

Obsah

Úvod	4
1 Úloha stochastického řízení	6
1.1 Formulace základní úlohy stochastického řízení	6
1.1.1 Systém a jeho popis	6
1.1.2 Ztrátová funkce a optimální řízení	6
1.2 Úloha stochastického řízení s aditivní ztrátou	7
1.2.1 Aditivní ztrátová funkce	7
1.2.2 Dynamické programování	7
1.2.3 Použití dynamického programování při řešení úlohy stochastického řízení s aditivní ztrátou	8
2 Úloha stochastického řízení s neúplným pozorováním	9
2.1 Formulace úlohy stochastického řízení s nepřesnými daty	9
2.1.1 Výstup systému a informační vektor	9
2.1.2 Optimální řízení pro úlohu s nepřesnými daty	9
2.1.3 Převod na úlohu s úplnými daty	10
2.2 Řízení systému s neznámými parametry	10
2.2.1 Systém s neznámými parametry, hyperstav	11
2.2.2 Převod na úlohu s nepřesnými daty	11
2.2.3 Kalmanův filtr	12
3 Suboptimální přístupy k úloze duálního řízení	14
3.1 Duální řízení	14
3.2 Certainty equivalent control	15
3.3 Opatrné řízení	16

3.4	Iterativní dynamické programování	16
3.4.1	Diskretizace prostoru	16
3.5	Metoda Monte Carlo	17
3.5.1	Použití metody Monte Carlo k výpočtu očekávané ztráty . . .	17
3.6	SIDP	18
3.6.1	Algoritmus SIDP	18
3.6.2	Detaily algoritmu	19
4	Srovnání suboptimální přístupů při řízení jednoduchého systému	22
4.1	Integrátor s neznámým ziskem	22
4.1.1	Popis systému	22
4.1.2	Transformace rovnic systému	23
4.2	Srovnávané řídicí algoritmy	24
4.2.1	Klasický přístup k dynamickému programování	24
4.2.2	Certainty equivalent control	25
4.2.3	Metoda opatrného řízení	25
4.2.4	SIDP	26
4.2.5	Popis experimentu	26
4.3	Srovnání jednotlivých přístupů	26
4.3.1	Výsledky simulace	27
4.3.2	Shrnutí výsledků simulace	28
	Závěr	34
	Seznam použitých zdrojů	35

Značení

V této bakalářské práci je použito následující značení:

t	diskrétní časový okamžik
a_t	hodnota veličiny v čase t
E_a	operátor střední hodnoty s rozdělením pravděpodobnosti P^a
$t:s$	posloupnost časů $(t, t + 1, \dots, s)$
$a_{t:s}$	posloupnost veličin $(a_t, a_{t+1}, \dots, a_s)$
$g_{t:s}(a_{t:s})$	posloupnost funkčních hodnot $(g_t(a_t), g_{t+1}(a_{t+1}), \dots, g_s(a_s))$
$ H $	počet prvků v množině H

Úvod

V technické praxi, stejně jako běžném životě, jsme nuceni dělat rozhodnutí. Ať už se jedná o řízení výrobní linky či hledání optimálního spojení mezi dvěma místy, naše rozhodnutí vycházejí ze znalostí, které o světě máme. Chceme-li činit úspěšná rozhodnutí, je třeba vyřešit dvě úlohy: 1) řízený objekt co nejlépe poznat a 2) dosáhnout cíle, který jsme si vytyčili. Tyto dva úkoly jsou však většinou v rozporu: systém se nejlépe pozná, když se nechová podle našich požadavků. V reálném světě navíc existují náhodné jevy, poruchy a nepředvídané situace, které jednotně nazýváme neurčitostí. Tato skutečnost způsobuje, že naše znalost systému není nikdy dokonalá.

Za účelem řízení systémů, které jsou buď natolik složité, že jejich deterministický popis je nemožný, nebo obsahují náhodné prvky již ze své podstaty, vzniklo stochastické řízení, nebo-li optimální řízení za neurčitosti. Cílem stochastického řízení je minimalizovat velikost odchylek systému od požadovaného stavu optimalizací řídicích zásahů.

Jeden z přístupů k řešení tohoto problému je dynamické programování, které navrhl americký matematik Richard Bellman [3]. Jedná se o metodu, která s využitím zpětného chodu minimalizuje hodnotu očekávané ztrátové funkce.

Přímá aplikace tohoto postupu je však bohužel i u poměrně jednoduchých značně komplikována složitostí výpočtu. K řešení úlohy je proto vhodné použít aproximačních metod.

V šedesátých letech 20. století navrhl Alexander Aronovich Feldbaum řešení použitím takzvaného duálního řízení [5]. Hlavní myšlenkou tohoto přístupu bylo, že řízení musí nejen minimalizovat aktuální ztrátu, ale rovněž musí získat o systému co nejvíce informací pro minimalizaci budoucích ztrát.

Další z možných aproximačních metod je použití stochastické iterativní aproximace řešení. To spočívá v použití iterativního dynamického programování a simulační metody Monte Carlo. Tento přístup byl popsán v článku [11]. Podstatou algoritmu je hledání řešení úlohy dynamického programování iterativně, za použití metody Monte Carlo pro simulaci neurčitosti v systému.

Tato bakalářská práce si klade následující cíle

- Formulace úlohy stochastického řízení
- Řešení úlohy stochastického řízení s aditivní ztrátovou funkcí pomocí dynam-

ického programování

- Formulace úlohy stochastického řízení s neúplným pozorováním a její převedení na úlohu s úplnými znalostmi systému
- Představení některých suboptimálních přístupů k úloze stochastického řízení
- Aplikace a porovnání zmíněných metod k nalezení optimální strategie na jednoduchém systému
- Na základě získaných výsledků diskutovat výhody a nevýhody algoritmu a jeho použitelnost při aplikaci na další úlohy.

Cílem práce je seznámení se s problémy, které aproximativní řešení úlohy stochastického řízení přináší. Přínosem je pak vytvoření konkrétní implementace algoritmu stochastického iterativního dynamického programování a jeho srovnání s jinými algoritmy. To umožňuje získat přehled o kladných a záporných stránkách jednotlivých přístupů a posoudit jejich aplikovatelnost na reálné úlohy. Očekávaným výsledkem srovnání algoritmu iterativního dynamického programování s jinými, v literatuře běžnějšími, je větší časová náročnost výpočtu a lepší robustnost a přesnost výsledného řízení.

Kapitola 1

Úloha stochastického řízení

1.1 Formulace základní úlohy stochastického řízení

1.1.1 Systém a jeho popis

Ústředním pojmem v teorii řízení je systém. Systém je část světa, kterou chceme poznat či řídit. Ovlivňování systému, ať už za účelem jeho lepšího poznání, či za účelem řízení, provádíme pomocí vstupů (řídících zásahů). Ve většině případů je řešení úlohy stochastického řízení prováděno numericky, je proto účelné pracovat s diskrétním časem. Budeme-li proto uvažovat diskrétní povahu času, stav systému v časovém okamžiku t podél konečného horizontu délky N popisuje soustava rovnic

$$x_{t+1} = f_k(x_t, u_t, w_t), \quad t = 0, 1, \dots, N - 1, \quad (1.1)$$

kde x_t je stav systému v čase t , u_t je řídicí zásah v čase t a w_t náhodná veličina reprezentující přítomnost šumu. Zde předpokládáme, že tvar rovnic f_t je nám znám, například z fyzikálního rozboru úlohy, či ze znalosti konstrukce stroje, který popisujeme. Dále předpokládáme, že stav systému můžeme přímo pozorovat. Případem neúplného pozorování se zabývá následující kapitola.

1.1.2 Ztrátová funkce a optimální řízení

Cílem je pro zadaný systém (1.1) navrhnout řízení, které bude systém udržovat co nejlépe požadovaného stavu. Pro tyto účely máme v úloze řízení k dispozici předepsanou ztrátovou (resp. účelovou) funkci

$$g(x_{1:N}, u_{0:N-1}), \quad (1.2)$$

kteřá určuje nakolik jsme vytyčených cílů dosáhli.

Označme $U(x_t)$ neprázdnou množinu přípustných řídicích zásahů pro systém nacházející se ve stavu x_t . Přípustnou řídicí strategií $\pi = \mu_{0:N-1}$ budeme rozumět posloupnost zobrazení

$$\mu_t(x_t) = u_t \quad t = 0, 1, \dots, N - 1, \quad (1.3)$$

kde $\mu_t(x_t) = u_t \in U(x_t)$ je přípustný řídicí zásah. Neprázdná množina Π pak bude značit množinu všech přípustných řídicích strategií.

Pro danou řídicí strategii označme očekávanou ztrátu jako

$$J_\pi(x_0) = \mathbb{E}_{w_{0:N-1}} \{g(x_{1:N}, \mu_{0:N-1}(x_{0:N-1}))\}. \quad (1.4)$$

Úlohou je potom najít takovou π^* , pro kterou platí

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0). \quad (1.5)$$

Celkově se tedy jedná o optimalizační úlohu nalézt takovou posloupnost funkcí (1.3), která minimalizuje očekávanou ztrátovu (1.4) za podmínek (1.1).

1.2 Úloha stochastického řízení s aditivní ztrátou

Úlohu stochastického řízení tak, jak byla definována v předchozí části, nelze obecně řešit. Je tedy potřeba úlohu nějak blíže specifikovat.

1.2.1 Aditivní ztrátová funkce

Jako vhodné se ukazuje omezit se na nějaký speciální tvar ztrátové funkce (1.4). Budeme proto dále uvažovat tzv. aditivní tvar ztrátové funkce, tedy že existují funkce g_t takové, že můžeme psát

$$g(x_{1:N}, u_{0:N-1}) = \sum_{t=1}^{N-1} g_t(x_{t+1}, u_t). \quad (1.6)$$

Očekávanou ztrátu (1.4) potom můžeme přepsat do tvaru

$$J_\pi(x_0) = \mathbb{E}_{w_{0:N-1}} \left\{ \sum_{t=0}^{N-1} g_t(x_{t+1}, \mu_t(x_t)) \right\}. \quad (1.7)$$

1.2.2 Dynamické programování

Takto specifikovaná úloha stochastického řízení se dá řešit použitím dynamického programování [3]. Dynamické programování je přístup k řešení optimalizačních úloh, na které se můžeme dívat jako na posloupnost rozhodnutí, pro které platí tzv. princip optimality. Ten říká, že optimální posloupnost rozhodnutí má tu vlastnost, že pro libovolný počáteční stav a rozhodnutí musí být všechna následující rozhodnutí optimální vzhledem k výsledkům rozhodnutí prvního.

