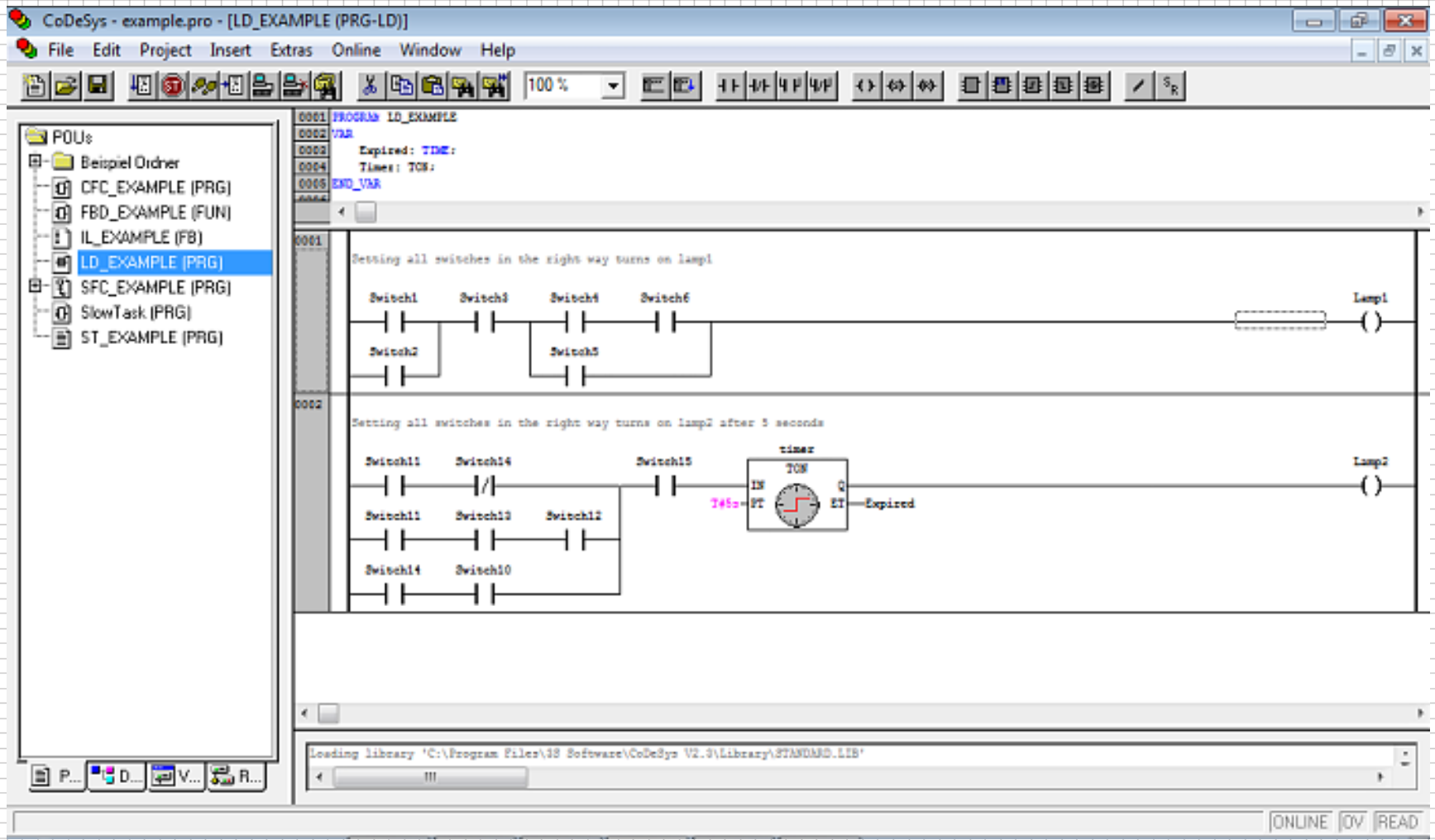
Výhody

+ Jasně definovaná posloupnost zápisu, kterou nelze porušit
+ Přehlednost zápisu (zvláště u menších programů)
+ Velmi rychlé programování logických operací s funkcemi čítání a časování
+ Ideální pro zpracování velkého počtu logických signálů (vstupů a výstupů)

Nevýhody

- Méně vhodný pro aritmetické operace a práce s daty (ASCII znaky)
- S rostoucí složitostí programu rychle narůstá jeho délka
- Hůře pochopitelný pro „klasické programátory“

Vhodný pro: realizaci logických řídicích sekcí a sekvencí programu a zpracování vstupních a výstupních signálů


CoDeSys - example.pro - [LD_EXAMPLE (PRG-LD)]
File Edit Project Insert Extras Online Window Help
100 %
POUs
Beispiel Ordner
CFC_EXAMPLE (PRG)
FBD_EXAMPLE (FUN)
IL_EXAMPLE (FB)
LD_EXAMPLE (PRG)
SFC_EXAMPLE (PRG)
SlowTask (PRG)
ST_EXAMPLE (PRG)
PROGRAM LD_EXAMPLE
VAR
Expired: TIME;
Timer: TON;
END_VAR
Setting all switches in the right way turns on lamp1
Switch1
Switch3
Switch4
Switch6
Switch2
Switch5
Lamp1
Setting all switches in the right way turns on lamp2 after 5 seconds
Switch11
Switch14
Switch15
timer
TON
IN
PT
Q
ET
T#5s
Expired
Lamp2
Switch11
Switch13
Switch12
Switch14
Switch10
ONLINE
OV
READ

Obrázek: IL

ST - Structured Text

Jazyk velmi podobný třeba jazyku Pascal

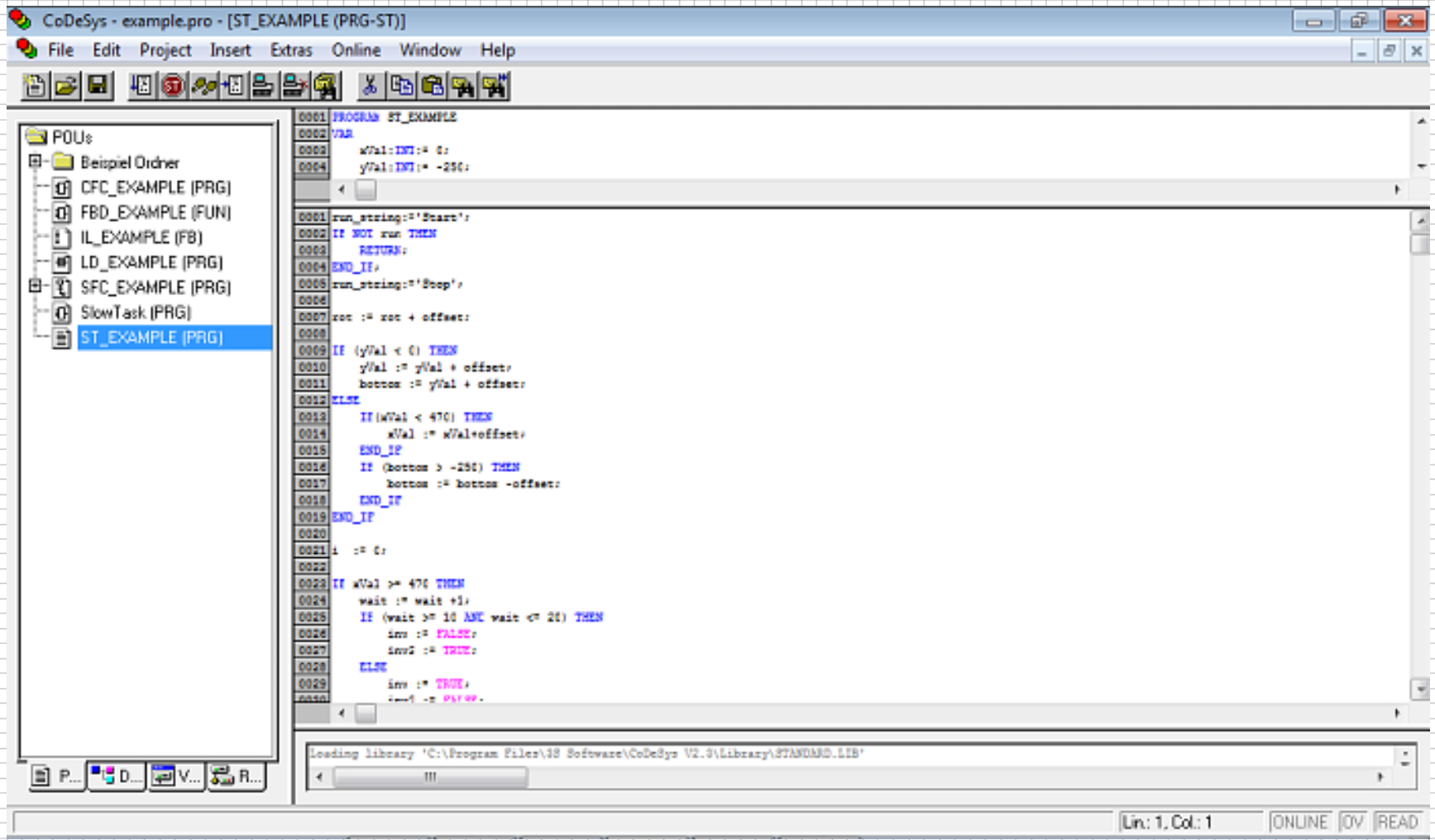
Výhody

+ Snadné programování složitých aritmetických operací a vzorců
+ Ideální pro složité zpracování analogových signálů
+ Ideální pro práci s velkými bloky dat a databázemi
+ Vhodný pro práci s textovými řetězci
+ Vhodný pro realizaci / zpracování datové komunikace

