Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Aplikované vědy a informatika
Kybernetika a řídící technika

Vypracované otázky ke státní závěrečné zkoušce

(Ing.)

9. června 2017 **Martin Bulín, MSc.**

Kapitola 1

Umělá inteligence [UISZ]

1.1 Učící se systémy a klasifikátory [USK]

vyučující:	Prof. Ing. Josef Psutka, CSc.
ročník/semestr studia:	3.ročník/LS
datum zkoušky:	. 4. 2014
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními metodami klasifikace předmětů a jevů, které jsou reprezentovány svými obrazy (vektory příznaků). Výuka bude zaměřena na klasifikátory, které jsou trénovány s podporou učitele (supervised) anebo bez učitele (unsupervised).

1.1.1 Kritérium minimální chyby.

Často nejsme schopni posoudit jednoznačně, do které třídy vektor příznaků X patří. Cílem je potom nastavit klasifikátor tak, aby ztráty způsobené chybným rozhodnutím byly minimální.

Definition 1 *Ztráta, která vznikne, jestliže obraz náležející do třídy ω_s zařadí klasifikátor do třídy ω_r: $l(\omega_r|\omega_s)$*

- předp., že obrazový prostor X obsahuje obrazy z R tříd: $\omega_1, ..., \omega_R$
- apriorní ppsti výskytu obrazů náležejících ke třídě $\omega_r => p(\omega_r), \qquad r = 1, ..., R$
- podmíněná hustota ppsti obrazu x ze třídy ω_r je $p(x|\omega_r), \qquad r = 1, ..., R$
- nechť je dána matice ztrátových funkcí:

$$l = \begin{bmatrix} l(\omega_1|\omega_1) & \dots & l(\omega_1|\omega_R) \\ \vdots & \ddots & \vdots \\ l(\omega_R|\omega_1) & \dots & l(\omega_R|\omega_R) \end{bmatrix} \tag{1.1}$$

Předpokládejme, že na vstup klasifikátoru přicházejí x pouze z ω_s a klasifikátor je bude zařazovat do ω_r podle diskriminační funkce $\omega_r = d(x, q)$.

Definition 2 *Podmíněná střední ztráta (střední ztráta podmíněná výběrem obrazů výlučně ze třídy ω_s:*

$$J(q|\omega_s) = \int_X l[d(x,q)|\omega_s] \cdot p(x|\omega_s)\, dx \tag{1.2}$$

Protože jednotlivé třídy ω_s se vyskytují s ppstí $p(\omega_s)$, bude celková střední ztráta:

$$J(q) = \sum_{s=1}^{R} J(q|\omega_s) \cdot p(\omega_s) = \int_X \sum_{s=1}^{R} l[d(x,q)|\omega_s] \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx \tag{1.3}$$

Hledáme q^*, které minimalizuje $J(q)$:

$$\begin{aligned} J(q^*) = \min_q J(q) &= \int_X \min_q \sum_{s=1}^{R} l[d(x,q)|\omega_s] \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx = \\ &= \int_X \min_r \sum_{r=1}^{R} l(\omega_r|\omega_s) \cdot p(x|\omega_s) \cdot p(\omega_s)\, dx = \int_X \min_r L_x(\omega_r)\, dx \end{aligned} \tag{1.4}$$

Místo minima $J(q)$ hledáme minimum $L_x(\omega_r) = \sum_{r=1}^{R} l(\omega_r|\omega_s) \cdot p(x|\omega_s) \cdot p(\omega_s), \qquad r = 1, ..., R.$

Při klasifikaci podle funkce $L_x(\omega_r)$ by se postupovalo tak, že pro daný x by se vyčíslily všechny $L_x(\omega_r), r = 1, ..., R$ a obraz x by se přiřadil do té třídy ω_s, pro kterou by byla ztráta minimální. Je zřejmé, že různou volbou ztrátové funkce $l(\omega_r|\omega_s)$ dostáváme různý tvar rozhodovacího pravidla. Předpokládejme, že ztrátová funkce je zvolena tak, že při správném rozhodnutí přiřadí ztrátu 0 a při jakémkoliv špatném rozhodnutí ztrátu 1 (penalta 0/1).

$$l(\omega_r|\omega_s) = 1 - \delta_{rs}, \qquad \delta_{rs} = \begin{cases} 1 & r = s \\ 0 & r \neq s \end{cases} \tag{1.5}$$

Po dosazení:

$$\begin{aligned} L_x(\omega_r) &= \sum_{s=1}^{R} (1 - \delta_{rs}) p(x|\omega_s) \cdot p(\omega_s) = \sum_{s=1}^{R} p(x|\omega_s) \cdot p(\omega_s) - \sum_{s=1}^{R} \delta_{rs}\, p(x|\omega_s) \cdot p(\omega_s) \\ &= \sum_{s=1}^{R} [p(x|\omega_s) \cdot p(\omega_s)] - p(x|\omega_r) \cdot p(\omega_r) \end{aligned} \tag{1.6}$$

Platí známý Bayesův vztah:

$$\boxed{p(\omega_s|x) = \frac{p(x|\omega_s) \cdot p(\omega_s)}{p(x)}} \qquad , \tag{1.7}$$

kde $p(\omega_s|x)$ je aposteriorní pravděpodobnost, která vyjadřuje ppst třídy ω_s za předpokladu, že je na vstupu klasifikátoru obraz x.

$p(x|\omega_s)$ ... ppst x za předpokladu, že patří do ω_s

$p(\omega_s)$ ... apriorní ppst třídy ω_s

$p(x)$ ... ppst obrazu x (celková hustota funkce do obrazového prostoru)

$$\sum_{s=1}^{R} p(\omega_s|x) \stackrel{!}{=} 1 = \sum_{s=1}^{R} \frac{p(x|\omega_s) \cdot p(\omega_s)}{p(x)} => p(x) = \sum_{s=1}^{R} p(x|\omega_s) \cdot p(\omega_s) \tag{1.8}$$

Dosadíme: $L_x(\omega_r) = p(x) - p(x|\omega_r) \cdot p(\omega_r)$. Hodnota $p(x)$ je pro všechny třídy konstantní a jedná se v podstatě o aditivní konstantu, takže lze definovat novou funkci $L'_x(\omega_r) = p(x|\omega_r) \cdot p(\omega_r)$. Klasifikace zde probíhá tak, že se hledá takové zařazení ω_s, pro které je $L'_x(\omega_r)$ maximální:

$$\omega_r^* = \underset{r}{\operatorname{argmax}}\, p(x|\omega_r) \cdot p(\omega_r), \qquad r = 1, ..., R \tag{1.9}$$

1.1.2 Pravděpodobnostní diskriminační funkce. Souvislost s klasifikátory podle lineární diskriminační funkce, podle nejmenší vzdálenosti, podle nejbližšího souseda a podle k-nejbližšího souseda.

Kritérium minimální chyby se často označuje jako Bayesovo kritérium. Klasifikaci lze zajistit s využitím diskriminačních funkcí:

$$g'_r(x) = p(x|\omega_r) \cdot p(\omega_r), \qquad r = 1, ..., R \tag{1.10}$$

Klasifikátor pracující podle Bayesova kritéria se nazývá Bayesův klasifikátor. Pro jeho konstrukci je třeba znát hodnoty apriorní pravděpodobnosti a hustoty pravděpodobnosti pro každou třídu. Rozhodnutí, do které třídy neznámý obraz x patří, se provede podle hodnoty $g'_r(x)$ výběrem maxima.

Velmi často se místo diskriminační funkce $g'_r(x)$ používá její přirozený logaritmus:

$$g_r(x) = ln\, g'_r(x) = ln\, p(x|\omega_r) + ln\, p(\omega_r), \qquad r = 1, ..., R \tag{1.11}$$

Např. pro $R = 3$ a jednosložkový vektor x:

Předpokládejme, že obrazy v jednotlivých třídách vyhovují normálnímu rozložení (velmi častý případ). Pro obrazy v r-té třídě nechť platí:

$$\boxed{p(x|\omega_r) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot \sqrt{det\, C_r}} \cdot e^{-\frac{1}{2} \cdot (x-\mu_r)^T \cdot C_r^{-1} \cdot (x-\mu_r)}} \tag{1.12}$$

$\mu_r = E\{x\}_{x\in\omega_r}$ ... vektor středních hodnot obrazů r-té třídy

$C_r = E\{(x-\mu_r)\cdot(x-\mu_r)^T\}_{x\in\omega_r}$ ... kovarianční matice r-té třídy

Dosadíme do diskriminační funkce:

$$g_r(x) = ln\, p(x|\omega_r) + ln\, p(\omega_r) = -\frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(det\, C_r) - \frac{1}{2}\cdot(x-\mu_r)^T\cdot C_r^{-1}\cdot(x-\mu_r) + ln\, p(\omega_r) \tag{1.13}$$

Podle tvarů kovariančních matic C_r, hodnot μ_r a $p(\omega_r)$ dostáváme typické tvary diskriminačních funkcí.

1. **Obecné kovarianční matice** $C_r, r = 1, ..., R$

   Hyperplochy konstantních hodnot diskriminačních funkcí $g_r(x)$ jsou n-dimenzionální hyperelipsoidy (různě natočené a různě velké). Rozdělující hyperplocha mezi třídou ω_r a ω_s, tj. plocha, která je geometrickým místem shodných hodnot diskriminačních funkcí $g_r(x)$ a $g_s(x)$.

   $$\begin{aligned}\varphi_{rs}(x) &= g_r(x) - g_s(x) \stackrel{!}{=} 0 \\ &= -\frac{1}{2}\, ln\left[\frac{det\, C_s}{det\, C_r}\right] + ln\left[\frac{p(\omega_r)}{p(\omega_s)}\right] - \frac{1}{2}\cdot(x-\mu_r)^T\cdot C_r^{-1}\cdot(x-\mu_r) + \frac{1}{2}\cdot(x-\mu_s)^T\cdot C_s^{-1}\cdot(x-\mu_s)\end{aligned} \tag{1.14}$$

   Rozdělující hyperplochy mohou být podle tvaru C_r a C_s např. n-dimenzionální hyperroviny, hyperelipsoidy, hyperparaboloidy apod.

2. **Všechny třídy mají stejnou kovarianční matici** $C_r = C, \forall r = 1, ..., R$

   Shluky vzorků všech tříd vytvářejí stejně orientované a velké n-dimenzionální elipsoidy.

   $$g_r(x) = -\frac{1}{2}x^T C_r^{-1}x + \mu_r^T C_r^{-1}x - \frac{1}{2}\mu_r^T C_r^{-1}\mu_r + ln\, p(\omega_r) - \frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(det\, C) \tag{1.15}$$

   Plochy konstantních velikostí diskriminačních funkcí jsou n-rozměrné elipsoidy, které mají stejný tvar a jsou stejně orientovány. Rozdělující plochu $\varphi_{rs}(x)$ mezi třídami ω_r a ω_s lze vyjádřit:

   $$\begin{aligned}\varphi_{rs}(x) &= (\mu_r-\mu_s)^T C^{-1}x - \frac{1}{2}\mu_r^T C^{-1}\mu_r + \frac{1}{2}\mu_s^T C^{-1}\mu_s + ln\, p(\omega_r) - ln\, p(\omega_s) = \\ &= \varphi_{rsn}x_n + \cdots + \varphi_{rs1}x_1 + \varphi_{rs0} = \varphi_{rs}^T x + \varphi_{rs0}\end{aligned} \tag{1.16}$$

   Rozdělující plocha mezi třídami ω_r a ω_s je n-dimenzionální *rovina*. Jedná se tedy o n-dimenzionální *lineární diskriminační funkci*.

3. **Všechny třídy mají stejnou diagonální kov. matici** $C_r = C = \delta^2 I, r = 1, ..., R$

   Předpokladem je, že obrazy každé třídy mají statisticky nezávislé příznaky a každý příznak má stejnou varianci δ^2. Geometricky to odpovídá situaci, kdy vzorky každé třídy vytváří shluky tvaru n-dimenzionálních koulí centrovaných kolem příslušné střední hodnoty. Potom $det\, C = \delta^{2n}$ a $C^{-1} = \frac{1}{\delta^2}I$. Předpokládejme, že všechny třídy jsou stejně pravděpodobné,

tj. $p(\omega_r) = p(\omega), \forall r = 1, ..., R$. Potom:

$$g_r(x) = -\frac{1}{2\delta^2}||x - \mu_r||^2 + ln\, p(\omega) - \frac{n}{2}\, ln\, 2\pi - \frac{1}{2}\, ln(\delta^{2n}) = -k_1 \cdot ||x - \mu_r||^2 + k_2 \quad (1.17)$$

Konstanty $k_1 > 0$ a k_2 jsou shodné pro všechny třídy a výraz $||x - \mu_r||^2$ představuje kvadrát Euklidovské vzdálenosti mezi vektorem x a střední hodnotou ω_r. Klasifikátor zařadí neznámý obraz x do té třídy ω_r, pro kterou je $g_r(x)$ maximální. Z výrazu pro $g_r(x)$ vyplývá, že $g_r(x)$ bude tím větší, čím bude $||x - \mu_r||^2$ menší ($k_1 > 0$). Jedná se tedy v podstatě o *klasifikátor podle minimální vzdálenosti.*

Klasifikace podle minimální vzdálenosti

Diskriminační funkce: $g_r^*(x) = ||x - \mu_r||^2$. Klasifikátor zařadí neznámý obraz x do té třídy, pro kterou bude $g_r^*(x)$ minimální ($\omega_r^* = \min_r d^2(x, \mu_r)$). Není to určitě nejlepší klasifikátor, ale je tu velká lákavost ho používat, protože stačí jediný obraz na třídu. Rozdělující nadrovina má tvar:

$$\begin{aligned} \varphi_{rs}(x) &= -k1 \cdot ||x - \mu_r||^2 + k_2 - \left[-k_1 \cdot ||x - \mu_s||^2 + k_2\right] = \\ &= k_1 \cdot \left[x^T x - 2\mu_s x + \mu_s^T \mu_s - x^T x - 2\mu_r x + \mu_r^T \mu_r\right] \stackrel{!}{=} 0 \\ &=> (\mu_r - \mu_s)^T x - \frac{1}{2}(\mu_r^T \mu_r - \mu_s^T \mu_s) = 0 \end{aligned} \quad (1.18)$$

Rozdělující nadplochy mezi třídami jsou *lineární*, jsou to n-dimenzionální roviny kolmé na úsečku $\mu_r - \mu_s$, kterou půlí [1]. Tento klasifikátor je velmi jednoduchý na implementaci - pro jeho nastavení stačí získat střední hodnoty každé třídy a pro neznámý obraz x ve fázi klasifikace vypočítat vzdálenost ke všem středním hodnotám (též nazýván *klasifikátor se vzorovými etalony.* Pro svou jednoduchost je často nasazován i v případech, kdy není zabezpečena jeho optimální funkce podle kritéria minimální chyby (např. je málo početná trénovací množina nebo není znám typ rozložení nebo není známa disperzní matice ap.). To vede k negativním vlivům klasifikátoru:

- vzhledem k tomu, že využívá pouze střední hodnoty, nerespektuje tvar shluků jednotlivých tříd (pokud je odlišný od $C_r = C = \delta^2 I$; tvar shluku koresponduje s tvarem disperzní matice).
- nerespektuje případné odlišné apriorní pravděpodobnosti jednotlivých tříd

Dobrých výsledků dosáhneme, když budou třídy dobře distribuované (střední hodnoty dostatečně vzdálené, shluky dostatečně kompaktní a jednotlivé třídy stejně pravděpodobné).

Klasifikace podle nejbližšího souseda (Nearest Neighbour Classifier)

Uvedené nevýhody klasifikátoru podle minimální vzdálenosti lze zmírnit často tím, že využijeme více vzorových etalonů pro každou třídu. Zvolíme-li pro každou třídu ω_r S_r vzorových etalonů:

[1] Klasifikátor podle minimální vzdálenosti je ekvivalentní co do struktury lineárnímu klasifikátoru s R diskriminačními funkcemi, který může vytvořit až $\frac{R(R-1)}{2}$ rozdělujících nadrovin. Má však obecně jiné parametry, tj. klasifikuje obecně jiným způsobem než lineární klasifikátor.

$\mu_{r1}, \mu_{r2}, \ldots, \mu_{rS_r}$, pak klasifikace probíhá podle pravidla vyjádřeného vztahem:

$$\omega_r^* = \underset{s,r}{\operatorname{argmin}} ||x - \mu rs|| = \underset{s,r}{\operatorname{argmin}}\, d(x, \mu rs) \tag{1.19}$$

Obraz x se tedy zařadí do té třídy ω_r, jejíž některý etalon má mezi všemi ostatními etalony nejmenší vzdálenost od x. Tento způsob klasifikace má tu výhodu, že při dostatečně rozsáhlé trénovací množině se tvar rozdělujících funkcí pro jednotlivé třídy "blíží"Bayesovskému klasifikátoru. Na druhou stranu to však znamená značné zvýšení výpočetních nároků (klasifikátor si musí neustále pamatovat celou množinu vzorových etalonů - celou trénovací množinu) a při klasifikaci musíme počítat vzdálenost neznámého obrazu x ke všem vzorům.

Klasifikace podle k-nejbližších sousedů (k-Nearest Neighbour Classifier)

Lepších výsledků lze často dosáhnout využitím tzv. rozhodovacího pravidla, kdy nejprve vyčíslíme všechny vzdálenosti $||x-\mu_{rs}|| \forall r = 1, ..., R \forall s = 1, ..., S_r$ a pak je pro každou třídu ω_r uspořádáme tak, aby pro nový soubor $||x - \mu_{r[s]}||$ platilo:

$$||x - \mu_{r[1]}|| \leq ||x - \mu_{r[2]}|| \leq \cdots \leq ||x - \mu_{r[S_r]}||$$

Klasifikátor pak zařadí obraz x do třídy ω_r^* podle minima průměrné vzdálenosti k-nejbližších sousedů:

$$\omega_r^* = \underset{r}{\operatorname{argmin}} \frac{1}{k} \sum_{k=1}^{k} ||x - \mu_{r[i]}||, \qquad r = 1, ..., R \tag{1.20}$$

Nevýhody:

- musím si pamatovat všechny obrazy
- při každé klasifikaci náročné výpočty

1.1.3 Klasifikátor s lineární diskriminační funkcí. Klasifikace do dvou a do více tříd.

Pokud obrazy v jednotlivých třídách podléhají normálnímu rozložení a všechny třídy vykazují stejnou kovarianční matici C, je optimální nastavení klasifikátoru podle kritéria minimální chyby (Bayesova kritéria) zabezpečeno *lineárními diskriminačními funkcemi*. Vzhledem k jejich výhodným analytickým vlastnostem se jich ovšem využívá i v případech, kdy výše uvedené podmínky splněny nejsou a nebo, a to je častější případ, kdy ověření platnosti těchto podmínek je nepřiměřeně náročné (např. nelze statisticky prokázat typ rozložení vzhledem k malému počtu obrazů ap.). Zvolíme-li v takovém případě rozhodovací pravidlo založené na lineárních diskriminačních funkcích, musíme mít vždy na paměti, že jsme nezvolili optimální řešení s hlediska Bayesova kriteria minimální chyby. Přesto je třeba říci, že v případech, kdy obrazy jednotlivých tříd jsou dobře distribuované, tj. vytvářejí kompaktní shluky, které jsou od sebe dostatečně vzdálené (lineárně separabilní třídy) toto zjednodušení dostatečně vyhovuje.

Uvažme lineární diskriminační funkci $g(x) = q_0 + q_1 x_1 + \ldots q_n x_n = q_0 + \sum_{i=1}^{n} q_i x_i$, kde q_i jsou váhy funkce a q_0 je práh funkce.

Dále mějme $||q|| = \sqrt{q_1^2 + q_2^2 + \cdots + q_n^2}$. Pro $n = 2$:

Klasifikace do dvou tříd (dichotomie)

Při klasifikaci do dvou tříd ω_1 a ω_2 stačí k rozhodnutí jediná diskriminační funkce:

$$g(x) = q_0 + \sum_{i=1}^{n} q_i x_i \tag{1.21}$$

Pro $g(x) > 0$ je $x \in \omega_1$, pro $g(x) < 0$ je $x \in \omega_2$.

Klasifikace do více tříd

(a) Předpokládejme, že obrazy každé třídy jsou *lineárně separovatelné* od obrazů všech ostatních tříd. Pak diskriminační funkce mezi třídami ω_r a $\bar{\omega_r}$ je:

$$g_r(x) = q_{r,0} + \sum_{i=1}^{n} q_{r,i} x_i \tag{1.22}$$

a platí, že pro $x \in \omega_r$ je $g_r(x) > 0$ a pro $x \in \bar{\omega_r}$ je $g_r(x) < 0$. Klasifikátor pak rozhodne o zařazení x do té třídy $\omega_r (r = 1, ..., R)$ pro níž je diskriminační funkce $g_r(x) > 0$. Problém je však v tom, že se může stát, že pro neznámé x bude hodnota více než jedné diskriminační funkce větší než 0. V takovém případě klasifikátor není schopen rozhodnout [2].

Př. ($n = 2$, $R = 3$):

(b) Předpokládejme, že obrazy každé třídy jsou *po dvojicích lineárně separovatelné* od všech ostatních tříd. V tomto případě existuje celkově $\frac{R(R-1)}{2}$ diskriminačních funkcí $\varphi_{rs}(x); r, s =$

[2] Tento způsob klasifikace má jistá omezení, např. v prostoru dimenze $n = 2$ lze takto rozdělit maximálně $R = 3$ dobře distribuované třídy.

$1, ..., R \wedge r \neq s$, které vytvářejí rozdělující roviny mezi obrazy všech dvojic tříd. Pro obraz $x \in \omega_r$ pak platí $\varphi_{rs}(x) > 0 \, \forall s \neq r$, viz[3].

Př. ($n = 2$, $R = 4$):

(c) Předpokládejme[4], že existují diskriminační funkce $g_r(x), r = 1, ..., R$ z případu ad a). Vytvoříme rozdělující hyperplochy mezi třídami r a s.

$$\varphi_{rs}(x) = g_r(x) - g_s(x) \stackrel{!}{=} 0 \tag{1.23}$$

Pro $\varphi_{rs}(x) > 0$ je $g_r(x) > g_s(x)$. Z toho vyplývá, že klasifikátor zařadí x do ω_r, jestliže $g_r(x) > g_s(x) \, \forall s = 1, ..., R$; $s \neq r$. Viz poznámky[5].

Hodnocení: Je zřejmé, že případ ad a) není vhodný k aplikování vzhledem k vytváření rozsáhlých oblastí, ve kterých nejsme schopni provést jednoznačné přiřazení. Rozhodnutí mezi případem ad b) a ad c) závisí do značné míry na intuici (zvláště v prostorech vyšší dimenze). Obecně lze říci, že případ ad b) vyžaduje určení $\frac{R(R-1)}{2}$ diskriminačních funkcí $\varphi_{rs}(x)$, kdežto případ ad c) požaduje nalezení pouze R diskriminačních funkcí $g_r(x)$. Jestliže se však počet tříd R blíží dimenzi n obrazového prostoru nebo se očekává, že obrazy jednotlivých tříd jsou špatně distribuované, bude možná postup podle ad b) lepším řešením.

1.1.4 Metody nastavování klasifikátorů (trénování klasifikátorů).

(a) Známe-li celou trénovací množinu apriori, můžeme se pokusit o *analytické řešení:*

- *pravděpodobnostní diskriminační funkce*: je třeba určit a prokázat typ rozložení a hodnoty parametrů rozložení včetně apriorní pravděpodobnosti tříd

[3] Samozřejmě platí $\varphi_{rs}(x) = -\varphi_{sr}(x)$. V mnoha případech se nevyužívá všech $\frac{R(R-1)}{2}$ diskriminačních funkcí. V tomto případě se opět objevují oblasti, pro které nejsme schopni rozhodnout a zařazení x.

[4] Vylepšení ad a).

[5] Rozdělující funkce $\varphi_{rs}(x)$ rozdělují obrazový prostor bezezbytku (nejsou hluché oblasti, kde není možno provést přiřazení).

- *klasifikátor dle minima vzdálenosti*: je třeba určit střední hodnotu obrazů v každé třídě
- *klasifikátor podle nejbližšího či k-nejbližšího souseda*

Pro prostory vyšší dimenze nepoužitelné, navíc je třeba si pamatovat celou trénovací množinu. Dále není umožněno dotrénování klasifikátoru, pokud se objeví nové informace o trénovací množině. Častou chybou je nerespektování apriorních pravděpodobností tříd.

(b) Trénovací množinu neznáme apriori: *metody učení* Učící se klasifikátor má dvě fáze:

- *fáze učení*: postupně předkládány dvojice $[x_k, \Omega_k]$ z trénovací množiny, nastavujeme parametry q tak, aby pro $k \to \infty$ bude $q \to q^*$ (optimální nastavení, minimální střední hodnota ztrát).
- *fáze klasifikace*: využívá se zkušenosti nashromážděné v parametrech q k predikci neznámých obrazů. Klasifikátor se chová jako jednoúčelový automat.

Střední ztráta:

$$J(q) = \int_{X\times O} Q(x, \Omega, q)\, dF(x, \Omega) = \sum_{r=1}^{R} p(\omega_r) \int_X Q(x, \Omega, q) p(x|\omega_r)\, dx \tag{1.24}$$

Úkolem je najít takové q^*, které minimalizuje $J(q)$.

$$\operatorname*{grad}_q J(q^*) = \int_{X\times O} \operatorname*{grad}_q Q(x, \Omega, q^*)\, dF(x, \Omega) \stackrel{!}{=} 0 \tag{1.25}$$

Běžně ovšem neznáme distribuční (ani hustotní) funkce, proto se obracíme na rekurentní algoritmy, které obcházejí přímé řešení rovnice. Existují dva přístupy:

- metody učení založené na odhadování hustot ppsti
- metody učení založené na přímé minimalizaci ztrát

Metody učení založené na odhadování hustot ppsti

Snaha stanovit distribuční funkci $dF(x, \Omega)$ z trénovací množiny, poté se využije kritérium minimální chyby a dostáváme soustavu diskriminačních funkcí $g_r(x) = p(x|\omega_r) \cdot p(\omega_r)$. Hledáme odhady $\hat{p}(x|\omega_r)$ a $\hat{p}(\omega_r)$. Žádané vlastnosti:

- *nestrannost*: má zaručovat, že se odhad bude v průměru pohybovat kolem neznámé odhadované veličiny
- *konzistence*: s rostoucím počtem obrazů trénovací množiny se odhad blíží stále více k odhadované veličině
- *eficience*: eficientní odhad je odhad s nejmenší disperzí

Je-li trénovací množina vybrána nezávisle, lze odhad $\hat{p}(\omega_r)$ apriorní ppsti určit podle:

$$\hat{p}(\omega_r) = \frac{K_r}{K}, \qquad r = 1, ..., R \tag{1.26}$$

kde K_r je počet obrazů trénovací množiny, které patří do třídy ω_r a K je celkový počet obrazů v trénovací množině. Podle velikosti apriorní informace o hledané hustotě rozdělujeme metody získávání odhadů hustotní funkce na *parametrické* a *neparametrické*.

1. *Parametrické metody*: známe informaci o tvaru hustotní funkce $p(x|\omega_r)$, ale neznáme parametry q, který rozložení blíže popisuje (např. u Gausse: μ_r a δ_r).

   - *Metoda momentů*

     Teoretické momenty náhodných veličin se porovnávají s výběrovými momenty do takového stupně l, kolik je neznámých parametrů.

     výběrový průměr: $\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x_k$

     výběrový rozptyl: $S = \frac{1}{K-1}\sum_{k=1}^{K}(x_k - \bar{x})(x_k - \bar{x})^T$

     l-tý výběrový moment: $M_l = \frac{1}{K}\sum_{k=1}^{K} x_k^l$

   - *Metoda nejlepších nestranných lineárních odhadů*

     Odhad parametrů q se uvažuje jako lineární funkce obrazů $x_1, ..., x_K$: $\hat{q} = c_1 x_1 + ... + c_K x_K$. Koeficienty $c_k, k = 1, ..., K$ se určují z podmínek nestrannosti a minimální disperze odhadu $\hat{q}$. Platí $E\,\hat{q} = q$ a odhad $\hat{q}$ parametru q je eficientní, jestliže minimalizuje stopu disperzní matice $tr\,D\,\hat{q}$.

   - *Metoda maximální věrohodnosti*

     Metoda je založena na maximalizaci tzv. Fisherovy funkce věrohodnosti:

     $$L(x_1, ..., x_K|q) = \prod_{k=1}^{K} p(x_k|q) \tag{1.27}$$

     Hledáme takový parametr q, pro který je funkce maximální. Místo tohoto maxima lze hledat maximum logaritmu této věrohodnostní funkce $\operatorname{grad}_q ln\, L(x_1, ..., x_K|q) \stackrel{!}{=} 0$. Např. pro normální rozdělení je nejlepším odhadem střední hodnoty aritmetický průměr $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ a nejlepším odhadem kovarianční matice je:

     $$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T \tag{1.28}$$

2. *Neparametrické metody*: máme nulovou apriorní informaci, musíme odhadovat celý tvar hustotní funkce.