Platnost principu optimality pro očekávanou ztrátu tvaru (1.7) je intuitivně snadno pochopitelná. Pokud by totiž nějaký úsek řídicí strategie nebyl optimální, pak očekávanou ztrátu snížíme přechodem ke strategii, ve které onu neoptimální část nahradíme

optimálním řešením podproblému na daném úseku. Přesný důkaz platnosti principu optimality pro očekávanou ztrátu tvaru (1.7) lze nalézt například v [4].

1.2.3 Použití dynamického programování při řešení úlohy stochastického řízení s aditivní ztrátou

Při řešení úlohy stochastického řízení s aditivní ztrátou je možné postupovat, jak je u úloh řešených pomocí dynamického programování zvykem. Ze tímto účelem označme $J_t(x_t)$ minimální hodnotu střední ztráty od okamžiku t do N v závislosti na x_t . Dle (1.7) pro ni můžeme psát

$$J_N(x_N) = 0 \quad (1.8)$$

$$J_t(x_t) = \min_{u_t \in U(x_t)} \mathbb{E}_{w_t} \{g_k(x_{t+1}, u_t) + J_{t+1}(x_{t+1})\} \quad t = 0, \dots, N-1. \quad (1.9)$$

Při konstrukci optimální řídicí strategie budeme postupovat od konce řídicího horizontu a postupně hledat $J_t(x_t)$. Pro výpočet x_{t+1} se použije rovnice (1.1). Libovolná řídicí strategie $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, která splňuje systém rovnic

$$J_t(x_t) = \mathbb{E}_{w_t} \{g_k(x_t, \mu_t(x_t), w_t) + J_{t+1}(f_t(x_t, \mu_t(x_t), w_t))\} \quad t = 0, \dots, N-1 \quad (1.10)$$

pak bude optimální posloupností rozhodnutí.

Kapitola 2

Úloha stochastického řízení s neúplným pozorováním

Při aplikaci matematického modelování na řešení nějaké konkrétní úlohy se obvykle potýkáme s problémem, jak určit konstanty, které daný model určují. Zkoumáme-li například nějaký fyzikální systém, z rozboru fyzikálních zákonitostí obvykle známe tvar rovnic, které určují jeho vývoj v čase, nicméně počáteční podmínky či parametry, které v rovnicích vystupují a jsou pro daný systém charakteristické, můžeme získat pouze nepřímo, obvykle měřením vhodných veličin. Tato kapitola se zabývá modifikací úlohy stochastického řízení pro případ přítomnosti neznámých parametrů.

2.1 Formulace úlohy stochastického řízení s nepřesnými daty

2.1.1 Výstup systému a infomační vektor

Informace o stavu systému x_t v čase t získáváme pomocí výstupu y_t , který je dán jako

$$y_0 = h_0(x_0, v_0), \quad y_{t+1} = h_{t+1}(x_{t+1}, u_t, v_{t+1}), \quad t = 1, \dots, N-1, \quad (2.1)$$

kde v_t je náhodná veličina charakterizující chybu měření. Počáteční stav x_0 je dán rozdělením pravděpodobnosti P^{x_0} a další vývoj systému určuje soustava (1.1).

Informace, které jsou v průběhu řízení k dispozici je zvykem psát ve formě tzv. informačního vektoru, který má tvar

$$I_0 = y_0, \quad I_{t+1} = (y_{0:t+1}, u_{0:t}), \quad t = 1, \dots, N-1. \quad (2.2)$$

2.1.2 Optimální řízení pro úlohu s nepřesnými daty

Řídící zásah nyní nemůže explicitně záviset na stavu systému, protože máme k dispozici pouze informační vektor. Podobně jako v předešlé kapitole proto zavádíme

neprázdnou množinu $U(I_t)$ všech přípustných řídicích zásahů za informace I_t . Přípustnou řídicí strategií $\pi = \mu_{0:N-1}$ bude posloupnost

$$\mu_t(I_t) = u_t \quad t = 0, 1, \dots, N-1, \quad (2.3)$$

kde $\mu_t(I_t) = u_t \in U(I_t)$ je přípustný řídicí zásah.

Úkolem je najít přípustnou strategii, která by minimalizovala očekávanou ztrátu

$$J_\pi = \mathop{\text{E}}_{x_0, w_{0:N-1}, v_{0:N}} \left\{ \sum_{t=0}^{N-1} g_t(x_{t+1}, \mu_t(x_t)) \right\}, \quad (2.4)$$

za podmínek (1.1) a (2.1).

2.1.3 Převod na úlohu s úplnými daty

Protože v čase t nemáme k dispozici přímo stav systému x_t , ale pouze informační vektor I_t , nemůžeme použít postup z předchozí kapitoly. Před tím je potřeba úlohu vhodně transformovat. Za tímto účelem zapíšeme informační vektor ve tvaru

$$I_0 = y_0, \quad I_{t+1} = (I_t, u_t, y_{t+1}), \quad t = 1, \dots, N-1. \quad (2.5)$$

Na tuto rovnost můžeme pohlížet jako na rovnice systému (1.1). Stav v čase t je nyní I_t , vstup u_t a y_{t+1} náhodná veličina podmíněná I_t a u_t přes (2.1).

Dále přejdeme k nové ztrátové funkci, kterou definujeme jako

$$\tilde{g}_t(I_{t+1}, u_t) = \mathop{\text{E}}_{x_{t+1}} \{g_t(x_{t+1}, u_t) | I_t, u_t\}, \quad t = 1, \dots, N-1, \quad (2.6)$$

kde x_{t+1} se počítá dle (1.1) a x_t se považuje za náhodnou veličinu podmíněnou informačním vektorem I_t .

Očekávanou ztrátu podproblému od času t do N nyní můžeme psát ve tvaru

$$J_N(I_N) = 0 \quad (2.7)$$

$$J_t(I_t) = \min_{u_t \in U_t} \mathop{\text{E}}_{w_t, y_{t+1}} \{ \tilde{g}_t(I_{t+1}, u_t) + J_{t+1}(I_{t+1}) | I_t, u_t \} \quad t = 0, \dots, N-1 \quad (2.8)$$

Tato úloha již může být řešena pomocí dynamického programování. Při řešení budeme postupovat od konce řídicího horizontu a postupně hledat $J_t(I_t)$. Potom libovolná $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, která nabývá minimální očekávané ztráty $J_0(y_0)$ je optimální řídicí strategie.

2.2 Řízení systému s neznámými parametry

Pokud chceme řídit systém, jehož výstup závisí na nějakém neznámém konstantním parametru θ , můžeme využít znalosti řešení problému s neúplným pozorováním. Parametr θ bude reprezentovat stav systému x_t , který se nyní v čase nemění.

2.2.1 Systém s neznámými parametry, hyperstav

V této úloze máme výstupy systému y_t popsány jako

$$y_0 = h_0(\theta, v_0), \quad y_{t+1} = h_t(I_t^{(d)}, \theta, u_t, v_{t+1}), \quad t = 0, \dots, N-1, \quad (2.9)$$

kde $I_t^{(d)} = (y_{t:t-d}, u_{t-1:t-d})$ a číslo d se nazývá řád modelu.

Označme T_t dostatečnou statistiku pro parametr θ založenou na informacích dostupných v čase t . Pokud dostatečná statistika neexistuje, pak bude T_t označovat nějakou její vhodnou aproximaci. Označme dále $H_t = (I_t^{(d)}, T_t)$ tzv. hyperstav systému.

Předpokládejme dále, že o parametru θ máme nějakou apriorní informaci v podobě hustoty pravděpodobnosti $f(\theta|T_0)$. Aposteriorní hustotu $f(\theta|T_{t+1})$ získáme pomocí Bayesova vzorce

$$f(\theta|T_{t+1}) = \frac{f(y_{t+1}|\theta, I_t^{(d)}, u_t)f(\theta|T_t)}{\int f(y_{t+1}|\theta, I_t^{(d)}, u_t)f(\theta|T_t)d\theta} \quad (2.10)$$

Rekurzivní použití vzorce (2.10) pro odhad parametru θ se nazývá postup Bayesovského učení [10].

Pro vývoj hyperstavu H_t v čase můžeme na základě (2.10) psát

$$H_{t+1} = f_t(H_t, u_t, y_{t+1}), \quad t = 1, \dots, N-1. \quad (2.11)$$

Rovnici (2.11) můžeme podobně jako (2.5) považovat za rovnici systému (1.1) pro stav H_t a vstup u_t s šumem y_{t+1} .

2.2.2 Převod na úlohu s nepřesnými daty

Ztrátová funkce je nyní

$$g(y_{1:N}, u_{0:N-1}) = \sum_{t=0}^{N-1} g_t(y_{t+1}, u_t). \quad (2.12)$$

Úlohou je nalezení řídicí strategie $\pi = \mu_{0:N-1}$, která by minimalizovala očekávanou ztrátu

$$J_\pi = \mathbb{E}_{\theta_0, v_{0:N-1}} \left\{ \sum_{t=0}^{N-1} g_t(y_{t+1}, \mu_t(H_t)) \right\}, \quad (2.13)$$

za apriorní informace $f(\theta|T_0)$, známého rozdělení šumu v_t a podmínek (2.11) a (2.9).

Rovnice (2.11), (2.9) a (2.12) potom představují úlohu stochastického řízení s nepřesnými daty.

Úlohu opět řešíme pomocí dynamického programování, tedy postupnou minimalizací očekávané ztráty od konce řídicího horizontu

$$J_N(H_N) = 0 \quad (2.14)$$

$$J_t(H_t) = \min_{u_t \in U_t} \mathbb{E} \{ g_t(y_{t+1}, u_t) + J_{t+1}(H_{t+1}) | H_t, u_t \}, \quad t = 0, \dots, N-1, \quad (2.15)$$

kde H_{t+1} se počítá dle (2.11). Střední hodnota vzhledem k y_{t+1} se počítá pomocí (2.9) a $f(\theta|T_t)$ jakožto aktuálního odhadu na parametr θ .

2.2.3 Kalmanův filtr

Pokud v rovnicích (2.9) popisujících výstup systému vystupuje aditivní gaussovský šum a neznámý parametr je separován jako lineární člen, můžeme vypočítat konkrétní tvar rovnice (2.11), tzv. Kalmanův filtr [7].