Nevýhody

- Je nutné znát příkazy a přesnou syntaxi zápisu
- Horší přehlednost zápisu logických operací
- Méně vhodný pro přehledné zpracování velkého množství logických signálů (vstupů / výstupů)

Vhodný pro: klasické programátory, realizaci složitých algoritmů a práci s daty a řetězci


CoDeSys - example.pro - [ST_EXAMPLE (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
Beispiel Ordner
CFC_EXAMPLE (PRG)
FBD_EXAMPLE (FUN)
IL_EXAMPLE (FB)
LD_EXAMPLE (PRG)
SFC_EXAMPLE (PRG)
SlowTask (PRG)
ST_EXAMPLE (PRG)
PROGRAM ST_EXAMPLE
VAR
xVal:INT:= 0;
yVal:INT:= -256;
END_VAR
run_string:='Start';
IF NOT run THEN
RETURN;
END_IF
run_string:='Stop';
xor := xor + offset;
IF (yVal < 0) THEN
yVal := yVal + offset;
bottom := yVal + offset;
ELSE
IF(xVal < 470) THEN
xVal := xVal+offset;
END_IF
IF (bottom > -256) THEN
bottom := bottom -offset;
END_IF
END_IF
i := 0;
IF xVal >= 470 THEN
wait := wait +1;
IF (wait >= 10 AND wait <= 20) THEN
inv := FALSE;
inv2 := TRUE;
ELSE
inv := TRUE;
Loading library 'C:\Program Files\3S Software\CoDeSys V2.3\Library\STANDARD.LIB'
Lin: 1, Col: 1
ONLINE
OV
READ

Obrázek: ST

CFC - Continuous Function Chart

Zápis programu v programovacím režimu CFC je naopak ideální pro toho, kdo vyznává skládání programu z jednotlivých „krabiček“ vzájemně propojených přes vývody. Tedy něco pro elektrokonstruktéry, kteří jsou zvyklí vytvářet klasická schémata zapojení součástek. Zde je právě prostě jen plocha, na kterou se postupně vkládají a propojují jednotlivé funkce v podobě „součástek“ s danými vlastnostmi.

CFC - Continuous Function Chart

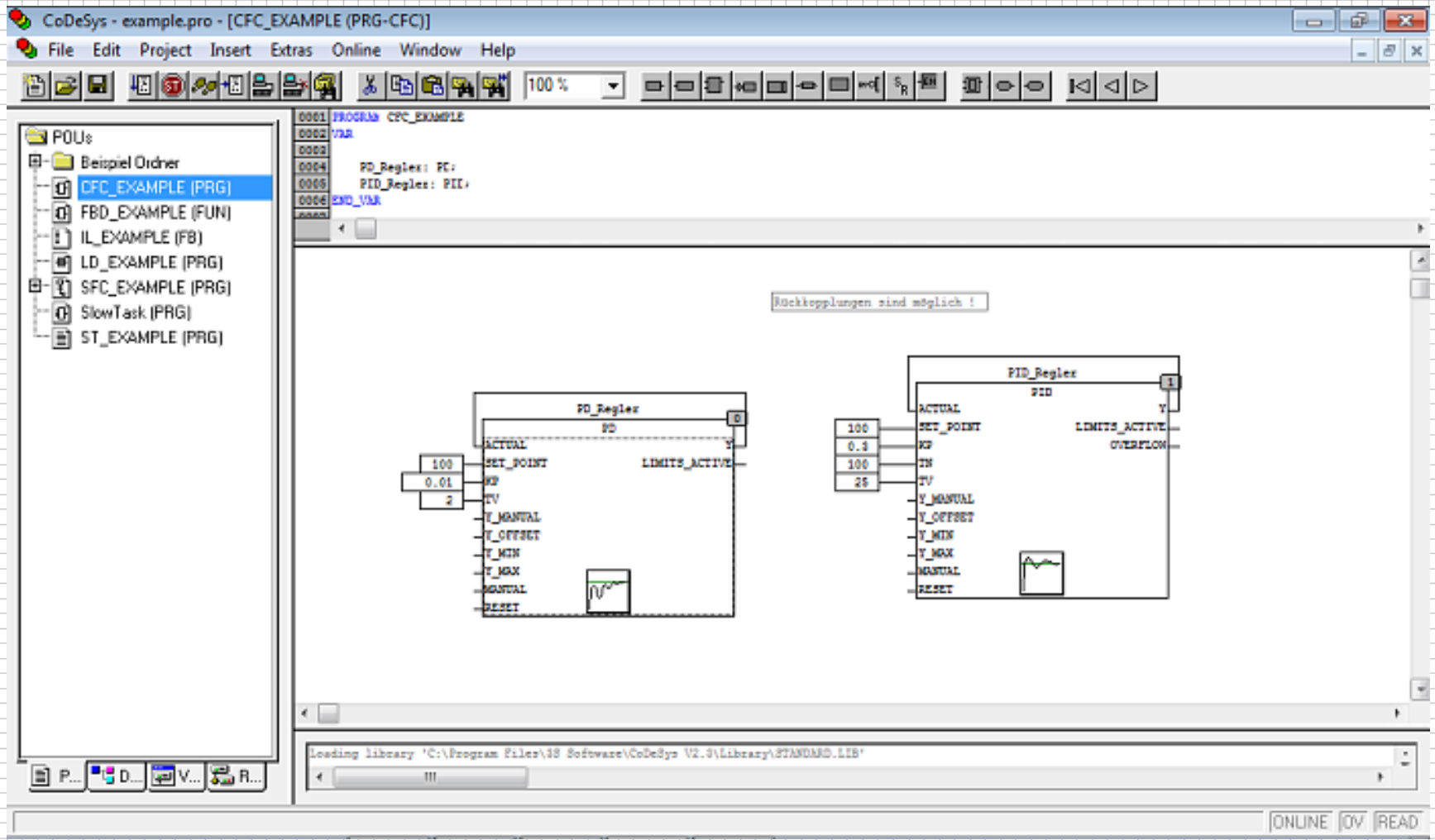
Výhody

+ Volné uspořádání programu dle potřeby programátora
+ Zápis připomíná hardwarové schéma zapojení součástek
+ Snadná realizace jednoduchého zpracování analogových signálů
+ Přehledný průchod signálu strukturou programu při zpracování
+ Přehledná realizace zpětných vazeb

Nevýhody

- Pro složitější programy se zápis stává nepřehledný
- Méně vhodný pro zpracování velkého množství logických signálů (vstupů / výstupů)
- Nevhodný pro realizaci manipulace s většími bloky dat a ASCII řetězci
- Nevhodný pro realizaci datové komunikaci

Vhodný pro: části programu zpracovávající analogově-digitální signály


CoDeSys - example.pro - [CFC_EXAMPLE (PRG-CFC)]
File Edit Project Insert Extras Online Window Help
POUs
Beispiel Ordner
CFC_EXAMPLE (PRG)
FBD_EXAMPLE (FUN)
IL_EXAMPLE (FB)
LD_EXAMPLE (PRG)
SFC_EXAMPLE (PRG)
SlowTask (PRG)
ST_EXAMPLE (PRG)
PROGRAM CFC_EXAMPLE
VAR
PD_Regler: PD;
PID_Regler: PID;
END_VAR
Rückkopplungen sind möglich !
PD_Regler
PD
ACTUAL
SET_POINT
KP
TV
Y_MANUAL
Y_OFFSET
Y_MIN
Y_MAX
MANUAL
RESET
Y
LIMITS_ACTIVE
100
0.01
2
PID_Regler
PID
TN
100
25
Y_MANUAL
Y_OFFSET
Y_MIN
Y_MAX
MANUAL
RESET
Y
LIMITS_ACTIVE
OVERFLOW
0.3
ONLINE
OV
READ