   - *Metoda histogramu*

Obrazový prostor rozdělíme na L disjunktních podmnožin $A_l, l = 1, ..., L$ (obvykle to jsou n-rozměrné intervaly). Symbolem c_l označíme počet obrazů x_k, které padnou do A_l, dělený číslem K. Za odhad $\hat{p}(x|\omega_r)$ pak bereme po částech konstantní funkci, která na intervalu A_l nabývá c_l. Nevýhodou je nutnost znalosti apriorního rozdělení obrazového prostoru na intervaly před fází učení. Odhad hustotní funkce:

$$\hat{p}(x|\omega_r) = \sum_{l=1}^{L} c_l \cdot \varphi_l(x), \qquad \varphi_l = \begin{cases} 1 & x \in A_l \\ 0 & jinak \end{cases} \tag{1.29}$$

Metody učení založené na přímé minimalizaci ztrát

Cílem je navrhnout parametry klasifikátoru, které budou minimalizovat ztrátu, rekurentním vypočítáváním odhadů $\hat{q}(0) \to \hat{q}(1) \to \cdots \to \hat{q}^*$.

$$\operatorname*{grad}_q J(q^*) = \int_{X \times O} \operatorname*{grad}_q Q(x, \Omega, q^*)\, dF(x, \Omega) = 0 \tag{1.30}$$

Nabízí se využití gradientních metod. V praktických úlohách neznáme distribuční funkci a nelze tak vyčíslit grad $J(q)$. Navíc, při pevném q hodnota funkce $Q(x, \Omega, q^*)$ náhodně kolísá v závislosti na x a Ω - to je u gradientních metod nepoužitelné. Řešení rovnice hledáme pomocí metody *stochastických aproximací*. Základní algoritmus přímé minimalizace ztrát:

$$q(k+1) = q(k) - C_{k+1} \operatorname*{grad}_q Q[x(k+1), \Omega(k+1), q(k)] \qquad k = 1, ..., K \tag{1.31}$$

C_k je čtvercová matice (obvykle $C_k = c_k \cdot I$), $[x(k), \Omega(k)]$ je k-tá dvojice trénovací množiny a q je vektor parametrů. Vhodnou volbou $Q(x, \Omega, q)$ a C_k lze získat téměř všechny algoritmy přímé minimalizace ztrát.

Položíme $x_0 = 1$: $x^T = [x_0, x_1, ..., x_n]$ a váhový vektor rozšíříme o práh q_0: $q^T = [q_0, q_1, ..., q_n]$. Diskriminační funkce má tvar $g(x) = q^T \cdot x$ a rozhodovací pravidlo $\omega = \operatorname{sign} g(x)$. Po zavedení pásma necitlivosti δ volíme:

$$\operatorname*{grad}_q Q(x, \Omega, q) = \begin{cases} 0 & q^T x \Omega \geq \delta \\ -x\Omega & q^T x \Omega < \delta \end{cases} \tag{1.32}$$

Dosadíme-li do vztahu pro rekurentní výpočet q, dostaneme algoritmus učení:

$$\begin{aligned} q(k+1) &= q(k) - C_{k+1} \operatorname*{grad}_q Q[x(k+1), \Omega(k+1), q(k)] = \\ &= \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq 0 \\ q(k) + C_{k+1}x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < 0 \end{cases} \end{aligned} \tag{1.33}$$

(a) *Rosenblattův algoritmus*

$C_k = C_{k+1} = 1$, $\delta = 0$:

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq 0 \\ q(k) + x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < 0 \end{cases} \tag{1.34}$$

Nedostatkem je, že nikdy nezjistíme absolutně přesný klasifikátor.

(b) *Metoda konstantních přírůstků*

$C_k = \frac{\beta}{||x(k)||^2}$, $\beta > 0$

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq \delta \\ q(k) + \frac{\beta}{||x(k+1)||^2}x(k+1)\Omega(k+1) & q^T(k)x(k+1)\Omega(k+1) < \delta \end{cases} \quad (1.35)$$

(c) *Upravená metoda konstantních přírůstků*

Připočítáváme vektor $\frac{\beta}{||x(k+1)||^2}x(k+1)\Omega(k+1)$ tolikrát, až pro určitý obraz $x(k+1)$ dosáhneme správné klasifikace, tj. bude platit $q^T(k+1)x(k+1)\Omega(k+1) \geq \delta$. Může se stát, že nikdy nedojdeme k řešení.

(d) *Relaxační metoda učení*

$$q(k+1) = \begin{cases} q(k) & q^T(k)x(k+1)\Omega(k+1) \geq \delta \\ q(k) + 2C_{k+1}x(k+1)\Omega(k+1)[\delta - q^T(k)x(k+1)\Omega(k+1)] & q^T(k)x(k+1)\Omega(k+1) < \delta \end{cases} \quad (1.36)$$

kde $c_k = \frac{\sigma}{2||x(k)||^2}$ a $\sigma \in (0, 2)$.

1.1.5 Metody shlukové analýzy (učení bez učitele).

Někdy se stane, že je k dispozici jen trénovací množina bez údajů o správné klasifikaci, někdy chybí i informace o počtu tříd. Úkolem je nalézt shluky obrazů, tj. takové skupiny jejichž prvky jsou si vzájemně podobné (geometricky blízké). Lze aplikovat pouze na data, kde ty shluky opravdu jsou.

Požadavky pro míry podobnosti: $d(x_i, x_i) = 0$; $d(x_i, x_j) \geq 0$, $i \neq j$; $d(x_i, x_j) = d(x_j, x_i)$.

Míry podobnosti mezi dvěma obrazy x a x':

- $d(x, x') = ||x - x'||$
- Euklidova vzdálenost: $d(x, x') = \sqrt{\sum_{i=1}^{n}(x_i - x'_i)^2}$

Míry podobnosti mezi dvěma shluky T_i a T_j:

- minimální míra: $D_{min}(T_i, T_j) = \min_{x \in T_i; x' \in T_j} d(x, x')$
- maximální míra: $D_{max}(T_i, T_j) = \max_{x \in T_i; x' \in T_j} d(x, x')$
- průměrná míra: $D_{mean}(T_i, T_j) = \frac{1}{s_i \cdot s_j} \sum_{x \in T_i} \sum_{x' \in T_j} d(x, x')$

- centroidní míra: $D_c(T_i, T_j) = ||\frac{1}{s_i}\sum_{x\in T_i} x - \frac{1}{s_j}\sum_{x'\in T_j} x'|| = d(\bar{x}, \bar{x}')$

Nehierarchické shlukovací metody

Snaha provést optimální rozklad dané množiny T do předem známého počtu R shluků. Hledáme takové rozdělení, pro které nabývá kritérium J extrém.

$$J = \sum_{i=1}^{R} J_i = \sum_{i=1}^{R}\sum_{x\in T_i} d^2(x, \mu_i) \tag{1.37}$$

1. **McQueenova metoda k-means (1967)** Algoritmus se snaží minimalizovat celkový ukazatel jakosti J rozdělením obrazů do shluků.

   Dáno:

   $T = \{x_i\}_{i=1}^{N}$ ... obrazy trénovací množiny

   R ... počet tříd

   $T_i(k)$ ... množina obrazů i-tého shluku v k-tém kroku algoritmu

   $\mu_i(k)$ ... střední (průměrná) hodnota i-tého shluku v k-tém kroku

   $J_i(k)$ ... hodnota kritéria i-tého shluku v k-tém kroku

   $s_i(k)$ ... počet obrazů ve shluku T_i v k-tém kroku

   Postup:

   (i) Zvol R počátečních středů shluků $\mu_1(1), ..., \mu_R(1)$.

   (ii) V k-tém iteračním kroku se rozdělují obrazy trénovací množiny do R shluků $T_1(k), ..., T_R(k)$ podle vztahu $x \in T_j(k)$ jestliže $d(x, \mu_j(k)) < d(x, \mu_i(k))\, \forall i, j = 1, ...R;\ i \neq j$. Vztah je postupně aplikován pro všechny obrazy z množiny T.

   (iii) Z výsledků předchozího kroku vypočti pro každý shluk nový střed, tj. $\mu_j(k+1)$, $j = 1, ..., R$ klasickým zprůměrováním (zaručí, že $J_j(k+1$ bude minimální):

   $$\mu_j(k+1) = \frac{1}{s_j(k)}\sum_{x\in T_j(k)} x, \qquad j = 1, ..., R \tag{1.38}$$

   (iv) Jestliže $\mu_j(k+1) = \mu_j(k)\, j = 1, .., R$, algoritmus dokonvergoval a procedura je ukončena. Jinak jdi na krok (ii). Alternativně lze proces ukončit v případě, že pokles hodnoty kriteriální funkce je již nevýznamný.

   Funkce metody je ovlivněna zejména specifickým počtem shluků a volbou počátečních středů shluků. Metoda sice pro vhodná data poskytuje přijatelné výsledky, ale dosažení globálního minima ukazatele jakosti procesu shlukování není zaručeno (konvergence nejčastěji končí v nějakém lokálním minimu).

2. **Iterativní optimalizace** V k-tém iteračním kroku změníme zařazení jednoho obrazu.

   Postup:

   (i) Proveď počáteční rozklad N obrazů do R shluků a vypočti hodnotu kriteriální funkce J a urči $\mu_i(1)$, $i = 1, ..., R$.

   (ii) Vyber obraz $\hat{x}$, kde např. $\hat{x} \in T_i$ (nejlépe nějaký systematický výběr).

   (iii) Jestliže $s_i(k) = 1$, jdi na bod (vi)

   (iv) Vypočti:

$$\begin{aligned} A_i &= \frac{s_i(k)}{s_i(k)-1} d^2(\hat{x}, \mu_i(k)) \\ A_j &= \frac{s_j(k)}{s_j(k)+1} d^2(\hat{x}, \mu_j(k)), \qquad \forall j \neq i \end{aligned} \tag{1.39}$$

   Urči $j^* = \underset{j}{\operatorname{argmin}} A_j$. Jestliže $A_i > A_{j^*}$, přesuň $\hat{x}$ do T_{j^*}. Jinak ponech $\hat{x}$ v T_i.

   (v) Aktualizuj J, μ_i a μ_{j^*}.

   (vi) Jestliže se J po pokusu přesunout postupně všech N obrazů trénovací množiny nezměnila, proces ukonči. Jinak přejdi do bodu (ii).

   Pokud bychom chtěli dosáhnout opravdu globálního minima J, mohli bychom vyzkoušet teoreticky všechny možnosti rozkladu, ale existuje obecně $\frac{R^N}{R!}$ možností (pro $R = 3$ a $N = 100$ je to $\approx 10^{47}$ možností).

Hierarchické shlukovací metody

Většinou aplikujeme, pokud není známa informace o počtu tříd.

(A) *Aglomerativní přístup* Vycházíme z jednotlivých obrazů, které postupně shlukujeme, až dojde ke spojení všech obrazů do jedné množiny.

1. **Metoda shlukové hladiny** Celé množině obrazů $T = \{x_1, ..., x_N\}$ postupně přiřazujeme posloupnost rozkladů B_0 až B_{N-1} a každému vzniklému shluku A přiřazujeme tzv. shlukovací hladinu $h(A) \geq 0$.

   Postup:

   (i) Rozklad B_0 tvoří jednotlivé obrazy, tj. $A_{0j} = \{x_j\}$ a $h(A_{0j}) = 0$, $j = 1, ..., N$.

   (ii) V i-tém kroku pro $1 \leq i \leq N-1$ vybereme jedinou dvojici shluků (obrazů) $A_{i-1,u}$ a $A_{i-1,v}$, pro kterou nastala nejmenší vzdálenost (ve smyslu zvolené metriky).

   - provedeme sjednocení shluků $A_{i-1,u} \cup A_{i-1,v} < A_{i,l}$, $l = 1, ..., N-1$;
   - utvoříme rozklad $B_i = \{A_{i,1}, ..., A_{i,N-1}\}$, kde všechny shluky s výjimkou sjednoceného $A_{i,l}$ přechází do B_i beze změny;
   - položíme $A_{i,l} = \min_{u,v} D(A_{i-1,u}, A_{i-1,v})$

   (iii) Bod (ii) opakujeme, až v posledním kroku procedury vznikne jediný shluk.

(B) *Divisní přístup* Celkový shluk obrazů určený ke klasifikaci postupně rozdělujeme a získáme hierarchický systém podmnožin.

1. **Jednoprůchodový heuristický algoritmus hledání shluků** Je zadána trénovací množina $T = \{x_1, ..., x_N\}$. Definujeme ve vztahu se zvolenou mírou podobnosti neznámý práh $t > 0$.

   (i) Z obrazů trénovací množiny vybereme jeden, např. $x_{(1)} \in T$ a ztotožníme ho se středem prvního shluku $\mu_1 = x_{(1)}$.

   (ii) Z trénovací množiny vybereme obraz $x_{(2)} \in T$ a vypočteme vzdálenost $d_{21} = d(x_{(2)}, \mu_1)$. Jestliže $d_{21} > t$, pak zavedeme nový shluk se středem $\mu_2 \coloneqq x_{(2)}$. Jinak přidělíme obraz $x_{(2)}$ do shluku se středem μ_1.

   (iii) Předpokládáme, že shluk se středem μ_2 byl zaveden, poté vybereme obraz $x_{(3)} \in T$ a vyčíslíme $d_{31} = d(x_{(3)}, \mu_1)$ a $d_{32} = d(x_{(3)}, \mu_2)$. Jestliže $d_{31} > t$ a $d_{32} > t$, zavedeme nový shluk se středem $\mu_3 \coloneqq x_{(3)}$. Jinak přidělíme obraz $x_{(3)}$ do shluku, k jehož středu má nejmenší vzdálenost.

   (iv) Postupujeme analogicky, až rozdělíme všechny obrazy trénovací množiny.

   Výsledky závisí na prvním vybraném středu shluku; pořadí, v němž jsou obrazy uvažovány; hodnotě t; geometrických vlastnostech dat. Algoritmus je velmi jednoduchý a rychlý a stanovuje první náhled na trénovací množinu obrazů. Podle velikosti prahu t se mění i výsledný počet shluků. Metoda vyžaduje pouze jediný průchod trénovací množinou, ale v praxi je potřeba rozsáhlé experimentování s hodnotou prahu a startovacím obrazem.

2. **Metoda řetězové mapy** Základním krokem při vytváření řetězové mapy je přeuspořádání dat. Z trénovací množiny vybereme libovolný "startovací"obraz $x_{(1)}$, najdeme jeho nejbližšího souseda $x_{(1)[1]}$ a položíme $x_{(2)} = x_{(1)[1]}$ (druhá položka seznamu). Další položkou bude nejbližší soused k $x_{(2)}$, atd. Nejbližšího souseda vždy vybíráme z množiny dosud neuspořádaných obrazů. Proces pokračuje, dokud nezískáme posloupnost ze všech obrazů trénovací množiny.

   $$\tilde{T} = \{x_{(1)}, ..., x_{(N)}\} \tag{1.40}$$

   S i-tým členem této posloupnosti potom spojíme vzdálenost $d_i = d(x_{(i),x_{i+1}}$, $i = 1, ..., N-1$. Řetězovou mapu získáme jako závislost $d_i = f(i)$. Znázorníme-li tuto závislost graficky, pak pokud jsou obrazy trénovací množiny dobře distribuovány, tj. vytvářejí v prostoru kompaktní shluky, které jsou od sebe dostatečně vzdálené, lze typicky na diagramu nalézt souvislé oblasti relativně malých hodnot d_i, které jsou od sebe odděleny (relativně) vyšší hodnotou d_j. Tyto zvýšené hodnoty d_j odpovídají hranicím shluků.

3. **Metoda MAXIMIN (maximum-minimum)** Jedná se o jednoduchý heuristický algoritmus, který je vhodný pro odhad počtu shluků v obrazovém prostoru. Je dána množina obrazů $T = \{x_k\}_{k=1}^{N}$ a volitelná konstanta $q > 0$.

   (i) Z trénovací množiny vybereme "startovací"obraz a ztotožníme ho se středem prvního shluku, tj. $\mu_1 \coloneqq x_{(1)}$.

(ii) Dále zvolíme (ve smyslu zvolené metriky) nejvzdálenější obraz k μ_1 a ztotožníme ho se středem druhého shluku, tj. $\mu_2 \coloneqq x_{(2)}$.

(iii) Pro každý obraz z trénovací množiny (bez $x_{(1)}$ a $x_{(2)}$) vyčíslíme vzdálenosti k μ_1 a k μ_2 a vždy vybereme tu minimální.

(iv) Z uchovaných minimálních vzdáleností vybereme tu maximální a porovnáme její velikost s velikostí vzdálenosti $d(\mu_1, \mu_2)$ a obraz, pro který tato maximální vzdálenost nastala, označíme jako $x_{(3)}$.

(v) Jestliže je ta maximální vzdálenost větší než $q \cdot d(\mu_1, \mu_2)$, zavedeme nový shluk se středem $\mu_3 \coloneqq x_{(3)}$, jinak shlukovací proces končí.

(vi) V dalším kroku algoritmu postupujeme stejně jako v kroku (iii), ale pro tři středy shluků. To znamená, že z minimálních vzdáleností obrazů trénovací množiny $(T \setminus \{x_{(1)}, x_{(2)}, x_{(3)}\})$ ke středům shluků μ_1, μ_2 a μ_3 vybereme maximální a opět posoudíme, je-li větší než q-násobek např. průměru vzdáleností mezi již vytvořenými středy shluků. Postup opakujeme pro vzrůstající počet (středů) shluků, dokud se nová maximální vzdálenost nedostane do sporu s podmínkou vytvoření nového středu shluku.

Po nalezení počtu shluků a jejich středů lze provést rozdělení obrazů trénovací množiny do jednotlivých shluků podle kritéria minimální vzdálenosti.

4. **Metody binárního dělení** Předpokládejme, že jsme v procesu shlukování získali R shluků. V dalším postupu obvykle provádíme zařazování neznámých obrazů do nejvhodnějších shluků podle pravidla nejbližšího souseda → musíme nalézt takový shluk, jehož centroid má k neznámému obrazu x_i nejmenší vzdálenost. Při vyčerpávajícím porovnání se všemi centroidy narážíme při velkém R na velké množství operací. Metoda binárního dělení umožňuje redukci množství operací.

   - **Rovnoměrné binární dělení** Požadovaný počet výsledných shluků R musí být mocninou 2. V prvním kroku dělení je prvotní shluk obrazů trénovací množiny rozdělen na dva subshluky a každý z těchto subshluků je poté v dalším kroku rozdělen na další dva subshluky, atd. Proces končí, až je dosažen požadovaný konečný počet shluků R. Platí $R = 2^B$, kde B je počet kroků dělení. Proces lze znázornit prohledávacím stromem. Při rovnoměrném dělení se mechanicky rozdělují všechny koncové shluky (reprezentované v prohledávacím stromu aktuálními koncovými uzly). Při tomto procesu se může stát, že v některém subshluku zůstane několik nebo dokonce jen jediný obraz → další dělení takových shluků výrazně nepřispívá k minimalizaci celkového zkreslení.

   - **Nerovnoměrné binární dělení** Zde výsledný počet shluků R nemusí být mocninou 2. Abychom zajistili nižší celkové zkreslení při daném počtu shluků, je vhodné nedělit prohledávací strom rovnoměrně, ale v každém kroku subdělícího procesu vyčíslit zkreslení všech koncových shluků a rozdělit ten shluk, který přispívá do celkového zkreslení největší měrou. Po každém kroku algoritmu, který měl za následek rozdělení nějakého subshluku zjistíme, zda již nebylo dosaženo stanoveného počtu shluků nebo zda velkost celkového zkreslení nepoklesla pod stanovenou mez. Pokud některá z těchto podmínek byla splněna, proces dělení končí.

1.1.6 Výběr informativních příznaků.

Velikost vektorů příznaků všech vzorků uvažovaných pro stejnou úlohu musí být shodná. Pokud není, lze přemýšlet o úpravě těchto vektorů, která obecně velmi závisí na fyzikální podstatě jednotlivých jevů (příznaků). Pokud to fyzikální podstata dovoluje, můžeme řetězce lineárně natáhnout, popř. lineárně smrštit, na požadovanou velikost. Plyne-li z fyzikální podstaty vektorů příznaků možnost lineárního kolísání (nejčastěji v časové ose), lze pro výpočet vzdálenosti takových vektorů využití metody tzv. nelineárního borcení jednoho z pozorovaných obrazů (DTW - *dynamic time warping*).

Obecně při výběru informativních příznaků je třeba zachovat minimální hodnoty disperzí uvnitř jednotlivých tříd a maximální hodnoty disperzí mezi třídami.

Extrakce

Metoda extrakce výběru minimálního počtu informativních příznaků je založena na nejlepší aproximaci původních obrazů z prostoru H_m o dimenzi m obrazy z prostoru H_n o dimenzi $n < m$, a to ve smyslu minima střední kvadratické odchylky (Karhunen-Loevův rozvoj).

Selekce

Definujeme míru diskriminativnosti množiny příznaků $J = \mathrm{tr}(\hat{T}^{-1}_{AKVI} \cdot \hat{T}_{AKT})$. Ta se rovná největšímu vlastnímu číslu matice $T^{-1}_{AKVI} \cdot T_{AKT}$. Výběr příznaků založený na míře diskriminativnosti se provádí iterativně:

(i) Vypočte se míra diskriminativnosti pro celý soubor příznaků.

(ii) Vypočte se míra diskriminativnosti při vynechání jednoho příznaku. Tento krok se provede n-krát (vynechávají se postupně příznaky $v_1, ..., v_n$)

(iii) Z původního souboru příznaků vyloučíme definitivně ten příznak, při jehož vynechání v kroku (ii) došlo k nejmenšímu poklesu míry diskriminativnosti. Takový příznak je zřejmě nejméně informativní.

1.2 Neuronové sítě [NEU]

vyučující:	Doc. Dr. Ing. Vlasta Radová
ročník/semestr studia:	5.ročník/ZS
datum zkoušky:	5. 1. 2017
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními typy umělých neuronových sítí a s možnostmi jejich využití.

1.2.1 Základní umělé modely neuronu, vlastnosti, souvislost s biologickým neuronem.

Lidský mozek se skládá ze 100 miliard neuronů, které mezi sebou komunikují prostřednictvím sítě vazeb. Podnět z receptorů (zrak, sluch, čich, chuť, hmat) je ve formě elektrických impulsů přenášen do centrálního nervového systému, kde je zpracováván.

Biologický neuron

Základní buňka biologických neuronových sítí.

- *soma* (tělo)
- *axon* (výstup): jediný, dlouhý až 60 cm
- *dendrity* (vstupy): krátké do 3 mm, je jich až několik tisíc na neuron
- *synapse* (rozhraní): jednosměrné brány umožňující přenos signálu pouze ve směru axon → dendrita, je jich asi 10000 na neuron

Přenášené signály jsou elektrické impulsy, jejich přenos je ovlivněn uvolňováním budicích (excitátory) a tlumicích (inhibitory) látek v synapsích. Překročí-li hodnota budicích signálů hodnotu tlumicích signálů o určitý *práh*, nastává tzv. *aktivace neuronu* - na jeho výstupu se objeví impuls. Po vygenerování impulsu se neuron na určitou dobu (tzv. období pauzy) dostane do stavu, kdy nereaguje na žádné podněty → chování neuronu lze popsat diskrétně v čase. Období pauzy není pro všechny neurony stejně dlouhé → neurony v mozku pracují asynchronně.

McCullochův-Pittsův model (1943)

Vstupy $x_1, ..., x_n$ mohou nabývat pouze hodnot 0 nebo 1 podle toho, zda je přítomen signál nebo ne. Váhy $w_1, ..., w_n$ nabývají hodnot 1 pro budící signály a -1 pro tlumící signály. Dále je definován práh b. Jedná se tedy o jednoduchý binární model. Váhy jsou pevně nastaveny a práh je rovněž pevně stanoven.

Perceptron (1958)

Diskrétní verze perceptronu byla poprvé prezentována Frankem Rosenblattem. Pro výstup platí:

$$y(k+1) = f\left[\sum_{i=1}^{n} w_i x_i(k) + b\right] = f(w^T \cdot x(k) + b) \tag{1.41}$$

w ... váhový vektor

x ... vstupní vektor

$z = w^T \cdot x(k) + b$ ... aktivační hodnota

$f(\cdot)$ ... aktivační funkce, nejčastěji používané:

- *bipolární binární*: $f(z) = \text{sgn}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$
- *unipolární binární*: $f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$
- *bipolární spojitá*: $f(z) = \frac{2}{1+e^{-\lambda \cdot z}} - 1$
- *unipolární spojitá*: $f(z) = \frac{1}{1+e^{-\lambda \cdot z}}$
- *lineární*: $f(z) = \lambda \cdot z$

1.2.2 Základní typy neuronových sítí. Způsoby činnosti a učení neuronových sítí.

Umělá neuronová síť vznikne spojením jednotlivých modelů neuronů. Výsledná funkce sítě je určena způsobem propojení jednotlivých neuronů (tzv. topologií sítě), váhami těchto spojení a způsobem činnosti jednotlivých neuronů (aktivačními funkcemi). Mezi základní typy patří:

- *vícevrstvé dopředné neuronové sítě (feedforward nets)*: výstup jedné vrstvy je připojen na vstup následující vrstvy a signál se šíří pouze ze vstupu sítě na její výstup
- *neuronové sítě se zpětnou vazbou (recurrent, feedback nets)*: oproti dopředným sítím se zde signál šíří také z výstupu sítě zpět na její vstup

Fáze činnosti

1. *Fáze nastavování*: cílem je nastavit váhy a prahy sítě tak, aby prováděla požadovanou činnost (předpokládá se, že je dána topologie sítě); Provádí se buď výpočtem (lze jen u naprosto jednoduchých sítí) a nebo učením - trénováním
2. *Fáze pracovní*: síť reaguje na předložené vstupy podle svého předchozího nastavení

Způsoby učení

- *učení s učitelem (supervised learning)*: předpokládáme, že máme k dispozici tzv. trénovací množinu, tj. množinu dvojic [input, desired_output], které jsou síti postupně předkládány. Pro každý předložený vstup neuronová síť vygeneruje skutečný výstup, který se porovná s požadovaným výstupem, a trénovací algoritmus upraví nastavení vah a prahů tak, aby rozdíl mezi skutečným a požadovaným výstupem byl minimální. Každá dvojice se síti předkládá opakovaně - trénování probíhá v trénovacích cyklech (epochách), kdy během jedné epochy jsou sítě předloženy všechny dostupné dvojice. Pořadí výběru dvojic má vliv na výsledek učení. Dobře natrénovaná síť je schopna to, co se naučila, zobecňovat (schopnost generalizace). Trénování může být *dávkové* (váhy a prahy se mění po předložení více dvojic) nebo *sekvenční* (váhy a prahy se mění po každé dvojici).
- *učení bez učitele (unsupervised learning)*: síti jsou předkládány pouze vstupy, informace učitele (požadovaný výstup) chybí. Síť se sama snaží najít zákonitosti ve vstupních datech a nastavit své váhy a prahy tak, aby na podobné vstupy reagovala podobnými výstupy. Podobnost je většinou definována eukleidovskou vzdáleností a dochází tak ke shlukování vstupních dat.

RBF (Radial Basis Function) sítě

Motivaci pro jejich studium lze nalézt v matematice u řešení aproximačních úloh, kdy řešení hledáme ve tvaru lineární kombinace vhodných bazických funkcí. Představena je dvouvrstvá síť bez zpětné vazby. Ve druhé (výstupní) vrstvě jsou neurony s lineární aktivační funkcí. V první (skryté) vrstvě mají všechny váhy hodnotu 1. Každý neuron této vrstvy je reprezentován tzv. *radiální funkcí*, která je funkcí vzdálenosti mezi vstupním vektorem a centroidem příslušného neuronu. Pro výstup j-tého neuronu ve skryté vrstvě tedy platí: $a_j = f(r_j)$, kde $f(\cdot)$ je radiální funkce a r_j je vzdálenost mezi vstupním vektorem x a centroidem j-tého neuronu c_j (obvykle eukleidovská vzdálenost: $r_j = \sqrt{(x - c_j)^T (x - c_j)}$).

Trénuje se pouze druhá (výstupní) vrstva, a to pomocí učení s učitelem. Centroidy první vrstvy lze najít jakýmkoli shlukovacím algoritmem, obvykle se používá *k-means*. Pomocí RBF-sítí lze aproximovat vícerozměrné nelineární funkce a je oproti klasické dopředné síti obecně rychleji natrénovatelná a výsledná aproximace je přesnější.

1.2.3 Algoritmus backpropagation.

Jedná se učení s učitelem, tj. předpokládá se znalost trénovací množiny o P dvojicích $[x_p, u_p]$. Cílem je nastavit váhy a prahy sítě tak, aby výstup sítě byl pro všechny vstupní vektory z trénovací množiny správný, tj. aby byla minimální chyba definována vztahem:

$$E = \frac{1}{2PM} \sum_{p=1}^{P} ||u_p - y_p||^2 = \frac{1}{2PM} \sum_{p=1}^{P} \sum_{m=1}^{M} (u_{pm} - y_{pm})^2 \tag{1.42}$$

y_p ... skutečný výstup sítě pro vstup x_p

u_p ... požadovaný výstup sítě pro vstup x_p

P ... počet trénovacích dvojic

M ... počet výstupů sítě

Váhy a prahy se mění úměrně gradientu chybové funkce, tzn. chyba se z výstupu sítě šíří zpět do jednotlivých vrstev, kde se podle ní mění váhy a prahy.