Dle předpokladu má výstup v čase t tvar

$$y_{t+1} = \tilde{h}_t(I_t, u_t) + A_t(I_t, u_t)\theta + v_{t+1}, \quad t = 0, \dots, N-1. \quad (2.16)$$

kde $\tilde{h}_t(I_t, u_t)$, resp. $A_t(I_t, u_t)$ je známá funkce, resp. matice závisící na informačním vektoru a aktuální vstupu. Dále předpokládáme gaussovské rozložení šumu v_{t+1} se známým rozptylem

$$v_{t+1} \sim N(0, Q_{t+1}), \quad (2.17)$$

gaussovské rozložení odhadu neznámého parametru θ_t a jejich nekorelovanost, tedy

$$\theta_t \sim N(\hat{\theta}_t, P_t), \quad (2.18)$$

$$\text{Cov}(v_{t+1}, \theta_t) = 0. \quad (2.19)$$

Dosazením do (2.10) se odvodí, že aposteriorní hustota pravděpodobnosti $f(\theta|T_{t+1})$ je rovněž gaussovská a její parametry $(\hat{\theta}_{t+1}, P_{t+1})$ splňují rovnice

$$K_t = P_t A_t (A_t^T P_t A_t + Q_t)^{-1} \quad (2.20)$$

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t (y_{t+1} - \tilde{h}_t(I_t, u_t) - A_t \hat{\theta}_t), \quad (2.21)$$

$$P_{t+1} = (I - K_t A_t) P_t. \quad (2.22)$$

Odvození lze nalézt v [10].

Alternativní odvození bez požadavku gaussovského šumu je možné provést za předpokladu, že odhadovací proceduru střední hodnoty $\hat{\theta}_{t+1}$ neznámého parametru θ budeme hledat ve tvaru lineární opravy střední hodnoty $\hat{\theta}_t$ úměrné neurčitosti v systému. Tedy že

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t (y_{t+1} - \mathbb{E}_{\theta, v_t} y_{t+1}), \quad (2.23)$$

kde K_t je neznámá matice, kterou určíme z požadavku minimalizace výsledné matice rozptylu P_{t+1} . Pro šum v_t budeme požadovat nulovou střední hodnotu a existenci druhého momentu. Matici rozptylu označíme opět Q_t .

Pro matici P_{t+1} jako funkci K_t můžeme psát

$$P_{t+1}(K_t) = \mathbb{E}[(\theta - \hat{\theta}_{t+1})(\theta - \hat{\theta}_{t+1})^T]. \quad (2.24)$$

Dosazením za $\hat{\theta}_{t+1}$ z (2.23) a za y_t ze (2.16) a úpravou dostaneme (pro libovolnou matici B budeme pro lepší čitelnost namísto BB^T psát zkráceně B^2)

$$\begin{aligned} P_{t+1}(K_t) &= \mathbb{E}_{\theta, v_t} \left\{ (\theta - \hat{\theta}_t - K_t (y_{t+1} - \tilde{h}_t(I_t, u_t) - A_t \hat{\theta}_t))^2 \right\} \\ &= \mathbb{E}_{\theta, v_t} \left\{ ((I - K_t A_t)(\theta - \hat{\theta}_t) - K_t v_t)^2 \right\} \\ &= (I - K_t A_t) \mathbb{E} \left\{ (\theta - \hat{\theta}_t)^2 \right\} (I - K_t A_t)^T - (I - K_t A_t) \text{Cov}(\theta, v_t) K_t^T - \\ &\quad - K_t \text{Cov}(\theta, v_t) (I - K_t A_t)^T + K_t \mathbb{E} \{ v_t^2 \} K_t^T. \end{aligned}$$

Použitím definice P_t , Q_t a předpokladu $\text{Cov}(\theta, v_t) = 0$ máme

$$P_{t+1}(K_t) = (I - K_t A_t) P_t (I - K_t A_t)^T + K_t Q_t K_t^T. \quad (2.25)$$

Protože požadujeme minimální rozptyl odhadu $\hat{\theta}_{t+1}$, určíme K_t z rovnice

$$\frac{\partial \text{tr}(P_t)}{\partial K_t} = 0. \quad (2.26)$$

K provedením derivace použijeme vzorce*ODVOZENI BUDE ASI AZ V DODATKU*

$$\frac{\partial \text{tr}(MXN)}{\partial X} = M^T N^T, \quad (2.27)$$

$$\frac{\partial \text{tr}(MXNX^T O)}{\partial X} = M^T O^T XN + OMXN, \quad (2.28)$$

kde M , N a O jsou konstantní matice.

Tím získáme lineární rovnici pro K_t tvaru

$$-P_t^T A_t - P_t A_t + K_t A_t P_t K_t + K_t A_t^T P_t K_t + 2Q_t K_t = 0, \quad (2.29)$$

která má řešení

$$K_t = P_t A_t (A_t^T P_t A_t + Q_t)^{-1} \quad (2.30)$$

Dosazením (2.30) do (2.25) po úpravě dostaneme

$$P_{t+1} = (I - K_t A_t) P_t \quad (2.31)$$

Rovnice (2.23), (2.30) a (2.31) představují rovnice Kalmanova filtru.

Kapitola 3

Suboptimální přístupy k úloze duálního řízení

Ačkoliv použití dynamického programování přináší významný pokrok v řešení úlohy stochastického řízení, analytické řešení obvykle není možné získat. V každém časovém kroku se totiž potýkáme se dvěma obecně obtížnými problémami: 1) výpočet střední hodnoty a 2) minimalizace vzhledem k u_t . Oba problémy obecně nemají analytické řešení a bez další specifikace úlohy je proto třeba přejít k aproximačním metodám.

V této kapitole se předkládá popis několika možných přístupů k aproximativnímu řešení úlohy duálního řízení. Připomeňme, že úlohou duálního řízení je nalezení řídicí strategie $\pi = \mu_{0:N-1}$, která by minimalizovala očekávanou ztrátu

$$J_\pi = \mathbb{E}_{\theta_0, v_{0:N-1}} \left\{ \sum_{t=0}^{N-1} g_t(y_{t+1}, \mu_t(H_t)) \right\}, \quad (3.1)$$

za apriorní informace θ_0 a podmíněk

$$H_{t+1} = f_t(H_t, u_t, y_{t+1}), \quad (3.2)$$

$$y_0 = h_0(\theta, v_0), \quad y_{t+1} = h_t(I_t^{(d)}, \theta, u_t, v_{t+1}), \quad t = 0, \dots, N-1, \quad (3.3)$$

kde $H_t = (I_t^{(d)}, T_t)$ je hyperstav systému a T_t dostatečná statistika pro neznámý parametr θ v čase t .

Úlohu řešíme pomocí dynamického programování, tedy postupnou minimalizací očekávané ztráty od konce řídicího horizontu

$$J_t(H_t) = \min_{u_t \in U_t} \mathbb{E} \{ g_t(y_{t+1}, u_t) + J_{t+1}(H_{t+1}) | H_t, u_t \}, \quad t = 0, \dots, N-1, \quad (3.4)$$

kde T_{t+1} a y_{t+1} se počítá dle (3.2) a (3.3).

3.1 Duální řízení

K navržení kvalitního řízení systému s neznámými parametry je potřeba eliminovat neurčitost, která způsobuje nepředvídané chování systému. Hledané řízení by tedy mělo zároveň

- minimalizovat aktuální ztrátu,
- získat o systému co nejvíce informací pro minimalizaci budoucích ztrát.

Tento postup, nazývaný duální řízení, navrhl Alexander Feldbaum v práci [5]. Název metody pochází z faktu, že požadavky, které na dobré duální řízení klademe, jsou v obecně rozporu. Abychom se o systému něco nového dozvěděli, je vhodné ho vybudit do stavu, kde může vykazovat nepředvídané chování. Naopak pro minimalizaci aktuální ztráty potřebujeme systém uvést do stavu předem známého.

Jak konkrétně volit tvar optimálního řídicího $\mu_t(H_t)$, aby odpovídal principům duálního řízení, je obvykle obtížná úloha. Často (například [12]) se tak volí aproximace tvaru

$$\mu_t(H_t) = \mu_t^{(1)}(H_t) + \mu_t^{(2)}(H_t), \quad (3.5)$$

kde $\mu_t^{(1)}$ je řídicí zásah snažící se o minimalizaci aktuální ztráty a $\mu_t^{(2)}$ je budící zásah, který slouží k excitaci systému za účelem identifikace neznámých parametrů.

Problém je nyní ve výrazu (3.5) vhodně určit funkce $\mu_t^{(1)}$ a $\mu_t^{(2)}$. Vhodným výběrem na řídicí složku $\mu_t^{(1)}$ duálního řízení je například řízení metodou opatrného řízení (v anglické literatuře "cautious control", zkráceně pak CC), nebo metodou certainty equivalence. Obě metody budou probrány dále. Pro budící složku je obvykle obtížné určit nějaký funkcionální tvar. Jako vhodná aproximace se ukazuje použití bílého šumu, který se zapne jen pokud neurčitost v systému překročí nějakou mezní hodnotu. Nevýhodou této přímočaré aproximace je nutnost určení amplitudy budícího signálu. K tomu obvykle nějakou systematickou metodu k dispozici nemáme.

3.2 Certainty equivalent control

Při použití metody Certainty equivalent control (CEC) (například [4]) se v rovnici pro očekávanou ztrátu (3.4) nahradí náhodná veličina y_{t+1} střední hodnotou \hat{y}_{t+1} . Ta se vypočítá z (3.3) pomocí známých rozdělání na v_t a postačující statistiky T_t pro parametr θ . Očekávaná ztráta (3.4) tak přejde v

$$J_t(H_t) = \min_{u_t \in U_t} \left\{ g_t(\hat{y}_t, u_t) + J_{t+1}(\hat{H}_{t+1}) | H_t, u_t \right\}, \quad t = 0, \dots, N-1, \quad (3.6)$$

Tím odpadne počítání střední hodnoty a zbývá pouze problém minimalizace vzhledem k u_t . Jedná se tedy o jednodušší úlohu, kterou se již může povést vyřešit. Nutno podotknout, že optimální řízení získané metodou separace nemusí být optimální řízení pro původní úlohu. Některé vlastnosti CEC jsou probírány v další při aplikaci na řízení jednoduchého systému. Podrobnější pojednání lze nalézt v [4].

3.3 Opatrné řízení

Metoda opatrného řízení (anglicky "cautious control", například [12]) spočívá v optimalizaci očekávané ztráty (3.4) na horizontu délky $N = 1$. Minimalizujeme tedy

$$J_0(H_0) = \min_{u_0 \in U_0} E \{g_0(y_1, u_0) | H_0, u_0\}. \quad (3.7)$$

Stačí tedy spočítat střední hodnotu $g_0(y_1, u_0)$ vzhledem k y_1 a výsledek minimalizovat vzhledem k u_t . Poznamenejme, že výsledné řízení nebude zcela jistě duální. To plyne z toho, že minimalizujeme očekávanou ztrátu pouze jeden krok dopředu a tedy se nemůže projevit výhoda identifikace parametrů pomocí vybudování systému mimo požadovaný stav.