Obrázek: CFC

FBD - Function Block Diagram

V principu podobný CFC

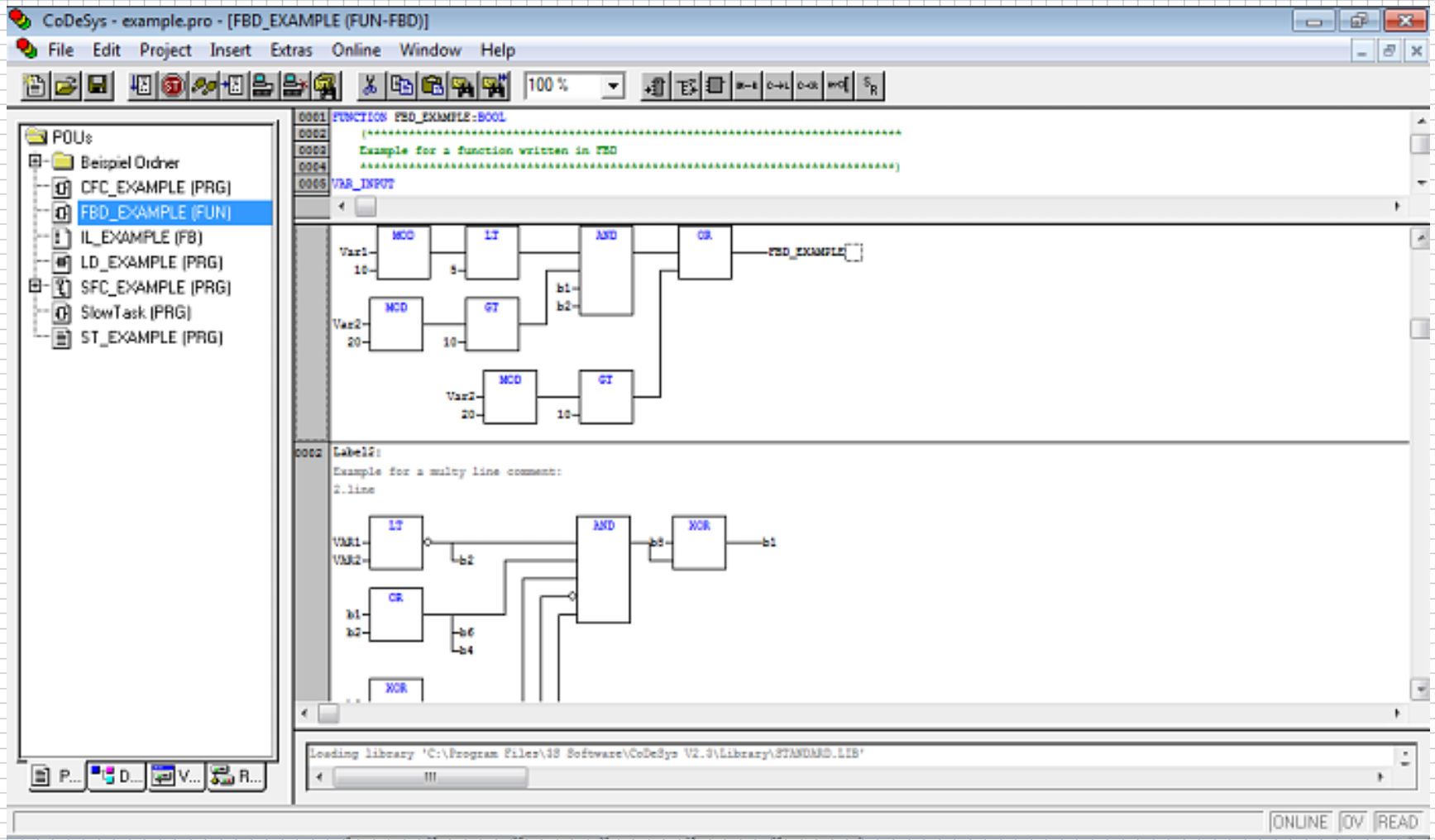
Výhody

+ Definované grafické členění programu do řádků
+ Logické operace v podobě hradel
+ Přehledný zápis programu
+ Ideální pro zpracování velkého počtu logických signálů (vstupů / výstupů)

Nevýhody

- Méně vhodný pro složitější zpracování analogových signálů
- Nevhodný pro hromadnou manipulaci s velkým množstvím dat a ASCII znaky (řetězci)
- Nevhodný pro programování složitých algoritmů (vzorců)

Vhodný pro: realizaci logických řídicích sekcí programu a zpracování vstupních a výstupních signálů


CoDeSys - example.pro - [FBD_EXAMPLE (FUN-FBD)]
File Edit Project Insert Extras Online Window Help
POUs
Beispiel Ordner
CFC_EXAMPLE (PRG)
FBD_EXAMPLE (FUN)
IL_EXAMPLE (FB)
LD_EXAMPLE (PRG)
SFC_EXAMPLE (PRG)
SlowTask (PRG)
ST_EXAMPLE (PRG)
FUNCTION FBD_EXAMPLE:BOOL
Example for a function written in FBD
VAR_INPUT
MOD
LT
AND
OR
GT
XOR
FBD_EXAMPLE
Label2:
Example for a multy line comment:
2.line
ONLINE
OV
READ

Obrázek: FBD

SFC - Sequential Function Chart

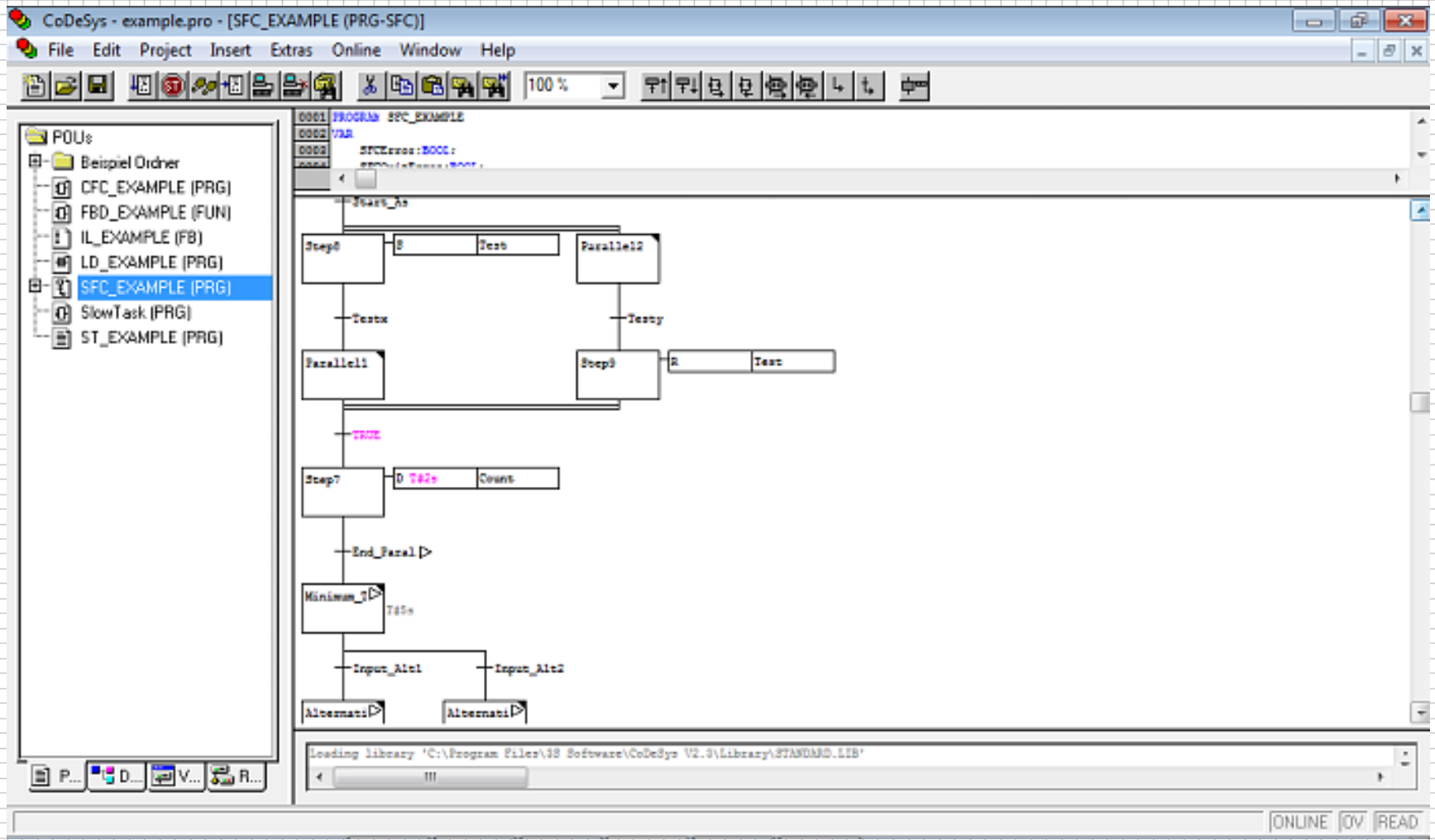
Výhody

+ Velmi přehledný zápis chování programu
+ Přehledné definování a ošetření různých stavů programu
+ Ideální pro realizaci sekvenční logiky
+ Vhodný pro jednoduchou práci s ASCII řetězci