Značení (předpokládáme síť s jednou skrytou vrstvou):

E ... chyba způsobená nesprávnou činností sítě během jednoho trénovacího cyklu

$f(\cdot)$ ... aktivační funkce ve skryté vrstvě

$g(\cdot)$ ... aktivační funkce ve výstupní vrstvě

Algoritmus:

(i) *Inicializace*

$W^{(1)}(0)$ ... váhová matice pro skrytou vrstvu $(j \times n)$ v kroku 0

$W^{(2)}(0)$ ... váhová matice pro výstupní vrstvu $(m \times j)$ v kroku 0

$B^{(1)}(0)$ ... prahový vektor pro skrytou vrstvu $(j \times 1)$ v kroku 0

$B^{(2)}(0)$ ... prahový vektor pro výstupní vrstvu $(m \times 1)$ v kroku 0

$c > 0$ ... konstanta učení

$E_{max} > 0$ ... maximální povolená chyba, pod kterou se chceme dostat

$ep_{max} > 0$ ... maximální počet trénovacích cyklů (epoch)

(ii) *výpočet výstupu pro vstup x_p v kroku k*

$$y_p = g\left[W^2(k) \cdot f(W^1(k) \cdot x_p + B^1(k))\right] \tag{1.43}$$

(iii) *výpočet chyby predikce pro vstup x_p*

$$E_p = \frac{1}{2}(u_p - y_p)^T(u_p - y_p) \tag{1.44}$$

(iv) *modifikace vah a prahů pro vstup x_p v kroku k*

$$\begin{aligned}
W_{ji}^{(1)}(k+1) &= W_{ji}^{(1)}(k) + c \cdot \sum_{m=1}^{M}(u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot W_{mj}^{(2)} \cdot f'(A_{j\cdot}^{(1)}(k)) \cdot x_i \\
B_j^{(1)}(k+1) &= B_j^{(1)}(k) + c \cdot \sum_{m=1}^{M}(u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot W_{mj}^{(2)} \cdot f'(A_{j\cdot}^{(1)}(k)) \\
W_{mj}^{(2)}(k+1) &= W_{mj}^{(2)}(k) + c \cdot (u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k)) \cdot A_{j\cdot}^{(1)}(k) \\
B_m^{(2)}(k+1) &= D_m^{(2)}(k) + c \cdot (u_{pm} - y_{pm}) \cdot g'(A_{m\cdot}^{(2)}(k))
\end{aligned} \tag{1.45}$$

(v) *zastavovací podmínky* Pokud $E(k) \leq E_{max}$, síť je úspěšně natrénována. Pokud počet trénovacích epoch přesáhl ep_{max}, trénování končí bez ohledu na chybu.

Malá hodnota konstanty učení c zpomaluje proces, ovšem pro velkou hodnotu zase hrozí nebezpečí nestability - je vhodné volit adaptivní c. Strmost aktivační funkce má podobný vliv jako konstanta učení. Algoritmus může skončit v nevyhovujících "mělkých"lokálních minimech chybové funkce (chybová funkce je závislost chyby na parametrech, tj. vahách a prazích sítě). Možná řešení:

- vyzkoušet různé inicializace vah a prahů
- zvýšit počet neuronů ve skrytých vrstvách, popř. počet skrytých vrstev (pouze částečné řešení)
- *momentová metoda*: Jejím smyslem je zabránit tomu, aby se proces trénování zastavil v mělkém lokálním minimu. Pro změnu vah použijeme:

$$w(k+1) = w(k) + \alpha \cdot \Delta w(k-1) - (1-\alpha) \cdot c \cdot \frac{\partial \epsilon}{\partial w} \tag{1.46}$$

  kde $\Delta w(t-1)$ je minulá změna váhy, tj. $\Delta w(t-1) = w(t) - w(t-1)$ a α je momentová konstanta. Analogicky se mění i prahy. Při inicializaci je třeba nastavit $\Delta w(0) = 0$, resp. $\Delta b(0) = 0$. Používá se zejména při dávkovém trénování.

1.2.4 Sítě se zpětnou vazbou. Hopfieldova neuronová síť.

S předpokladem čtvercové matice W se výstup sítě počítá:

$$y_j(k+1) = f\left[\sum_{i=1}^{J} w_{ji} y_j(k) + b_j\right] \tag{1.47}$$

Přenos může být:

- *synchronní*: všechny výstupy přepočítám naráz, tedy pro vstupy v kroku $(k+1)$ používám výstupy z kroku (k)
- *asynchronní*: přepočítávám jeden výstup po druhém, tedy např. pro vstup $y_2(k+1)$ používám výstup $y_1(k+1)$, tj. v každém kroku se vypočítává výstup pouze jednoho neuronu

Proces samovolného přechodu končí buď v *rovnovážných stavech*, kdy $y(k+1) = y(k)$, nebo v *rovnovážných cyklech* tvořených stavy, mezi kterými síť kmitá. Jestliže rekurentní síť pracuje v asynchronním režimu, váhová matice je symetrická a prvky na diagonále jsou nezáporné $\rightarrow$ síť vždy konverguje do rovnovážného stavu. Jestliže rekurentní síť pracuje v synchronním režimu a váhová matice je symetrická, pak síť vždy konverguje do rovnovážného stavu nebo cyklu délky 2.

Hopfieldova síť

Jedná se o jednovrstvou rekurentní síť s neurony s bipolární binární aktivační funkcí, symetrickou váhovou maticí s nulami na diagonále a nulovým prahovým vektorem ($w_{ij} = w_{ji} \forall i, j, i \neq$

$j; \wedge w_{ii} = 0$). Pro výstup i-tého neuronu platí:

$$y_i(k+1) = \operatorname{sgn}\left[\sum_{i=1}^{n} w_{ij} y_j(k)\right] \tag{1.48}$$

$y(k)$ ... výstup sítě v čase (k) = stav sítě v čase (k)

$y(0)$ ... inicializační stav sítě

Po inicializace v čase $k = 0$ přechází síť samovolně z jednoho stavu do druhého (jedná se o dynamický systém). Ke změně stavu sítě může docházet synchronně či asynchronně. Při asynchronním přenosu se neuron, jehož výstup se bude přepočítávat, obvykle vybírá náhodně → *stochastická asynchronní rekurze.*

U Hopfieldovy sítě nedochází k učení, váhy sítě jsou určovány pomocí tzv. *záznamového algoritmu*, během kterého se do sítě zaznamenávají požadované rovnovážně stavy. Pro váhovou matici platí:

$$W = \left[\sum_{p=1}^{P} u_p \cdot u_p^T\right] - P \cdot I \tag{1.49}$$

u_p ... tzv. prototypy, tj. rovnovážné stavy, které mají být do sítě zaznamenány (dimenze n)

P ... počet zaznamenaných prototypů

I ... identická matice řádu n

Pro Hopfieldovu síť lze definovat tzv. *výpočetní energii* ve tvaru:

$$E(y) = -\frac{1}{2} y^T \cdot W \cdot y \tag{1.50}$$

Lze ukázat, že při přechodu sítě z jednoho stavu do jiného se tato energie nezvyšuje a v rovnovážném stavu (resp. cyklu) je minimální. Pro každý rovnovážný stav u existuje tzv. *komplementární stav* u', pro který platí $u' = -u$. Tento stav je rovněž rovnovážným stavem, i když v průběhu záznamového algoritmu nebyl do sítě zaznamenán. Zda proces přechodu sítě z jednoho stavu do jiného skončí v požadovaném nebo komplementárním stavu závisí při asynchronním režimu na pořadí přepočítávání výstupů jednotlivých neuronů → nelze ovlivnit. Do Hopfieldovy sítě lze spolehlivě zaznamenat maximálně $P \leq 0.14 \cdot n$ rovnovážných stavů (n je počet neuronů sítě). Pokud je v síti zaznamenáno rovnovážných stavů více, může proces synchronní i asynchronní rekurze skončit v tzv. *falešném rovnovážném stavu*, který neodpovídá žádnému zaznamenanému ani žádnému komplementárnímu stavu.

V praxi se Hopfieldova síť příliš nevyužívá, právě z důvodů možné existence falešných rovnovážných stavů. Dá se však využít jako rekonstruktor zašuměných dat či pro řešení optimalizačních úloh (např. hledání nejkratší cesty). Princip: Optimalizační úloha se popíše vhodnou funkcí, která se převede do tvaru výpočetní energie sítě. Hopfieldova síť pak samovolně hledá minimum této funkce. Existují i sítě pracující v čase spojitě, ale při jejich analýze je třeba řešit nelineární diferenciální rovnice.

1.2.5 Samoorganizující se sítě.

Samy se snaží objevit zákonitosti a souvislosti ve vstupních datech (tzv. proces samoorganizace), tzn. během učení (bez učitele) se snaží nastavit své váhy a prahy tak, aby na podobné vstupy reagovaly podobnými výstupy → dochází k tzv. shlukování vstupních dat. Míra podobnosti se obvykle posuzuje pomocí Eukleidovské vzdálenosti

$$||a - b|| = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{1.51}$$

Značení:

$S_m(k)$ ... množina vektorů, které jsou zahrnuty ve shluku S_m před přidáním vektoru x

$w_m(k)$ ... centroid shluku S_m před přidáním vektoru x

$P_m(k)$ ... počet vektorů ve shluku S_m před přidáním vektoru x

$S_m(k+1)$ ... množina vektorů, které jsou zahrnuty ve shluku S_m po přidání vektoru x, tzn. $S_m(k+1) = S_m(k) \cup x$

$w_m(k+1)$ ... centroid shluku S_m po přidání vektoru x

$P_m(k+1)$ ... počet vektorů ve shluku S_m po přidání vektoru x, tzn. $P_m(k+1) = P_m(k) + 1$

Předpokládejme, že máme množinu P vektorů dimenze n: $\{x_1, x_2, ..., x_P\}$, které chceme rozdělit do R shluků $S_1, S_2, ..., S_R$, přičemž každý shluk je reprezentován svým centroidem $w_1, w_2, ..., w_R$, pro které platí:

$$w_r = \frac{1}{P_r}\sum_{x_j \in S_r} x_j, \qquad r = 1, ..., R \tag{1.52}$$

kde P_r je počet vektorů ve shluku S_r. Vektor x chceme zařadit do jednoho z R shluků → zařadíme ho do toho shluku S_m, k jehož centroidu je x nejblíže:

$$||x - w_m|| = \min_{r=1,...,R} ||x - w_r|| \tag{1.53}$$

Po přidání vektoru x do shluku S_m se změní poloha centroidu w_m, centroidy ostatních shluků zůstávají beze změny.

Kohonenova síť

Určena pro shlukování vstupních vektorů dimenze n do R shluků. Je to jednovrstvá dopředná síť s R výstupy, n vstupy a nulovým prahovým vektorem. Každý řádek váhové matice je normalizován tak, že má velikost 1, tzn., že pro i-tý řádek váhové matice platí:

$$||w_i|| = \sqrt{\sum_j w_{ij}^2} = 1 \tag{1.54}$$

Síť se učí bez učitele. Učení probíhá na základě tzv. *pravidla vítěze* ("vítěz bere vše") → na základě vstupního vektoru x se změní váhy m-tého neuronu podle vztahu:

$$\hat{w}_m(k+1) = w_m(k) + c \cdot (x^T - w_m(k)) \tag{1.55}$$

kde $c \in \langle 0.1, 0.7 \rangle$ je konstanta učení a m-tý neuron je tzv. *vítěz*, pro kterého platí:

$$w_m \cdot x = \max_{r=1,\ldots,R} w_r \cdot x \tag{1.56}$$

To znamená, že se mění váhy neuronů s nejvyšší aktivační hodnotou, váhy ostatních neuronů se nemění. Po změně vah je třeba váhový vektor m-tého neuronu normalizovat, tzn.

$$w_m = \frac{\hat{w_m}}{||\hat{w_m}||} \tag{1.57}$$

Váhová matice $W(0)$ se inicializuje náhodnými čísly, poté je třeba provést normalizaci každého řádku.

Vlastnosti Kohonenovy sítě:

- Váhy neuronů se při trénování blíží do středu shluků, které reprezentují.
- Síť má pouze jednu vrstvu → lze s ní najít pouze lineárně oddělitelné shluky.
- I v případě lineárně oddělitelných shluků nemusí být tyto shluky vždy nalezeny.
- Váhy měnící se podle výše zmíněného vztahu při konstantní hodnotě c nekonvergují ke konstantním hodnotám. Proto se často využívá proměnlivá konstanta učení $c(t)$, která má po určitou dobu t_p (např. 1000 kroků) konstantní hodnotu c_0, a pak se snižuje, např. dle vztahu:

  $$c(t) = c_0 \cdot e^{-\frac{t-t_p}{t_q}}, \qquad t > t_p \tag{1.58}$$

  kde t_q udává rychlost poklesu. Pokud předem neznáme počet shluků, předpokládáme, že shluků je velké množství. Během trénování pak některé neurony pravděpodobně nebudou měnit své váhy → nemají význam a lze je vypustit.
- Po natrénování sítě bude pro zadaný vstupní vektor největší hodnota na výstupu toho neuronu, jehož váhy reprezentují centroid shluku, ke kterému má daný vstupní vektor nejblíže.

Využití: hledání shluků ve vstupních datech; vektorová kvantizace.

Kohonenova mapa (feature map)

Jedná se o speciální případ Kohonenovy sítě. Neurony jsou uspořádány tak, že tvoří jednorozměrné či dvojrozměrné pole. Učení probíhá bez učitele dle pravidla vítěze. Mění se ovšem nejen váhy vítězného neuronu, ale i váhy neuronů okolních. Pro změnu vah platí:

$$w_i(k+1) = w_i(k) + c \cdot (x^T - w_i(k)), \qquad i \in N_m(k) \tag{1.59}$$

$w_i(k)$ ... i-tý řádek matice W (není normalizovaný)

$N_m(k)$ ... okolí m-tého (vítězného) neuronu v čase k

Pro vítězný neuron s indexem m přitom platí:

$$||x - w_m^T|| = \min_{i=1,\ldots,R} ||x - w_i^T|| \tag{1.60}$$

Řádky váhové matice se nenormalizují! V procesu trénování sítě se velikost okolí zmenšuje. Na začátku trénování se velikost okolí obvykle volí tak, že zahrnuje všechny neurony sítě. Pak se velikost okolí lineárně snižuje až na nulu, kdy okolí obsahuje pouze vítězný neuron. Doba trénování se obvykle volí tak, aby doba, po kterou je velikost okolí rovna 0, byla přibližně 3-krát větší než doba, po kterou docházelo ke snižování velikosti okolí. Inicializace vah se provádí malými náhodnými čísly, obvykle z intervalu $\langle -0.1, 0.1 \rangle$. Konstanta učení c se podobně jako u Kohonenovy sítě volí proměnlivá v čase. Kohonenova mapa se využívá pro redukci počtu příznaků pro klasifikaci či pro vizualizaci vektorů velké dimenze.

1.2.6 Oblasti použití neuronových sítí.

Obecně lze aplikace matematicky formulovat jako aproximaci funkce (tj. optimalizační úlohu):

$$y = f(x, parametry) \tag{1.61}$$

- *Klasifikátory*: úkolem je zařadit vstupní data do skupin (tříd) podle vzájemné podobnosti (nelineární sítě); Vstupem jsou jednotlivé klasifikované obrazy (tj. vektory příznaků), výstupem je informace o zařazení vstupního obrazu do určité třídy. Je-li R počet tříd, potom počet výstupů je sítě je obvykle roven buď R nebo $log_2 R$. Obecně platí, že počet výstupu sítě může být jakékoli číslo z intervalu $\langle log_2 R, R \rangle$. Většinou se ve všech vrstvách využívá některá ze sigmoidálních aktivačních funkcí, která se po natrénování ve výstupní vrstvě nahradí odpovídající binární aktivační funkcí.
- *Aproximátory funkcí (regresory)*: z několika zadaných (naměřených) hodnot je třeba sestavit funkční závislost (nelineární sítě); Jednotlivé vstupy nelineárních sítí představují nezávisle proměnné aproximované funkce, výstupy představují závisle proměnné. Pro aproximaci libovolné funkce postačují 2 až 3 skryté vrstvy neuronů:
  - jedna vrstva rozdělí vstupní prostor nadrovinou (ve 2D přímkou)
  - dvě vrstvy jsou schopny oddělit konvexní oblast
  - tři vrstvy jsou schopny oddělit i nekonvexní oblasti

  Počet skrytých vrstev a počet neuronů v nich má vliv na schopnost sítě zobecňovat (generalizovat) vstupní body trénovací množiny. Velké množství vrstev a neuronů může vést k přetrénování (overfitting) sítě, malé množství k nedotrénování (underfitting) sítě.
- *Paměti a rekonstruktory*: na základě předloženého vstupního obrazu je síť schopna "vybavit si"odpovídající výstupní obraz (asociativní paměť - nelineární sítě), popř. je schopna zrekonstruovat zašuměný vstupní obraz do původní podoby (Hopfieldova síť); Asociativní paměť je paměť adresovaná obsahem, adresou je klíčová hodnota ukládaná s informací. Vstupem sítě je vstupní obraz (tzv. klíč), kterým se přistupuje do paměti, výstupem sítě je příslušný asociovaný obraz. Zapamatovaná informace je v síti uložena v hodnotách vah a

prahů a je rozprostřena po celé síti. Asociativní paměť je schopna vykonávat svou funkci dostatečně správně i při částečném poškození (například při odstranění části neuronů skryté vrstvy). U pamětí adresovaných adresou se obsah z poškozené části ztrácí úplně.

- *Optimalizace*: úkolem je minimalizovat určitou ztrátovou funkci, která je obvykle definována uživatelem (rozvrhování činností, hledání optimální cesty, apod.);
- *Shlukování a redukce příznaků*: objevování shluků ve vstupních datech a redukce počtu příznaků. Základní charakteristikou těchto sítí je tzv. samoorganizace (Hopfieldova síť).

1.3 Zpracování digitalizovaného obrazu [ZDO]

vyučující:	Doc. Ing. Miloš Železný Ph.D.
	Ing. Petr Neduchal
ročník/semestr studia:	4.ročník/LS
datum zkoušky:	13. 7. 2015
hodnocení:	1

cíl předmětu (STAG):
Porozumět principům zpracování digitalizovaného obrazu a počítačového vidění. Analyzovat vlastnosti obrazové informace a interpretovat tyto informace, navrhnout a vytvořit algoritmus pro zpracování obrazové informace s cílem rozpoznání objektů, jevů či vlastností scény v obraze obsažené.

1.3.1 Bodové jasové transformace.

Jasové korekce

Cílem je stanovit nový jas daného pixelu (bodu), který je funkcí původního jasu a polohy:

$$p_{ij}^{new} = f(i, j, p_{ij}^{old}) \tag{1.62}$$

Nejčastěji se používá matice opravných koeficientů OPR a dostáváme tedy:

$$p_{ij}^{new} = p_{ij}^{old} \cdot OPR_{ij} \tag{1.63}$$

Matici opravných koeficientů získáme *kalibrací* - na snímacím zařízení nesnímáme obraz se známými hodnotami. Z těchto známých správných hodnot (actual) a z naměřených hodnot (measured) vypočteme matici opravných koeficientů:

$$OPR_{ij} = \frac{actual_{ij}}{measured_{ij}} \tag{1.64}$$

Využití se najde například při opravě systematických chyb snímacího řetězce.

Jasové transformace

V tomto případě je funkce stejná pro všechny body (pixely) obrazu, nezáleží na poloze v obrázku:

$$p_{ij}^{new} = f(p_{ij}^{old}) \tag{1.65}$$

Využívá se:

- vyhledávací/převodní tabulky (LookUp Table)
- prahování (segmentace - rozdělení obrazu na oblasti, které souvisejí s reálnými objekty)
- ekvalizace histogramu

1.3.2 Geometrické transformace.

Obecná definice:

$$i' = u(i,j); \qquad j' = v(i,j); \qquad g(i',j') = f(i,j) \tag{1.66}$$

- *vztah známe*: např. rotace, posun, zvětšení, ...
- *vztah hledáme* na základě původního a transformovaného obrazu: např. pomocí korespondence souřadnic na družicovém snímku a na mapě, používají se tzv. vlícovací body

Proces se skládá ze dvou kroků:

1. *plošná transformace*

$$x' = \sum_{r=0}^{m}\sum_{k=0}^{m-r} a_{rk}\cdot x^r \cdot y^k; \qquad y' = \sum_{r=0}^{m}\sum_{k=0}^{m-r} b_{rk}\cdot x^r \cdot y^k \tag{1.67}$$

   Jedná se o polynom m-tého stupně. Např.:

   - bilineární: $x' = a_0 + a_1\cdot x + a_2\cdot y + a_3\cdot xy; \qquad y' = b_0 + b_1\cdot x + b_2\cdot y + b_3\cdot xy$
   - afinní: $x' = a_0 + a_1\cdot x + a_2\cdot y; \qquad y' = b_0 + b_1\cdot x + b_2\cdot y$ - sem patří např. rotace, posun, zvětšení, zkosení, atd.

2. *interpolace jasu*: Nejbližší soused - většinou se postupuje tak, že se udělá inverzní transformace a odečte se nejbližší hodnota v původním obrazu

Geometrické transformace jsou z principu ztrátové. V některých speciálních případech (jako např. otočení o násobek 90°) je informace zachována, obecně to však neplatí. Využití: dálkový průzkum Země, desktop publishing.

1.3.3 Filtrace šumu.

Obecná diskrétní konvoluce:

$$g(i,j) = \sum_{m,n}, \sum_{i,j} f(i-m, j-n) \cdot h(m,n) \tag{1.68}$$

Někdy se setkáme s problémem, jak počítat konvoluci co nejrychleji (např. při požadavku na zpracování v reálném čase). Tento problém částečně řeší tzv. box-algorithm (Šlesinger).

Vyhlazování (filtrace)

Cílem je potlačení šumu v obrazu (např. aditivní šum ν se střední hodnotou μ). Myšlenka je založena na využití n bodů v okolí.

$$f_i' = f_i + v_i; \qquad \frac{f_1 + f_2 + \cdots + f_n}{n} + \frac{\nu_1 + \nu_2 + \cdots + \nu_n}{n} \tag{1.69}$$

Průměrovat můžeme přes více snímků: $g(i,j) = \frac{1}{n}\sum_{k=1}^{n} f_k(i,j)$, kde f_k je obrazová funkce k-tého snímku a n se řádově pohybuje v desítkách $(30-50)$. Alternativně můžeme průměrovat lokálně - v daném okolí, to však přináší nevýhodu rozostření hran (ztratí se detaily) $\rightarrow$ velikost masky by měla být menší, než je nejmenší detail v obraze, který chceme zachovat.

Používané masky:

$$h_1 = \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}; \qquad h_2 = \frac{1}{10}\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix}; \qquad h_3 = \frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}; \qquad h_4 = \frac{1}{18}\begin{bmatrix}1&1&1\\1&0&1\\1&1&1\end{bmatrix}$$

V případě h_1 jde o rovnoměrnou masku (Average), h_2 zvýhodňuje středový bod, h_3 zvýhodňuje středový bod a body v hlavních směrech (Gauss) a maska h_4 naopak znevýhodňuje středový bod. Poslední dvě masky mají teoretickou výhodu výpočetní rychlosti, neboť dělení 8, resp. 16, lze v počítači realizovat jako bitový posun a tedy velmi rychle.

Alternativně se dá využít tzv. *maximální zastoupení* (Modus) - výsledkem filtrace je jas, který se v daném okolí vyskytuje nejčastěji. Dále můžeme využít nějakého *výběrového kvantilu* (jako např. Medián). Ten dobře řeší problém výskytu jedné nebo více vychýlených hodnot (např. Salt&Pepper Noise), ovšem je nelineární, porušuje tenké čáry a "trhá"rohy.

1.3.4 Gradientní operátory.

Též se označují jako diferenciální operátory či hranové detektory. Dokáží např. detekovat nespojitosti šedé úrovně v obraze, lze je tedy využít pro segmentaci. Hledáme velikost a směr gradientu.

Gradient ve spojitém případě:

$$|grad\, g| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{1.70}$$

V diskrétním případě:

$$\Delta x g(i,j) = g(i,j) - g(i,j-1); \qquad \Delta y g(i,j) = g(i,j) - g(i-1,j) \tag{1.71}$$

$$|grad\, g| = \sqrt{(\Delta x g)^2 + (\Delta y g)^2} \tag{1.72}$$

$$\varphi = \operatorname{arctg}\left(\frac{\Delta y g}{\Delta x g}\right) \tag{1.73}$$

Máme tři typy gradientních operátorů:

1. *Aproximace derivací diferencemi*
   - Roberts: $g(i,j) = |f(i,j) - f(i+1,j+1)| + |f(i,j+1) - f(i+1,j)|$
   - Laplace: $h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ → aproximace Laplaceova operátoru ($\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$)

   $$\Delta_x^2 f(i,j) = f(i,j+1) + f(i,j-1) - 2f(i,j) \tag{1.74}$$

   $$\Delta_y^2 f(i,j) = f(i+1,j) + f(i-1,j) - 2f(i,j) \tag{1.75}$$

   $\nabla^2 f(i,j) = \Delta_x^2 f(i,j) + \Delta_y^2 f(i,j) = f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j) - 4f(i,j)$

   Udává pouze velikost hrany, ale ne její směr. Chceme-li znát i směr hrany, použijeme směrově závislý gradientní operátor.

2. *Srovnání s parametrickým modelem hran* Nejprve označíme směry (např. 8). Vztahy:

   $$|grad\, g| \hat{=} \max_{k=0,\ldots,7}\{g * h_k\} \tag{1.76}$$

   $$\varphi = \operatorname*{argmax}_{k=0,\ldots,7}\{g * h_k\} \tag{1.77}$$

   Používají se masky *Prewitt*, *Sobel*, *Kirsch*.

3. *Průchody nulou 2. derivace obrazové funkce (Marrova teorie)* Hledáme průchody nulou druhé derivace obrazové funkce. Používají se masky pro detekci čáry a masky pro detekci bodu.

1.3.5 Metody segmentace.

Vstupem je intenzitní obraz a výstupem obraz rozčleněný na části, které mají souvislost s objekty reálného světa.

- *Kompletní segmentace*: Vytvořené oblasti jednoznačně korespondují s objekty ve vstupním obraze. Obecně je nezbytná spolupráce s vyšší úrovní zpracování a využívají se znalosti o řešeném problému. V případě, kdy je obraz tvořen kontrastními objekty na pozadí konstantního jasu, přicházejí dobré výsledky i na nižší úrovni zpracování. Např. text, krevní buňky, počítání šroubků.
- *Částečná segmentace*: Vytvořené oblasti jsou homogenní vzhledem k určitým zvoleným vlastnostem (jas, barva, textura, apod.). Oblasti se obecně mohou překrývat a je třeba aplikovat další postupy na vyšší úrovni zpracování. Např. scéna s polem a lesem při pohledu z okna - po segmentaci neodpovídá jedna oblast jednomu objektu.

Další příklady využití: hledání lodí na moři, typické vlastnosti železničních tratí a dálnic (např. maximální zakřivení), řeky se neprotínají

Prahování

Nejjednodušší a nejčastěji používaná metoda segmentace, je nenáročná na hardwarovou realizaci a je nejrychlejší (lze provádět v reálném čase). Důležitá je volba prahu T - úloha, kterou lze obecně jen velmi obtížně provést automaticky. Také lze použít jen pro určitou třídu obrazů (objekty a pozadí musí být jasově snadno rozlišitelné).

$$g(i,j) = \begin{cases} 1 & f(i,j) \geq T \\ 0 & f(i,j) < T \end{cases} \tag{1.78}$$

Možná modifikace:

$$g(i,j) = \begin{cases} 0 & f(i,j) \in D \\ 1 & \text{jinak} \end{cases} \tag{1.79}$$

kde D je množina jasů odpovídajících pozadí (např. u snímků krevních buněk - cytoplazma se jeví v určitém intervalu jasů, pozadí je světlejší, jádro je tmavší).

Dále můžeme použít prahovaní více prahy či poloprahování (odstraníme pozadí, v objektech však zachováme rozložení jasů). Používá se při vizuálním hodnocení výsledků člověkem. $f(i,j)$ nemusí být pouze jasová funkce (ale také třeba hodnota gradientu, hloubková mapa, barva, ...).

Metody určování prahu:

- *Histogram*: Hledáme lokální minimum mezi dvěma největšími dostatečně vzdálenými lokálními maximy.
- *Procentní prahování*: Máme apriorní znalost o tom, kolik procent plochy obrazu pokrývají objekty (např. průměrné pokrytí plochy stránky textem se pohybuje okolo 5%). Práh potom nastavíme tak, aby právě tolik procent obrazových bodů mělo barvu objektů, zbytek barvu pozadí.

