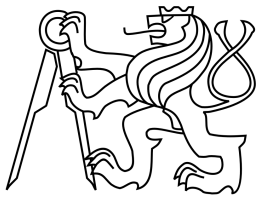
České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky
Obor: Matematické inženýrství
Zaměření: Matematické modelování



Stochastické iterativní aproximace dynamického programování
Stochastic iterative approximations in dynamic programming

BAKALÁŘSKÁ PRÁCE

Vypracoval: Miroslav Zima
Vedoucí práce: Ing. Václav Šmídl , Ph.D
Rok: 2010

Před svázáním místo téhle stránky vložíte zadání práce s podpisem děkana (bude to jediný oboustranný list ve Vaší práci) !!!!

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Miroslav Zima

Poděkování

Děkuji Ing. Václavu Šmídlovi, Ph.D. za vlídné a odpovědné vedení, stejně jako za podnětné návrhy, kterými přispěl ke zvýšení úrovně výsledné práce.

Miroslav Zima

Název práce:
Stochastické iterativní aproximace dynamického programování

Autor: Miroslav Zima

Obor: Matematické inženýrství
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Václav Šmídl , Ph.D
Ústav teorie informace a automatizace Akademie věd České republiky
Konzultant: —

Abstrakt: Tato práce se zabývá teorií řízení za neurčitosti a jejím speciálním případem principialně řešitelným dynamickým programováním. Cílem bylo aplikovat metodu stochastické iterativní aproximace na řešení úlohy dynamického programování. Tento přístup byl implementován pro jednoduchý systém a dosažené výsledky byly porovnány s řízením získaným pomocí jiných metod. V závěru práce jsou diskutovány vlastnosti a použitelnost algoritmu na složitější sytémy.

Klíčová slova: dynamické programování, duální řízení, iterativní dynamické programování, metoda Monte Carlo

Title:
Stochastic iterative approximations in dynamic programming

Author: Miroslav Zima

Abstract: This thesis is concerned with the control theory under uncertainty and with one of its special case in principle solvable by dynamic programming. The aim was to apply the method of stochastic iterative approximations in dynamic programming for solving the dynamic programming task. This approach has been implemented for simple system and obtained results have been compared with a control designed by other methods. In the conclusions the features and usability of the algorithm on more complex systems are discussed.

Key words: dynamic programming, dual control, iterative dynamic programming, Monte Carlo method

Obsah

Značení

V této bakalářské práci je použito následující značení:

t	diskrétní časový okamžik
a_t	hodnota veličiny a v čase t
$\mathop{\mathrm{E}}_{a}$	operátor střední hodnoty s rozdělením pravděpodobnosti P^a
$t\!:\!s$	posloupnost časů $(t, t+1, \ldots, s)$
$a_{t:s}$	posloupnost veličin $(a_t, a_{t+1}, \ldots, a_s)$
$g_{t:s}(a_{t:s})$	posloupnost funkčních hodnot $(g_t(a_t), g_{t+1}(a_{t+1}), \ldots, g_s(a_s))$
$\lvert H \rvert$	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 řídících zásahů.

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

Přímá aplikace tohoto postupu je však bohužel i u poměrně jednoduchých značně komplikována složitostí výpočtu. K řešení úlohy je proto vhodné požít 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. Nicméně duální řízení je pouze obecný přístup k návrhu řízení. Konkrétní návrh je obvykle dílem další aproximace.

Jednou z možných aproximačních metod k návrhu duálního řízení je použití stochastické iterativní aproximace řešení úlohy dynamického programování. To spočívá v použití iterativního dynamického programování (IDP, [8]) a simulační metody Mante Carlo, například [6] . 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

- formulat úlohu stochastického řízení,

- řešit úlohu stochastického řízení s aditivní ztrátou pomocí dynamického programování,
- formulovat úlohu stochastického řízení s neúplným pozorováním a její převedení na úlohu s úplnými znalostmi systému,
- představit některé suboptimální přístupy k úloze stochastického řízení, zejména pak algoritmus stochastického iterativního dynamického programování (SIDP),
- porovnat uvedené řídící algoritmy na jednoduchém systému,
- na základě získaných výsledků diskutovat výhody a nevýhody algoritmu SIDP a jeho použitelnost při aplikaci na složitější úlohy.

Kapitola 1

Úloha stochastického řízení

1.1 Základní úloha stochastického řízení

Tento oddíl se zabývá obecnou formulací úlohy stochastického řízení a pojmy s tím spojenými.

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

$$x_{t+1} = f_k(x_t, u_t, w_t), \qquad t = 0, 1, \ldots, N-1. \tag{1.1}$$

Zde x_t představuje stav systému v čase t, u_t řídící zásah v čase t a w_t náhodnou veličinu reprezentující přítomnost šumu. 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. Případ neúplného pozorování bude probrán později.