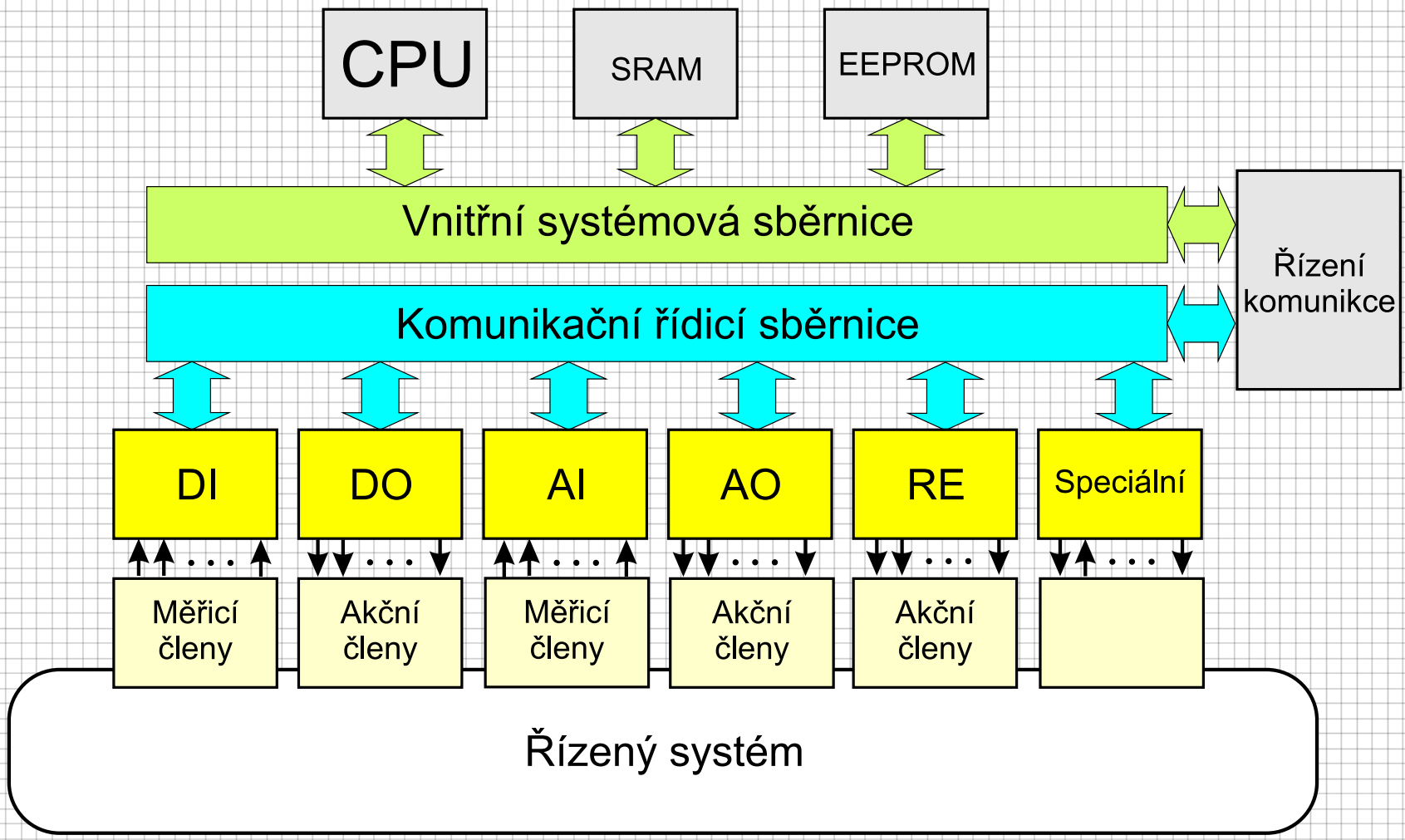
Nevýhody

- Nevhodný pro přímou realizaci zpracování analogových signálů
- Nevhodný pro zpracování velkého počtu logických signálů
- Nevhodný pro programování složitých algoritmů

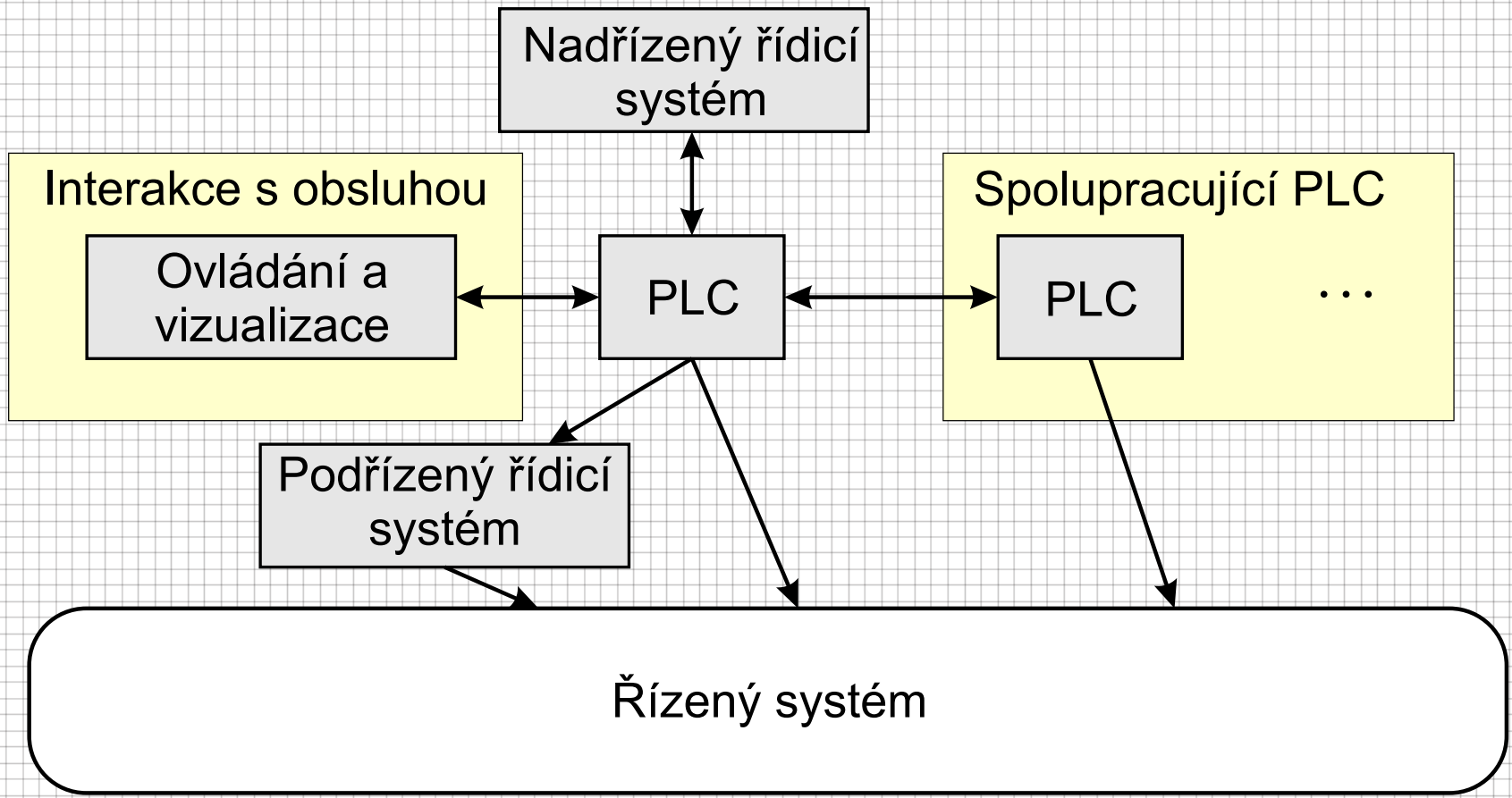
Vhodný pro: vytváření páteřní (hlavní) větve programu, odkud se volají podprogramy


CoDeSys - example.pro - [SFC_EXAMPLE (PRG-SFC)]
File Edit Project Insert Extras Online Window Help
100 %
POUs
Beispiel Ordner
CFC_EXAMPLE (PRG)
FBD_EXAMPLE (FUN)
IL_EXAMPLE (FB)
LD_EXAMPLE (PRG)
SFC_EXAMPLE (PRG)
SlowTask (PRG)
ST_EXAMPLE (PRG)
PROGRAM SFC_EXAMPLE
VAR
SFCErrors:BOOL;
Start_As
Step0
S
Test
Parallel2
Testx
Testy
Parallel1
Step9
R
Test
INIT
Step7
D T#2s
Count
End_Paral
Minimum_7
T#5s
Input_Alt1
Input_Alt2
Alternati
Alternati
Loading library 'C:\Program Files\3S Software\CoDeSys V2.3\Library\STANDARD.LIB'
ONLINE
OV
READ

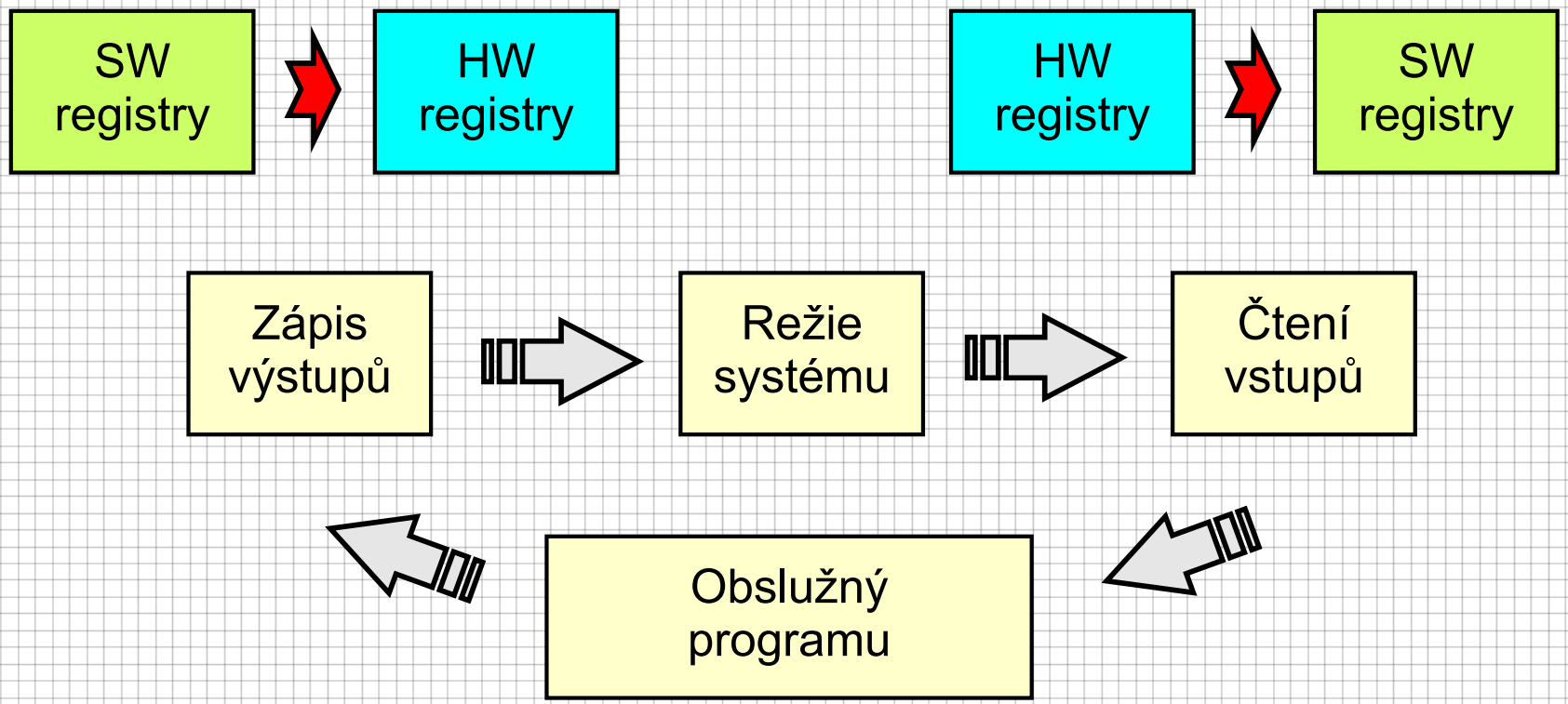
Obrázek: SFC



Obrázek: Blokové schéma PLC



Obrázek: PLC - zapojení do řídicího procesu



Obrázek: Cyklus zpracování SW v PLC

Pro program PLC je typické, že nepracuje s aktuálními hodnotami vstupů a výstupů, ale s jejich obrazy uloženými v zápisníkové paměti