Segmentace na základě detekce hran

Hrany jsou místa obrazu, kde dochází k určité nespojitosti, většinou v jasu, ale také v barvě, textuře, apod. Obraz hran vznikne aplikací některého hranového operátoru (Canny). Obvykle jen málo míst v obraze má nulovou hodnotu velikosti hrany (šum), je třeba tedy např. prahováním potlačit nevýrazné hrany a zachovat pouze ty významné. Často se využívá apriorní informace o možné poloze hran, popř. znalosti koncových bodů.

Segmentace narůstáním oblastí (region growing)

Metoda uplatnitelná v obrazech se šumem, kde se obtížně hledají hranice. Významnou vlastností je homogenita. Kritérium homogenity je založeno na jasových vlastnostech, komplexnějších způsobech popisu nebo dokonce na vytvářeném modelu segmentovaného obrazu. Většinou pro oblasti požadujeme splnění těchto podmínek:

- $H(R_i) = \text{TRUE}$, $i = 1, 2, ..., l$
- $H(R_i \cup R_j) = \text{FALSE}$, $i, j = 1, 2, ..., l$; $i \neq j$; (R_i soused R_j)

l ... počet oblastí

R_i ... jednotlivé oblasti

$H(R_i)$ ... dvouhodnotové vyjádření kritéria homogenity, oblasti musí být homogenní a maximální

Spojování oblastí: Na začátku každý obrazový element (2x2, 4x4, 8x8, ...) představuje samostatnou oblast, dále spojujeme vždy dvě sousední oblasti, pokud vyhovuje kritérium homogenity. Výsledek závisí na pořadí spojování. Proces končí v okamžiku, kdy nelze spojit žádné dvě oblasti.

Štěpení a spojování (split and merge): Využívá pyramidální reprezentaci obrazu. Oblasti jsou čtvercové a odpovídají elementu dané úrovně pyramidální datové struktury. Na počátku určíme nějaké počáteční rozložení obrazu. Platí-li pro obraz R i-té úrovně pyramidální struktury $H(R) = \text{FALSE}$ (oblast není homogenní), rozdělíme R na 4 oblasti $(i + 1)$. úrovně. Existují-li sousední oblasti R_i a R_j takové, že $H(R_i \cup R_j) = \text{TRUE}$, spojíme R_i a R_j do jedné oblasti. Nelze-li žádnou oblast spojit ani rozdělit, algoritmus končí.

Segmentace srovnáváním se vzorem (template matching)

Úloha má za úkol nalézt známé objekty (vzory) v obraze. Kromě hledání objektu této metody lze využít také např. pro srovnávání dvou snímků z různých míst či k určování relativního pohybu objektů. Nepřítelem je zde opět šum. Jako míru souhlasu většinou využíváme vzájemnou

korelaci:

$$C_1(u,v) = \frac{1}{\max\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|}$$
$$C_2(u,v) = \frac{1}{\sum\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|} \tag{1.80}$$
$$C_3(u,v) = \frac{1}{\sum\limits_{i,j\in V}|f(i+u,j+v) - h(i,j)|^2}$$

Testujeme souhlas obrazu f se vzorem h umístěným v poloze (u, v). Pro každou polohu vzoru h v obraze f určíme hodnotu míry souhlasu vzoru s danou částí obrazu. Problémy nastanou, pokud se vzor v obraze vyskytuje natočený, s jinou velikostí nebo s geometrickým zkreslením. Můžeme to řešit zkoušením více případů či hledáním nejprve menších částí vzoru. Metodu lze urychlit zrychleným prováděním testů v hrubším rozlišení a v místě lokálního maxima pak přesným doměřením.

1.3.6 Matematická morfologie.

Oblast analýzy obrazu, která se opírá o teorii bodových množin (binární obraz, obraz s více úrovněmi jasu). Ve středu pozornosti je tvar objektů → identifikace tvaru, optimální rekonstrukce tvaru, který je porušen. Obrazy jsou nejprve předzpracovány např. metodami segmentace jsou nalezeny objekty (binární obraz) a poté jsou použity morfologické postupy. Použití především pro: odstranění šumu, zjednodušení tvaru objektů, zdůraznění struktury objektů (kostra, ztenčování, zesilování, výpočet konvexního obalu, označování objektů), popis objektů číselnými charakteristikami (plocha, obvod, projekce)

Relace s menší bodovou množinou (strukturním elementem): Morfologickou transformaci si představíme, jako bychom pohybovali strukturním elementem systematicky po celém obraze. Bod obrazu, který se shoduje s počátkem souřadnic strukturního elementu, nazýváme okamžitý bod, a právě do něj zapíšeme výsledek relace. Ke každé morfologické transformaci $\Phi(x)$ existuje duální transformace $\Phi^*(x)$. Mezi základní transformace patří posunutí, dilatace, eroze, otevření a uzavření

Dilatace a eroze

Dilatace $\oplus$ skládá body dvou množin pomocí vektorového součtu

$$X \oplus B = \{d \in E^2; d = x + b;\, x \in X,\, b \in B\} \tag{1.81}$$

Nejčastěji používán strukturní element (3x3). Objekty se rozrostou o jednu slupku na úkor pozadí, díry a zálivy tloušťky 2 se zaplní.

Eroze $\ominus$ skládá dvě bodové množiny s využitím rozdílu vektorů, je duální (ale ne inverzní) transformací k dilataci.

$$X \ominus B = \{d \in E^2; d + b \in X;\, \forall b \in B\} \tag{1.82}$$

Nejčastěji se opět používá element (3x3). Zmizí objekty (čáry) tloušťky 2 a osamělé body, objekty se zmenší o 1 slupku. Odečteme-li od původního obrazu jeho erozi, dostaneme obrysy objektu.

Otevření a uzavření

V obou případech jde o kombinaci dilatace a eroze a výsledný obraz obsahuje méně detailů.

Otevření $\circ$ je eroze následovaná dilatací:

$$X \circ B = (X \ominus B) \oplus B \tag{1.83}$$

Oddělí objekty spojené úzkou šíjí, odstraní malé detaily.

Uzavření $\bullet$ je naopak dilatace následovaná erozí:

$$X \bullet B = (X \oplus B) \ominus B \tag{1.84}$$

Spojí objekty, které jsou blízko u sebe, zaplní malé díry a úzké zálivy.

Pokud se obraz po otevření/uzavření elementem B nezmění, říkáme, že je otevřený/uzavřený vzhledem k B. Obě transformace jsou *idempotentní*, tj. opakovaným použitím těchto operací se nezmění výsledek.

Skelet

Skelet $S(Y)$ je množina bodů - středů kružnic, které jsou obsaženy v Y a dotýkají se Y alespoň ve dvou bodech. Lze vytvořit pomocí erozí a dilatací, ale potom může skelet být tlustší než jeden bod. Často se skelet nahrazuje množinou zpracovanou sekvenčním homotopickým zpracováním (hit or miss transformace).

Aplikací funkcí *ztenčování*, resp. *zesilování*, dojde k okleštění skeletu o izolované body a krátké čáry z konců skeletu. V konečném stavu skelet obsahuje pouze uzavřené křivky.

Kapitola 2

Teorie řízení [TŘSZ]

2.1 Lineární systémy 1-2 [LS1], [LS2]

vyučující:	Doc. Ing. Jiří Melichar, CSc. Ing. Martin Čech, Ph.D. Ing. Jiří Mertl, Ph.D.
ročník/semestr studia:	2.ročník/ZS-LS
datum zkoušky:	X. 1. 2013/X. X. 2013
hodnocení:	1/2

cíl předmětu (STAG):

LS1: Student by měl získat přehled o typech, struktuře a chování reálných dynamických systémů, obeznámit se s metodikou tvorby matematických modelů reálných dynamických systémů a s metodami analýzy jejich vlastností a chování v časové i frekvenční oblasti. Student by měl také porozumět základním principům řízení dynamických systémů a metodám pro získávání potřebných dat z reálných procesů.

Cílem předmětu LS2 je, aby student:

- získal přehled o klasických regulačních úlohách, o struktuře regulačních obvodů a o základních typech dynamických i nedynamických regulátorů;
- dokázal analyzovat reálnou regulační úlohu v její celistvosti, uměl formulovat požadavky na kvalitu regulace v časové i frekvenční oblasti při současném respektování všech omezení;
- byl schopen použít vhodné metody pro návrh spojitých i číslicových regulátorů a získávat potřebná data z reálného procesu;
- byl schopen analýzy nelineárních dynamických systémů a základní orientace v problémech jejich řízení.

2.1.1 Matematické modely spojitých a diskrétních lineárních dynamických systémů.

Za model považujeme *stavovou reprezentaci* dynamických systémů a nebo dynamiku vyjádřenou *diferenciálními rovnicemi*. K modelu dojdeme buď *experimentálně* a nebo (raději) na základě *matematicko-fyzikálního* modelování (znalost fyzikálních zákonů z dané vědní oblasti)

- *klasická mechanika*
  (a) Newtonovská mechanika - silové zákony, rovnováha sil

(b) Lagrangeova mechanika - znalost kinetické a potenciální energie + elektromechanická analogie

- *elektrické obvody* - Kirchhoffovy zákony
- *hydrodynamika* - rovnice kontinuity, Bernoulliho zákon
- jiné získání struktury a parametrů modelu

Z matematicko-fyzikálního modelování získáme diferenciální rovnice vyššího řádu, které lze převést na soustavu diferenciálních rovnic prvního řádu (lineární či nelineární). Odtud potom získáme stavovou reprezentaci.

Spojitý systém:

$$\begin{aligned} \dot{x}(t) &= A \cdot x(t) + B \cdot u(t) \\ y(t) &= C^T \cdot x(t) + D \cdot u(t) \end{aligned} \tag{2.1}$$

Diskrétní systém:

$$\begin{aligned} x(k+1) &= A \cdot x(k) + B \cdot u(k) \\ y(k+1) &= C^T \cdot x(k) + D \cdot u(k) \end{aligned} \tag{2.2}$$

Newtonova mechanika

Newtonův zákon síly:

$$\boxed{F = m \cdot a} \tag{2.3}$$

K sestavení stavového modelu se zavedou složky vektoru stavu $x_1(t), x_2(t)$:

$$\begin{aligned} y(t) &\coloneqq x_1(t) \\ \dot{y}(t) &= \dot{x}_1(t) \coloneqq x_2(t) \end{aligned} \tag{2.4}$$

Víme, že $x_1(t)$ odpovídá poloze. Derivace polohy, tedy $x_2(t)$, odpovídá rychlosti. Zrychlení poté odpovídá derivace rychlosti, tedy $\dot{x}_2(t) = a = \frac{F(t)}{m}$. Nyní už můžeme sestavit stavový model založený na zákonu síly:

$$\begin{aligned} \dot{x}(t) &= A \cdot x(t) + b \cdot F(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot x(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \cdot F(t) \\ y(t) &= C^T \cdot x(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot x(t) \end{aligned} \tag{2.5}$$

Lagrangeova mechanika

Založena na znalosti kinetické a potenciální energie.

Lagrangeova funkce:

$$L(y, \dot{y}) \coloneqq T(\dot{y}) - V(y) \tag{2.6}$$

Lagrangeova rovnice I. druhu:

$$F(t) = \frac{d}{dt}\frac{\partial L(y,\dot{y})}{\partial \dot{y}} - \frac{\partial L(y,\dot{y})}{\partial y} = \frac{d}{dt}\frac{\partial T(\dot{y})}{\partial \dot{y}} + \frac{\partial V(y)}{\partial y} \tag{2.7}$$

Analogie pro hmotný bod z Newtonova zákonu síly:

$$\begin{aligned} F_1(t) &= m \cdot \ddot{y}(t) = \frac{d}{dt}(m \cdot \dot{y}) = \frac{d}{dt}\left[\frac{\partial}{\partial \dot{y}}(\frac{1}{2} \cdot m \cdot \dot{y}^2)\right] = \frac{d}{dt}\frac{\partial T(\dot{y})}{\partial \dot{y}} \\ F_2(t) &= m \cdot g = \frac{\partial}{\partial y}(m \cdot g \cdot y) \end{aligned} \tag{2.8}$$

Další příklady

- *Př.: Matematické kyvadlo*

  (a) Newton:

$$\begin{aligned} m \cdot a(t) &= m \cdot g \cdot sin(\varphi(t)); \qquad ds(t) = l \cdot d\varphi(t) \\ v(t) &= \frac{ds(t)}{dt} = l \cdot \dot{\varphi}(t) \\ a(t) &= \frac{d}{dt} \cdot v(t) = l \cdot \ddot{\varphi}(t) \\ \rightarrow m \cdot l \cdot \ddot{\varphi}(t) &= m \cdot g \cdot sin(\varphi(t)),\ \varphi(0) \\ \ddot{\varphi}(t) &= \frac{g}{l} \cdot sin(\varphi(t)) \end{aligned}$$

  (b) Lagrange:

$$\begin{aligned} T(\cdot) &= \frac{1}{2} \cdot m \cdot v^2 = \frac{1}{2} \cdot m \cdot (V_x^2 + V_y^2) \\ x(t) &= l \cdot sin(\varphi(t)); \qquad y(t) = l - l \cdot cos(\varphi(t)) \\ \dot{x}(t) &\coloneqq V_x(t) = l \cdot cos(\varphi(t)) \cdot \dot{\varphi}(t); \qquad \dot{y}(t) \coloneqq V_y(t) = l \cdot sin(\varphi(t)) \cdot \dot{\varphi}(t) \\ \rightarrow T(\cdot) &= \frac{1}{2} \cdot m \cdot \left[l^2 \cdot cos^2(\varphi(t)) \cdot \dot{\varphi}^2(t) + l^2 \cdot sin^2(\varphi(t)) \cdot \dot{\varphi}^2(t)\right] = \frac{1}{2} \cdot m \cdot l^2 \cdot \dot{\varphi}^2(t) \end{aligned}$$

$$\begin{aligned} V(\cdot) &= m \cdot g \cdot y(t) = m \cdot g \cdot l - m \cdot g \cdot l \cdot cos(\varphi(t)) \\ \rightarrow L(\cdot) &= T(\cdot) - V(\cdot) = \frac{1}{2} \cdot m \cdot l^2 \dot{\varphi}^2(t) - m \cdot g \cdot l + m \cdot g \cdot l \cdot cos(\varphi(t)) \end{aligned}$$

  Dosadíme do Lagrangeovy rovnice:

$$\begin{aligned} \frac{d}{dt} \cdot \frac{\partial L}{\partial \dot{\varphi}(t)} - \frac{\partial L}{\partial \varphi} &= 0 \\ l \cdot \ddot{\varphi}(t) + g \cdot sin(\varphi(t)) &= 0 \\ \rightarrow \ddot{\varphi}(t) + \frac{g}{l} \cdot sin(\varphi(t)) &= 0,\ \varphi(0) \end{aligned}$$

- *Př.: RLC obvod*

  Kirchhoff: $\boxed{\sum u = 0}$

$$u(t) = L\frac{di(t)}{dt} + \frac{1}{C}\int i(t)dt = Ri(t)$$
$$\frac{du(t)}{dt} = L\frac{d^2i(t)}{dt^2} + \frac{1}{C}i(t) + R\frac{di(t)}{dt}$$

  Volba stavových proměnných:

$$x_1(t) \coloneqq i(t); \qquad x_2(t) \coloneqq u_c = \frac{1}{C}\int i(t)dt$$

$$\rightarrow \dot{x}_1(t) = -\frac{R}{L}x_1(t) - \frac{1}{L}x_2(t) + \frac{1}{L}u(t); \qquad \dot{x}_2(t) = \frac{1}{C}x_1(t); \qquad y(t) = Rx_1(t)$$

- *Př.: Interaktivní model dravec-kořist*

- *Př.: Levitace kuličky v magnetickém poli*

- *Př.: Stejnosměrný motor řízený do kotvy*

- *Př.: Jednoduchý tlumič*

2.1.2 Linearizace nelineárních dynamických systémů, rovnovážné stavy. Harmonická linearizace.

Pro lineární dynamické systémy (LDS) máme:

- *rovnovážný stav* x_r při $u(t) = 0\,\forall t \iff 0 = A \cdot x_r \begin{cases} x_r = 0 & h[A] = n \\ \text{integrator} & h[A] = k < n \end{cases}$
- *ustálený stav* x_r při $u(t) = \text{konst}\,\forall t \iff 0 = A \cdot x_r + B \cdot u_{konst}$

Pro nelineární dynamické systémy (NDS):

- *rovnovážný stav* x_r při $u(t) = 0\,\forall t \iff 0 = f(x_r) \to y_r = g(x_r)$ (obecně více rovnovážných stavů)
- *ustálený stav* x_r při $u(t) = \text{konst}\,\forall t \iff 0 = f(x_r, u_{konst}) \to y_r = g(x_r)$

Linearizace NDS

Bodová (vs. exaktní) linearizace (v okolí rovnovážných či ustálených stavů).

NDS:

$$\dot{x}(t) = f(x, u); \qquad y(t) = h(x, u)$$

Rovnovážné stavy: x_r, u_{konst}, y_r. Linearizovaný model v okolí x_r (bude akceptovatelný v "blízkém okolí"rovnovážných stavů):

$$x(t) = x_r + \Delta x(t); \qquad y(t) = y_r + \Delta y(t); \qquad u(t) = u_{konst} + \Delta u(t)$$
$$\dot{x}_r + \Delta\dot{x}(t) = f(x_r + \Delta x, u_{konst} + \Delta u) = f(x_r, u_{konst} + \frac{\partial f(x,u}{\partial x}\Delta x(t) + \frac{\partial f(x,u)}{\partial u}$$
$$\Delta y = \frac{\partial fC}{\partial x}\Delta x + \frac{\partial fC}{\partial u}$$

O dynamice LDS či linearizovaného DS rozhodují vlastní čísla matice A: $\lambda_i(A)$, $i = 1, ..., n$, resp. rozhodují o chování trajektorií systému vzhledem k rovnovážným stavům.

$$\{\lambda_i(A)\}_{i=1}^n = \begin{cases} \text{realna} & \begin{cases} \text{kladna} \\ \text{zaporna} \end{cases} \\ \text{ryze imaginarni} & \\ \text{komplexne sdruzena} & \begin{cases} Re\{\lambda_i\} > 0 \\ Re\{\lambda_i\} < 0 \end{cases} \\ \lambda_i \neq 0 \to \infty \text{ rovnovaznych stavu} & \end{cases} \tag{2.9}$$

Typy rovnovážných stavů (LDS II. řádu)

(a) *"střed"*: $\lambda_{1,2} = \pm i$

(b) *"ohnisko"*: $\lambda_{1,2} = a \pm bi$

(c) *"uzel"*: $\lambda_{1,2} \in R$

(d) *"sedlo"*: $\lambda_{1,2} \in R$, $\lambda_1 < 0 \wedge \lambda_2 > 0$

Explicitní řešení stavové rovnice

$$x(t) = e^{A(t-t_0)} \cdot x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)} \cdot \beta \cdot u(\tau) d\tau; \qquad \tau \in [t_0, t] \tag{2.10}$$

Výpočet stavové matice přechodu $e^{A(t)}$:

1. Rozvoj v řádu: $e^{A(t)} = I + \frac{A \cdot t}{1!} + \frac{A^2 \cdot t^2}{2!} + \cdots = \sum_{i=0}^{\infty} A^i \cdot \frac{t^i}{i!}$
2. Využití Laplaceovy transformace: $e^{At} = \mathscr{L}^{-1}\{(pI - A)^{-1}\}$
3. Každá čtvercová matice může být převedena na diagonální: $(A = V \cdot D \cdot V^{-1})$

$$e^{At} = V \cdot e^{Dt} \cdot V^{-1} \tag{2.11}$$

4. Využití Clay-Hamiltonovy věty (*"Každá čtvercová matice (A) vyhovuje své charakteristické rovnici"*): λ_i jsou řešením charakteristické rovnice pro matici A.

Harmonická linearizace

Pro analýzu uzavřeného obvodu vycházíme z předpokladů:

1. V obvodu vznikly ustálené kmity (autooscilace) ze základní frekvencí ω_0

2. Lineární systém má charakter dolnofrekvenční propust, $|F_s(j\omega_0)| >> |F_s(jk\omega_0)|$, $k \geq 2$.
3. Nelinearita je symetrická vůči nulovému bodu.

Za těchto předpokladů je na výstupu nelinearity *periodický signál* se základní frekvencí ω_0 a můžeme jej rozložit ve Fourierovu řadu. Protože chceme nahradit statickou nelinearitu v regulační smyčce tzv. *ekvivalentním přenosem*, budeme tento přenos definovat analogicky jako u frekvenčních přenosů lineárních dynamických systémů. Ekvivalentní přenos statické nelinearity definujeme poměrem výstupního a vstupního signálu nelinearity, respektive poměrem jejich Fourierových obrazů.

2.1.3 Vlastnosti lineárních dynamických systémů. Řiditelnost, pozorovatelnost, kriteria. Vnitřní a vnější stabilita, kriteria.

(a) *stabilita*

a/ *vnitřní*: stabilita rovnovážného stavu měřeného systému $\dot{x}(t) = Ax(t)$, $x(t_0)$, $x_r = 0$

- LDS je stabilní $\iff \forall\, x_i : Re\{x_i\} < 0$
- LDS je asymptoticky stabilní $\iff \forall x(t_0) : \lim_{t\to\infty} ||x(t) - x_r|| = 0$

b/ *vnější*: BIBO: $y(t) = g(t) = c^T e^{A(t-t_0)} x(t_0) + c^T \int_{t_0}^{t} e^{A(t-\tau)} Bu(\tau) d\tau$

(b) *řiditelnost, dosažitelnost stavu*: Matice dosažitelnosti (řiditelnosti): $Q_D \coloneqq [b, Ab, ..., A^{n-1}b]$.
Systém je řiditelný (dosažitelný) $\iff h[Q_D] = n = dim(x)$.

(c) *pozorovatelnost a rekonstruovatelnost*: "Jsem schopný získat počáteční stav?"Matice pozorovatelnosti: $Q_P \coloneqq [C, A^T \cdot C, ..., (A^{n-1})^T \cdot C]^T$.
Systém je pozorovatelný $\iff h[Q_P] = n = dim(x)$.

Kritéria polynomiálně (Hautus):

- LDS je řiditelný (dosažitelný) $\iff h[pI - A, b] = n$, $\forall p = \lambda_i, i = 1, ..., n$ ($< n$ pro neřiditelný)
- LDS je pozorovatelný (rekonstruovatelný) $\iff h[pI - A, C^T]^T = n$, $\forall p = \lambda_i, i = 1, ..., n$ ($< n$ pro nepozorovatelný)

(d) *stabilizovatelnost* (oslabená řiditelnost - nebude řiditelný, ale aspoň stabilní): $u(x) = -kx(t)$, $\dot{x}(t) = (A - bk^T)$

(e) *detekovatelnost* (oslabená pozorovatelnost): $u(y) = -hy(t)$

(f) *Duální systém*: $S(A', b, C^T) = S_{dual}(A^T, C, b^T)$. Systém je v minimální realizaci, pokud je řiditelný a zároveň pozorovatelný.

(Kalmanova dekompozice systému)

2.1.4 Časové a frekvenční odezvy elementárních členů regulačních obvodů.

2.1.5 Základní typy spojitých a diskrétních regulátorů (P,PI,PID, stavové regulátory a stavové regulátory s integračním charakterem), popis, vlastnosti.

2.1.6 Struktura regulačních obvodů s jedním a dvěma stupni volnosti, přenosy v regulačním obvodu, princip vnitřního modelu.

2.1.7 Problém umístitelnosti pólů a nul nedynamickými a dynamickými regulátory. Požadavky na umístění pólů, konečný počet kroků regulace.

2.1.8 Požadavky na funkci a kvalitu regulace (přesnost regulace, dynamický činitel regulace, kmitavost, robustnost ve stabilitě a j.), omezení na dosažitelnou kvalitu regulace.

2.1.9 Metoda geometrického místa kořenů, pravidla pro konstrukci a využití při syntéze regulátorů, příklady.

2.1.10 Přístup k syntéze regulátorů v klasické teorii regulace, klasické metody, heuristické metody.

2.1.11 Deterministická rekonstrukce stavu, stavový regulátor s rekonstruktorem stavu.

2.1.12 Ljapunovova teorie stability. Ljapunovova rovnice.

2.2 Teorie odhadu [TOD]

vyučující:	Prof. Ing. Miroslav Šimandl, CSc.
	Ing. Jindřich Duník, Ph.D.
ročník/semestr studia:	3.ročník/ZS
datum zkoušky:	28. 4. 2014
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je obeznámit studenty s možnostmi odhadu parametrů, náhodných veličin a náhodných procesů v podmínkách neurčitosti z apriorních informací a měřených dat.

2.2.1 Problémy odhadu, základní etapy vývoje teorie odhadu, náhodné veličiny, náhodné procesy a jejich popis, stochastický systém.

2.2.2 Optimální odhad ve smyslu střední kvadratické chyby. Odhad ve smyslu maximální věrohodnosti.

2.2.3 Jednorázové a rekurzivní odhady.

2.2.4 Odhad stavu lineárního diskrétního systému – filtrace (Kalmanův filtr).

2.2.5 Úlohy odhadu stavu lineárního diskrétního stochastického systému – predikce a vyhlazování.

2.2.6 Odhad stavu lineárního systému se spojitým či diskrétním měřením (Kalman-Bucyho filtr).

2.3 Optimální systémy [OPS]

vyučující:	Ing. Miroslav Flídr, Ph.D.
	Ing. Ivo Punčochář, Ph.D.
ročník/semestr studia:	4.ročník/LS
datum zkoušky:	15. 7. 2015
hodnocení:	3

cíl předmětu (STAG):
Cílem předmětu je seznámení studentů s různými typy optimalizačních úloh. Studenti se naučí řešit jednak základní statické optimalizační úlohy tak především úlohy optimalizace dynamických systémů. Důraz je kladen především na pochopení řešení následujících problémů:

- časově optimální řízení;
- Pontrjaginův princip minima;
- dynamické programování a Bellmanova optimalizační rekurze;
- lineárně - kvadratická úloha optimálního řízení.

2.3.1 Optimální programové řízení diskrétních dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení.

Formulace úlohy

Dáno:

1. Systém: $x_{k+1} = f_k(x_k, u_k), \qquad k = 0, ..., N-1, \qquad x_0$ známé, $x_k \in R^n, u_k \in R^m$
2. Kritérium: $J(x_0) = \Phi(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k)$, kde $\Phi(x_N)$ je ohodnocení koncového stavu a $L_k(x_k, u_k)$ je váhová funkce.

Cílem je nalézt posloupnost řízení u_0^{N-1} takovou, aby kritérium nabývalo své minimální hodnoty při splnění vazbových podmínek $x_{k+1} - f_k(x_k, u_k) = 0$.

Rozšířené kritérium a Hamiltonova funkce

Diferenční rovnice popisující dynamiku systému jsou chápány jako vazbové podmínky. Definujeme rozšířené kritérium (v podstatě Lagrangeovu funkci):

$$\bar{J}(x_0) = \Phi(x_N) + \sum_{k=0}^{N-1} \left[L_k(x_k, u_k) + \lambda_{k+1}^T (f_k(x_k, u_k) - x_{k+1}) \right] \tag{2.12}$$

kde λ_1^N představuje Lagrangeovy multiplikátory. Při splnění vazbových podmínek nabývá rozšířené kritérium extrémů ve stejných bodech jako původní kritérium , tj. $\min J(x_0) = \min \bar{J}(x_0)$. Toto

kritérium můžeme alternativě vyjádřit ve tvaru:

$$\bar{J}(x_0) = \Phi(x_N) - \lambda_N^T x_N + \sum_{k=0}^{N-1} \left[H_k(x_k, u_k, \lambda_{k+1}) - \lambda_{k+1}^T x_{k+1} \right] + H_0(x_0, u_0, \lambda_1), \tag{2.13}$$

kde funkce $H_k(x_k, u_k, \lambda_{k+1}) = L_k(x_k, u_k) + \lambda_{k+1}^T f_k(x_k, u_k)$ se nazývá Hamiltonova funkce (Hamiltonián).

Nutné podmínky optimálního řízení

Hledané řešení vyhovuje podmínce $d\bar{J} = 0$, kde $d\bar{J}$ vyjadřuje totální diferenciál vzhledem ke všem veličinám x_0^N, u_0^{N-1} a λ_1^N. "Nulováním"jednotlivých členů definujících $d\bar{J}$ obdržíme nutné podmínky extrému. Je možné ukázat, že k vyjádření nutných podmínek je potřeba znát jen Hamiltonián. Hamiltonovy kanonické rovnice:

$$\begin{aligned}
x_{k+1} &= \left(\frac{\partial H_k}{\partial \lambda_{k+1}} \right)^T = f_k(x_k, u_k), \qquad x_0 \\
\lambda_k &= \left(\frac{\partial H_k}{\partial x_k} \right)^T = \left(\frac{\partial L_k}{\partial x_k} \right)^T + \left(\frac{\partial f_k}{\partial x_k} \right)^T \lambda_{k+1}, \qquad \lambda_N = \left(\frac{\partial \Phi}{\partial x_N} \right)^T \\
0 &= \left(\frac{\partial H_k}{\partial u_k} \right)^T = \left(\frac{\partial L_k}{\partial u_k} \right)^T + \left(\frac{\partial f_k}{\partial u_k} \right)^T \lambda_{k+1}
\end{aligned}$$

Řešení představuje dvoubodový okrajový problém, který je v obecném případě obtížně analyticky řešitelný!