3.4 Iterativní dynamické programování

Iterativní dynamické programování [8] je jednou z variant klasického přístupu k nalezení optimální strategie, která minimalizuje očekávanou ztrátu (3.4). Standardní numerický přístup k dynamickému programování lze shrnout následovně

1. prostor proměnných H_t se diskretizuje do mřížky,
2. postupně se od konce horizontu napočítává minimální očekávaná ztráta $J_t(H_t)$ pro každý bod diskretizace H_t . K výpočtu se používají již napočtené minimální očekávané ztráty v následujících časech,
3. optimální strategie bude ta, na které bude nabyto minimální očekávané ztráty z počátečního stavu na konec řídicího horizontu.

Tento postup je přímočarou aplikací principu dynamického programování. Bohužel je velmi citlivý na dimenzi stavového prostoru H_t , který je potřeba diskretizovat, neboť počet bodů potřebných k disretizaci roste exponenciálně s dimenzí prostoru. Tato skutečnost se v anglické literatuře označuje jako "curse of dimensionality".

Oproti klasickému dynamickému programování iterativní dynamické programování problém řeší v sérii iterací. V každé iteraci se vychází ze strategie spočtené v předchozím běhu a prostřednictvím perturbací tohoto (suboptimálního) řešení se hledá strategie, pro kterou bude očekávaná ztráta nižší. Tato se použije v následující iteraci. Výhodnost iterativního přístupu spočívá ve snížení citlivosti na dimenzi úlohy.

3.4.1 Diskretizace prostoru

Při hledání optimální strategie $\mu_t(H_t)$ je pro přesné vyčíslení očekávané ztráty (3.8) na úseku řídicího horizontu $t : N$ nutné znát její analytické vyjádření. To ale není obvykle možné. Je proto nutné přejít k nějaké aproximaci, například

1. předpokládat nějaký tvar optimální strategie a při výpočtu určit pouze konstanty, které výslednou strategii určí jednoznačně,
2. diskretizovat prostor (H_t) a počítat $\mu_t(H_t)$ jen v bodech diskretizace a jinde se uchýlit k interpolaci (popřípadě extrapolaci).

Jakým způsobem efektivně diskretizovat prostor nezávislých proměnných pro aproximativní výpočet očekávané ztráty (3.8) je při použití dynamického programování obtížná otázka. Bude-li bodů v diskretizace příliš málo, bude výpočet nespolehlivý, naopak pro příliš jemnou diskretizaci bude počet bodů v diskretizaci hyperstavu rychle stoupat a časová náročnost výpočtu pak prakticky znemožní jeho řešení.

Zde se ukazuje výhodnost použití iterativního dynamického programování. Při něm totiž stačí diskretizovat jen tu část prostoru, která bude potřebná v následující iteraci. Pomocí perturbací strategie spočtené v předchozím kroku se určí ta část prostoru, která je pro bezprostřední výpočet podstatná. Díky tomu stačí k dostatečně jemné diskretizaci podstatně méně bodů. Iterativní dynamické programování tak problém nárůstu časové náročnosti se zvyšující se dimenzí hyperstavu neřeší, nicméně díky tomu, že stačí používat výrazně méně bodů k diskretizaci prostoru, ho alespoň částečně redukuje.

KONVERGENCE = CCA 2 STRANY

3.5 Metoda Monte Carlo

Metoda Monte Carlo [6] je statistická simulační metoda. Její princip spočívá ve vzorkování nějaké náhodné veličiny za účelem odhadu její hledané charakteristiky, např. střední hodnoty. V této práci je metoda Monte Carlo použita k výpočtu očekávané ztráty (3.4).

3.5.1 Použití metody Monte Carlo k výpočtu očekávané ztráty

Při běžném použití dynamického programování máme při výpočtu $J_t(H_t)$ k dispozici předpis pro následující očekávanou ztrátu $J_{t+1}(H_{t+1})$. Metoda monte Carlo by nám však dala k dispozici pouze odhad očekávané ztráty. Použití těchto aproximací v dalším výpočtu by chybu výpočtu navyšovalo.

Namísto $J_t(H_t)$ je proto vhodné pro další výpočet uchovávat $\mu_t(H_t)$. Očekávanou ztrátu v čase t pak lze počítat jako průměr n realizací náhodných veličin přes které je prováděna střední hodnota $(\theta_{t:N-1}, v_{t:N})$, tedy

$$\frac{1}{n} \sum_{i=1}^n \left(g_j(y_{j+1}^i, \mu_j(H_j)) + \sum_{j=t+1}^{N-1} g_j(y_{j+1}^i, \mu_j(H_j^i)) \right), \quad (3.8)$$

kde y_{j+1}^i se počítá podle (3.3) jako

$$y_{j+1}^i = h_j(I_j^i, \theta_j^i, \mu(H_j^i), v_{j+1}^i), \quad j = t, \dots, N-1, \quad i = 1 \dots, n, \quad (3.9)$$

počet opakování algoritmu	n_{pass}
počet iterací algoritmu	n_{iter}
počet bodů v diskretizaci každé dimenze H_t	n_g
apriorní řídicí strategie	$\mu_{0:N-1}$
počet kadnidátů na změnu řídicího zásahu	m
počáteční rozsah pro hledání optimálního řídicího zásahu	β^{in}
parametr pro redukcí β^{in} při opakování algoritmu	γ
parametr pro redukcí β^{in} při iterování algoritmu	λ
počet realizací pro odhad metodou Monte Carlo	n

Tabulka 3.1: Parametry algoritmu SIDP

a index i označuje i -tou realizaci dané veličiny. Realizace $\theta_{t:N-1}$ se generují podél trajektorie (3.3). To znamená, že θ_{j+1}^i se generuje až ve chvíli, kdy je známé I_j^i , u_j^i , postačující statistika T_j^i a y_{j+1}^i a tedy přes (3.2) i postačující statistika T_{j+1}^i .

Výpočet je při uchování $\mu_t(H_t)$ namísto $J_t(H_t)$ časově náročnější. Namísto přechodu $J_t(H_t)$ je totiž nutné přechít hodnotu $\mu_t(H_t)$ a následně vygenerovat trajektorii od času t do konce horizontu. To obnáší vygenerovat náhodnou realizaci šumu v_t a neznámého parametru θ (pomocí T_t), aplikovat řídicí zásah, tedy dle (3.3) vypočítat y_{t+1} a následně (3.2) dle vypočítat T_{t+1} . Tím bude určen bod v H_{t+1} . Zde pak pomocí interpolace (a extrapolace) určíme optimální zásah, který aplikujeme. Podobně jako prve tak určíme následující bod v H_{t+2} , až nakonec se dostaneme na konec řídicího horizontu. Při výpočtu postupně napočítáváme hodnotu ztrátové funkce.

3.6 SIDP

Metoda stochastického iterativního dynamického programování (SIDP) [11] spočívá v současném použití metody Monte Carlo k získání aproximace pro očekávanou ztrátu a iterativního dynamického programování k nalezení optimální strategie. Při použití iterativního dynamického programování se uchýlíme k diskretizaci prostoru hyperstavů a budeme používat interpolaci (popřípadě extrapolaci) napočtených hodnot. Poznamenejme, že díky předpokladu gaussovského rozdělení odhadu neznámého parametru θ , diskretizace vzhledem k T_t znamená diskretizaci vzhledem k $(\hat{\theta}_t, P_t)$.

3.6.1 Algoritmus SIDP

V tomto odílu je popsán algoritmus SIDP, tak jak byl navržen v [11]. Parametry algoritmu jsou uvedeny v tabulce 3.1. Jak plyne z popisu 1, časová složitost algoritmu SIDP vzhledem k jeho parametrům je $O(n_{pass}n_{iter}N^2mnn_g^{\dim H_N})$ (časová náročnost metody Monte Carlo je úměrná vzdálenosti od konce horizontu, proto je časová složitost úměrná druhé mocnině N).

Algorithm 1 Schéma algoritmu SIDP

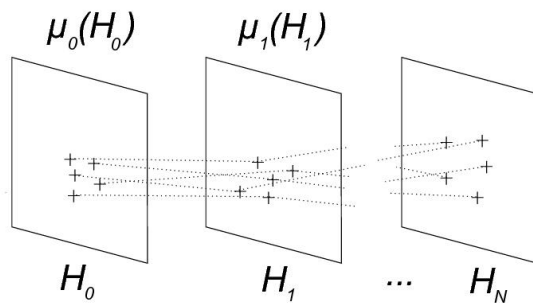
```
for  $i = 1$  to  $n_{pass}$  do
  for  $j = 1$  to  $n_{iter}$  do
     $\beta_{i,j} := \gamma^{j-1} \lambda^{i-1} \beta^{in}$ 
    for  $k = 1$  to  $|H_t|$  do
      spočti trajektorii  $H_{0,k}$ , použij aktuální  $\pi^*$ , její interpolace a extrapolace a
      realizace neznámého parametru  $\theta_0, \dots, \theta_{N-1}$  podél této trajektorie
    end for
    for  $t = N - 1$  to  $0$  do
      vytvoř  $\tilde{H}_t$  jakožto rovnoměrnou síť v oblasti bodů  $H_t$ 
      interpoluj (extrapoluj)  $\mu_t^*(H_t)$  na  $\mu_t^*(\tilde{H}_t)$ 
      for  $k = 1$  to  $|H_t|$  do
        for  $m = -\lfloor \frac{m-1}{2} \rfloor$  to  $\lfloor \frac{m}{2} \rfloor$  do
          pro  $\tilde{H}_{t,k}$  vygeneruj kandidáta na řízení  $\mu_t(\tilde{H}_{t,k}) = \mu_t^*(\tilde{H}_{t,k}) + m\beta_{i,j}$ 
          pomocí metody Monte Carlo spočti očekávanou ztrátu
        end for
        rozhodnutí s nejnižší očekávanou ztrátou uchovej jako nové optimální
        rozhodnutí pro  $\tilde{H}_{t,k}$ .
      end for
    end for
  end for
end for
```

3.6.2 Detaily algoritmu

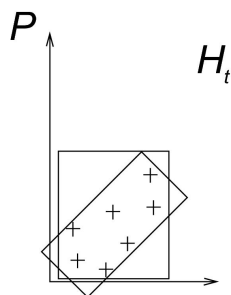
Část prostoru, která se bude v následující iteraci algoritmu diskretizovat se určí pomocí aktuálního suboptimálního řešení a náhodných realizací šumu $v_{0:N}$ a neznámého parametru $\theta_{0:N}$. Pomocí těchto realizací vygenerujeme trajektorie v $H_{0:N}$. V každé časové úrovni pak diskretizujeme jen tu část prostoru, kterou takto vygenerované trajektorie prochází. Schématicky je situace znázorněna na obrázku 3.1

V této práci se pro diskretizaci zasažené části prostoru volí jednoduchá metoda, ve které se ve směru souřadných os spočte nejmenší hyperkvádr obsahující vygenerované body. Prostor se poté diskretizuje pouze v této oblasti. V práci [11], kde je metoda SIDP navržena, je pro diskretizaci prostoru použit hyperkvádr s obecnou orientací. Metodu k jeho určení převzali autoři z [2]. Tento postup by měl vést k ještě efektivnější diskretizaci prostoru. Nicméně metoda, která je v naší práci použita se ukázala jako postačující. Navíc je implementačně podstatně jednodušší a vyhledávání v tabulce s orientací ve směru souřadných je rychlejší. Výhodou je i to, že můžeme snadno zaručit požadavek na kladný rozptyl P_t neznámého parametru θ , viz obrázek 3.2 .