Druhy regulace

Regulace na konstantní hodnotu

Regulace při $w = konst$. Př.: žehlička.

Regulace programová

Regulace při $w = f(t)$. Požadovanou veličinu w měníme v čase podle programu. Př.: vypalování keramiky.

Regulace vlečná

Regulace při $w = f(q)$. Požadovanou veličinu w měníme v závislosti na jiné fyzikální veličině. Př.: vytápění domu.

Druhy regulace

Regulace spojitá

Regulace při níž se signály mění spojitě v čase. Př.: PID regulace.

Regulace nespojitá

Regulace diskrétní v čase, hodnotě. Př.: žehlička.

Fuzzy

Fuzzy v angličtine znamená nezřetelný, mlhavý, neurčitý, neurčitý.

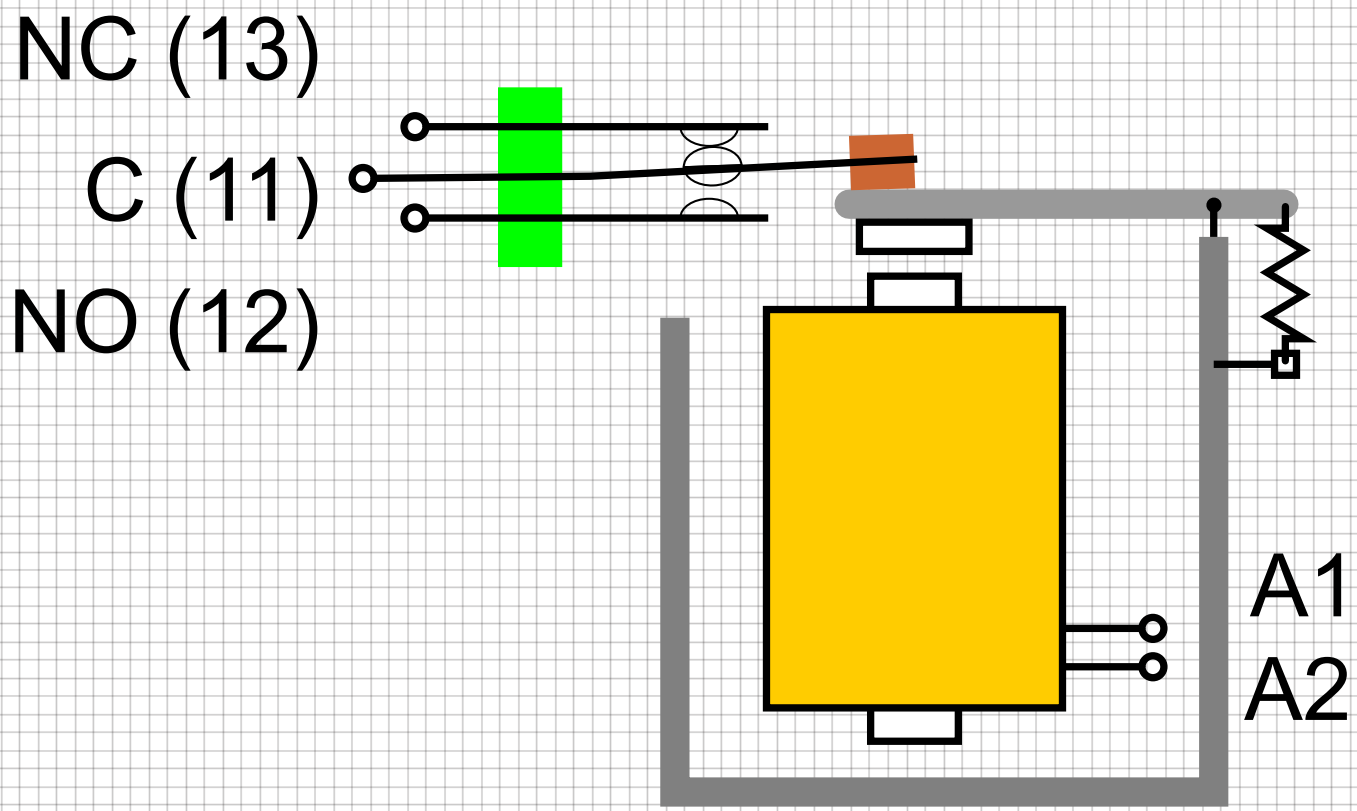
Fuzzy množina

Množina s nezřetelnou (neurčitou) hranicí. Rozumí se tím, že u některých prvků takovéto množiny je příslušnost k množině neurčitá nebo nejednoznačná. Př.: zdravý a nemocný člověk.

Akční členy řídicího obvodu

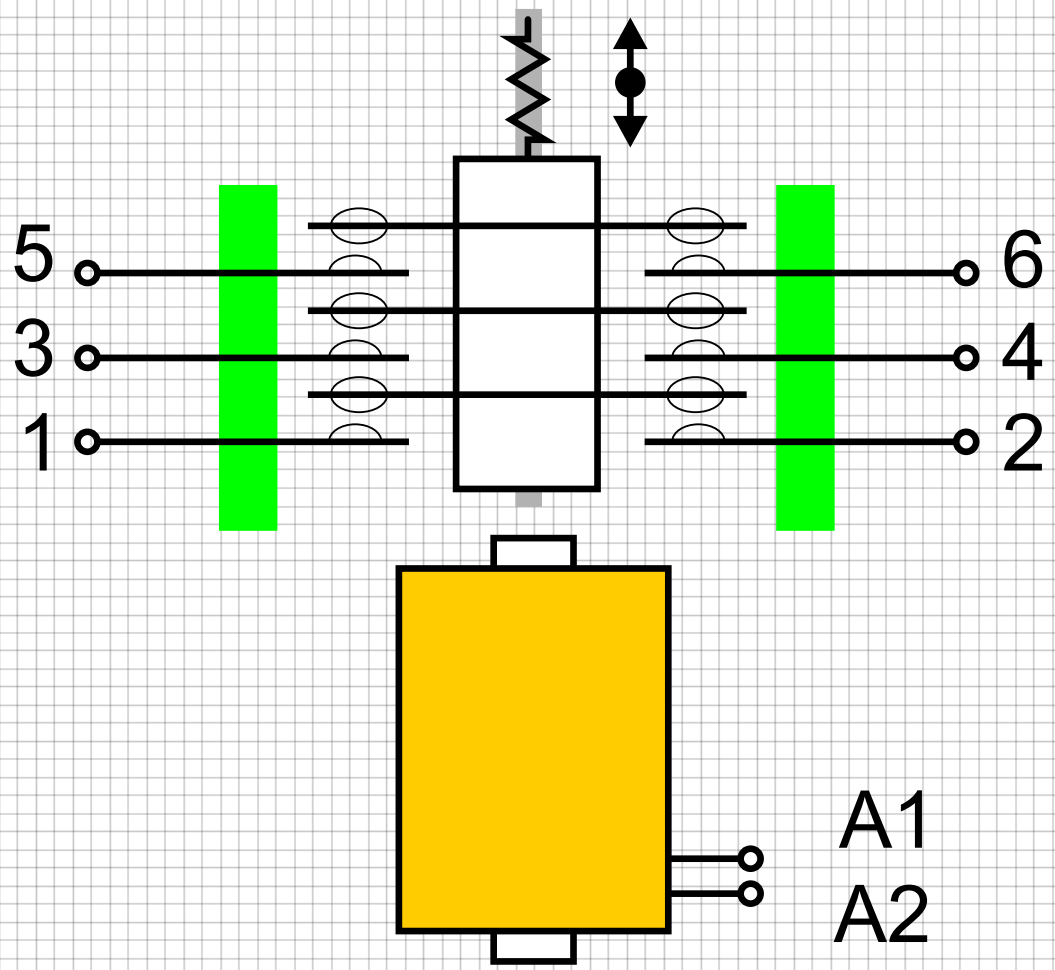
Stavební prvky logického řízení:
relé
stykače
tlačítka
přepínače
časové relé
SSR