Modifikace úlohy pro pevný konec

Předchozí formulace a řešení jsou platné pro úlohu s pevným časem a volným koncem. V případě pevného konce je definována dodatečná sada vazbových podmínek specifikujících koncovou podmínku pro stav ve tvaru $\Psi(x_N) = 0$, např. $\Psi(x_N) = x_N - x_F = 0$, tj. požadavek na $x_N = x_F$ pro pevně daný finální stav x_f. Rozšířené kritérium pro tuto úlohu je dáno vztahem:

$$\bar{J}(x_0) = \Phi(x_N) + \nu^T \Psi(x_N) + \sum_{k=0}^{N-1} \left[L_k(x_k, u_k) + \lambda_{k+1}^T (f_k(x_k, u_k) - x_{k+1}) \right], \tag{2.14}$$

kde ν mají též význam (dalších) Lagrangeových multiplikátorů. Nutné podmínky optimálního řízení (tj. Hamiltonovy kanonické rovnice) zůstávají nezměněny. Rozšířena je pouze koncová okrajová podmínka

$$\lambda_N = \left(\frac{\partial \Phi}{\partial x_N} \right)^T + \left(\frac{\partial \Psi}{\partial x_N} \right)^T \nu \tag{2.15}$$

2.3.2 Optimální programové řízení spojitých dynamických systémů. Formulace úlohy. Hamiltonova funkce. Nutné podmínky pro optimální řízení. Podmínky transverzality. Pontrjaginův princip minima.

Formulace úlohy s pevným časem

Dáno:

1. Systém: $\dot{x}_t = f(x(t), u(t), t), \qquad t \in \langle t_0, t_f \rangle, \qquad x(t_0)$ známé, $x(t) \in R^n, u(t) \in R^m$
2. Kritérium: $J(x(t_0)) = \Phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt$, kde $\Phi(x(t_f), t_f)$ je ohodnocení koncového stavu a $L(x(t), u(t), t)$ je váhový funkcionál.

Cílem je nalézt funkci řízení $u(t)$ jako funkci času $t \in \langle t_0, t_f \rangle$ takovou, aby kritérium nabývalo své minimální hodnoty při splnění vazbových podmínek $f(x(t), u(t), t) - \dot{x}(t) = 0$.

Rozšířené kritérium a Hamiltonova funkce

Diferenciální rovnice popisující dynamiku systému jsou chápány jako vazbové podmínky. Definujeme rozšířené kritérium (v podstatě Lagrangeovu funkci):

$$\bar{J}(x(t_0)) = \Phi(x(t_f), t_f) + \int_{t_0}^{t_f} \left[L(x(t), u(t), t) + \lambda^T(t)(f(x(t), u(t), t) - \dot{x}(t)) \right] dt \tag{2.16}$$

kde $\lambda(t)$ představuje funkci Lagrangeových multiplikátorů. Při splnění vazbových podmínek nabývá rozšířené kritérium extrémů ve stejných bodech jako původní kritérium , tj. $\min J(x_0) = \min \bar{J}(x_0)$. Toto kritérium můžeme alternativě vyjádřit ve tvaru:

$$\bar{J}(x(t_0)) = \Phi(t(x_f), t_f)) - \lambda^T(t_f)x(t_f) + \lambda^T(t_0)x(t_0) + \int_{t_0}^{t_f} \left[H(x(t), u(t), \lambda(t), t) + \lambda^T(t)x(t) \right] dt, \tag{2.17}$$

kde funkce $H(x(t), u(t), \lambda(t), t) = L(x(t), u(t), t) + \lambda^T(t) f(x(t), u(t), t)$ se nazývá Hamiltonova funkce (Hamiltonián).

Nutné podmínky optimálního řízení

Hledané řešení vyhovuje podmínce $\partial \bar{J} = 0$, kde $\partial \bar{J}$ vyjadřuje variaci funkcionálu $\bar{J}$ vzhledem k variacím funkcí $x(t)$, $u(t)$ a $\lambda(t)$. Vhodnou volbou funkce multiplikátorů a s využitím faktu že variace $\partial x(t_0) = 0$ je možné se vyhnout vyšetřování závislosti na variaci stavu $\partial \dot{x}(t) = \frac{\partial f}{\partial x}\partial x + \frac{\partial f}{\partial u}\partial u$ a zároveň obdržíme nutné podmínky optimálního řízení. Je možné ukázat, že k vyjádření nutných podmínek je potřeba znát jen Hamiltonovu funkci.

Hamiltonovy kanonické rovnice:

$$\dot{x}(t) = \left(\frac{\partial H}{\partial \lambda}\right)^T = f(x(t), u(t), t), \qquad x(t_0)$$
$$\dot{\lambda}(t) = -\left(\frac{\partial H}{\partial x}\right)^T = -\left(\frac{\partial L}{\partial x}\right)^T - \left(\frac{\partial f}{\partial x}\right)^T \lambda, \qquad \lambda(t_f) = \left(\frac{\partial \Phi(t_f)}{\partial x(t_f)}\right)^T$$
$$0 = \left(\frac{\partial H}{\partial u}\right)^T = \left(\frac{\partial L}{\partial u}\right)^T + \left(\frac{\partial f}{\partial u}\right)^T \lambda$$

Řešení představuje dvoubodový okrajový problém, který je v obecném případě obtížně analyticky řešitelný!

Úloha s volným časem - podmínka transverzality

Pro úlohu s volným časem je nutné uvažovat i vliv diferenciální změny času dt_f na variaci rozšířeného kritéria $\partial \bar{J}$. Řešení problému je opět dáno dříve uvedenými Hamiltonovými kanonickými rovnicemi. Modifikována je ovšem koncová okrajová podmínka pro určení trajektorie multiplikátorů (tato nová koncová podmínka se nazývá podmínka transverzality):

$$\left[\left(\frac{\partial \Phi}{\partial x} - \lambda^T\right) dx(t_f) + \left(\frac{\partial \Phi}{\partial t} + H\right) dt_f\right]_{t=t_f} = 0 \tag{2.18}$$

Speciální případy úlohy a příslušné podmínky transverzality:

- *Volný konec trajektorie*: Pro libovolné diferenciální změny dt_f a libovolné $dx(t_f)$ musí platit:
$$\left(\frac{\partial \Phi}{\partial x} - \lambda^T\right)_{t=t_f} = 0; \qquad \left(\frac{\partial \Phi}{\partial t} + H\right)_{t=t_f} = 0 \tag{2.19}$$

- *Pevný čas a volný konec trajektorie*: Je-li koncový čas určen, je $dt_f = 0$ a podmínky transverzality se redukují na:
$$\left(\frac{\partial \Phi}{\partial x} - \lambda^T\right)_{t=t_f} = 0 \tag{2.20}$$

- *Volný čas a pevný konec trajektorie*: Např. úloha časově optimálního řízení. Pokud je koncový stav $x(t_f)$ pevně určený v libovolném konečném čase t_f, je $dx(t_f) = 0$ a podmínky transverzality přejdou do tvaru:
$$\left(\frac{\partial \Phi}{\partial t} + H\right)_{t=t_f} = 0 \tag{2.21}$$

- *Volný čas a konec trajektorie na nadploše*: Je-li koncový bod dán jako bod nadplochy $\Psi(x(t_f)) = 0$, pak je tato podmínka chápána jako další vazbová podmínka a podmínka transvezality je dána jako:

$$\left[\left(\frac{\partial\Phi}{\partial x} - \lambda^T + \frac{\partial\Psi}{\partial x}\nu^T\right)dx(t_f) + \left(\frac{\partial\Phi}{\partial t} + H\right)dt_f\right]_{t=t_f} = 0 \tag{2.22}$$

- *Koncový bod leží na křivce* $x = \Psi(t)$: V tomto případě nejsou diferenciální změny dt_f a $dx(t_f)$ nezávislé, ale zjevně platí $dx(t_f) = \frac{d\Psi(t)}{dt}dt_f = \dot{\Psi}(t)dt_f$. Dosazením do obecné podmínky transverzality dostaneme:

$$\left[\left(\frac{\partial\Phi}{\partial x} - \lambda^T\right)\dot{\Psi}\right]_{t=t_f} = \left(\frac{\partial\Phi}{\partial t} + H\right)_{t=t_f} \tag{2.23}$$

  Je zřejmé, že pokud je křivka popsaná konstantní funkcí času $\Psi(t) = konst$, platí $\dot{\Psi} = 0$, a dostáváme úlohu s pevným koncem a volným časem.

Časově optimální řízení

Pro tuto úlohu je váhový funkcionál a ohodnocení koncového stavu dáno jako $L = 1$, a $\Phi = 0$ a kritérium optimality má tedy tvar $J = t_f - t_0$. Optimální trajektorie je funkcí času a získáme ji vyřešením dvouokrajové úlohy:

$$\begin{aligned}
&\dot{x} = f(x,u,t), \qquad x(t_0), x(t_f)\\
&\dot{\lambda} = -\left(\frac{\partial f}{\partial x}\right)^T\lambda, \qquad (\lambda^T f)_{t=t_f} = -1\\
&0 = \left(\frac{\partial f}{\partial u}\right)^T\lambda
\end{aligned}$$

Pokud řešíme *časově invariantní problém*, lze ukázat, že pro t-invariantní systém a kritérium platí, že $\dot{H}(x^*(t), u^*(t), \lambda^*(t)) = 0$, tj. H je konstantní podél optimální trajektorie.

Pontrjaginův princip minima

Máme-li dány optimální trajektorie stavu $x^*(t)$ a multiplikátorů $\lambda^*(t)$, tj. jsou splněny následující nutné podmínky:

$$\begin{aligned}
&\dot{x}(t) = \left(\frac{\partial H}{\partial\lambda}\right)^T = f(x(t),u(t),t), \qquad x(t_0) = x_0\\
&\dot{\lambda} = -\left(\frac{\partial H}{\partial x}\right)^T = -\left(\frac{\partial L}{\partial x}\right)^T - \left(\frac{\partial f}{\partial x}\right)^T\lambda, \qquad \left[\left(\frac{\partial\Phi}{\partial x} - \lambda^T\right)dx(t_f) + \left(\frac{\partial\Phi}{\partial t} + H\right)dt_f\right]_{t=t_f} = 0
\end{aligned}$$

s počáteční podmínkou $x(t_0)$ a koncovou okrajovou podmínkou danou obecnou podmínkou transverzality. Pak optimální řízení jako funkce času je hledáno jako

$$\boxed{u^*(t) = \underset{u(t)\in U\subset R^m}{\operatorname{argmin}} H(x^*(t), u(t), \lambda^*(t), t)} \tag{2.24}$$

Toto je nutná podmínka globálního minima a plně nahrazuje podmínku lokálního minima $\left(\frac{\partial H}{\partial u}\right)^T = 0$. Největší uplatnění Pontriaginův princip minima přináší pro případy, kdy jsou uvažovány omezení funkce řízení a stavu. V takovém případě platí, že $\partial \bar{J} \geq 0$ (protože již není možné libovolně variovat ∂u).

2.3.3 Deterministický diskrétní systém automatického řízení. Princip optimality. Bellmanova funkce. Bellmanova optimalizační rekurze.

Formulace úlohy

Dáno:

1. Systém: $x_{k+1} = f_k(x_k, u_k), \qquad k = 0, ..., N-1, \qquad x_0$ známé, $x_k \in R^n, u_k \in R^m$
2. Kritérium: $J(x_0, u_0^{N-1}) = \left[\Phi(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k)\right]_{\text{P.T.S}}$, kde $\Phi(x_N)$ je ohodnocení koncového stavu a $L_k(x_k, u_k)$ je váhová funkce.

Cílem je nalézt posloupnost strategií řízení g_0^{N-1}, které generují posloupnost řízení $u_0^{N-1} \in U_0^{N-1}$ takovou, aby kritérium nabývalo své minimální hodnoty.

Koncová část kritéria optimality - princip optimality

Koncová část kritéria optimality je definována jako:

$$J_k(x_k, u_k^{N-1}) = \left[\Phi(x_N) + \sum_{l=k}^{N-1} L_l(x_l, u_l)\right]_{\text{P.T.S.}} \tag{2.25}$$

To může být vyjádřeno rekurentním vztahem:

$$J_k(x_k, u_k^{N-1}) = \left[L_k(x_k, u_k) + J_{k+1}(x_{k+1}, u_{k+1}^{N-1})\right]_{\text{P.T.S.}} = \left[L_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k), u_{k+1}^{N-1})\right]_{\text{P.T.S.}}$$

Mějme optimální posloupnost strategií řízení g_0^{N-1} generující optimální posloupnost řízení u_0^{N-1}. Rozdělíme-li trajektorii v čase k na neměnnou minulost a neuskutečněnou budoucnost, pak bude platit, že:

$$J^*(x_0) = J_0(x_0, u_0^{*N-1}) + \left[\sum_{l=0}^{k-1}(L_l(x_l^*, u_l^*)) + J_k(x_k^*, u_k^{*N-1})\right]_{\text{P.T.S.}}$$

Princip optimality: *Každý koncový úsek optimální trajektorie je optimálním koncovým úsekem trajektorie.*

Bellmanova funkce

Bellmanova funkce je definována jako minimální hodnota koncového úseku kritéria:

$$V_k(x_k) = \min_{u_k^{N-1} \in U_k^{N-1}} J_k(x_k, u_k^{N-1}) \tag{2.26}$$

S využitím rekurentního vztahu pro koncovou část kritéria lze Bellmanovu funkci vyjádřit jako:

$$V_k(x_k) = \min_{u_k \in U_k} \left[L_k(x_k, u_k) + \min_{u_{k+1}^{N-1} \in U_{k+1}^{N-1}} J_{k+1}(x_{k+1}, u_{k+1}^{N-1}) \right] \tag{2.27}$$

Bellmanova optimalizační rekurze

$$\begin{aligned} V_k(x_k) &= \min_{u_k \in U_k} \left[L_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k)) \right], \qquad k = N-1, N-2, ..., 0 \\ u_k &= g_k(x_k) = \operatorname*{argmin}_{u_k \in U_k} \left[L_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k)) \right], \qquad k = N-1, N-2, ..., 0 \\ V_N(x_N) &= J_N(x_N) = \Phi(x_N) \end{aligned}$$

Minimální hodnota kritéria je dána jako $J^*(x_0) = J_0(x_0, u_0^{*N-1}) = V_0(x_0)$

2.3.4 Syntéza optimálního deterministického systému automatického řízení pro diskrétní lineární řízený systém a kvadratické kritérium. Formulace a řešení. Asymptotické řešení a jeho stabilita.

Formulace úlohy

Dáno:

1. Systém: $x_{k+1} = A_k x_k + B_k u_k, \qquad k = 0, ..., N-1, \qquad x_0$ známé, $x_k \in R^n, u_k \in R^m$
2. Kritérium: $J(x_0, u_0^{N-1}) = \left[x_N^T Q_N x_N + \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right]_{\text{P.T.S}}$,

   kde je $\begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} = \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix}^T, \qquad Q_k \geq 0, R_k > 0.$

Cílem je nalézt posloupnost strategií řízení $g_k(x_k) \, \forall k$, které generují posloupnost řízení $u_0^{N-1} \in U_0^{N-1}$ takovou, aby kritérium nabývalo své minimální hodnoty.

Bellmanova funkce

Indukcí lze dokázat, že Bellmanova funkce je dána kvadratickou formou $V_k(x_k) = x_k^T P_k x_k \, \forall k$, kde $P_k = P_k^T$ je pozitivně semidefinitní matice. Pro koncový čas pak platí $V_N(x_N) = x_N^T Q_N x_N$, z čehož vyplývá, že $P_N = Q_N$.

Bellmanova optimalizační rekurze

Předpokládáme-li, že v čase $k+1$ je Bellmanova funkce dána kvadratickou formou $V_{k+1}(x_{k+1}) = x_{k+1}^T P_{k+1} x_{k+1}$, pak je optimalizační rekurze dána vztahy:

$$V_k(x_k) = \min_{u_k \in U_k} \left[\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + V_{k+1}(A_k x_k + B_k u_k) \right], \qquad k = N-1, N-2, ..., 0$$

$$u_k = g_k(x_k) = \operatorname*{argmin}_{u_k \in U_k} \left[\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + V_{k+1}(A_k x_k + B_k u_k) \right], \qquad k = N-1, N-2, ..., 0$$

$$V_N(x_N) = x_N^T Q_N x_N$$

Řešení lineárně-kvadratické úlohy (LQ úlohy)

1. Úloha má vždy řešení a toto řešení je jediné.
2. Bellmanova funkce je pozitivně semidefinitní kvadratická forma $V_k(x_k) = x_k^T P_k x_k \, \forall k$, $P_k = P_k^T \geq 0$
3. Řešení je dáno pro okamžiky $k = N-1, ..., 0$ řešením *Riccatiho diferenční rovnice*:

   $$P_k = A_k^T P_{k+1} A_k + Q_k - (A_k^T P_{k+1} B_k + S_k)(B_k^T P_{k+1} B_k + R_k)^{-1}(A_k^T P_{k+1} B_k + S_k)^T \quad (2.28)$$

   s koncovou okrajovou podmínkou $P_N = Q_N$.
4. Optimální zpětnovazební strategie řízení při měřeném stavu je:

   $$u_k(x_k) = g_k(x_k) = -(B_k^T P_{k+1} B_k + R_k)^{-1}(A_k^T P_{k+1} B_k + S_k)^T x_k = -K_k x_k$$

5. Hodnota kritéria optimality podél trajektorie optimálního (uzavřeného) systému je:

   $$J^*(x_0) = x_0^T P_0 x_0$$

Asymptotická LQ úloha

- Uvažujeme t-invariantní systém a nekonečný časový horizont:

  $$x_{k+1} = A_k x_k + B_k u_k, \qquad k = 0, 1, ... \quad (2.29)$$

  kde dvojice (A, B) je řiditelná a počátek je rovnovážným stavem, tj. $f(0,0) = 0$.
- Kritérium je též uvažováno jako t-invariantní:

  $$J(x_0, u_0^\infty) = \left[\sum_{k=0}^{\infty} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right]_{\text{P.T.S}} \quad (2.30)$$

  Pro rovnovážný stav v počátku platí, že $L(0,0) = 0$ a tedy, že i při nekonečném horizontu řízení nabývá kritérium konečné hodnoty.

Pro řešení asymptotické LQ úlohy platí:

1. Bellmanova funkce je pozitivně semidefinitní *časově invariantní* kvadratická forma $V_k(x_k) = x_k^T P x_k \, \forall k,\ P = P^T \geq 0$
2. Matice P definující kvadratickou formu Bellmanovy funkce je dáno řešením *Riccatiho diferenční rovnice*:

$$P = A^T P A + Q - (A^T P B + S)(B^T P B + R)^{-1}(A^T P B + S)^T \tag{2.31}$$

3. Optimální zpětnovazební strategie řízení při měřeném stavu je též *časově invariantní*:

$$u_k(x_k) = g(x_k) = -(B^T P B + R)^{-1}(A^T P B + S)^T x_k = -K x_k$$

4. Hodnota kritéria optimality podél trajektorie optimálního (uzavřeného) systému je:

$$J^*(x_0) = x_0^T P x_0$$

Podmínky stability asymptotického řešení LQ úlohy:

- Je-li Bellmanova funkce Ljapunovovou funkcí, tj. řešení Riccatiho algebraické rovnice P je pozitivně definitní, pak je rovnovážný stav v počátku ($x = 0$) globálně asymptoticky stabilní.
- Je-li dvojice (A, B) řiditelná a dvojice $(C, \bar{A})$ pozorovatelná, pak je rovnovážný stav globálně asymptoticky stabilní. Matice C a $\bar{A}$ jsou definovány následovně:

$$C^T C = Q - S R^{-1} S^T; \qquad \bar{A} = A + B R^{-1} S^T$$

  Poznámky:

  - Zavádí se fiktivní pozorování $z_k = C x_k$ skrze kritérium $J(x_0, u_0^\infty = \sum_{k=0}^{\infty} z_k^T z_k + u_k^T R u_k$.
  - Je-li tato podmínka splněna je matice P zároveň pozitivně definitní.
  - Stabilita je zaručena i pro případ detekovatelné dvojice $(C, \bar{A})$, tj. jsou-li nepozorovatelné póly stabilní, a matice P je pak pozitivně semidefinitní.

2.3.5 Deterministický spojitý systém automatického řízení. Kontinualizace Bellmanovy optimalizační rekurze.

Formulace úlohy

Dáno:

1. Systém: $\dot{x}(t) = f(x(t), u(t), t), \qquad t \in \langle t_0, t_f \rangle, \qquad x(t_0)$ známé, $x(t) \in R^n, u(t) \in R^m$
2. Kritérium: $J(x(t_0), u_{t_0}^{t_f}) = \left[\Phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) dt\right]_{\text{P.T.S}}$, kde $\Phi(x(t_f), t_f)$ je ohodnocení koncového stavu a $L(x(t), u(t), t)$ je váhový funkcionál.

Cílem je nalézt funkci strategií řízení $g(x(t),t)$, pro $t \in \langle t_0, t_f \rangle$, která generuje řízení $u(t)$ pro $t \in \langle t_0, t_f \rangle$ takové, aby kritérium nabývalo své minimální hodnoty.

Využití známého řešení pro diskrétní systém pomocí kontinualizace

Řešení hledáme pro diskrétní systém: $x(t_k + h) \approx x(t_k) + f(x(t_k), u(t_k), t_k) \cdot h$, kde $h = \frac{(t_f - t_0)}{N}$ a $t_k = t_0 + kh$, $k = 0, ..., N-1$. Kritérium kvality uvažujeme ve tvaru:

$$J(x_0, u_{t_0}^{t_f}) \approx \left[\Phi(x(t_N), t_N) + \sum_{k=0}^{N-1} L(x(t_k), u(t_k), t_k) h \right] \tag{2.32}$$

Bellmanovu optimalizační rekurzi pro tento problém vyjádříme následovně:

$$V(x(t),t) = \min_{u(t) \in U} [L(x(t), u(t), t)h + V(x(t+h), t+h)]_{\text{P.T.S.}}$$
$$g(x(t),t) = \operatorname*{argmin}_{u(t) \in U} [L(x(t), u(t), t)h + V(x(t+h), t+h)]_{\text{P.T.S.}}$$

s okrajovou podmínkou $V(x(t_f), t_f) = \Phi(x(t_f), t_f)$. Problém pak řešíme pro $h \to 0+$, tj. kontinualizací diskrétního řešení.

Bellmanova funkce pro spojitý deterministický systém

Je dána ve formě parciální diferenciální rovnice:

$$-\frac{\partial V(x,t)}{\partial t} = \min_{u(t) \in U} \left[L(x,u,t) + \frac{\partial V(x,t)}{\partial x} f(x,u,t) \right] \tag{2.33}$$

s okrajovou podmínkou $V(x(t_f), t_f) = \Phi(x(t_f), t_f)$. Tato parciální diferenciální rovnice je obvykle nazývána Hamiltonova-Belmanova-Jacobiho parciální diferenciální rovnice. *Strategie řízení* je pak hledána jako:

$$g(x,t) = \operatorname*{argmin}_{u(t) \in U} \left[L(x,u,t) + \frac{\partial V(x,t)}{\partial x} f(x,u,t) \right] \tag{2.34}$$

Asymptotická úloha optimálního řízení pro spojitý systém

Předpoklady pro asymptotickou úlohu:

1. Uvažujeme t-invariantní systém a nekonečný časový horizont:

$$\dot{x}(t) = f(x(t), u(t)), \qquad t = \langle 0, \infty \rangle, \qquad f(0,0) = 0$$

2. Kritérium je též uvažováno jako t-invariantní:

$$J(x(t_0), u_{t_0}^{\infty}) = \left[\int_{t_0}^{\infty} L(x(t), u(t)) dt \right]_{\text{P.T.S.}}, \qquad L(x(t), u(t)) \geq 0 \, \forall t \wedge L(0,0) = 0$$

Bellmanova funkce a strategie řízení (t-invariantní, tj. $\frac{\partial V}{\partial t} = 0$):

$$0 = \min_{u \in U} \left[L(x,u,t) + \frac{\partial V(x)}{\partial x} f(x,u) \right]$$
$$g(x) = \operatorname*{argmin}_{u \in U} \left[L(x,u,t) + \frac{\partial V(x)}{\partial x} f(x,u) \right]$$

Asymptotická LQ úloha pro spojitý deterministický systém

1. Úloha má řešení a to jediné.
2. Matice P Bellmanovy funkce je časově invariantní symetrická pozitivně semidefinitní matice, která je řešením *Riccatiho diferenční rovnice*:
$$0 = PA + A^T P - (PB + S)R^{-1}(PB + S)^T + Q \tag{2.35}$$
3. Je-li navíc pozorovatelná dvojice $(\bar{A}, C)$, kde:
$$C^T C = Q - SR^{-1}S^T; \qquad \bar{A} = A + BR^{-1}S^T$$
je optimální systém *globálně asymptoticky stabilní* a matice P pozitivně definitní.
4. Hodnota kritéria podél optimální trajektorie je:
$$J^*(x_0) = x_0^T P x_0$$

2.3.6 Optimální stochastický systém automatického řízení. Strategie řízení. Bellmanova funkce a Bellmanova optimalizační rekurze.

Formulace úlohy

Dáno:

1. Systém: $\varphi_k(x_k, y_k, x_{k-1}, y_{k-1}, u_{k-1})$, kde $\begin{bmatrix} x_k \\ y_k \end{bmatrix} \in R^n$ představuje stav, dále $u_k \in U_k \subset R^m$ a $\varphi_0(x_0, y_0, 0)$ představuje počáteční podmínku.
2. Kritérium:
$$J(\gamma_0^N) = E\left[L(x_0^N, y_0^N, u_0^N), \gamma_0^N\right] = \int\int\int L(x_0^N, y_0^N, u_0^N) p(x_0^N, y_0^N, u_0^N, \gamma_0^N) dx_0^N dy_0^N du_0^N$$
$$L(x_0^N, y_0^N, u_0^N) = \sum_{k=0}^{N-1} L_k(x_k, u_k), \gamma_0^N$$

Cílem je nalézt posloupnost strategií řízení $\gamma_k(y_0^k, u_0^{k-1}) \in \Gamma_k^N$, která pro $k = 0, ..., N-1$ generuje řízení u_k takové, aby kritérium nabývalo své minimální hodnoty.

Kritérium optimality koncového úseku strategie řízení

$$J_k(\gamma_k^N, y_0^k, u_0^{k-1} = E\left[\sum_{l=k}^{N} L_l(x_l, y_l, u_l), y_0^k, u_0^{k-1}, \gamma_k^N\right]$$

které může být vyjádřeno rekurentním vztahem

$$J_k(\gamma_k^N, y_0^k, u_0^{k-1} = \int_{\infty}^{\infty} E\left[L_k(x_k, y_k, u_k) + J_{k+1}(\gamma_{k+1}^N, y_0^{k+1}, u_0^k), y_0^k, u_0^k\right], \gamma_k(u_k, y_0^k, u_0^{k-1}) du_k$$

V případě deterministické strategie řízení $\gamma_k^N = \delta(u_k - g_k(y_0^k, u_0^{k-1}))$ je kritériem optimality koncového úseku strategie řízení daného vztahem:

$$J_k(\gamma_k^N, y_0^k, u_0^{k-1} = E\left[L_k(x_k, y_k, u_k) + J_{k+1}(\gamma_{k+1}^N, y_0^{k+1}, u_0^k), y_0^k, u_0^k\right]$$

Okrajová podmínka je v obou případech $J_{N+1} = 0$. Hodnota kritéria podél optimální trajektorie systému je definována jako $J(\gamma_0^N) = E\left[J_0(\gamma_0^N, y_0)\right]$.

Bellmanova funkce

Bellmanova funkce je definována jako minimální hodnota koncového úseku kritéria:

$$V_k(y_0^k, u_0^{k-1}) = \min_{\gamma_k^N \in \gamma_k^N} J_k(\gamma_k^N, y_0^k, u_0^{k-1})$$
$$\gamma_0^N = \operatorname*{argmin}_{\gamma_k^N \in \gamma_k^N} J_k(\gamma_k^N, y_0^k, u_0^{k-1})$$

Optimalizace je opět založena na rekurzivním výpočtu hodnoty Bellmanovy funkce.

Bellmanova optimalizační rekurze

$$V_k(y_0^k, u_0^{k-1}) = \min_{u_k \in u_k} E\left[L_k(x_k, y_k, u_k) + V_{k+1}(y_0^{k+1}, u_0^k), y_0^k, u_0^k\right]$$
$$g_k(y_0^k, u_0^{k-1}) = \operatorname*{argmin}_{u_k \in u_k} E\left[L_k(x_k, y_k, u_k) + V_{k+1}(y_0^{k+1}, u_0^k), y_0^k, u_0^k\right]$$
$$\gamma_0^*(u_k, y_0^k, u_0^{k-1}) = \delta(u_k - g_k(y_0^k, u_0^{k-1}))$$

Okrajová podmínka rekurze je $V_{N+1}(y_0^N, u_0^N) = 0$ a hodnota kritéria optimality pro optimální strategii je $J^*(\gamma_0^{*N}) = E[V_0(y_0)]$.