Máme-li diskretizovanou požadovanou část prostoru, je nutné na ni namapovat dosavadní napočtené optimální řízení. K tomu se použije interpolace, popřípadě extrapolace napočteného řešení. V této práci je interpolace/extrapolace realizována jednoduše pomocí nejbližšího již napočteného bodu. Možným vylepšením by byla například lineární projekce či vážený průměr nejbližších napočtených bodů.



Obrázek 3.1: Trajektorie v hyperstavu – jednotlivá realizace trajektorie je napočítávána pomocí realizací šumu a neznámého parametru



Obrázek 3.2: Oblast určená k diskretizaci H_t – ačkoliv je objem obecně orientovaného hyperkvádrů menší, body v něm nesplňují požadavek na kladný rozptyl P

Pro každý z bodů (nejprve pro ty na konci řídicího horizontu) se optimální řídicí zásah hledá pomocí perturbace stávajícího suboptimálního řešení. Pro daný bod se proto vygeneruje m kandidátů na optimální zásah, rovnoměrně kolem optimálního zásahu z předcházející iterace. Jako jeden z kandidátů na optimální řízení se vždy ponechá stávající suboptimální řešení z minulé iterace.

Kandidáti na řízení se nyní porovnají pomocí metody Monte Carlo. Jak již bylo popsáno výše, pro každého kandidáta se vygeneruje n realizací ztráty, přes které se dle (3.8) spočte průměr.

Namísto jednoduchého porovnání pomocí průměru lze kandidáty na optimální řídicí zásah porovnat nějakým sofistikovanějším víceúrovňovým algoritmem. Jedno z možných vylepšení je použito i v [11]. Konkrétně se jedná o dvouúrovňový algoritmus popsaný v [9]. V první úrovni tohoto algoritmu se nejprve pro každého kandidáta u_t vygeneruje n_0 realizací. Na jejich základě se vyberou ti, na který je nabyto minima s pravděpodobností větší než je daná mez α_0 . Pro tyto se v druhé fázi vygeneruje dostatečný počet realizací tak, aby bylo možné nejlepší rozhodnutí zvolit s pravděpodobností alespoň rovné zadané mezi α_1 . Takto upravený algoritmus metody Monte Carlo je robustnější. Navíc umožňuje efektivní porovnání většího množství kandidátů, neboť počet realizací v první fázi může být poměrně nízký, slouží pouze k odfiltrování zjevně horších kandidátů na řízení. Pro účely této práce postačuje základní verze metody Monte Carlo a proto je v následující implementaci SIDP použita.

Výstupem algoritmu je poté tabulka optimálních řídicích zásahů bodech diskretizace. Samotné řízení systému je pak prováděno aplikací pedpočtených optimálních zásahů.

Kapitola 4

Srovnání suboptimální přístupů při řízení jednoduchého systému

V této kapitole je popsán jednoduchý systém, na kterém jsou porovnány některé řídicí algoritmy založené na principech uvedených v předešlé kapitole.

4.1 Integrátor s neznámým ziskem

Integrátor s neznámým ziskem, jakožto systém vhodný k ilustraci výhod duálního řízení, byl podrobně zkoumán v [1]. Pro srovnání uvádíme rovněž tamější výsledky.

4.1.1 Popis systému

Výstup systému je popsán jako

$$y_{t+1} = y_t + \theta u_t + v_{t+1} \quad t = 0, \dots, N-1, \quad (4.1)$$

$$v_{t+1} \sim N(0, \sigma^2), \quad (4.2)$$

kde $\theta \neq 0$ je neznámý parametr a rozptyl šumu $\sigma^2 = 0, 1$. Počáteční hodnota výstupu je nastavena na $y_0 = 1$.

O neznámém parametru θ máme v čase t informaci v podobě dostatečné statistiky $T_t = (\hat{\theta}_t, P_t)$, tvořené střední hodnotou a rozptylem. Předpokládáme nekorelovanost θ s šumem, tedy že

$$\text{Cov}(v_{t+1}, \theta) = 0. \quad (4.3)$$

Optimální řízení je takové, které udrží výstupy systému na nulové hodnotě. Ztrátová funkce je kvadratická v y_{t+1} , čili

$$g(y_{1:N}, u_{0:N-1}) = \sum_{t=0}^{N-1} y_{t+1}^2. \quad (4.4)$$

Odhadovací procedurou pro parametr θ je Kalmanův filtr. Pro systém (4.1) má tvar

$$K_t = \frac{u_t P_t}{u_t^2 P_t + \sigma^2} \quad (4.5)$$

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t(y_{t+1} - u_t \hat{\theta}_t), \quad (4.6)$$

$$P_{t+1} = (1 - K_t u_t) P_t. \quad (4.7)$$

Hyperstav systému H_t tvoří vektor $(y_t, \hat{\theta}_t, P_t)$. Očekávaná ztráta je

$$J_t(H_t) = \min_{u_t \in U_t} \mathbb{E} \{y_{t+1}^2 + J_{t+1}(H_{t+1}) | H_t, u_t\}, \quad t = 0, \dots, N-1. \quad (4.8)$$

Ta po dosažení z (4.1) a částečném provedení střední hodnoty přejde na tvar

$$J_t(H_t) = \min_{u_t \in U_t} \left\{ (y_t + \hat{\theta}_t u_t)^2 + u_t^2 P_t + \sigma^2 + \mathbb{E}_{y_{t+1}} (J_{t+1}(H_{t+1})) | y_t, \hat{\theta}_t, P_t, u_t \right\}. \quad (4.9)$$

4.1.2 Transformace rovnic systému

Před samotnou aplikací nějakého řídicího algoritmu lze úlohu vhodnou transformací proměnných zjednodušit. Dle [1] je takovou transformací přechod od stavového popisu $(y_t, \hat{\theta}_t, P_t, u_t)$ k popisu pomocí nových proměnných $(\eta_t, \beta_t, \zeta_t, \nu_t)$ dle vztahů

$$\eta_t = \frac{y_t}{\sigma}, \quad (4.10)$$

$$\beta_t = \frac{\hat{\theta}_t}{\sqrt{P_t}}, \quad (4.11)$$

$$\zeta_t = \frac{1}{\sqrt{P_t}}, \quad (4.12)$$

$$\nu_t = \frac{u_t \sqrt{P_t}}{\sigma}. \quad (4.13)$$

Současně můžeme neurčitost ve výstupu (4.1) reprezentovat jedinou normalizovanou náhodnou veličinou podle

$$s_t = \frac{y_{t+1} - y_t + \hat{\theta}_t u_t}{\sqrt{u_t^2 P_t + \sigma^2}} \sim N(0, 1). \quad (4.14)$$

Rovnice pro výstup (4.1) a následující odhad neznámého parametru (4.6) tak přejde v

$$\eta_{t+1} = \eta_t + \beta_t \nu_t + \sqrt{1 + \nu_t^2} s_t \quad (4.15)$$

$$\beta_{t+1} = \sqrt{1 + \nu_t^2} \beta_t + \nu_t s_t \quad (4.16)$$

Přejdeme-li k vhodně upravené očekávané ztrátě, dostaneme

$$V_t(\eta_t, \beta_t, \zeta_t) = \frac{J_t(y_t, \hat{\theta}_t, P_t)}{\sigma^2} \quad (4.17)$$

$$= \min_{\nu_t} \left\{ (\eta_t + \beta_t \nu_t)^2 + \nu_t^2 + 1 + \mathbb{E}_{s_t} (V_{t+1}(\eta_{t+1}, \beta_{t+1}, \zeta_{t+1})) \right\}. \quad (4.18)$$

Nyní spočteme očekávanou ztrátu pro $N - 1$ jako

$$V_{N-1}(\eta_{N-1}, \beta_{N-1}, \zeta_{N-1}) = \min_{\nu_{N-1}} \{(\eta_{N-1} + \beta_{N-1}\nu_{N-1})^2 + \nu_{N-1}^2 + 1\}. \quad (4.19)$$

Pomocí diferenciálního počtu pak získáme optimální zásah ve tvaru

$$\nu_{N-1} = -\frac{\eta_{N-1}\beta_{N-1}}{1 + \beta_{N-1}^2} \quad (4.20)$$

a očekávanou ztrátu rovnou

$$V_{N-1}(\eta_{N-1}, \beta_{N-1}, \zeta_{N-1}) = \frac{\eta_{N-1}^2 + 1}{\beta_{N-1}^2 + 1}. \quad (4.21)$$

Protože optimální zásah ν_{N-1} ani očekávaná ztráta V_{N-1} nezávisí na ζ_{N-1} , díky tvaru V_t nebude rovněž optimální zásah ν_t a očekávaná ztráta V_t záviset na ζ_t . K nalezení optimálního řízení tedy stačí v každém čase t uvažovat pouze dvourozměrný hyperstav $H_t = (\eta_t, \beta_t)$. Navíc můžeme bez újmy na obecnosti počítat optimální zásah pouze pro kladné hodnoty η_t a β_t . Optimální řízení tedy napočteme pouze pro kladné hodnoty y_t a θ_t , pro ostatní možnosti pak díky tvaru (4.1) dostaneme požadovaný řídicí zásah vhodnou volbou znaménka napočteného optimálního zásahu.

4.2 Srovnávané řídicí algoritmy

V tomto oddílu jsou popsány řídicí algoritmy, které budou posléze porovnány při řízení systému (4.1). Algoritmy jsou založené na principech uvedených v předešlé kapitole.

4.2.1 Klasický přístup k dynamickému programování

V článku [1] je problém řízení systému (4.1) řešen pomocí přímé aplikace numerických metod na řešení úlohy dynamického programování. Jde o schéma popsané v sekci 3.4. Prostor hyperstavů byl diskretizován do mřížky 64x64. Pro každý bod hyperstavu se napočítala očekávaná ztráta, mimo body mřížky se použila kubická interpolace. K numerické integraci byla použita klasická Simpsonova metoda. K nalezení minima se pak použila jednoduchá metoda při níž se každými třemi body na mřížce proložila parabola, našlo její minimum a to se pak testovalo, zda-li je globálním minimem očekávané ztráty.