Relé



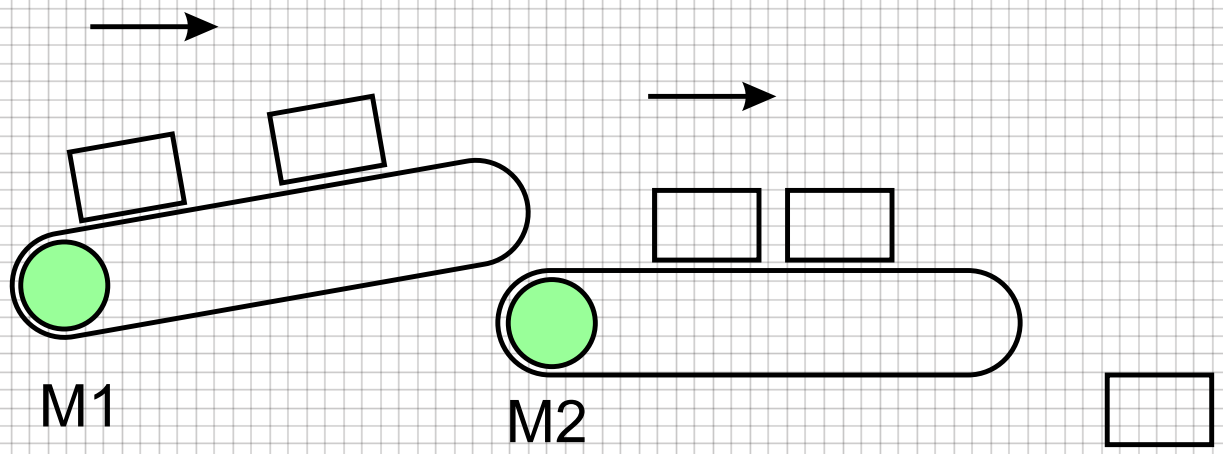
Obrázek: Princip a označení kontaktů

Sykač



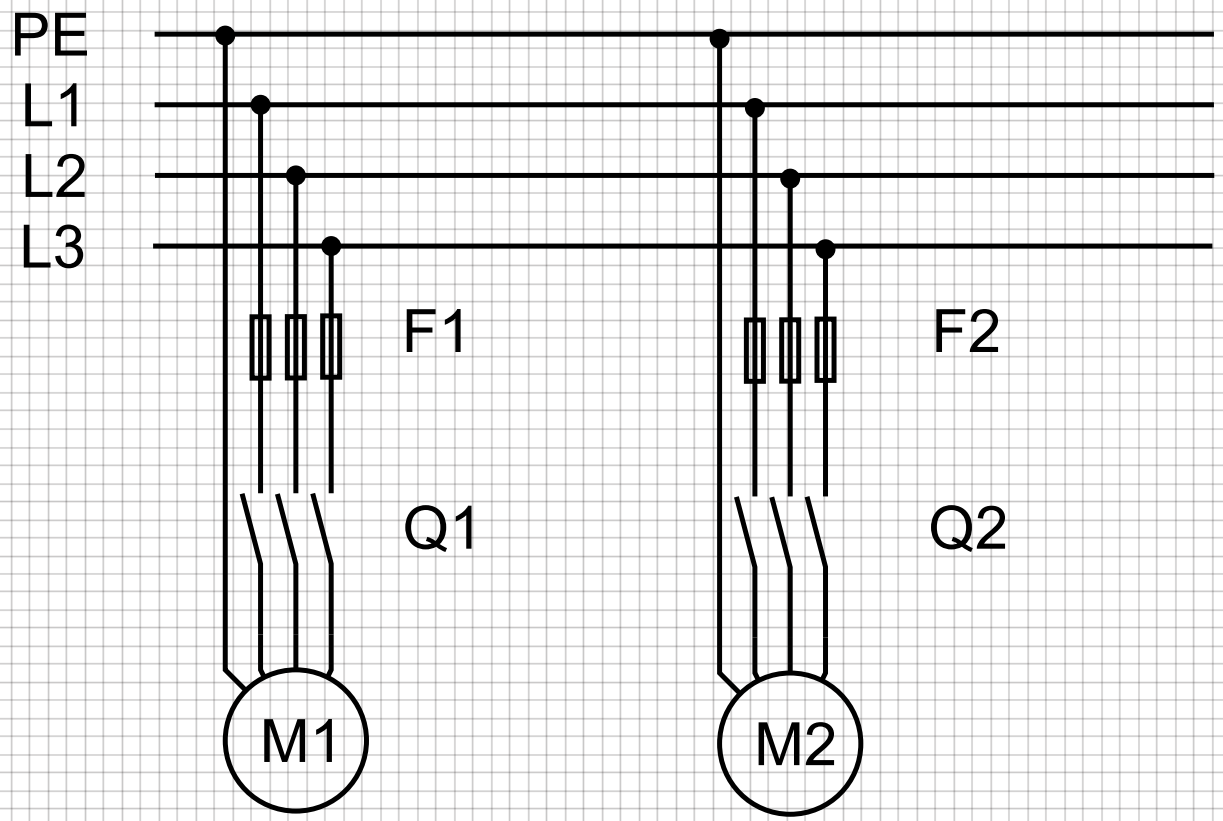
Obrázek: Princip a označení kontaktů

Příklad



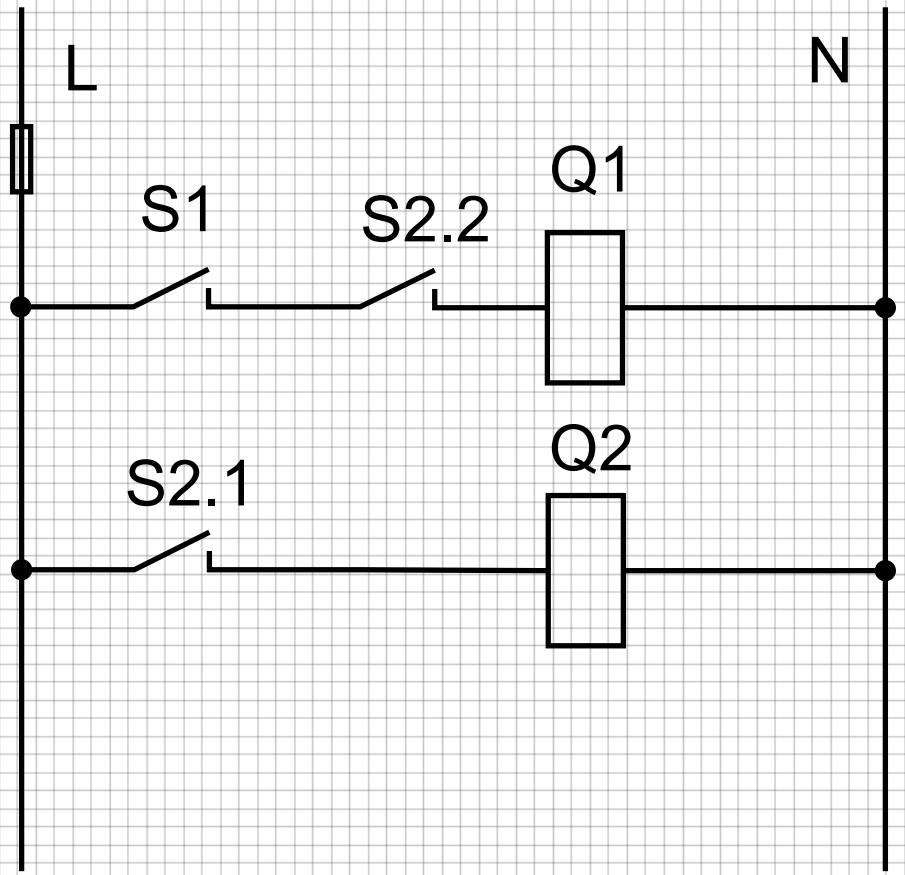
Obrázek: Ilustrativní příklad

Příklad



Obrázek: Zapojení výkonové

Příklad



Obrázek: Zapojení řízení

Festo

Bionic

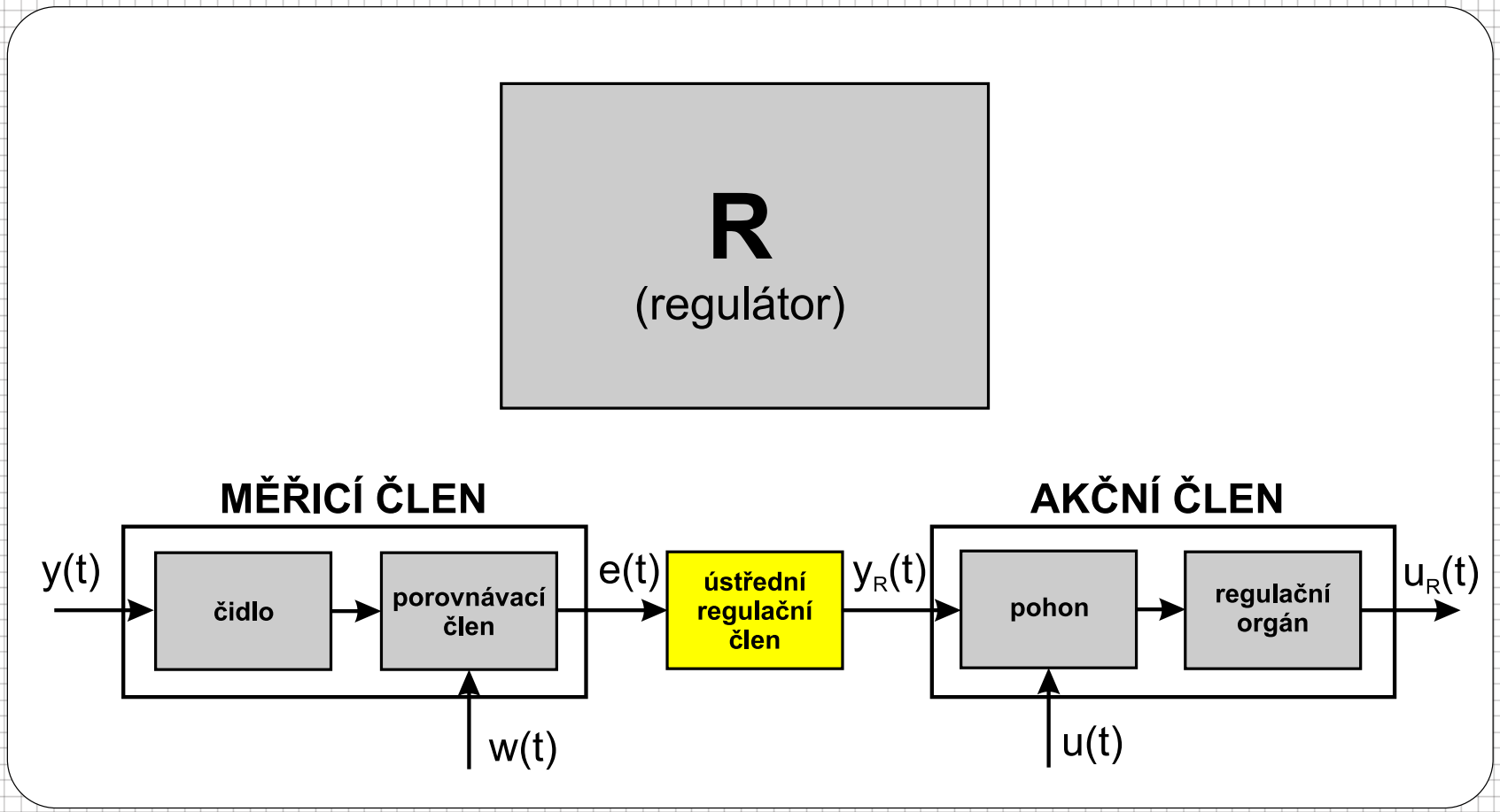
Kooperativní robot:
https://youtu.be/54u3H69tcgM
Různé typy robotů:
https://www.youtube.com/watch?v=7-JvyzOddTM
Pavouk:
https://www.youtube.com/watch?v=jGP5NxcCyjE

Komunikace

Sběrnice

- RS 232, RS 485, ProfiBus
- Ethernet
- I^2C, SPI, 1-wire

Části zpětnovazebního obvodu



Obrázek: Části regulačního obvodu

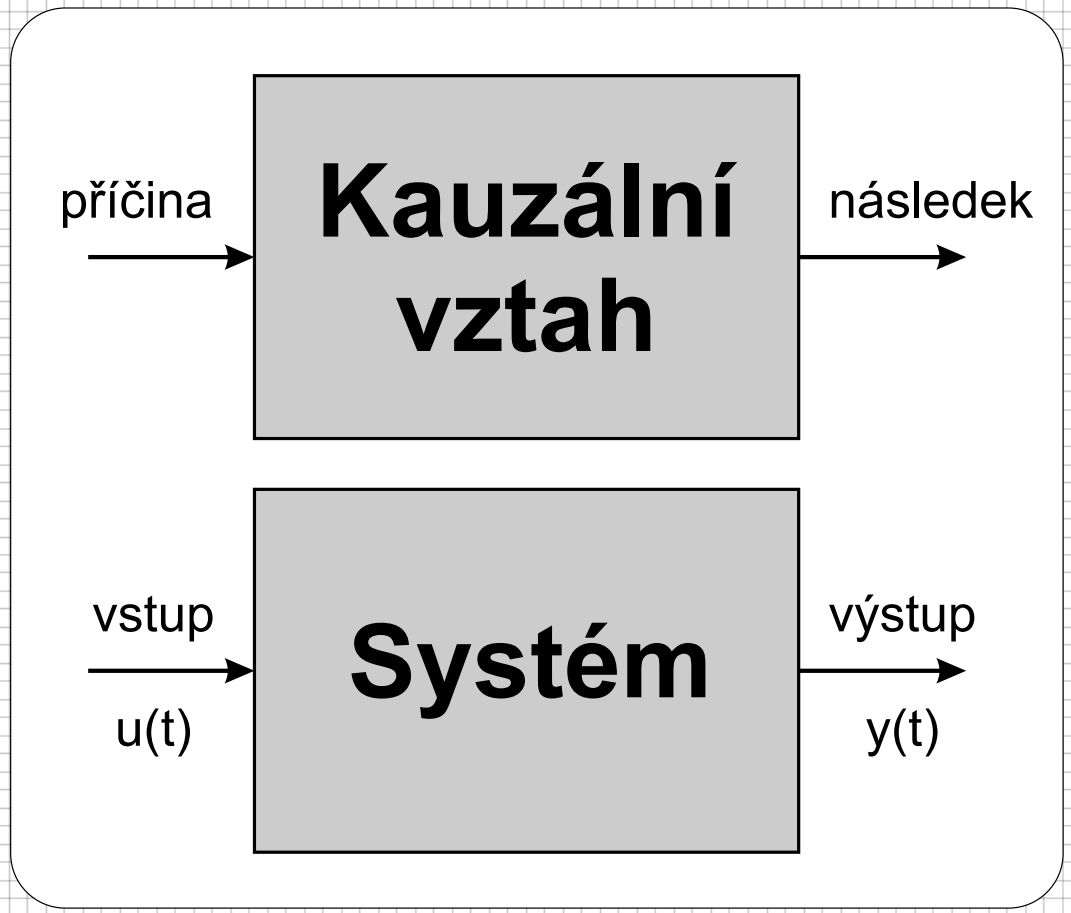
Řízení úzce souvisí s pochopením kauzálních - příčinných vztahů, které jsou charakteristické pro činnost (fungování) dotyčného systému (objektu). Pro správně navržené řízení daného systému je nezbytné správné pochopení kauzálních vztahů mezi jednotlivými částmi tohoto systému a také našich řídicích zásahů.
Tyto vztahy nazveme jako kauzální (orientovaná) relace.

Kauzalita