2.3.7 Syntéza optimálního systému automatického řízení pro lineární gaussovský řízený systém a kvadratické kritérium. Formulace a řešení. Separační teorém.

Formulace problému

Dáno:

1. Systém:

$$\begin{aligned}\varphi_k(x_k : x_{k-1}, u_{k-1}) &= \mathscr{N}(A_{k-1}x_{k-1} + B_{k-1}u_{k-1}, G_{k-1}G_{k-1}^T\\ \varphi_0(x_0 : 0) &= \mathscr{N}(\hat{x}_0, \Pi_0)\\ \Psi_k(y_k : x_k &= \mathscr{N}(C_k x_k, H_k H_k^T)\end{aligned}$$

2. Kritérium:

$$J(g_0^{N-1}) = E\left[x_N^T Q_N x_N + \sum_{k=0}^{N-1} \begin{bmatrix}x_k\\u_k\end{bmatrix}^T \begin{bmatrix}Q_k & S_k\\S_k^T & R_k\end{bmatrix} \begin{bmatrix}x_k\\u_k\end{bmatrix}; g_0^{N-1}\right]$$

Cílem je nalézt posloupnost strategií řízení $g_k(x_k)\,\forall k$, které generují posloupnost řízení $u_0^{N-1} \in U_0^{N-1}$ takovou, aby kritérium nabývalo své minimální hodnoty.

Alternativní ekvivalentní popisy systému

Popis pomocí stochastické rovnice s gaussovskými normovanými šumy:

$$\begin{aligned}x_{k+1} &= A_k x_k + B_k u_k + G_k \xi_k\\ y_k &= C_k x_k + H_k \eta_k\\ p(x_0) &= \mathscr{N}(\hat{x}_0, \Pi_0)\end{aligned}$$

kde $\xi_k \in R^n$ a $\eta_k \in R^p$ jsou navzájem nezávislé gaussovské normované absolutně náhodné diskrétní procesy s rozložením $\mathscr{N}(0, I)$. Oba tyto procesy jsou též nezávislé s náhodným počátečním stavem x_0.

Popis pomocí stochastické rovnice s gaussovskými nenormovanými šumy:

$$\begin{aligned}x_{k+1} &= A_k x_k + B_k u_k + w_k\\ y_k &= C_k x_k + v_k\\ p(x_0) &= \mathscr{N}(\hat{x}_0, \Pi_0)\end{aligned}$$

kde $w_k \in R^n$ a $v_k \in R^p$ jsou navzájem nezávislé gaussovské absolutně náhodné diskrétní procesy s rozložením $\mathscr{N}(0, G_{k-1}G_{k-1}^T)$ a $\mathscr{N}(0, H_{k-1}H_{k-1}^T)$. Oba tyto procesy jsou opět nezávislé s náhodným počátečním stavem x_0.

Kalmanův filtr

Stav systému x_k není přímo měřitelný a proto je nutné nalézt jeho odhad. Pro lineární gaussovský systém poskytuje optimální odhad stavu Kalmanův filtr.

Filtrační hustota pravděpodobnosti:

$$\begin{aligned} p(x_k|y_0^k, u_0^k) &= \mathscr{N}(u_k(y_0^k, u_0^{k-1}), \Pi_{k|k}) \\ \mu_k &= \hat{x}_k + K_k^F(y_k - C_k\hat{x}_k) \\ K_k^F &= \Pi_{k|k-1}C_k^T(H_kH_k^T + C_k\Pi_{k|k-1}C_k^T) \\ \Pi_{k|k} &= \Pi_{k|k-1} - K_k^F C_k \Pi_{k|k-1} \end{aligned}$$

Prediktivní hustota pravděpodobnosti:

$$\begin{aligned} p(x_{k+1}|y_0^k, u_0^k) &= \mathscr{N}(\hat{x}_{k+1}(y_0^k, u_0^{k-1}), \Pi_{k+1|k}) \\ \hat{x}_{k+1} &= A_k\mu_k + B_k u_k \\ \Pi_{k+1|k} &= A_k\Pi_{k|k}A_k^T + G_kG_k^T \end{aligned}$$

Počáteční podmínka: $p(x_0) = \mathscr{N}(\hat{x}_0, \Pi_0))$.

Bellmanova funkce

Indukcí lze dokázat, že Bellmanova funkce je dána kvadratickou formou:

$$V_k(y_0^k, u_0^{k-1} = \mu_k^T(y_0^k, u_0^{k-1})P_k\mu_k(y_0^k, u_0^{k-1}) + K_k, \forall k$$

kde $P_k = P_k^T$ je pozitivně semidefinitní matice. Pro koncový čas pak platí:

$$V_N(y_0^N, u_0^{N-1}) = \mu_N^T P_N \mu_N + K_N$$

Bellmanova optimalizační rekurze

Pro časové okamžiky $k = N-1, ..., 0$ platí rekurentní vztahy, po jejichž úpravě dostaneme Bellmanovu funkci a zákon řízení ve tvaru:

$$\begin{aligned} V_k(y_0^k, u_0^{k-1}) &= \min_{u_k \in u_k} \left[\begin{bmatrix} u_k \\ u_k \end{bmatrix}^T \begin{bmatrix} A_k^T P_{k+1} A_k + Q_k & A_k^T P_{k+1} B_k + S_k \\ B_k^T P_{k+1} A_k + S_k^T & B_k^T P_{k+1} B_k + R_k \end{bmatrix} \begin{bmatrix} u_k \\ u_k \end{bmatrix} \right] + K_k \\ g_k(y_0^k, u_0^{k-1}) &= \operatorname*{argmin}_{u_k \in u_k} \left[\begin{bmatrix} u_k \\ u_k \end{bmatrix}^T \begin{bmatrix} A_k^T P_{k+1} A_k + Q_k & A_k^T P_{k+1} B_k + S_k \\ B_k^T P_{k+1} A_k + S_k^T & B_k^T P_{k+1} B_k + R_k \end{bmatrix} \begin{bmatrix} u_k \\ u_k \end{bmatrix} \right] \end{aligned}$$

Je zřejmé, že $V_k(y_0^k, u_0^{k-1}) = V_k(\mu_k)$ a $g_k(y_0^k, u_0^{k-1}) = g_k(\mu_k)$.

Separační teorém

Porovnáním s řešením deterministické LQ úlohy optimálního řízení zjistíme, že platí tzv. Separační teorém: *Řešení LQG úlohy lze rozdělit na řešení dvou úloh a to:*

1. Návrh optimálního stavového regulátoru pro ekvivalentně určitý (deterministický) systém.
2. Návrh optimálního estimátoru stavu generujícího jeho střední hodnotu podmíněnou dostupným pozorováním.

Tj. v zákonu řízení nahradíme skutečné hodnoty stavu v regulátoru jejich odhady.

Řešení LQG úlohy

2.4 Adaptivní systémy [AS]

vyučující:	Ing. Jindřich Duník, Ph.D.
	Ing. Ladislav Král, Ph.D.
ročník/semestr studia:	5.ročník/ZS
datum zkoušky:	12. 12. 2016
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je obeznámit studenty s adaptivními systémy automatického řízení a adaptivními systémy zpracování signálů.

2.4.1 Základní přístupy k syntéze adaptivních řídicích systémů, schematické vyjádření, srovnání s předpoklady a návrhem standardních regulátorů.

2.4.2 Adaptivní řízení s referenčním modelem, MIT pravidlo, využití Ljapunovovy teorie stability.

2.4.3 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku řízení, přiřazení pólů, diofantické rovnice, minimální variance.

2.4.4 Samonastavující se regulátory, charakteristika a základní přístupy k návrhu bloku poznávání, parametrické metody odhadu.

2.4.5 Adaptivní systémy na zpracování signálu, adaptivní prediktor, adaptivní filtr, analogie se samonastavujícími se regulátory.

2.4.6 Adaptivní řízení a strukturální vlastnost stochastického optimálního řízení, duální řízení, neutralita, separabilita, ekvivalence určitosti.

Kapitola 3

Aplikovaná kybernetika [AKSZ]

3.1 Umělá inteligence [UI]

vyučující:	Prof. Ing. Josef Psutka, CSc.
	Ing. Aleš Pražák, Ph.D.
ročník/semestr studia:	2.ročník/ZS
datum zkoušky:	X. X. 2012
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními problémovými oblastmi umělé inteligence (UI) a naučit je aplikovat vybrané metody řešení úloh, reprezentace znalostí v UI a hraní her.

3.1.1 Metody řešení úloh v UI

3.1.2 Logické formalizmy pro reprezentaci znalostí. Predikátový počet 1. řádu. Rezoluční metoda.

3.1.3 Produkční systém. Báze znalostí a báze dat. Dopředné a zpětné šíření.

3.1.4 Síťové formalizmy pro reprezentaci znalostí. Sémantické sítě. Rámce. Scénáře.

3.1.5 Metody hraní her v UI. Procedura minimax, alfa-beta prořezávání.

3.2 Modelování a simulace 1 [MS1]

vyučující:	Ing. Václav Hajšman, Ph.D. Ing. Jindřich Liška, Ph.D. Ing. Miloš Fetter
ročník/semestr studia:	2.ročník/ZS
datum zkoušky:	X. X. 2012
hodnocení:	1

cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními principy modelování dynamických systémů.

3.2.1 Systém, model, modelování, simulace, systémová analýza.

Systém

Objekty reálného světa jsou složité, těžko pozorovatelné. Pro studium objektů však máme určitý důvod, sledujeme určitý cíl a tedy nemusíme studovat objekty zcela obecně v plné složitosti – provádíme zjednodušení. Systémy jsou nástrojem studia objektů reálného světa, jsou zjednodušeným, abstraktním pohledem na objekty reálného světa. Systém je složitá entita tvořená vzájemně působícími prvky sloužící společnému účelu nebo podřízená společnému cíli (je-li podstata působení předávání informací, hovoříme o kybernetickém systému).

- *Abstrakce*: zanedbání, vyloučení těch skutečností ze studia systému, které nejsou z hlediska sledovaného účelu (cíle) podstatné
- *Dekompozice*: rozklad systému na dílčí části – subsystémy a prvky
- *Hierarchie*: zachycení souvislostí mezi částmi systému vyjádřením sounáležitost částí systému ve smyslu nadřazenosti a podřízenosti, princip hierarchie v kombinaci s principem dekompozice vede k analýze systému od celku k částem, k postupnému, interaktivnímu zpřesňování a zjemňování popisu systému.
- *Modularita*: specifikace částí systému vykazujících jistou míru samostatnosti a minimální počet vzájemných vazeb

Model, modelování

Jsou dány dva objekty X a Y a pozorovatel. Objektu X říkáme, že je modelem objektu Y, jestliže pozorovatel může použít objektu X k získání odpovědí na otázky, které se týkají objektu Y. Modelování není samostatným vědním oborem, jedná se o soubor principů, přístupů, metod k tvorbě modelů (definování systémů).

Simulační model, simulace

Metoda získávání nových znalostí o systému na základě řízeného experimentování s jeho modelem. Simulační model – model určený (vhodný) pro simulaci (napodobení) chování reálného

objektu. Provedeme-li nějaké řízené pozorování na reálném objektu, říkáme, že jsme provedli experiment. Provedeme-li takové řízené pozorování na modelu, označujeme ho simulací.

- *Výhody*: cena, rychlost, bezpečnost, možnosti popisu velmi složitých systémů, u kterých by bylo použití jiných metod velmi složité nebo nemožné (například analytické řešení – aplikovatelné na jednoduché systémy nebo zjednodušené popisy složitých systémů)
- *Nevýhody*: náročnost tvorby simulačních modelů, problém ověření validity modelu, problém nepřesnosti a nestability numerických metod, výpočetní náročnost, jednorázovost

Vytvoření abstraktního modelu – formulace zjednodušeného popisu zkoumaného systému. Vytvoření simulačního modelu – zápis abstraktního modelu ve formě počítačového programu. Simulace – řízené experimentování se simulačním modelem.

Systémová analýza

- *izomorfní* vs. *homomorfní* systémy: Izomorfní systémy jsou nerozlišitelné od sebe pro pozorovatele, který sleduje pouze jejich vstupy a výstupy. Při homomorfismu existuje jednoznačné přiřazení mezi stavem systému A a B a nejednoznačné zpětné přiřazení – systém B získaný ze systému A zjednodušením, je homomorfním modelem systému A (shoda ve vybraných veličinách, nesymetrická relace). Vytváření homomorfních systémů je principem modelování, abstraktní model je homomorfní k modelovanému systému, simulační model je zpravidla izomorfní k abstraktnímu modelu.
- *spojité* vs. *diskrétní* systémy: Dělení dle povahy interakce jejich prvků v čase.
- *bezsetrvačné* vs. *dynamické* systémy: Dělení dle struktury. Dynamické systém obsahují setrvačnost (paměť) - např. integrátor, sekvenční logické obvody (sítě) vs. zesilovač, kombinační logické obvody
- *deterministické* vs. *stochastické*: Dělení dle chování. U deterministických systémů jsou hodnoty proměnných v každém okamžiku přesně definovány, stochastické obsahují náhodu.

3.2.2 Modelování systému diskrétních událostí, diskrétní simulace.

Diskrétní simulace je simulace prováděná s diskrétními simulačními modely. V anglické literatuře se pro diskrétní simulaci používá výhradně termínu "discrete event simulation", tedy simulace diskrétních událostí. Diskrétní simulační modely jsou modely diskrétní v čase i veličinách. Nejčastějším případem diskrétních modelů jsou aplikace teorie hromadné obsluhy.

- diskrétní v čase, časové okamžiky zpravidla neekvidistantní
- diskrétní ve veličinách

- stochastické chování
- proměnný počet prvků systému
- výskyt front reprezentovaných zpravidla spojovými seznamy
- vysoký stupeň paralelismu a z toho vyplývající vysoké nároky na řízení programu

Událost je změna, jež je elementární a okamžitá (s nulovou dobou trvání). Události odpovídají diskrétním časovým okamžikům, v nichž se něco děje (dochází ke změně stavu systému). Proces je posloupnost logicky na sebe navazujících událostí. Neprovádí se celý najednou, v jednom časovém okamžiku se provádí pouze část odpovídající právě jedné události (reakce na událost). Dílčí části procesu jsou od sebe odděleny tzv. plánovacími příkazy, které způsobí pozastavení procesu spojené s předáním řízení jinému procesu. Opětovná aktivace procesu vede k pokračování od místa posledního pozastavení.

- *Aktivní*: Právě běžící proces, v daném okamžiku může být jen jeden
- *Ukončený*: Proces, který ukončil svoji operační část, nemůže být již nikdy aktivován
- *Pozastavený*: Proces naplánovaný k provedení v určitém čase, pokud nedojde ke zrušení zaplánování jiným procesem nebo nedojde k ukončení simulace, bude v tomto čase proveden
- *Pasivní*: Přímo (přímá aktivace) nebo nepřímo (prostřednictvím zaplánování) aktivovatelný proces jiným procesem

Plánování událostí je obecně vázáno na splnění určité podmínky (dosažení určité hodnoty simulárního času, dosažení určitého stavu modelovaného systému) - časové vs. podmínkové plánování. Nástroje: Simula, J-Sim, JavaSim, javaSimulation, ...

3.2.3 Simulační experiment, studie, analýza rizika, náhoda v simulačních úlohách.

Simulační pokus (experiment): Jeden experiment se simulačním modelem (fixované parametry simulačního modelu, fixovaná násada generátoru pseudonáhodných čísel).

Simulační studie: Posloupnost simulačních pokusů majících stejný účel s cílem zjistit a analyzovat chování simulovaného systému při různých modifikacích nebo působení různých vlivů (proměnný určitý parametr simulačního modelu, fixovaná násada generátoru pseudonáhodných čísel).

Analýza rizika (posouzení relevantnosti, důvěryhodnosti získaných výsledků):

- Posouzení vlivu změny parametru na získané výsledky.
- Posouzení vlivu náhody na získané výsledky. Po nalezení optimální hodnoty parametru simulačního modelu provádíme ověřování vlivu náhody – měníme násadu generátoru pseudonáhodných čísel pro fixovanou hodnotu parametru simulačního modelu.

$\text{mira rizika} = \frac{\text{pocet pokusu s I<I}_{\text{mez}}}{\text{pocet vsech pokusu}}$, kde I_{mez} je mezní hodnota kritéria optimality.

Náhoda v simulačních úlohách

Jednou ze základních charakteristických vlastností systémů diskrétních událostí je stochastický charakter chování. Simulační model musí tuto skutečnost respektovat → nutnost specifikace základních parametrů stochastických procesů (příchody zákazníků, doby obsluh, poruchy, ...).

- Pro simulační modely koncepčních systémů (systémů definovaných nad fiktivními, neexistujícími objekty) – odhad získaný na základě zkušeností s chováním obdobných reálně existujících systémů
- Pro simulační modely reálných systémů (systémů definovaných nad reálnými objekty) - odhad získaný na základě analýzy získaných experimentálních dat nebo jejich přímé využití

Implementace náhody v simulačním modelu, získání hodnot náhodných veličin:

- volba a parametrizace teoretické distribuční funkce, tj. funkce dané exaktním vzorcem (zpravidla využívána standardní rozložení pravděpodobnosti - normální, exponenciální, Poissonovo, rovnoměrné, ...)
- specifikace empirické distribuční funkce (zpravidla schodovité nebo po částech lineární), jejíž hodnoty se získají analýzou experimentálních dat
- přímé využití experimentálních dat

Diskrétní náhodné proměnné nabývají konečně nebo spočetně mnoho různých hodnot. U spojitých rozložení pravděpodobnosti hodnoty spojitě vyplňují určitý interval.

3.2.4 Modelování v netechnických oborech (kompartmenty, buněčné automaty, ...).

Speciální simulační techniky a nástroje - jednoduché účelově orientované prostředky pro řešení simulačních úloh v technické praxi a především v netechnických oborech. Popis reálného světa v logice a pojmech blízkých uživateli – odborníkovi z dané oblasti (biologie, lékařství, ...). Jejich užití nevyžaduje detailní znalosti z matematiky, programování, ... – problém řešen intuitivně v grafickém rozhraní. Pracovat se musí velmi opatrně, podmínky nikdy zcela neodpovídají reálným!

Dva přístupy k analýze a vytváření modelu:

- *deduktivní*: potřeba přesné znalosti vyšetřovaných jevů a vstupních podmínek (teoretický přístup – problém!)
- *induktivní*: neznáme přesně fyzikální zákonitosti či nejsou odpovídající podmínky (medicína), jde o znalost dynamiky daného děje, ne o matematická pravidla

Techniky užívané v inženýrské biologii a lékařství

(a) *Metoda řešení diferenční či diferenciální rovnice*

(i) *Forresterova (systémová) dynamika*: povaha problému - porodnost, úmrtnost (matematické rovnice se sestavují dle grafického zobrazení)

(ii) *Ekvivalence elektrickým schématem*: vhodné pro modelování systémů, u kterých dochází k transportu látky v prostoru i času, např. nervové vlákno

(iii) *Populační modely*: slouží k popisu populace, porovnávání a odhadu budoucího vývoje (nutnost matematického modelování); jednodruhové vs. vícedruhové

(iv) *Epidemiologické modely*: modely časoprostorového šíření infekčních chorob - i pro modelování principiálně blízkých procesů; spojité (deterministické) vs. diskrétní v úrovni (stochastické)

(b) *Kompartmentové modelování*: Popis zkoumaného systému prostřednictvím diskrétních oblastí (zón) mezi nimiž protéká kanály určitá látka. Rychlost změny určité látky v čase závisí na množství látky, jež do kompartmentu vstoupilo a vystoupilo (např. změny v endokrinním systému).

- Kompartment – diskrétní oblast (zóna) určitého systému, kterou je možné nějakým způsobem logicky či kineticky odlišit od okolí, homogenní
- Kanály – propojení kompartmentů, kterými protéká určitá látka, jejíž dynamika nás zajímá, idealizujeme (nulový objem)
- Vstup kompartmentu – reprezentován přivedením látky z jeho okolí nebo syntézou této látky uvnitř kompartment
- Výstup kompartmentu – pohyb látky mimo prostor kompartmentu nebo její transformací do jiné formy

(i) *Modelování systému příjmu potravy*: Chceme sledovat dynamiku koncentrace nějaké látky, která je součástí potravy.

(ii) *Modelování funkce ledvin*: Zbavování organismu nadbytečné vody.

(iii) *Distribuce dýchacích plynů v organismu (1967)*: Kompartmenty propojeny cirkulující krví.

(c) *Celulární (buněčné) automaty*: Dynamické systémy s diskrétním prostorem a časem, které jsou charakterizovány čtyřmi základními vlastnostmi:

- Geometrií buněčné mřížky - zpravidla pravidelné N rozměrné soustavy buněk
- Specifikací okolí buňky (von Neumann (4), Moor (8), rozšířený Moor (24), ...)
- Množinou stavů buňky – často dvoustavové buňky (aktivní x neaktivní, živá x mrtvá)
- Algoritmem vypočtu příštího stavu buňky na základě současného stavu této buňky a jejího okolí (pravidla)

Příklady: hra Life, šíření epidemie

3.2.5 Konstrukce modelů na základě měření, zpracování signálu v časové, frekvenční a časo-frekvenční oblasti, modely periodických procesů.

- sledované veličiny: dynamické (krátkodobé) chování systému (dráha, rychlost, zrychlení)
- provozní parametry: statické (dlouhodobé) chování systému (teplota, tlak, otáčky, ...)
- analýza dat: určení stavu a vlastností systému v časové, frekvenční, časo-frekvenční oblasti
- vyhodnocení dat: funkce hodnotící analyzovaná data; překročení určitého prahu ukazuje na změnu stavu sledovaného systému (alarm)

Dělení signálu:

- deterministický (periodický vs. neperiodický) vs. stochastický (stacionární vs. nestacionární)
- spojitý v čase vs. diskrétní v čase
- spojitý v hodnotách vs. diskrétní v hodnotách

Deterministické signály lze popsat pomocí přesného analytického vyjádření. Pro periodické signály platí $x(t) = x(t+T)$. Harmonické signály s konstantní frekvencí a amplitudou lze popsat vztahem $x(t) = Asin\left(\frac{2\pi}{T}t + \varphi\right) = Asin(2\pi ft + \varphi)$

Zpracování signálů v časové oblasti

Základní charakteristiky:

- *Energie signálu*: $E = \int_{-\infty}^{\infty} |x(t)|^2 dt$. Signály s konečnou energií nazýváme energetické.
- *Výkon signálu*: $P = \lim_{T\to\infty} \frac{1}{2T} \int_{-\infty}^{\infty} |x(t)|^2 dt$. Signály s konečným výkonem nazýváme výkonové (musí být periodické), ty lze dále charakterizovat stejnosměrnou složkou (bez abskv) a efektivní hodnotou ($\sqrt{P}$).
- *Korelační funkce*
  - autokorelační funkce: $R(\tau) = \int_{-\infty}^{\infty} x(t)x(t+\tau)dt$. Autokorelační funkce periodického signálu je také periodická.
  - vzájemná korelační funkce: $R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)y(t+\tau)dt$. Ze vzájemné korelační funkce lze určit závislost dvou funkcí, např. posun v čase.

Zpracování signálů ve frekvenční oblasti

Z časového průběhu často nelze získat dostatečné množství informace - změny vlastností systému jsou skryty v šumu pozadí (strukturální vibrace, akustické rezonance, ...) → parametrické metody pro odhad spektra – Burg, Yule-Walker (průměrování spekter - v diagnostice nejpoužívanější metoda ke snížení šumu ve spektru). Pokud 1) počet nespojitostí signálu $x(t)$ je konečný a 2) $\int_0^\tau |x(t)|dt < \infty$, lze signál $x(t)$ rozvinout v trigonometrickou řadu se základní frekvencí $\omega = \frac{2\pi}{T}$ (lze ji zapsat také v komplexním tvaru).

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [a_k cos(k\omega t) + b_k sin(k\omega t)] = \frac{a_0}{2} + \sum_{k=1}^{\infty} [A_k cos(k\omega t + \varphi_k)] \tag{3.1}$$

Člen před sumou odpovídá stejnosměrné složce dané funkce (signálu). Harmonický člen pro $k = 1$ se nazývá první nebo základní harmonická a členům pro $k > 1$ říkáme vyšší harmonické. Hodnota A_k je amplitudou k-té harmonické a φ_k odpovídá počátečnímu fázovému posuvu dané harmonické.

Fourierova transformace je limitním případem Fourierovy řady pro neperiodické signály.

$$\boxed{X(\omega) = F[x(t)] = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt} \tag{3.2}$$

- linearita: $F[ax(t) + by(t)] = aX(\omega) + bY(\omega)$
- posun signálu v čase má vliv pouze na fázové spektrum: $F[x(t - t_0)] = X(\omega)e^{-j\omega t_0}$
- posun ve frekvenci: $F[x(t)e^{j\omega_0 t}] = X(\omega - \omega_0)$
- součin signálů odpovídá konvoluci jejich Fourierových obrazů: $F[x(t) \cdot y(t)] = X(\omega) * Y(\omega)$
- konvoluce signálů odpovídá součinu jejich Fourierových obrazů: $F[x(t) * y(t)] = X(\omega) \cdot Y(\omega)$

Veškeré signály, které v praxi naměříme, mají konečnou délku (dáno trváním měření). Vzorkováním spojitého signálu získáme diskrétní signál definovaný v časových okamžicích $x(kT_s)$. Diskrétní Fourierova transformace (**DFT**):

$$x\left[\frac{n}{NT_s}\right] = T_s \sum_{i=0}^{N-1} x[kT_s]e^{-j\frac{2\pi kn}{N}}, \qquad k = 0, 1, ..., N-1 \tag{3.3}$$

Analýza signálů ve frekvenční oblasti: reálná diagnostická data z většiny procesů jsou ve své podstatě nejčastěji nestacionární, nelineární a příliš krátká (neopakovatelná). Fourierova transformace je obecná metoda pro analýzu celkového rozložení amplitudy v závislosti na frekvence, ale, systém musí být lineární, data musí být stacionární a nelze rozlišit puls a šum.

Zpracování signálů v časo-frekvenční oblasti

Frekvenční analýza nestacionárních signálů, tj. signálů, jejichž parametry se mohou měnit v čase. Nutnou podmínkou pro korektní vyhodnocení výsledků DFT je právě neměnnost parametrů signálů a jeho periodicita. Jestliže signál není periodický, dochází k tzv. úniku ve spektru (prosakování ve spektru).

Krátkodobá Fourierova transformace (STFT) je vypočtena z krátkých úseků analyzovaného signálu:

$$X(t,\omega) = \sum_{-\infty}^{\infty} x(\tau)h(\tau - t)e^{-j\omega t}d\tau \tag{3.4}$$

kde $x(t)$ je analyzovaný signál a $h(t)$ je okénková funkce (pravoúhlé, Hamming,...). Předpokladem je, že ten krátký úsek signálu, vzniklý převážením původního signálu okénkovou funkcí, je stacionární, tj., že se jeho parametry nemění. Amplitudové spektrum krátkodobé Fourierovy transformace lze zobrazit třeba spektrogramem. Umožňuje sledovat změnu amplitudy (energie) signálu v závislosti na frekvenci a čase, ale analýza je ovlivněna Heisenberg-Gaborovým principem neurčitosti.

Wavelet transformace využívá k dekompozici množinu ortonormálních funkcí (bází) a poskytuje analýzu s vícenásobným rozlišením (multiresolution) - použití okna proměnné délky s jednou vlnkou (waveletem) → podrobnější časové rozlišení ve vyšších frekvencích (na úkor frekvenčního rozlišení).

Dosud jsme se bavili o metodách lineární časo-frekvenční dekompozice. Mezi metody využívající okamžitou frekvenci patří Hilbertova-Huangova transformace a Kalmanův filtr.

Modely periodických procesů

Rubbing je ve většině případů až důsledkem některé závady stroje. Mezi nejčastější příčiny rubbingu patří síly, které způsobují ohyb hřídele. Změna geometrie hřídele může vést ke kontaktu mezi rotorem a statorem. Zvýšením teploty v místě kontaktu dochází k dilataci kovového materiálu což způsobuje další průhyby hřídele. Druhou častou příčinou rubbingu jsou nadměrné vibrace, kdy jejich amplituda přesáhne povolenou mez mezi rotorem a statorem. K tomuto jevu může docházet například během přechodu stroje přes kritické otáčky, a jedná se o děj přechodný.

Projevy rubbingu:

- změna tuhosti spojení systému rotor-ložisko-stator → změna přirozené frekvence, změna amplitudy a fáze
- nárazy → vznik sub-a super-synchronních komponent, vznik vysokofrekvenčních komponent, šum
- tření → vznik vysokofrekvenčních komponent, šum, povrchové opotřebení

Typy rubbingu: částečný vs. úplný. Typy úplného rubbingu:

- synchronní (s dopřednou precesí): Je buzen silou nevývažku rotoru. Precese rotoru je shodná se smyslem jeho otáčení okolo osy rotace.