Výslednou aproximaci optimálního řízení pak autoři uvádějí jako

$$\nu_t = -\frac{0,56 + \beta_t}{2,2 + 0,08\beta_t + \beta_t^2}\eta_t - \frac{1,9}{1,7 + \beta_t^4} \quad (4.22)$$

První člen v (4.22) můžeme interpretovat jako modifikované opatrné řízení, druhý člen pak jako budící složku řízení.

4.2.2 Certainty equivalent control

Očekávaná ztráta (4.17) přejde v

$$V_t(\eta_t, \beta_t) = \min_{\nu_t} \{ \hat{\eta}_{t+1}^2 + V_{t+1}(\eta_{t+1}, \beta_{t+1}) \} \quad (4.23)$$

Střední hodnota výstupu je dle (4.15) rovna

$$\hat{\eta}_{t+1} = \eta_t + \beta_t \nu_t, \quad (4.24)$$

Optimální řídicí zásah bude tedy pro každé $\beta_t \neq 0$ roven

$$\nu_t = -\frac{\eta_t}{\beta_t}. \quad (4.25)$$

Pokud $\beta_t = 0$, pak to dle (4.11) znamená, že i $\hat{\theta}_t = 0$ (to se může stát ačkoliv $\theta \neq 0$). Aktuální očekávaná ztráta nezávisí na ν_t a můžeme tedy volit libovolný řídicí zásah bez vlivu na hodnotu očekávané ztráty. V takovém případě volíme za řídicí zásah realizaci bílého šumu (velikost rozptylu bude upřesněna u konkrétního experimentu). Jedná se vlastně o jednoduchou aplikaci principu duálního řízení popsaného rovnicí (3.5).

Z tvaru optimálního řídicího zásahu (4.25) snadno zjistíme, že CEC je metodou neduálního. Pokud je totiž výstup systému na požadované hodnotě, řídicí zásah nebude systém vychylovat za účelem lepší identifikace parametru θ . Nicméně díky tomu, že máme k dispozici analytické vyjádření ν_t , může být řízení pomocí metody CEC prováděno s minimálními výpočetními nároky. To může být v některých aplikacích rozhodující výhodou.

Aplikace metody CEC v podobě řídicího zásahu (4.25) je zjevně nevhodná pro malé hodnoty β_t . Metoda bude totiž generovat, příliš velké řídicí zásahy bez ohledu na možnou přítomnost neurčitosti. Konkrétní výsledky budou prezentovány dále v práci.

4.2.3 Metoda opatrného řízení

Optimální řídicí zásah je pro metodu opatrného řízení (caution control, CC) dán dle (4.20) jako

$$\nu_t = -\frac{\eta_t \beta_t}{1 + \beta_t^2}. \quad (4.26)$$

Všimněme se, že pro velké hodnoty β_t přejde optimální zásah pro metodu opatrného řízení (4.26) v optimální zásah metody CEC (4.25). Naproti tomu pro malé hodnoty β_t (tedy pro velké hodnoty neurčitosti v identifikaci parametru θ_t , viz (4.11)) bude řízení podstatně konzervativnější. Opět se podařilo získat analytické vyjádření ν_t a výpočet optimálního řízení pomocí metody CC může být prováděn velmi efektivně.

počet opakování algoritmu	n_{pass}	4
počet iterací algoritmu	n_{iter}	8
počet bodů v diskretizaci každé dimenze H_t	n_g	20
apriorní řídicí strategie	$\mu_{0:N-1}$	0
počet kadnidátů na změnu řídicího zásahu	m	7
počáteční rozsah pro hledání optimálního řídicího zásahu	β^{in}	1
parametr pro redukcí β^{in} při opakování algoritmu	γ	0,9
parametr pro redukcí β^{in} při iterování algoritmu	λ	0,8
počet realizací pro odhad metodou Monte Carlo	n	20

Tabulka 4.1: Konkrétní volba parametrů algoritmu SIDP

4.2.4 SIDP

Dle popisu (a následné transformace) systému (4.1) je pro výpočet optimální strategie pomocí algoritmu SIDP nutné diskretizovat dvoudimenzionální prostor nezávisle proměnných $H_t = (\eta_t, \beta_t)$. Implementace algoritmu byla provedena v souladu s oddílem 3.6.2. Konkrétní nastavení parametrů algoritmu zachycuje tabulka 4.1. Výpočet za daných parametrů trval v řádech desítek minut.

4.2.5 Popis experimentu

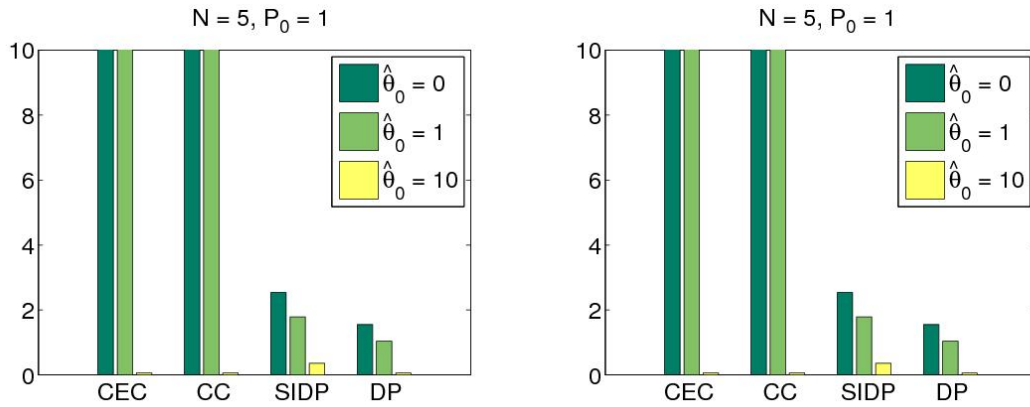
Systém (4.1) je možné v případě potřeby vhodně posunout či přeskálovat, není tedy potřeba uvažovat jiné hodnoty referenčního signálu a počátečního výstupu. Kvalitu výsledného řízení posuzujeme z hlediska celkové ztráty vygenerované podél horizontu délky N aplikací dané řídicí strategie.

Řídicí horizont byl v první sérii experimentů zvolen jako $N_1 = 5$ v druhé pak $N_2 = 10$. Kvalita řízení byla testována vzhledem k různým hodnotám počátečního odhadu θ , tedy k různým hodnotám $(\hat{\theta}_0, P_0)$. Ty byly postupně voleny z množiny $\{10; 1; 0\} \times \{10; 1; 0, 1\}$. Každá simulace byla opakována 1000x, uvedené hodnoty celkové ztráty jsou pak průměrem přes jednotlivé realizace.

Pro každé jednotlivé opakování simulace byla skutečná hodnota parametru θ pro počáteční hodnoty $(\hat{\theta}_0, P_0)$ zvolena jako první nenulová realizace náhodné veličiny s rozdělením $N(\hat{\theta}_0, P_0)$. Pro takto vygenerovanou hodnotu θ byly postupně aplikovány výše popsané řídicí algoritmy. Pro snížení vlivu náhodnosti při porovnání kvality řízení byly všechny realizace šumu (v rámci jednoho opakování simulace) v průběhu řízení jednotlivými algoritmy voleny stejně.

4.3 Srovnání jednotlivých přístupů

V této sekci jsou porovnány popsané algoritmy při řízení systému (4.1). Očekávaným výsledkem bylo, že v případě velké počáteční neznalosti systému bude duální řízení získané pomocí SIDP výrazně výhodnější oproti neduálním metodám. Ty zde zastupuje metoda opatrného řízení (CC) a metoda caution control (CEC). Pro srovnání



Obrázek 4.1: Výsledky simulace pro délku horizontu $N = 5$, rozptyl $P_0 = 1$ a $P_0 = 10$ a různé střední hodnoty $\hat{\theta}_0$

je uveden výsledek získaný klasickým numerickým přístupem k dynamickému programování (DP) převzatý z [1]. Jak vyplývá z tvaru (4.22), jedná se rovněž o duální metodu. Očekává se, že výsledky získané pomocí klasického DP budou srovnatelné s výsledky získanými metodou SIDP. Nicméně jak bylo prověřeno v článku [11], srovnatelné výsledky dává SIDP již při desetinovém výpočetním čase. To je způsobeno tím, že se při použití metody SIDP diskretizuje jen ta část hyperstavu, která je v další iteraci algoritmu. Pro efektivní diskretizaci tedy stačí výrazně méně (ve výsledcích zde použitých 64x64 pro metodu DP, zatímco pro SIDP jen 20x20).

4.3.1 Výsledky simulace

Kvantitativní výsledky simulací zachycují obrázky 4.1 a 4.2. Kvalitativní pak série obrázků 4.3–4.6.

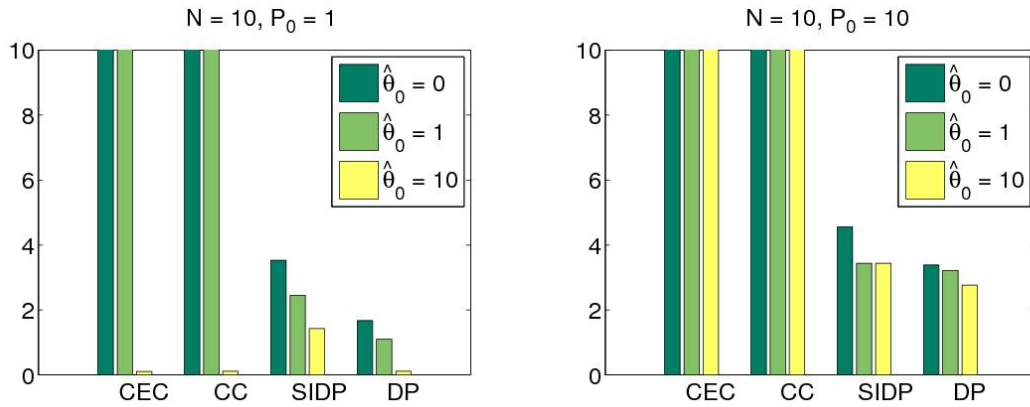
Na obrázku 4.1 jsou zachyceny výsledky pro délku řídicího horizontu $N = 5$, počáteční hodnoty rozptylu $P_0 = 1$ a $P_0 = 10$ a různé hodnoty $\hat{\theta}_0$, konkrétně pro hodnoty $\{0, 1, 10\}$.

První pozorovatelný výsledek pro případ $P_0 = 1$ je, že neduální metody (CEC a CC) poskytují v průměru použitelné výsledky pouze pokud je hodnota parametru θ dostatečně vzdálena od nuly, tedy pro počáteční odhad $\hat{\theta}_0 = 10$. V ostatních případech je řízení silně závislé na náhodných realizacích šumu a v průměru je takové řešení nepoužitelné. Oproti tomu duální metody (SIDP a DP) poskytují dobré řízení pro všechny testované kombinace $(\hat{\theta}_0, P_0)$.