$$\text{příčina} \rightarrow \text{následek}$$

Šipkou je vyznačen neměnný „směr (orientace)“ závislosti. Směr nelze obecně otočit.

Graficky znázorňujeme kauzální relaci následovně:



Obrázek: Kauzalita

Algebra blokových schémat

Rozbor

V praxi bývají systémy spojené z mnoha podsystémů. Bloková algebra je soubor pravidel pro určení výsledného popisu systému na základě znalosti popisů jednotlivých podsystémů (vnitřních či vnějších).

Podmínky použití

Pro použití blokové algebry musí být splněny následující podmínky

- všechny členy v systému jsou lineární - platí princip superpozice
- signál se šíří pouze jedním směrem - vstupy neovlivňují zpětně výstupy (vysoký vstupní a nízký výstupní odpor)

Algebra blokových schémat

Algebra blokových schémat

K čemu je to dobré

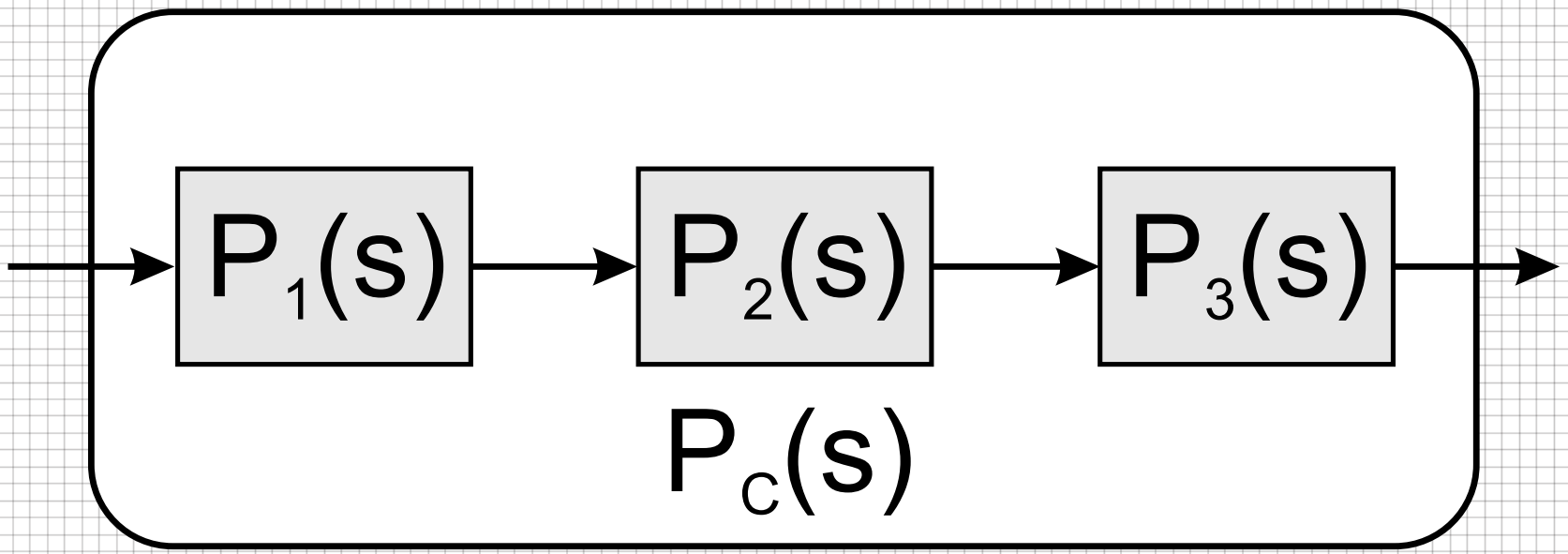
Znalost blokové algebry umožňuje zjednodušovat složitá bloková schémata a stanovit výsledný přenos zapojení.
Zjednodušovat je nejlépe tak, že uvnitř blokového schématu hledáme některé z uvedených základních zapojení a postupně nahrazujeme tato zapojení jediným členem. [Švarc, I. 1992]