- samobuzený (se zpětnou precesí): Precese rotoru není shodná se smyslem otáčení rotoru okolo osy rotace. Může vzniknout v oblasti rezonanční

3.2.6 Modely vibrací a kmitání, experimentální modální analýza.

(a) **Statistické metody**

(i) *Činitel výkmitu - crest-faktor*: Metoda je založena na faktu, že periodicky se opakující vibrační ráz lze s postačujícím rozlišením vyhodnotit z výkmitu měřeného vibračního signálu. Tento výkmit je však prakticky neměřitelný jako efektivní hodnota v daném kmitočtovém rozsahu. Zhoršující se technický stav se projevuje nárůstem jak četnosti rázů tak jejich výkmitů. Efektivní hodnota vibračního signálu se zvětšuje se zvyšující se četností rázů, zatímco se hodnota výkmitů stabilizuje. Podíl mezi maximální a efektivní hodnotou vibrací (zrychlení). Vyhodnocuje se z časového signálu (frekvenční pásmo 10Hz – 10kHz): $c_f = \frac{\hat{a}}{a_{eff}}$. Metoda je rychlá a laciná, ale není příliš přesná co se týče stanovení stupně poškození. Je navíc nevýhodná při parazitních zdrojích kmitů.

(ii) *K-hodnota*: Zohledňuje efektivní a maximální (špičkové) hodnoty při stavu bez poškození a v aktuálním stavu: $K(t) = \frac{a_{eff}(0)\cdot\hat{a}(0)}{a_{eff}(t)\cdot\hat{a}(t)}$. K tomu, aby mohlo být rozhodnuto o aktuálním stavu, jsou stanoveny hladiny parametru $K(t)$, které vycházejí z dlouholetých zkušeností a poznatků. Metoda parametru K(t) je rychlá a nenáročná a oproti srovnatelným metodám spočívá její výhoda v diagnostikovatelnosti většího množství zdrojů poškození. Nebylo zjištěno omezení její platnosti.

(iii) *Kurtosis*: Nespoléhá se na měření absolutní velikosti vibrací. Způsob výpočtu veličiny je je založen na rozdělení amplitud signálu: $K = \frac{\int(x-x^{-4})p(x)dx}{\sigma^4}$. Nepoškozené ložisko emituje stochastické kmitání, které má gaussovo normální rozdělení pravděpodobnosti a hodnota K pro toto rozdělení je K=3. Tuto hodnotu můžeme naměřit v širokém pásmu od 2,5 kHz do 80 kHz s odchylkami +- 8% (nezávisle na zatížení a otáčkách). S postupujícím poškozením roste koeficient K v nižším frekvenčním pásmu. Velké poškození vede k nárůstu hodnoty K ve vyšších frekvencích, zatímco v nízkých frekvencích se koeficient K vrací zpět na svou původní hodnotu.Míra poškození určovaná Kurtosis faktorem se proto odhaduje v pěti frekvenčních pásmech.

(b) **Rezonanční metody** (rezonance snímače)

(i) *Metoda rázových pulsů (SPM/BCU)*: Princip spočívá v měření a posouzení rázových pulzů. V měřícím zařízení je signál filtrován pásmovou propustí se střední frekvencí pásma v okolí rezonanční frekvence snímače. (nejčastěji cca 35 kHz). K určení vlastního stavu je monitorována maximální hodnota špičky impulsu (shock value) v signálu, jejíž nárůst poukazuje na počínající poškození v drahách ložiska. V prahové hodnotě (carpet value) je shrnut vibrační šum ložiska. Nárůst této hodnoty zpravidla poukazuje na problémy s mazáním. Aby mohl být stav ložiska posuzován objektivněji a aby mohla být posuzována různá ložiska navzájem, je nutné provádět měření ložiska v dobrém stavu (nově osazené ložisko) a hodnoty metody normovat. Diference špičkové a prahové hodnoty se blíží svými vlastnostmi činiteli výkmitu (crest- factor).

(ii) *Metoda špičkové energie (Spike Energy)*: Zpracováván je signál z akcelerometru v oblasti od 100 Hz až do 65 kHz. Dále je definována očekávaná frekvence poškození ložiska f_D, požadovaný počet harmonických n_{SE}, které budou k dispozici ve výsledném spektru. Pokud není nastavena očekávaná frekvence poškození f_D, pak je pro další výpočet použita maximální frekvence f_{max} obsažená v měřeném signálu. Pro každý vzorek signálu z A/D převodníku se počítá odstup aktuálního vzorku od předchozího maximálního záporného impulsu. Pokud je hodnota aktuálního vzorku menší než hodnota maximálního záporného impulsu, pak bude maximální záporný impuls nastaven na tuto hodnotu. Odstup aktuálního vzorku od maximálního záporného impulsu je označován jako peak-to-peak a je v každém kroku násoben konstantou tlumení.

(c) **Frekvenční metody**

(i) *Obálková analýza*: Metoda, kterou mohou být detekovány a monitorovány opakující se rázy již v raném stadiu jejich vzniku a vývoje. Velmi krátké a rychle doznívající pulsy nejsou rozpoznatelné přímou frekvenční analýzou měřeného signálu. K tomu slouží vytvoření obálkové křivky časového signálu. Měřený signál je vlastně superpozicí vibračního signálu ložiska se stacionárním základním kmitáním a rovněž s cizími vibracemi, které se šíří materiálem z okolních částí stroje. Filtrace horní propustí, nebo pásmovou propustí proto, aby se odstranily vlivy například otáček rotoru a dalších složek, které se projevují v nízkých frekvencích. Signál na výstupu obsahuje jen složky s vysokými kmitočty, ke kterým zaručeně patří i kmitání, impulsy v důsledku rázů. Následně je filtrovaný signál usměrněn a filtrován dolní propustí. Tento postup odstraňuje nosný signál (vlastní kmitavé pohyby stroje) a výsledkem je obálka původního časového signálu. Signál obálky se dále analyzuje např. pomocí FFT ve frekvenčním spektru.

(ii) *Kepstrální analýza*: Slouží k rozpoznání periodicit ve frekvenčním spektru: $c(\tau) = F[log[]F(x(t))[]]$

(d) **podíl KRMS/LRMS**: Podíl mezi krátkodobou a dlouhodobou efektivní hodnotou - pro ohodnocení navýšení intenzity signálu (odhalení nestacionarit z časového signálu). Při vzniku nestacionarity dojde k navýšení intenzity signálu a tím i k nárůstu efektivní hodnoty. Efektivní hodnota může ale vlivem změn stavu zařízení značně kolísat. Pro odstranění těchto relativně pomalých změn je krátkodobá efektivní hodnota K-RMS (Kurzzeit-RMS) dělena dlouhodobou efektivní hodnotou L-RMS (Langzeit-RMS). Délka okna LRMS je řádově větší než u KRMS (jedná se řádově o sekundy, kdežto okno KRMS je v řádu milisekund).

Využití: monitoring volných částí v primárním okruhu JE, detekce pádu keramického obkladu ve spalovací komoře plynové turbíny, včasné určení poruchy ložiska.

Shrnutí:

- akcelerometry: senzory pro snímání vibrací

- analýza vibračních signálů v časové a frekvenční oblasti odděleně je dobře použitelná v mnoha aplikacích, ale je vždy nutné zhodnotit jejich použití vzhledem k charakteru úlohy
- časo-frekvenční analýza je důležitým nástrojem pro řešení problémů s analýzou nestacionárních a nelineárních signálů
- v časo-frekvenční oblasti je možné selektivně analyzovat krátkodobé změny na určitých frekvencích a v určitém čase odděleně od rezonancí
- pro potlačení nežádoucích frekvencí v zobrazení je používáno tzv. normování (adaptivní pro krátkodobé změny a na určitý stav systému pro dlouhodobé změny)

Experimentální modální analýza

Cílem modální analýzy je určení vlastních frekvencí dílu nebo soustavy dílů (součástí mechanické konstrukce). Tyto vlastní frekvence slouží k informaci o měřeném objektu, které se potom využívají k hodnocení provozních stavů, kdy by případná rezonance některé z provozních frekvencí s vlastní frekvencí vedla k totálnímu zničení objektu. Možnost určení vlastních frekvencí jsou podle způsobu zjišťování: 1) *výpočet* - teoretická (analytická modální analýza); 2) *experiment* - experimentální modální analýza (modální zkouška).

Vytvoření matematického modelu popisujícího dynamické chování testované struktury experimentální cestou. Získáme: 1) vlastní frekvence, 2) modální tlumení, 3) vlastní tvar → modální model (fyzikální/modální/odezvový).

Proč se provádějí modální zkoušky? Většina průmyslových odvětví běžně navrhuje své struktury na počítačích a často mají jejich analytický model. Dynamické vlastnosti vypočítané pomocí tohoto modelu je potřeba ověřit pomocí modální zkoušky na prototypu → doladění konečnoprvkového (MKP) modelu, snížení vibrací, simulace scénáře "co se stane, když..."(letecký, automobilový průmysl).

Shrnutí:

1. Vytvoření modelu pro modální zkoušku. Často už před zkouškou existuje počítačový geometrický model. Potom je úkolem definovat, které body na struktuře se mají měřit a ve kterých směrech. Tento krok je velmi důležitý, protože cílem je omezit množství měření a neprovádět zbytečnou, časově náročnou, přípravnou práci. Na druhé straně je však třeba vybrat správné body, aby byl vytvořen použitelný modální model. Často se použije část MKP modelu a výhodou je, že MKP analýza nám dává množství hodnotných informací pro tvorbu co nejlepšího modelu pro modální zkoušku.
2. Měření je často časově nejnáročnější krok. Čas zabírá instalace snímačů a jejich nastavení, což může často zabrat 80% času celé zkoušky. Jak už bylo diskutováno, pro prokládání křivek se používají frekvenční odezvové funkce, ale často jsou pro ověření důležité i jiné funkce. Můžeme zmínit koherenci, autospektrum aj.
3. Mnoho softwarových balíků má implementováno množství různých aproximačních metod a je důležité vybrat tu správnou pro řešenou úlohu.
4. Je důležité provést ověření, než data použijeme pro simulaci nebo pro srovnání s MKP daty.

3.2.7 Generování náhodných čísel, metoda Monte Carlo a odhad přesnosti simulačních výsledků.

Generování náhodných čísel

Náhodné číslo je realizace náhodné proměnné. Náhodné veličiny používáme pro modelování dějů, které jsou příliš komplexní, abychom je mohli popsat (vývoj počasí, hod kostkou).

- *Fyzikální/hardwarové metody*: výsledky fyzikálního pokusu, hod mincí/kostkou, ruleta, radioaktivní rozpad, počet fotonů dopadající na plochu, ...
- *Výpočetní/softwarové metody - pseudonáhodná čísla*: deterministický markovský systém, stav určuje celou posloupnost náhodných čísel.

  Požadavky na generátor pseudonáhodných čísel:

  - stabilita a nezávislost na vnějších podmínkách (hardwarové)
  - nízké nároky na implementaci, výpočet a paměť (softwarové)
  - realizace požadovaného rozdělení af využití celého definičního oboru rozdělení
  - není autokorelace, dlouhá perioda

Lineární kongruentní generátor: $x_{i+1} = (ax_i + c) mod m$. Často se také transformuje z rovnoměrného rozdělení (generujeme $F_y(y)$, nalezneme y). Další používaná: Gaussovo, exponenciální, fázová rozdělení (použití, pokud je aproximace exp. rozdělením příliš hrubá, definována jako doba absorpce markovského řetězce).

Metoda Monte Carlo (40.léta 20.století; S.M.Ulman and J. von Neumann)

Numerická výpočetní metoda založena na využití náhodných veličin a teorie pravděpodobnosti. *Použití*: řešení úloh, které nelze exaktně vyřešit, ověření analytických výpočtů. *Výhody*: jednoduché nasazení metody, obecnost. *Nevýhody*: výpočetní náročnost, poskytuje pouze bodové odhady a odhady chyb výpočtu. Minimální velikost statistického souboru je okolo 30. Metodu lze rozdělit do tří kroků:

1. Rozbor problému a návrh simulačního modelu
2. Vygenerování dostatečného množství realizací
3. Statistické vyhodnocení – střední hodnoty, kvantily, histogramy...

- *Neanalogový model*: úlohy, ve kterých se nevytváří model reálného děje (výpočet π, integrálů, řešení rovnic)
- *Analogový model*: založeno na simulaci procesu na počítači (výpočet rozdělení veličin v el. sítích, výpočet kvantilů velikosti napětí, plán údržby tepelné elektrárny, ...)

Analýza chyb modelů a simulací

Centrální limitní věta: rozdělení výběrového průměru se blíží k normálnímu rozdělení $\bar{x} = \mathcal{N}(\mu, \frac{\sigma^2}{n})$. Pro zlepšení odhadu 10x je nutné počítat 100x více iterací.

3.3 Programové prostředky řízení [PP]

vyučující: Ing. Pavel Balda, Ph.D.
ročník/semestr studia: 3.ročník/LS
datum zkoušky: X. X. 2014
hodnocení: 1
cíl předmětu (STAG):
Cílem předmětu je naučit studenty aplikovat některé vybrané techniky programování řídicích a informačních systémů především prostředky jazyka C#. V rámci předmětu je podána klasifikace operačních systémů a jejich základní vlastnosti. Dále je vysvětlena hierarchie programového vybavení typických řídicích systémů od čidel a akčních členů až po podnikové systémy.

3.3.1 Architektura podnikových řídicích systémů; používané programovací jazyky.

Programové prostředky v řídících systémech:

L0 : Inteligentní čidla a akční členy

- FPGA (Field Programmable Gate Array) - speciální návrhové systémy
- jednočipové procesory - assembler, jazyk C
- signálové procesory - assembler, jazyk C

L1 : Řízení v reálném čase, snímání dat

- PLC (Programmable Logic Controller) - programování podle normy IEC 61131-3 - Ladder diagramy, strukturovaný text (STL), instruction language (IL), function block chart (FBC), sequential flow chart (SFC)
- IPC (Industrial PC) - operační systémy reálného času (Windows CE, Phar Lap ETS, VxWorks, QNX, RT Linux), C/C++, .NET Compact Framework, Java
- kompaktní regulátory - nejčastěji i s jednočipy, jazyk C

L2 : Uživatelské rozhraní (HMI, SCADA), archivace, trendy

- HMI (Human-Machine Interface), SCADA (Supervisory Control and Data Acquisition) - Windows, I/O ovladače pro HW v L1, OPC, C++, C#, .NET, Java
- Export do SQL databází - ODBC, ADO.NET, C#
- DCS (Distributed Control Systems - Unix, X Window

L3 : Management, řízení výroby, vzdálený přístup po internetu

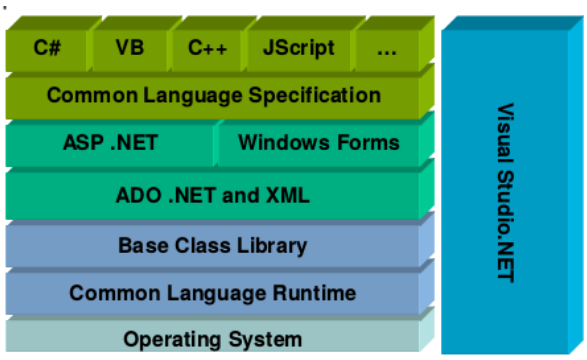
- Podnikové informační systémy, SQL databáze
- přístup přes internet - ASP.NET, tenký klient (Java), tlustý klient (ActiveX)

Programovací jazyky pro řízení:

- **Jazyk C**: Pracuje s datovými strukturami, ale není objektový. Je rozšířen na většinu HW platforem i OS. Často pracuje s ukazateli (pointry), nekontroluje souhlas typů, představuje základ syntaxe pro C++, Java, C#
- **C++**: Plně objektově orientovaný, vícenásobná dědičnost, rozšířen na mnoho platforem a OS, používaný pro většinu rozsáhlých systémů.
- **C#**: Moderní jazyk vycházející ze syntaxe C, C++, Java. potřebuje .NET Framework (Windows, Windows CE), generuje jen řízený kód (managed code). Velmi přísná typová kontrola (minimalizace chyb).
- Další: Java, Pascal/Delphi, Visual Basic, **Python** what the hell

3.3.2 Architektura .NET Frameworku; řízený modul, metadata, běh řízeného kódu.

Soubor technologií v softwarových produktech, které tvoří celou platformu, která je dostupná nejen pro Web, Windows i Pocket PC. Pro vývoj .NET aplikací vydal Microsoft Visual Studio .NET.



Common Language Runtime (CLR) je jádrem (společná báze) pro spouštění kódu všech programovacích jazyků v .NET. CLR nepozná v jakém jazyku byl kód vytvořen. Programy vytvořené pro .NET jsou překládány pro CLR. Výsledkem překladu je tzv. řízený modul (managed module).

Řízený modul (managed module)

Řízený modul je standardní PE (Portable Executable) soubor, potřebující pro běh CLR. Skládá se ze 4 částí:

- PE header: hlavička informující o typu souboru (GUI, CUI, DLL) , času překladu, ...
- *CLR header*: hlavička interpretovaná CLR a utilitami. Obsahuje požadovanou verzi CLR, umístění a velikost metadat, zdroje (resources), silná jména (strong names), příznaky, ...
- *metadata*: tabulky údajů udávající 1) typy a členy definované ve zdrojovém kódu modulu; 2) typy a členy, na něž se zdrojový kód odkazuje (které využívá). Odstraňují potřebu hlavičkových souborů (.h) a knihoven (.lib) pro překlad. Celá informace o typech a členech

spolu s kódem v IL je obsažena v modulech. Tuto informaci přímo využívají překladače. Visual Studio .NET je využívá pro IntelliSense (nápovědu během psaní kódu, které členy lze použít a jaké parametry mají dané metody). CLR je využívá pro kontrolu, že kód obsahuje jen „bezpečné“ operace. Jsou užívána pro serializaci dat v paměti, pro přenos na vzdálený počítač a obnovení stavu objektů na vzdáleném počítači. Umožňují garbage collectoru sledovat dobu života (lifetime) objektů. Pro každý objekt může garbage collector určit jeho typ a z metadat zjistí které položky se odkazují na jiné objekty.

- *Intermediate Language (IL)*: kód generovaný překladačem. Později je tento kód dále CLR překládán do instrukcí procesoru. Je založen na práci se zásobníkem, jeho instrukce vkládají (push) operandy na zásobník a vytahují (pop) výsledky ze zásobníku. Má omezenou množinu instr ukcí a nepracuje přímo s registry procesoru, což zjednodušuje vývoj překladačů. Největším přínosem je robustnost aplikací. Během překladu z IL do instrukcí procesoru CLR provádí tzv. verifikaci. Během ní se zkoumá IL kód, zda vše co provádí je „bezpečné“ (safe).

Exekuce řízeného IL kódu

Při zavedení do paměti se pro všechny metody vytvoří náhradní metody (stub). Při prvním zavolání jakékoliv metody skáče náhradní metoda do CLR. CLR načte kód v IL a přeloží metodu do kódu procesoru (native CPU code) pomocí tzv. Just-in-time překladače (JITCompiler). Náhradní metoda je přepsána kódem volajícím přeloženou metodu. Přeložená metoda se zavolá. Při druhém a dalším volání se volá přímo přeložená metoda.

Špatně detekovatelné chyby v programech jsou často spojeny se správou paměti. V jazycích C, C++ často dochází k paměťovým únikům (memory leaks) – paměť se naalokuje, ale neuvolní. Řešení v CLR: Garbage Collector (x cyklické odkazy, časová náročnost). CLR nepracuje přímo s moduly, ale se assembly - logická skupina jednoho nebo několika řízených modulů a zdrojů (resources), nejmenší samostatný kus SW pro instalování, má svou verzi (private/shared assembly).

Typy v CLR (Common Type System - CTS): kód napsaný v jednom jazyku může pomoci nich komunikovat s kódem z jiného jazyka: field, method, property, event. Common Language Specification (CLS): CLR integruje všechny jazyky tak, že objekty vytvořené v jednom z nich mohou být plně využívány v úplně jiném jazyku. CLS definuje minimální množinu rysů, které musí podporovat každý překladač pro CLR.

3.3.3 Jazyk C#: hodnotové a referenční typy; jednoduché typy, implicitní konverze; výrazy a operátory; příkazy; výjimky.

Datové typy (Types): hodnotové, referenční (odkazové), zabalení a rozbalení

Hodnotové typy

Proměnná hodnotového typu obsahuje hodnotu tohoto typu (např. typ:int, proměnná:v, hodnota:123). Přiřazení vytváří kopii přiřazované hodnoty. Od hodnotových typů nelze dále dědit. Patří mezi ně:

- struktury (struct): Struktury jsou podobné třídám v tom, že mohou obsahovat datové členy a funkční členy. Na rozdíl od tříd jsou struktury hodnotové typy! Jsou vhodné zejména pro malé objekty, které nevyžadují dědičnost.
- jednoduché typy: rezervovaná slova (byte, short, int, long, char, float, bool, ...)
- výčtové typy (enum): deklarují množinu pojmenovaných konstant (pro každý enum je v pozadí nějaký celočíselný typ. Např. enum Color {Red, Green, Blue}.

Referenční typy

Proměnná referenčního typu obsahuje referenci (odkaz) na instanci typu objekt. Přiřazením do proměnné referenčního typu ukazuje proměnná na stejnou instanci objektu jako přiřazovaná. Referenční typy jsou:

- třídy (class)
- rozhraní (interface): Rozhraní definuje „kontrakt“. Třída nebo struktura, implementující interface musí dodržovat tento kontrakt. Interface může dědit z několika bázových interfaců. Interface neimplementuje členy, které definuje. Implementují je třídy nebo struktury, které jsou od daného rozhraní odvozeny.
- pole (array): Pole jsou datové struktury, k jejichž prvkům se přistupuje pomocí vypočtených indexů. Dimenze pole = počet indexů; jedno a vícedimenzionální pole (dvou, třídimenzionální, ...). Prvkem pole může být libovolný typ, včetně pole.
- delegáty (delegates): Umožňují implementovat scénáře známé z jazyků C, C++, Pascal, atd. jako ukazatele na funkce. Na rozdíl od C++ (atd.) jsou plně objektově orientovány a zapouzdřují (encapsulates) jak instanci objektu, tak i metodu. Deklarace delegáta definuje třídu odvozenou od System.Delegate a delegáty jsou implicitně sealed. Daná metoda a delegát jsou kompatibilní, mají-li stejný počet parametrů stejných typů ve stejném pořadí a se stejnými modifikátory parametrů a stejný návratový kód. Delegáty jsou v C# ekvivalentní podle jména, ne podle struktury deklarace. Dva různé typy delegátů jsou odlišné typy, i když mají shodný seznam parametrů i návratový kód. Seznam volání (invocation list) je posloupnost metod zapouzdřená do instance delegátu.

Převod z hodnotového typu na referenční se nazývá zabalení (boxing). Provádí se automaticky, když se hodnotový typ vyskytuje v místě, kde má být object. Opačný převod je rozbalení (unboxing). Je třeba použít explicitní přetypování, typ před zabalením a při rozbalení se musí přesně shodovat! Konverze mezi dvěma typy může být implicitní (long a = b) nebo explicitní (int c = (int) b).

Výrazy a operátory

Výraz je posloupnost operátorů a operandů. Existují 3 druhy operátorů:

- unární (např. x++)
- binární (např. x+y)
- ternární (např. c ? x : y)

Příkazy

Patří sem bloky, prázdné příkazy, deklarační příkaz (deklaruje lokální proměnné nebo konstanty), výrazový příkaz (vyhodnocení výrazu), výběrové příkazy (if, switch), iterační příkazy (for, while, do, foreach), skokové příkazy (break, continu, goto, return, throw), příkaz try, příkazy checked a unchecked (určují, jak bude ošetřeno přetečení celočíselných aritmetických operací a konverzí), příkaz lock (realizuje vzájemně výlučný (mutual-exclusive) přístup k danému objektu, vykoná příkaz a přístup uvolní), příkaz using (usnadňuje užívání jmenných prostorů a typů v nich definovaných).

Všechny příkazy kromě příkazů s návěštím a deklaračních příkazů se nazývají vložené (embedded). Vložené příkazy se používají v rámci jiných příkazů.

Výjimky

Výjimky jsou v C# strukturovaným a typově bezpečným způsobem zpracování systémových i aplikačních chyb. Mohou být vyvolány dvěma způsoby: 1) příkazem throw; 2) Během zpracování příkazů a výrazů jazyka C#, kdy nějaká operace nemůže být dokončena normálně. Jsou obsluhovány příkazem try.

3.3.4 Jazyk C#: Členy a přístup k nim; jmenné prostory; třídy, metody, vlastnosti, konstruktory, destruktory; struktury; pole; delegáty; atributy.

Členy a přístup k nim

Jmenné prostory a typy mají členy. Členy dané entity (objektu, jmenného prostoru) jsou dostupné pomocí kvalifikovaného jména: entita.jméno-člena.

- Jmenný prostor (namespace): členy jsou jmenné prostory a typy deklarované uvnitř něj.
- Struktura (struct): členy deklarované ve struktuře (např. jednoduché typy) a členy zděděné od třídy object.
- Vyjmenovaný typ (enum): deklarované konstanty a členy zděděné z třídy System.Enum.
- Třída (class): deklarované členy v této třídě a členy zděděné od bázové třídy kromě konstruktorů a destruktorů.
- Rozhraní (interface): deklarované členy v interfacu, členy všech bázových interfaců a členy zděděné od třídy object.
- Pole (array) : členy zděděné z třídy System.Array
- Delegát (delegate): členy zděděné z třídy System.Delegate

Přístup ke členům je určen v deklaraci pomocí modifikátorů:

- public: přístup není omezen
- protected: přístup z typu v němž je deklarace a z odvozených typů

- internal: přístup omezený na daný program
- protected internal: přístup omezený na daný program nebo odvozené typy
- private: přístup omezen na typ, v němž je deklarace uvedena

Jmenné prostory (Namespaces)

Kód programů v C# je organizován do jmenných prostorů (direktiva using). Deklarace jmenného prostoru se může nacházet v nejvyšší úrovni ve zdrojovém souboru (takový jmenný prostor se stává členem globálního jmenného prostoru) nebo se nachází uvnitř (vnitřní jmenný prostor) jiného jmenného prostoru (vnější jmenný prostor). V obou případech musí být jméno jmenného prostoru jedinečné uvnitř jej obsahujícího jmenného prostoru.

Třídy (Classes)

Třída je datová struktura, která může obsahovat: 1) datové členy (konstanty a položky); 2) členské funkce (metody, vlastnosti, události, operátory, konstruktory a destruktory); 3) vnořené typy. Mohou využívat dědičnost (inheritance) - odvozená třída může rozšiřovat a specializovat třídu základní (od sealed nelze odvozovat nové třídy). Modifikátor abstract označuje neúplnou třídu, která je určena pouze jako bázová třída.

- *Metody*: Člen implementující výpočet nebo akci s danou třídou (objektem). Může mít hodnotové, referenční (odpovídá stejnému místu v paměti jako proměnná, která je argumentem v okamžiku volání) a výstupní parametry. Dále také pole parametrů deklarované modifikátorem params (pole je vždy posledním parametrem). Statické vs. instanční (this) metody. Virtuální metody mohou být nahrazeny odvozenými třídami vs. override metody mění (přebíjí) implementaci existující zděděné virtuální metody. Abstraktní metody zavádí virtuální metodu, ovšem bez implementace - tu musí dodat odvozené třídy. Externí metody (extern) jsou typické implementovány v jiném programovacím jazyku.
- *Vlastnosti*: Člen umožňující přístup k charakteristikám dané třídy (objektu), např. délka řetězce, velikost fontu, apod.
- *Konstruktory instancí*: člen implementující akce nezbytné pro inicializaci instance třídy. Nedědí se.
- *Destruktory*: Člen, implementující akce nezbytné pro zrušení (destrukci) instance dané třídy. Destruktory se nedědí a nelze je volat explicitně. Instance se stává způsobilou pro destrukci, pokud ji nemůže používat žádný kód. Instanci uklízí garbage collector. Nemohou být přetížené (nemají parametry), proto třída může mít nejvýše jeden destruktor.
- *Atributy*: Deklarace v C# umožňují používat „značky“ vkládané do zdrojového kódu pro specifikaci „přídavné“ informace. Tuto informaci lze získat při běhu programu pomocí tzv. reflexe. Atributy se vkládají do zdrojového kódu do hranatých závorek [ ].
- *Indexery*: Člen umožňující, aby byl objekt indexován stejným způsobem jako pole (array).

3.3.5 Softwarové komponenty: DLL, RPC, COM; interface; OPC.

Dynamicky linkované knihovny (.DLL)

Moduly obsahující funkce a data. Jsou zaváděny do paměti svými volajícími moduly (.exe nebo .dll) a jsou mapovány do paměti volajícího procesu (programu). Mohou definovat dva druhy funkcí:

- *exportované funkce* - mohou být volány jinými moduly
- *interní funkce* - mohou být volány jen uvnitř DLL, kde jsou definovány

DLL umožnily rozdělit aplikace do více modulů tak, aby kód mohl být snadněji sdílen a aktualizován. Existují dva způsoby, jak volat funkci z DLL:

- *Load-time dynamic linking* – modul přímo explicitně volá funkce exportované z DLL. To vyžaduje linkování daného modulu s tzv. import library z DLL, která obsahuje informace o tom, kde leží dané exportované funkce.
- *Run-time dynamic linking* – modul používá funkci LoadLibrary() nebo LoadLibraryEx() k zavedení DLL knihovny za běhu. Pak volá funkci GetProcAddress() pro získání adres exportovaných funkcí z DLL. Tento způsob nepotřebuje import library.