Poznamenejme však, že pro hodnoty $(\hat{\theta}_0, P_0) = (10, 1)$ je použití neduálních metod velmi vhodné. Dávají totiž srovnatelné výsledky s duálními metodami (v případě SIDP dokonce lepší) a to za mnohem nižších výpočetních nároků.

Pro počáteční hodnotu rozptylu $P_0 = 10$ nebyly neduální metody úspěšné pro žádnou testovanou hodnotu $\hat{\theta}_0$. Obě duální metody pak v průměru opět dávají srovnatelné výsledky.

Srovnávací studie pro délku horizontu $N = 10$ (obrázek 4.2) dopadla velice podobně



Obrázek 4.2: Výsledky simulace pro délku horizontu $N = 5$, rozptyl $P_0 = 1$ a $P_0 = 10$ a různé střední hodnoty $\hat{\theta}_0$

jako v předchozím případě. Neduální metody v průměru opět fungovaly pouze pro nastavení $(\hat{\theta}_0, P_0) = (10, 1)$, tedy když byl skutečný parametr θ dostatečně vzdálen od nuly. V tomto případě však dosahovaly velmi nízké ztráty.

Série obrázků 4.3 zachycuje výsledky řízení metodou certainty equivalence při jednotlivých simulacích pro různé kombinace $(\hat{\theta}_0, P_0)$. Uvedené histogramy ukazují, že ač se v mnoha případech podařilo pomocí metody CE navrhnout řídicí strategii s celkově nízkou ztrátou, ve většině případů bylo řízení neúspěšné a vedlo k vysoké ztrátě. Jak již bylo zmíněno výše, výjimku tvoří pouze případ $(\hat{\theta}_0, P_0) = (10, 1)$, kdy je použití metody CE velmi vhodné.

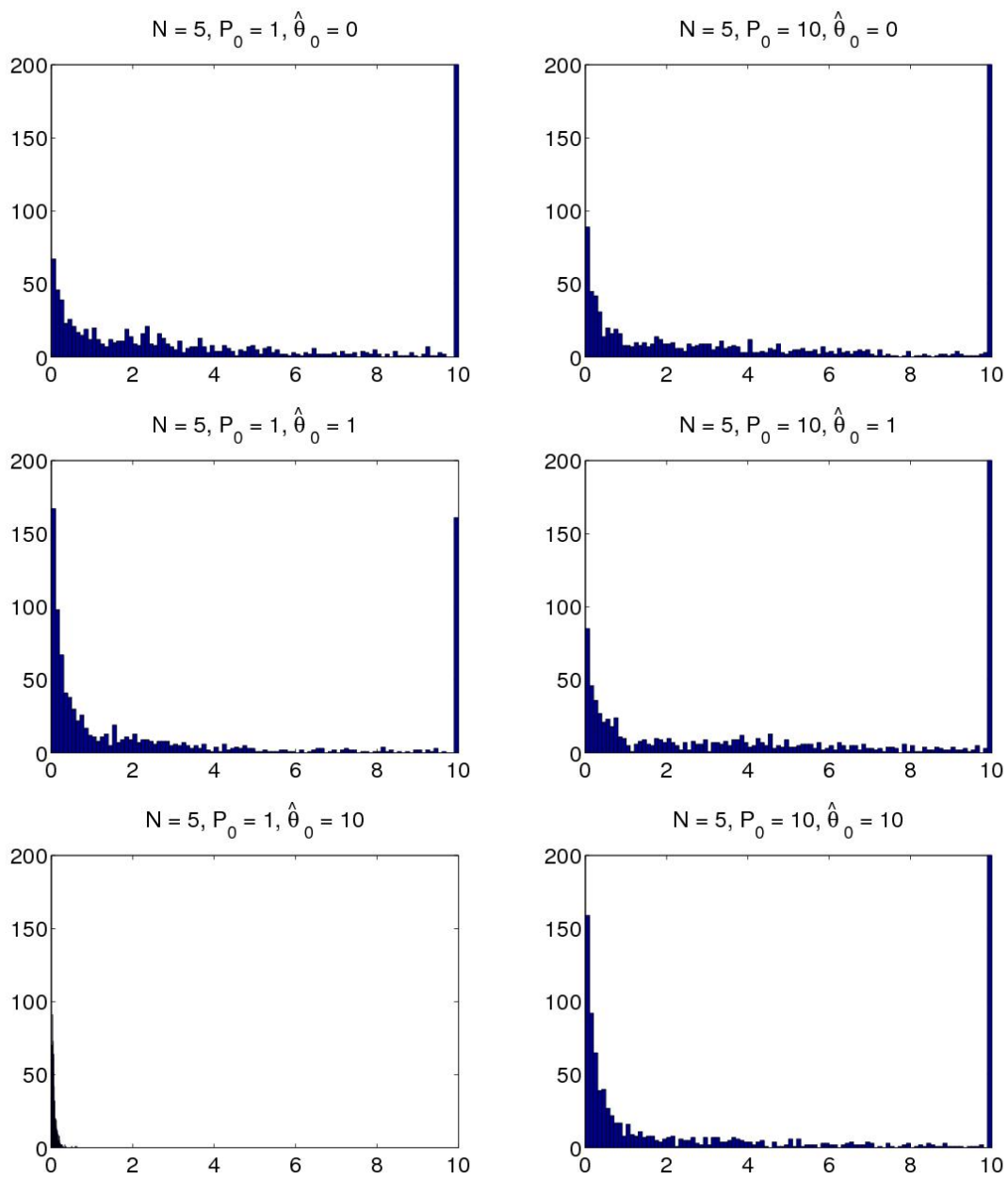
Na 4.4 jsou zobrazeny obdobné výsledky pro metodou cautious control. Oproti metodě CE můžeme pozorovat pokles výrazně špatného řízení pro hodnoty $(\hat{\theta}_0, P_0) = (1, 1)$ a $(\hat{\theta}_0, P_0) = (10, 10)$. Nicméně metoda opatrného řízení se opět ukazuje reálně použitelná pouze pro případ $(\hat{\theta}_0, P_0) = (10, 1)$.

Výsledky pro jednotlivé simulace při řízení metodou SIDP jsou na obrázku 4.5. Hlavní rozdíl oproti předchozím dvěma metodám je zřetelný pokles v počtu totálního selhání řízení. To se objevuje pouze při hodnotách rozptylu $P_0 = 10$ a i tehdy je podstatně nižší než v předchozích případech. Naproti tomu je v mnoha případech dosaženo ztráty, která je vyšší než při použití řízení pomocí CE a CC. Algoritmus SIDP tedy vychází v průměru lépe díky robustnosti vůči výrazně špatnému řízení.

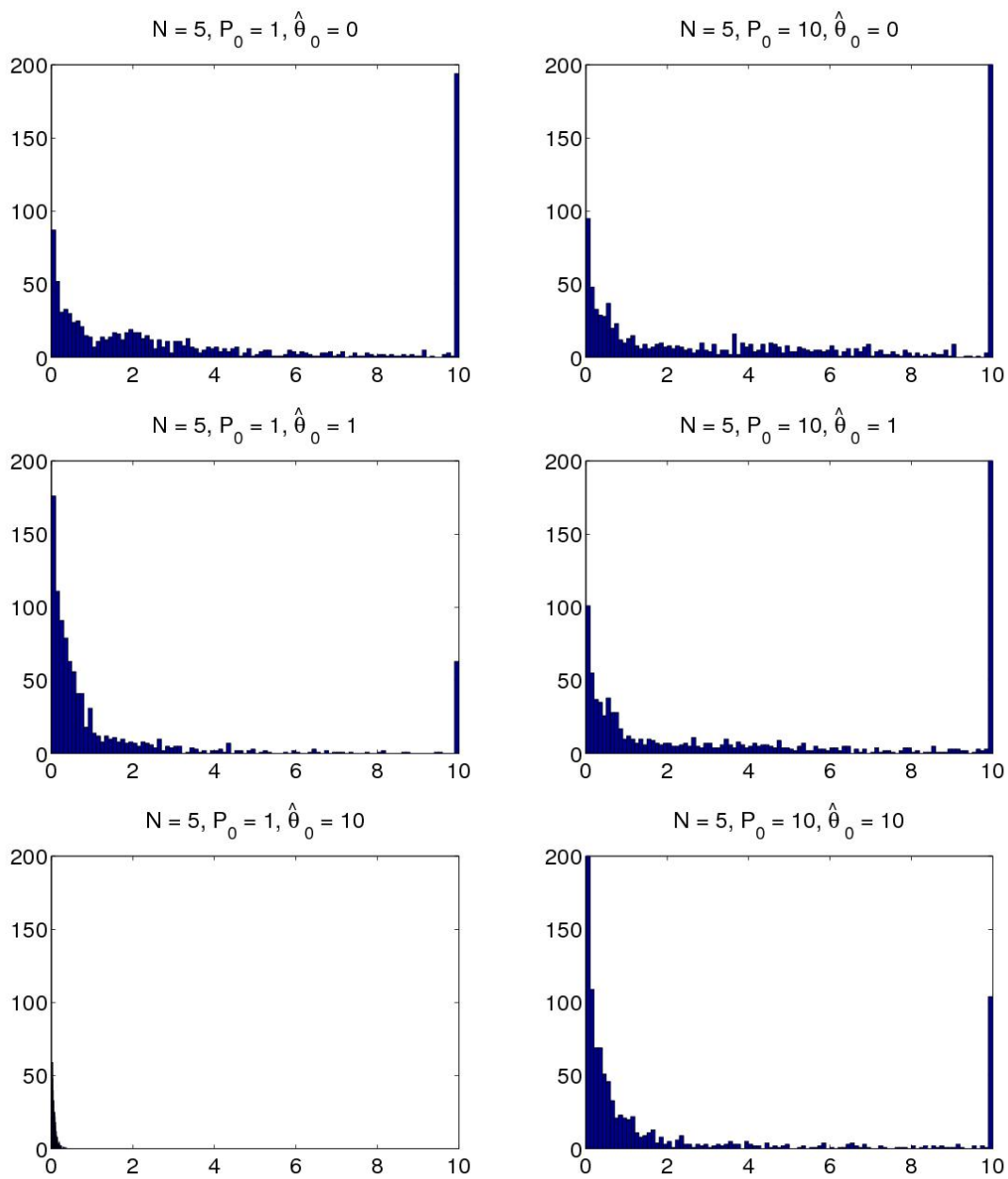
Histogramy pro metodu klasického numerického přístupu k řešení DP zobrazuje 4.6. Ačkoliv kvantitativně vychází řízení pomocí SIDP a DP velmi podobně, při kvalitativním porovnání pomocí histogramů si můžeme všimnout, že řízení metodou DP častěji nabývá nižší celkové ztráty než SIDP, ale rovněž i častěji selhává a dosahuje tak vysoké ztráty.