Základní zapojení

- Sériové
- Paralelní
- Zpětnovazební (antiparalelní)

Předpoklady:

$$P_C(s)=P_C$$



Obrázek: Sériové spojení

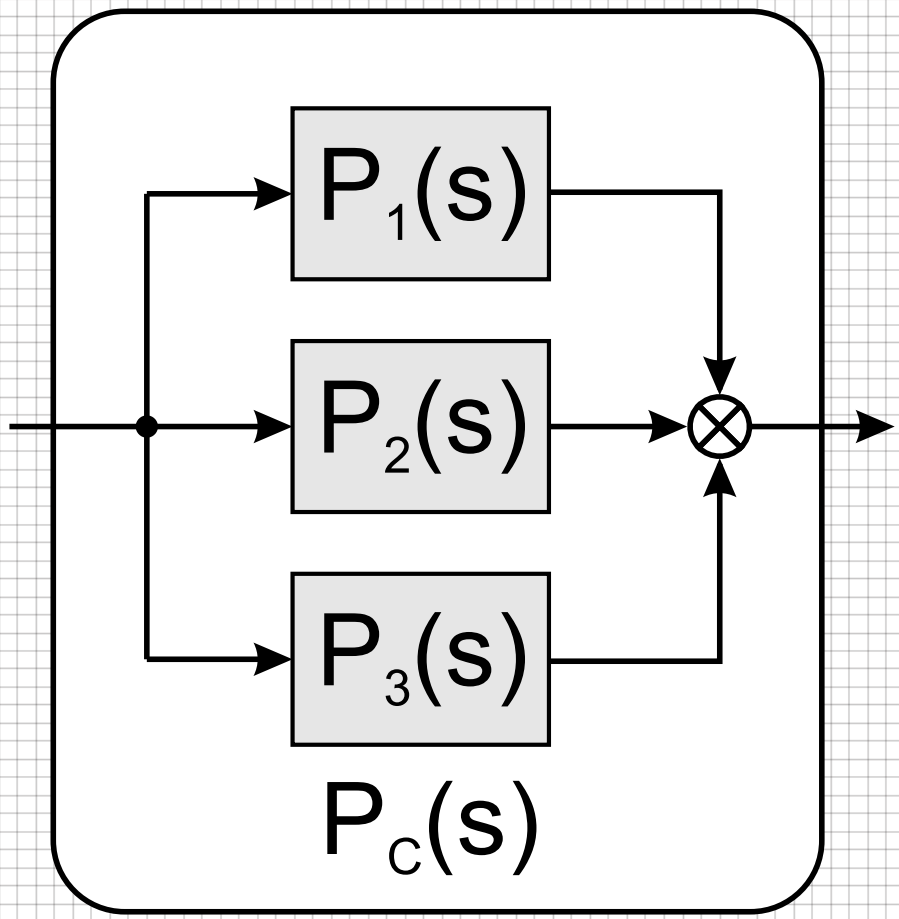
$$P_C(s) = P_3(s) \cdot P_2(s) \cdot P_1(s)$$

Algebra blokových schémat

$$P_C(s) = P_3(s) \cdot P_2(s) \cdot P_1(s)$$

$$P_C(s) = \prod_{i=1}^{n} P_i$$

Algebra blokových schémat



Obrázek: Paralelní spojení

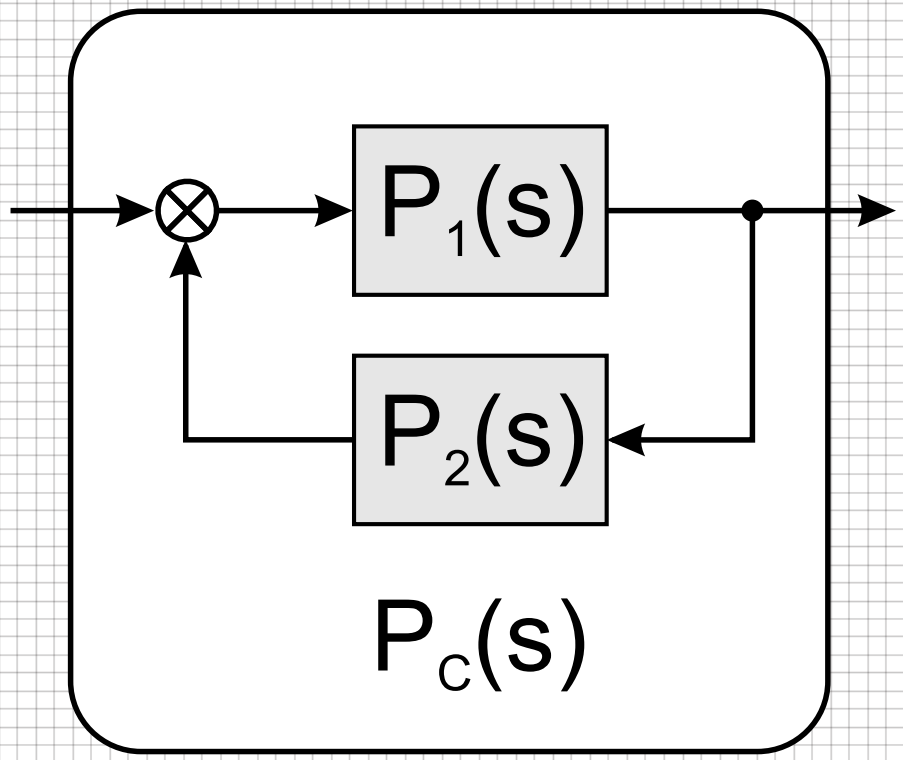
$$P_C(s) = P_3(s) + P_2(s) + P_1(s)$$

Algebra blokových schémat

$$P_C(s) = P_3(s) + P_2(s) + P_1(s)$$

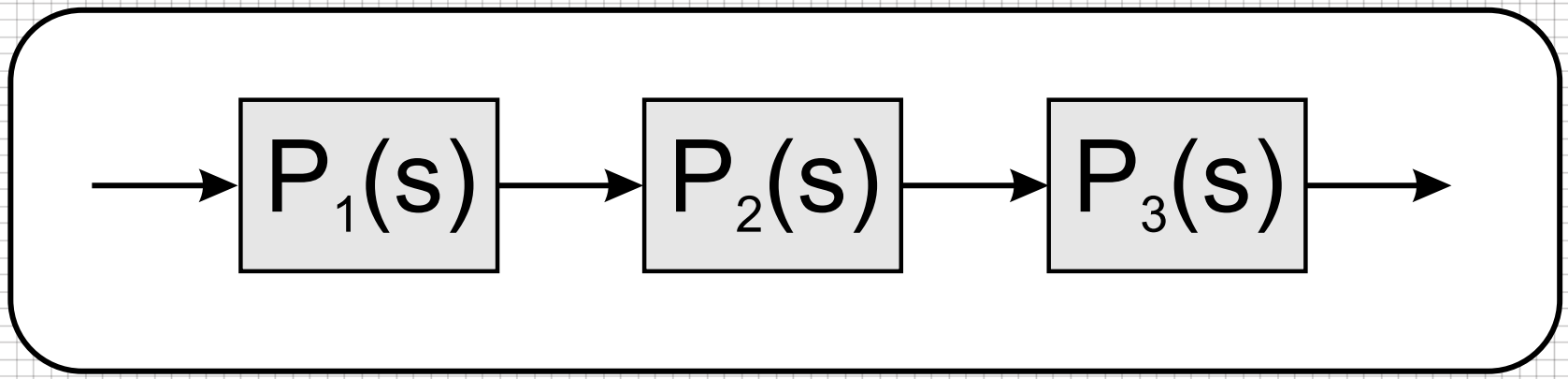
$$P_C(s) = \sum_{i=1}^{n} P_i$$

Algebra blokových schémat



Obrázek: Antiparalelní (zpětnovazební) spojení

$$P_C(s) = \frac{P_1(s)}{1 \mp P_1(s) \cdot P_2(s)}$$



Obrázek: Sériové spojení

Masonovo pravidlo pro výpočet přenosu

Přímá větev je jakákoliv posloupnost větví ze vstupu na výstup ve směru šipek, která neprochází žádným uzlem více než jednou. Přenos přímé větve je součin přenosů všech větví, které jsou obsaženy v dané přímé větvi. Přenos budeme označovat Vi, kde i je index přímé větve.
Smyčka je jakákoliv uzavřená posloupnost větví ve směru šipek, která neprochází žádným uzlem více než jednou.
Přenos smyčky je součin přenosů všech větví obsažených ve smyčce.
Říkáme, že se dvě smyčky dotýkají, pokud mají nějaký společný uzel. Jinak říkáme, že se nedotýkají. Stejně tak je to s dotýkáním či nedotýkáním u přímých větví.

Sample frame title

In this slide, some important text will be highlighted beause it's important. Please, don't abuse it.

Remark

Sample text

Important theorem

Sample text in red box

Examples

Sample text in green box. "Examples"is fixed as block title.