Remote Procedure Call (RPC)

Volání funkcí na vzdáleném počítači v architektuře klient-server. V obvyklých aplikacích architektury klient-server se programátor musí naučit detaily komunikace pro dané sítě včetně způsobu obsluhy chyb, převádět data do různých interních formátů a komunikovat s různými rozhraními.Při použití RPC se nemusí psát žádný kód obsluhující síťovou komunikaci v daném protokolu.

Jak RPC funguje? RPC „se tváří“, jako by klient přímo volal proceduru v programu serveru, ve skutečnosti však volá stejnojmennou proceduru z „Client Stub“:

- získává parametry z adresového prostoru klienta
- převádí parametry do standardní reprezentace NDR (network data representation)
- volá funkce pro RPC z Client Run-Time Library

Server pro volání vzdálené procedury dělá:

- „Server Run-Time Library“ přijme požadavek a zavolá proceduru ze „Server Stub“
- Tato procedura vyjme parametry z bufferu a převede je z NDR do potřebného tvaru
- „Server Stub“ zavolá proceduru na serveru

Po provedení procedury se výsledná data vrací klientovi obdobným způsobem. RPC je jednou částí DCE (Distributed Computing Environment) - prostředí pro distribuované počítání definované sdružením Open Software Foundation (OSF).

Component Object Model (COM)

Motivace pro používání komponent: Monolitické aplikace musely být aktualizovány jako celek, komponentové aplikace mohou se aktualizovat (upgrade) po částech. Komponenta je jakási miniaplikace. Celá aplikace se skládá z pospojovaných komponent („LEGO“). Vylepšování aplikace je často záležitostí nahrazování starých komponent novými.

Požadavky na komponenty: dynamické linkování - možnost výměny komponenty za běhu; zapouzdření (encapsulation) - Aby bylo možno nahrazovat komponenty bez nutnosti překladu musí být dodrženo rozhraní (interface) mezi komponentou a jejím klientem (programem, který ji využívá). Jsou dodávány v binární formě a nezveřejňují jazyk, v kterém byly napsány.

Co je COM? COM je nástroj, který umožnil „rozbít“ monolitické aplikace na komponenty. COM je specifikace (standard) - říká, jak vytvářet (psát) komponenty, které mohou být dynamicky zaměňovány. COM má své API, tzn COM knihovnu, které poskytuje služby pro práci s komponentami. COM používá DLL, aby komponenty měly schopnost dynamického linkování.

Historie: Původní cíl byla podpora koncepce známé jako „Object Linking and Embedding“ (OLE) - např. vložení a editace tabulky Excelu do Wordu.

COM Interface (rozhraní: Interface v COMu je množinou funkcí, které implementuje komponenta a klient je volá. Přesněji, interface je paměťová struktura obsahující pole ukazatelů na funkce. Interface lze reprezentovat obdélníkem s naznačeným vysunutým „jackem“.

OLE for Process Control (OPC)

OPC Foundation - neziskové sdružení, které zavedlo několik standardizovaných protokolů na bázi OLE/COM. Snaha o zlepšení interoperability mezi aplikacemi v oblasti automatizace a řízení, řídicími systémy a kancelářskými aplikacemi v oblasti řízení procesů. Definují standardní objekty, metody a vlastnosti pro servery poskytující informace v reálném čase, např. distribuované řídicí systémy, programovatelné automaty (PLC), inteligentní snímače, apod. Informace se v reálném čase komunikují do zařízení podporujících OLE/COM (servery, aplikace, apod.).

Komunikace prostřednictvím OPC má architekturu Klient-Server. Server poskytuje data, komunikuje s fyzickými zařízeními a klient komunikuje se serverem, který plní jeho požadavky. OPC server poskytuje přístup (čtení, zápis, komunikaci) k množině datových zdrojů (sources). OPC klient se připojuje k OPC serveru a komunikuje s ním prostřednictvím interfaců.

Specifikace OPC obsahují vždy dvě sady interfaců:

- *Custom interfacy* - základní rozhraní implementovaná OPC servery, poskytují maximální výkon
- *Automation interfacy* - nepovinná rozhraní, jsou vhodné pro skriptovací jazyky (Visual Basic, Excel, ...)

Kde je OPC vhodné? OPC je primárně navrženo pro přístup k datům ze síťového serveru. OPC interfacy však mohou být použity na mnoha místech v aplikaci (SCADA). Architektura a návrh OPC dovoluje klientské aplikaci přistupovat k datům z mnoha OPC serverů, mnoha dodavatelů, běžících na různých počítačích v síti prostřednictvím jediného objektu serveru.

3.3.6 Operační systémy: procesy a thready, synchronizace, deadlock, inverze priorit; správa paměti; vstupně-výstupní systém, programované vstupy/výstupy, přerušení, DMA, ovladače zařízení; souborové systémy.

Na operační systém (OS) se lze dívat dvěma způsoby:

- *Rozšířený stroj (extended machine), virtuální stroj (virtual machine)*: Zavádí abstrakci poskytovaných služeb, skrývá pro uživatele implementační detaily. Sjednocuje rozhraní pro programování pomocí tzv. systémových volání. Abstrakce pro příklad harddisku: otevření souboru, čtení/zápis souboru, zavření souboru. Abstrakce skrývá skutečnou nízkoúrovňovou (low-level) obsluhu zařízení (interrupty, časovače, správa paměti).
- *Manažer zdrojů (resource manager)*: Multiplexování v čase - sdílení jednoho zařízení více uživateli, např. přidělování CPU (Central Processing Unit) různým programům. Multiplexování v prostoru - rozdělení jednoho zdroje několika programům po částech, např. přidělování paměti nebo prostoru na disku a udržování informací o tom, jakou část má který program přidělenu.

Historie OS:

- *1945-1955: 1. generace počítačů* - založené na relé, pomalé, žádné programovací jazyky
- *1955-1965: 2. generace počítačů (mainframes)* - transistorové, spolehlivější, dávkový systém, FORTRAN, assembler
- *1965-1980: 3. generace počítačů* - integrované obvody (IBM), přenositelný kód, vznik UNIX OS
- *1980 → 4. generace počítačů* - mikroprocesory, osobní počítače, zpočátku OS CP/M, Bill Gates a jeho Basic, DOS (Disc Operating System) -¿ Microsoft, Apple Macintosh (first GUI), Windows, v současnosti síťové a distribuované OS

Procesy

Abstrakce běžícího programu, klíčový koncept OS. Moderní počítače umí dělat několik věcí najednou, dělá se to dvěma způsoby:

- přepínání mezi programy po desítkách ms (pseudoparalelismus)
- skutečný hardwarový paralelismus víceprocesorových systémů, kdy více procesorů sdílí společnou fyzickou paměť

Veškerý spouštěný software na daném počítači (někdy včetně OS) se skládá z několika sekvenčních procesů. Proces je prováděný program, včetně aktuálních hodnot programového čítače (program counter), registrů a proměnných. Multiprogramming je rychlé přepínání mezi procesy. Procesy jsou principiálně vytvářeny následujícími způsoby:

- při inicializaci (boot) systému - typicky několik procesů (foreground/background)
- vykonáním systémového volání pro vytvoření procesu běžícím procesem (child/parent proces)
- požadavkem uživatele na vytvoření nového procesu (kliknutí na ikonu)

Každý proces jednou skončí jedním z následujících způsobů:

- normální ukončení (exit) - dobrovolně
- fatální chyba - dobrovolně (proces sám narazí na fatal error)
- ukončení chybou - nedobrovolně (ilegální instrukce, dělení nulou, ...)
- ukončení jiným procesem - nedobrovolně (kill(), nutná oprávnění)

Z hlediska provádění mohou být procesy ve třech stavech:

- *běžící (running)* - právě v daném okamžiku používá CPU
- *připravený (ready)* - může být spuštěn, ale v daném okamžiku neběží
- *blokovaný (blocked)* - nemůže běžet, dokud nenastane nějaká externí událost

Thready (vlákna)

Procesy mají v tradičních OS svůj adresový prostor a v něm běží jedna posloupnost instrukcí (thread of control). Často se vyskytují situace, kdy by bylo vhodné v jednom adresovém prostoru kvazi paralelně spouštět více posloupností instrukcí, jako by byly samostatné procesy (až na sdílený paměťový prostor). Takový „odlehčený“ proces (lightweight process) se nazývá thread. Multithreading označuje možnost spouštění více threadů v jednom procesu Příklad: a) tři procesy, každý s jedním threadem, b) jeden proces se třemi thready Největší výhoda threadů: jednoduché vytvoření (cca 100x rychlejší než u procesu) a velká výkonnost. Thready spolupracují v rámci procesu, proto jsou některá data dostupná v celém procesu (adresový prostor, globální proměnné, otevřené soubory), některá vlastní pouze v daný thread (čítač instrukcí, registry).

Inter-Process Communication - přenos informací z jednoho procesu do druhého, synchronizace přístupu ke sdíleným datům a synchronizace běhu (race conditions - bez koordinace).

- *kritická sekce*: části programu, kde se přistupuje ke sdíleným datům
- *semafory*: Dijkstra (1965): počet vzbuzení (wakeups)
- *mutexy*: zjednodušená verze semaforu, když není potřeba čítání (někdy je nazýván binárním semaforem - locked/unlocked)
- *deadlock*: K některým zdrojům (resources) v operačním systému musí být umožněno přistupovat exkluzivně. Deadlock nastává, když dva nebo více procesů chce současně exkluzivně přistupovat k několika zařízením a blokují se navzájem.
- *plánovač (scheduler)*: součást operačního systému, určující, jaký proces má být spuštěn (Round-Robin scheduling, priority scheduling).
- *inverze priorit*

Správa paměti (Memory Management)

Memory Manager sleduje, jaké části paměti jsou používány a jaké jsou volné. Přiděluje paměť procesům a uvolňuje ji při jejich skončení. Zařizuje „přehazování“ (swapping) paměti na disk, není-li dostatek místa pro všechny procesy. Multiprogramování přináší dva zásadní problémy:

- *Přemisťování paměti (relocation)*: Program může být vždy umístěn do jiné oblasti paměti, proto musí být vyřešeno přeadresování všech skoků (např. modifikací instrukcí).
- *Ochrana paměti (protection)*: Je velmi nežádoucí, aby jeden program mohl zapisovat do oblasti paměti vyhrazené jinému programu (úmyslně nebo v důsledku chyby).

Řešení zavedením speciálních registrů:

- *Base register*: Při přepnutí do procesu je do registru zavedena počáteční adresa paměti procesu a k ní jsou všechny adresy přičítány.
- *Limit register*: Je naplněn délkou oblasti paměti. Každý přístup do paměti je hardwarově kontrolován vzhledem k tomuto registru.

Swapping je jednou ze strategií pro řešení nedostatku hlavní paměti. Každý proces je přenesen jako celek do hlavní paměti, na chvíli je spuštěn a pak je vrácen (jako celek) na disk. V paměti pak vznikají "díry", kterých se lze zbavit přesunem procesů co nejvíce k nižším adresám (memory compaction). Alternativně se dá pracovat s *virtuální pamětí* pro řešení nedostatku hlavní paměti (v paměti není celý program najednou, ale jen jeho část). Memory management unit (MMU) potom mapuje virtuální adresy na fyzické adresy (velikost stránek 4 až 64 kB).

Vstupně-výstupní systém (I/O)

Řízení I/O devices je jedním z hlavních úkolů OS. OS posílá k zařízením příkazy, obsluhuje přerušení (interrupts), chyby a měl by poskytovat co „nejjednotnější“ (pokud možno stejné) rozhraní ke všem zařízením (device independence). Obsluha chyb (error handling) by měla být prováděna co nejblíže hardwaru. I/O zařízení lze zhruba rozdělit do dvou typů:

- *Bloková zařízení*: Pracují s informací organizovanou do bloků pevné délky (typicky od 512 do 32768 bytů), každý blok má svou adresu. Nejtypičtějšími zařízeními tohoto typu jsou disky (HDD, FDD, CD, DVD).
- *Znaková zařízení*: Dodávají nebo přijímají posloupnost (stream) znaků, které nemají blokovou strukturu. Typickými představiteli jsou tiskárny, síťové adaptéry, myši, sériové a USB porty a většina „nediskových“ zařízení.
- zařadit nelze např. časovač (clock) či pamět videokarty

I/O zařízení se typicky skládá z mechanické a elektronické součásti. Elektronická část se nazývá řadič zařízení (device controller). Každý řadič má několik registrů pro komunikaci s CPU. Existují dva základní způsoby komunikace CPU s I/O zařízením:

- *I/O porty*: čtení/zápis pomocí instrukcí
- *Mapování do paměti*: Každému řídicímu registru je přiřazena jedinečná adresa v paměti. Není třeba speciálních instrukcí IN a OUT pro přístup k I/O portům, nemusí být proto ve vyšších programovacích jazycích užíván assembler. Také není třeba žádný dodatečný mechanismus ochrany paměti. Ovšem většina CPU podporuje práci s cacheováním paměti. Je-li užito na řídicí registry, může způsobit katastrofu. Pro zabránění tomuto problému musí být HW vybaven selektivním zakazováním cacheování, např. pro jednotlivé stránky.

Direct Memory Access (DMA) - přímý přístup do paměti: Často používaný mechanismus přímého přístupu do paměti, který je nezávislý na CPU. Vyžaduje přítomnost řadiče DMA

(má jej většina systémů), který má přístup na sběrnici (nezávisle na CPU). Jeho registry jsou programovány z CPU.

Přerušení (interrupts): Způsob signalizace ukončení I/O operace. Interrupt generuje řadič interruptů (interrupt controller). I/O řízené interrupty signalizují dokončení každé operace vygenerováním interruptu. Zřejmou nevýhodou I/O řízených interrupty je výskyt interruptu za každý znak (spotřebovává čas CPU) → řešením je využití DMA (ale obvykle je pomalejší než CPU).

Buffering: Ukládání do bufferů - často nemohou být data ukládána přímo na cílovou adresu, dokud nejsou zpracována.

Programované I/O: technika, kde všechnu práci dělá CPU (programově). Např. tisk řetězce na tiskárně.

Ovladače zařízení (device drivers)

Každý řadič zařízení má nějaké registry pro posílání příkazů a nějaké registry pro čtení stavu. Počet registrů a jednotlivé příkazy se radikálně liší v závislosti na jednotlivých zařízeních. Každé I/O zařízení potřebuje specifický kód pro svou obsluhu. Tento kód se nazývá ovladačem zařízení (device driver). Většina OS definuje standardní rozhraní, které musí dodržet všechna bloková zařízení a druhé rozhraní, které musí dodržet všechna znaková zařízení. Ovladače zařízení mají několik funkcí:

- Musí akceptovat abstraktní požadavky na čtení (read) a zápis (write) z nadřazené vrstvy SW nezávislého na zařízení.
- Musí inicializovat zařízení (je-li třeba) a mohou též spravovat požadavky na napájení.
- Ovladače musí být reentrantní! Běžící ovladač musí očekávat, že bude zavolán podruhé ještě předtím, než bylo ukončeno první volání.

Časovače (clocks, timers): Nejčastěji je časovač složen ze tří komponent:

- krystalový oscilátor: generuje periodický signál vysoké přesnosti a frekvence (typicky stovky MHz)
- čítač: dekrementuje svou hodnotu na každém pulsu. V okamžiku dosažení nuly generuje interrupt CPU (jednorázový/periodický)
- uchovávací registr: slouží pro zavedení počáteční hodnoty do čítače

Hardware pouze generuje tiky s danou periodou. Všechno ostatní řeší SW (virtuální časovače odvozené od 1 fyzického, aktualizace času v daném dni, ...).

Souborové systémy

Všechny počítačové aplikace potřebují ukládat a načítat informace. Při běhu procesu může být omezené množství informace uloženo v adresním prostoru procesu, ale po ukončení běhu procesu, restartu počítače nebo „spadnutí“ programu je informace z paměti ztracena! Ideou je odstranění závislosti mezi procesy a dlouhodobě ukládanou informací. Musí být možno ukládat velké množství informace, informace musí přežít spadnutí a několik procesů musí mít možnost

přistupovat k informaci současně. Obvykle je problém řešen ukládáním informace na disk nebo jiné externí médium do jednotek nazývaných soubory (files) - spravovány OS (file system). Soubory jsou tedy abstrakcí, jak ukládat informaci a později ji číst.

- *file naming*
- *struktura*: poslupnost bytů
- *typy souborů*: normální/adresáře
- *přístup k souborům*: sekvenční/libovolný
- *atributy souborů*: příznaky (flags), velikost, časové údaje, ochrana, ...
- *operace*: create, delete, open, close, read, write, append, seek, ...

3.3.7 Operační systémy reálného času: statické a dynamické plánovací algoritmy.

OS reálného času (RTOS) jsou takové OS, kde čas hraje podstatnou roli. Na podnět generovaný externími zařízeními musí reagovat do určitého pevného času. Správná reakce, ale pozdě je totéž co žádná reakce! Systémy reálného času lze rozdělit na:

- Systémy pevného reálného času (hard real time) – mezní termíny (deadlines) musí být dodrženy.
- Systémy měkkého reálného času (soft real time) – občasné nesplnění mezního termínu je nežádoucí, avšak tolerované.

Chování systému reálného času musí být predikovatelné a známé předem! Mohou vyskytovat dva typy událostí: periodické/aperiodické (nelze předvídat).

Systém může obsluhovat několik posloupností periodických událostí. Rozvrhovatelný (schedulable) RTOS: $U = \sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$.

Deadline (mezní termín): čas do kterého musí být daná úloha vykonána.

Real-time scheduling (rozvrhování reálného času): rozvrhování několika úloh „soupeřících“ (competing) o CPU, z nichž některé mají mezní termíny, kterým musí vyhovět - v případě periodických úloh se často mezním termínem rozumí okamžik, kdy má být spuštěna následující perioda dané úlohy.

Rate Monotonic Scheduling (RMS

Statický rozvrhovací algoritmus: Preemptivní spouštění periodických úloh podle priorit. Čím je daná úloha častěji spouštěna, tím má větší prioritu (proporcionálně). Liu a Layland (1973) stanovili 5 podmínek, za kterých lze algoritmus použít:

- Každý periodický proces musí skončit během své periody.
- Procesy jsou vzájemně nezávislé.
- Každý proces trvá stejnou dobu při každém spuštění.
- Žádné neperiodické procesy nemají deadline.
- Přepnutí procesů (preemption) se provede okamžitě s nulovou režií.

Earliest Deadline First (EDF

Dynamický rozvrhovací algoritmus, nevyžadující periodické úlohy ani stejnou dobu trvání úloh při každém spuštění (jako RMS).

- Každá úloha, které má být přidělena CPU, oznamuje svůj deadline.
- Scheduler vždy spouští úlohu s nejbližším časem deadline.
- K tomuto účelu má scheduler úlohy uložené v seznamu setříděném podle mezních časů.
- Kdykoliv se nějaká úloha dostane do připraveného stavu, scheduler zjistí, zda její deadline není dřív než u právě prováděného procesu. Pokud ano, odstaví prováděný proces a spustí tuto úlohu.

Podstatný rozdíl se ukáže, vzroste-li zatížení CPU $\rightarrow$ RMS selže. RMS zaručeně funguje, pokud $U \leq m(2^{\frac{1}{m}} - 1)$

Příklady RTOS:

- *Windows CE*: malý, konfigurovatelný, dobře „propojený“ operační systém reálného času; multithreadový programový model, důmyslná správa paměti, rozsáhlá množina ovladačů, podpora RT aplikací. Na rozdíl od stadardních Windows nepodporují více procesorů. Windows CE řeší inverzi priorit do hloubky 1. Rozlišovací schopnost časovače: 1ms.
- *Phar Lap ETS (Embedded Tools Suite)*: podpora 32-bitové procesorové platformy x86 firmy Intel, největší výhodou je vývoj aplikací v Microsoft Visual Studiu.
- *VxWorks*: asi nejslavnější a nejdražší RTOS, používá jej např. NASA

3.3.8 Struktury vzdálených a virtuálních laboratoří.

Oblast automatického řízení lze podle Fleminga (1989) charakterizovat následovně:

- Automatické řízení je ze své podstaty interdisciplinárním oborem.
- Matematika hrála a v budoucnu bude ještě více hrát zásadní roli v rozvoji automatického řízení.
- Od samého začátku je vztah mezi matematikou a automatickým řízením vztahem vzájemného „proplétání“.
- Pokroků v oblasti automatického řízení se dosahuje prostřednictvím směsi matematiky, modelování, výpočetní techniky a experimentování.

Nové technologie ve výpočetní technice podstatně rozšiřují možnosti praktických aplikací z automatického řízení (inteligentní čidla a akční členy, vložené řízení, pokročilé řídicí algoritmy, průmyslové komunikace, internetové řídicí systémy). Výjimečné postavení internetu má vliv nejen na výuku automatického řízení ale umožňuje i **experimentování na dálku**.

Vzdálená laboratoř: Uživatel není fyzicky přítomen, ale je ve spojení s jejím zařízením přes komunikaci, nejčastěji internet.

Virtuální laboratoř: Experimenty se neprovádějí na skutečném zařízení, ale na simulačním modelu (virtuálním zařízení).

Vytvoření kvalitní vzdálené nebo virtuální laboratoře je komplexní interdisciplinární úkol (didaktická/technická oblast).

Používané softwarové technologie:

- *RTOS*: Windows CE, Phar Lap ETS, RTX, Hyperkernel, RTLinux, QNX, VxWorks
- *Komunikační protokoly*: TCP/IP, UDP/IP, FTP, HTTP, RTP, RPC, DCOM
- *Softwarové komponenty*: DLL, COM, OLE, ActiveX, OPC
- *Scripty pro klienty*: VBScript, Jscript, Java applety
- *Skripty pro servery*: CGI, Perl, PHP, ASP, ASP.NET

Ad hoc vs. koncepční řešení: Ad hoc řešení neuvažují širší souvislosti (nedoporučeno). Koncepční řešení do návrhu systematicky zahrnují společné rysy řešených úloh (2 typy: 1) založená na standardním firemním řešení; 2) založená na vlastním výzkumu autorů).

Offline vs. Online interaktivita: Offline (batch) úlohy - Klient odešle na server např. název a parametry úlohy, server danou úlohu s těmito parametry spustí a výsledky odešle klientovi formou HTML stránky (např. Matlab Web Server). Výhoda snadná implementace, komunikace požadavek (request) - odpověď (reply) je zabudována do HTTP protokolu. Zásadní nevýhoda - uživatel nemůže do probíhajícího experimentu zasahovat. Online interaktivita je zásadní rys určující iluzi přítomnosti v laboratoři.

Omezené vs. otevřené experimenty: Experimenty s omezenou interakcí - změna jednoho či několika parametrů, doporučeno pro počáteční fáze seznamování se s daným modelem. Otevřené experimenty - mnohem větší rozsah možností, tvorba vlastní řídících systému nebo uživatelského rozhraní, zajímavější práce, větší motivace, příprava experimentů je však náročnější, nutno zabránit zničení modelu neodborným zásahem.

Simulace vs. řízení v reálném čase: Pro daný fyzikální model je vhodné mít i odpovídající simulační úlohy. U otevřených úloh je důležité, aby návrh algoritmů pro simulaci i řízení probíhal jednotným způsobem.

Měkký vs. pevný reálný čas: Operační systémy s měkkým reálným časem (soft real-time) nedosahují příliš dobré přesnosti periody vzorkování (Windows XP, Linux). Problematická řešení spouštění řídicích úloh z Matlabu pomocí ovladačů vstupně-výstupních karet. OS s pevným reálným časem (hard real-time) mají malou toleranci periody vzorkování, vysoká opakovatelnost měření (VxWorks, RTLinux, ...). Pro řízení rychlých experimentů jsou třeba vždy.

Tenký vs. tlustý klient: Tloušťka klienta závisí na množství programového vybavení nezbytného pro jeho provoz nad rámec internetového prohlížeče. Tenký klient využívá pouze internetový prohlížeč, funguje na všech operačních systémech a běžných prohlížečích, využívá HTML a Java applety. Tlustý klient se buď instaluje jako speciální aplikace nebo zásuvné moduly (komponenty ActiveX) do prohlížeče (nevýhody: bezpečnostní riziko, běh jen v Internet Exploreru; výhoda: snadné vytváření kvalitní vizualizace v dlouho vyvíjených vizualizačních systémech jako Labview, Genesis, InTouch, ...).

3.4 Převodníky fyzikálních veličin [PFV]

vyučující: Ing. Libor Jelínek Ph.D.
ročník/semestr studia: 4.ročník/LS
datum zkoušky: 16. 6. 2016
hodnocení: 2
cíl předmětu (STAG):
Cílem předmětu je seznámit studenty se základními principy, vlastnostmi a modely senzorů a akčních členů pro potřeby automatizace, monitorování a diagnostiky.

3.4.1 Struktura a parametry senzorů pro automatizaci, statické a dynamické modely a chyby, metody snižování chyb senzorů.

3.4.2 A/D a D/A převodníky, obvody pro úpravu signálů, frekvenční filtry.

3.4.3 Senzory teploty a tepla, obvody pro měření odporu, kapacity, indukčnosti a frekvence.

3.4.4 Senzory polohy a vzdálenosti (odporové, indukční, kapacitní, ultrazvukové, optické).

3.4.5 Senzory síly, hmotnosti, deformace, tlaku, rychlosti, zrychlení a vibrací (tenzometrické, piezoelektrické, kapacitní a elektrodynamické).

3.4.6 Senzory průtoku, množství, hustoty, viskozity, koncentrace a chemického složení.

Senzory průtoku

Senzory průtoku tekutin (tj. kapalin i plynů) určují množství objemové Q_V nebo hmotnostní Q_m tekutiny proteklé zvoleným průřezem za jednotku času.

- *Plovákové senzory*: Používají plovák pohybující se v kuželovité nádobě jako indikátor rovnováhy sil. Tekutina proudící zespodu nadnáší plovák (drážky na obvodu vyvolávají stabilizační rotaci) a mění se štěrbina mezi nádobkou a plováčkem. Tím klesá tlakový spád na plováčku. K ustálení jeho polohy dojde, když síly působící směrem dolů (gravitační síla zmenšená o vztlak) jsou v rovnováze se sílou působící nahoru (účinek tlakové diference mezi spodní a vrchní plochou plováčku).
- *Dávkovací senzory*: Využívají přenosu elementárního množství objemu v uzavřených komorách, např. mezi zuby ozubených kol. Jsou to v podstatě varianty rotačních čerpadel. Jako příklad poslouží senzor se dvěma ozubenými oválnými písty, které se otáčejí ve válcových komorách. Je poháněn rozdílem točivých momentů vyvolaných tlaky p_1 a p_2 na příslušné průměty činné plochy pístů.
- *Turbínkové a lopatkové senzory*: Protékající tekutina uvádí do rotačního pohybu soustavu vhodně uspořádaných ploch - pro šroubovicový pohyb optimalizovaných lopatek turbiny nebo plochých lopatek vodního kola. Turbínkové senzory při minimalizaci ztrát třením mají široký rozsah lineární závislosti úhlové rychlosti rotoru ω_r na rychlosti proudění v. Úhlová rychlost se snímá počítáním průchodů lopatek pod senzorem polohy. Nejčastěji se užívá senzoru indukčního, v němž změna magnetického toku při otáčení lopatek pod snímací cívkou s permanentním magnetem generuje impulsy napětí.
- *Vírové senzory*: Vhodně formovaný objekt v cestě proudící tekutiny může vyvolat její oscilační pohyb, jehož parametry jsou úměrné objemovému průtoku. Pro měření průtoku se využívá dvou typů oscilací tekutiny: nucené a přirozené oscilace. Pod nucenými oscilacemi se rozumí generace vírů těsně za žebrovitou překážkou na straně vtoku a jejich spirálový pohyb ve směru proudění. Většina vírových senzorů pracuje však s přirozenými oscilacemi, kdy víry jsou oddělovány za překážkou (střídavě na horní a dolní straně), jelikož proudící tekutina není schopna sledovat tvar překážky (tzv. Kármánovy vírové stezky).
- *Ultrazvukové senzory*: Jsou založeny na skládání vektoru rychlosti v tekutiny a ultrazvukové vlny c_0. Ultrazvuková vlna se od měniče (V_2, P_2) k měniči (V_1, P_1) bude šířit rychlostí $c_0 + v \cdot cos\alpha$ a zmenšenou rychlostí$c_0 - v \cdot cos\alpha$, když postupuje proti směru v k měniči (V_2, P_2)
- *Indukční senzory*:

.

3.4.7 Elektrické akční členy a jejich budiče (stejnosměrné, střídavé, krokové motory, PWM zesilovače, frekvenční měniče).

3.4.8 Hydraulické a pneumatické akční členy (pracovní a řídicí mechanizmy a zdroje tlakového média).