4.3.2 Shrnutí výsledků simulace

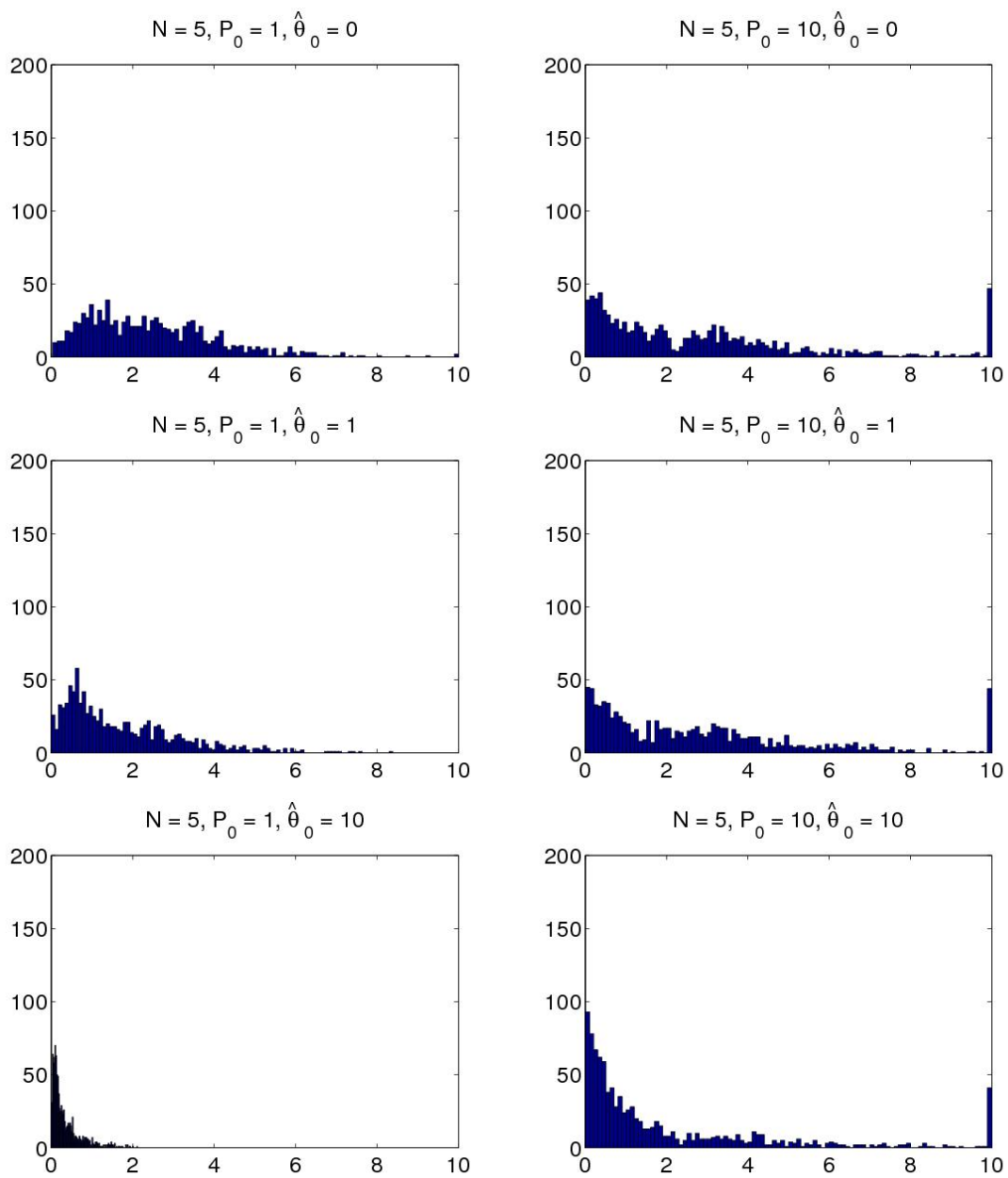
Dle provedených simulací vychází neduální metody (certainty equivalence a cautious control) použitelné pouze pro hodnoty parametrů $(\hat{\theta}_0, P_0) = (10, 1)$. V ostatních



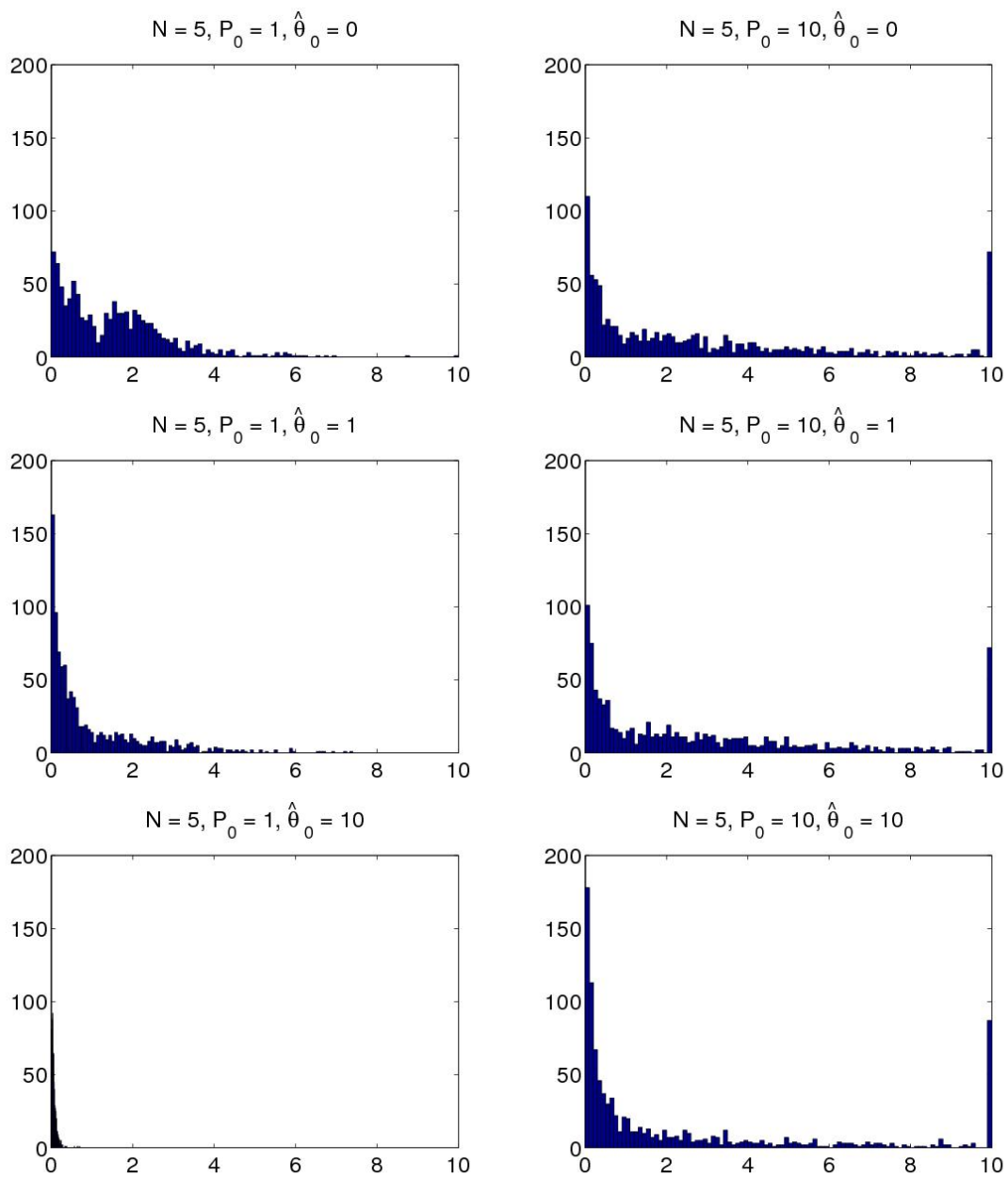
Obrázek 4.3: Výsledky jednotlivých simulací při řízení metodou certainty equivalence



Obrázek 4.4: Výsledky jednotlivých simulací při řízení metodou cautious control



Obrázek 4.5: Výsledky jednotlivých simulací při řízení metodou SIDP



Obrázek 4.6: Výsledky jednotlivých simulací při řízení metodou DP

případech, se metody ukázaly jako nepoužitelné. Podle kvalitativního zkoumání to bylo způsobeno tím, že v mnoha případech, ztráta výrazně překročila rozumnou mez. Přesto se občas podařilo navrhnout úspěšné řízení.

Duální metody (numerické řešení DP převzaté z [1] a SIDP) dokázaly navrhnout úspěšné řízení pro libovolné hodnoty parametrů $(\hat{\theta}_0, P_0)$. Obě zmiňované metody pak dosahovaly obvykle kvantitativně stejných výsledků. Jediný výraznější rozdíl byl při řízení na horizontu délky $N = 10$ pro rozptyl $P_0 = 1$. Tehdy si vedla lépe metoda DP. Metoda SIDP se ukázala jako robustnější, naproti tomu však čatěji dosahovala mírně vyšší ztráty než DP. Chování obou duálních metod tak vychází průměrně velmi podobné. Nutno však připomenout výsledek z [11], že SIDP dosahuje srovnatelného výsledku s DP již při zhruba desetinovém výpočetním čase.

Závěr

Sem přijde zaver

Seznam použitých zdrojů

- [1] K. J. Åström and A. Helmersson. Dual control of an integrator with unknown gain. *Computers & Mathematics with Applications*, 12(6):653–662, 1986.
- [2] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38:91–109, 2001.
- [3] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [4] D.P. Bertsekas. *Dynamic Programming and Optimal Control, vol. 1*. Athena Scientific, 1995.
- [5] AA Feldbaum. *Optimal control systems*. Academic Press, New York, 1965.
- [6] J.M. Hammersley and D.C. Handscomb. *Monte carlo methods*. Taylor & Francis, 1964.
- [7] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.
- [8] R. Luus. *Iterative dynamic programming*. CRC Press, 2000.
- [9] B.L. Nelson, J. Swann, D. Goldsman, and W. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49(6):950–963, 2001.
- [10] V. Peterka. Bayesian system identification. *Automatica*, 17(1):41–53, 1981.
- [11] A.M. Thompson and W.R. Cluett. Stochastic iterative dynamic programming: a Monte Carlo approach to dual control. *Automatica*, 41(5):767–778, 2005.
- [12] B. Wittenmark. Adaptive dual control. *Control Systems, Robotics and Automation, Encyclopedia of Life Support Systems (EOLSS), Developed under the auspices of the UNESCO*, 2002.