1.1.2 Ztrátová funkce a optimální řízení

Cílem řízení je pro systém popsaný soustavou (1.1) navrhnout regulátor (posloupnost řídících zásahů), který bude stav systému udržovat co nejblíže požadované hodnotě. Pro tyto účely máme v úloze řízení k dispozici předepsanou ztrátovou (resp. účelovou) funkci

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

která určuje nakolik jsme vytyčených cílů dosáhli.

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

$$\mu_t(x_t) = u_t \qquad t = 0, 1, \ldots, N-1, \tag{1.3}$$

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

Pro danou řídící strategii označme očekávanou ztrátu jako

$$J_\pi(x_0) = \underset{w_{0:N-1}}{\mathrm{E}} \left\{ g(x_{1:N}, \mu_{0:N-1}(x_{0:N-1})) \right\}. \tag{1.4}$$

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

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

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

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

Očekávanou ztrátu (1.4) potom můžeme přepsat do tvaru

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

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 rozhudnutí 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 řídící strategie nebyl optimální, pak očekávanou ztrátu snížíme přechodem ke strategii, ve které onu neoptimální část nahradíme optimálním řešením podproblému na tomtéž 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

$$J_N(x_N) = 0 \tag{1.8}$$

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

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

$$J_t(x_t) = \mathrm{E}_{w_t} \{g_k(x_t, \mu_t(x_t), w_t) + J_{t+1}(f_t(x_t, \mu_t(x_t), w_t))\} \qquad t = 0, \ldots, N-1 \tag{1.10}$$

pak bude optimální posloupností rozhodnutí.

1.3 Úloha stochastického řízení s nepřesnými daty

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.

Tento oddíl se zabývá modifikací úlohy stochastického řízení pro případ přítomnosti neznámých parametrů.

1.3.1 Výstup systému a infomační vektor

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

$$y_0 = h_0(x_0, v_0), \qquad y_{t+1} = h_{t+1}(x_{t+1}, u_t, v_{t+1}), \qquad t = 1, \ldots, N-1, \tag{1.11}$$

kde v_t je náhodná veličina charakterizující chybu měření. Předpokládáme znalost funkcí h_t. 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

$$I_0 = y_0, \qquad I_{t+1} = (y_{0:t+1}, u_{0:t}), \qquad t = 1, \ldots, N-1. \tag{1.12}$$

1.3.2 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 řídících zásahů za informace I_t. Přípustnou řídící strategií $\pi = \mu_{0:N-1}$ bude posloupnost

$$\mu_t(I_t) = u_t \qquad t = 0, 1, \ldots, N-1, \tag{1.13}$$

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

Úkolem je najít přípustnou strategii, která by minimalizovala očekávanou ztrátu

$$J_\pi = \mathop{\mathrm{E}}_{\substack{x_0,\ w_{0:N-1},\\ v_{0:N}}} \left\{ \sum_{t=0}^{N-1} g_t(x_{t+1}, \mu_t(x_t)) \right\}, \tag{1.14}$$

za podmínek (1.1) a (1.11).

1.3.3 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

$$I_0 = y_0, \qquad I_{t+1} = (I_t, u_t, y_{t+1}), \qquad t = 1, \ldots, N-1. \tag{1.15}$$

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 (1.11).

Dále přejdeme k nové ztrátové funkci, kterou definujeme jako

$$\tilde{g}_t(I_{t+1}, u_t) = \mathop{\mathrm{E}}_{x_{t+1}} \left\{ g_t(x_{t+1}, u_t) | I_t, u_t \right\}, \qquad t = 1, \ldots, N-1, \tag{1.16}$$

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

$$J_N(I_N) = 0 \tag{1.17}$$

$$J_t(I_t) = \min_{u_t \in U_t} \mathop{\mathrm{E}}_{w_t, y_{t+1}} \{\tilde{g}_t(I_{t+1}, u_t) + J_{t+1}(I_{t+1}) | I_t, u_t\} \qquad t = 0, \ldots, N-1 \tag{1.18}$$

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

1.4 Úloha řízení systému s neznámými parametry

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

1.4.1 Systém s neznámými parametry, hyperstav

V této úloze máme výstupy systému y_t popsány jako

$$y_0 = h_0(\theta, v_0), \qquad y_{t+1} = h_t(I_t^{(d)}, \theta, u_t, v_{t+1}), \qquad t = 0, \ldots, N-1, \tag{1.19}$$

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

$$f(\theta|T_{t+1}) = \frac{f(y_{t+1}|\theta, I_t^{(d)}, u_t) f(\theta|T_t)}{\int f(y_{t+1}|\theta, I_t^{(d)}, u_t) f(\theta|T_t) \mathrm{d}\theta} \tag{1.20}$$

Rekurzivní použití vzorce (1.20) pro odhad parametru θ se nazývá postup Bayesovského učení [10].

Pro vývoj hyperstavu H_t v čase můžeme na základě (1.20) psát

$$H_{t+1} = f_t(H_t, u_t, y_{t+1}), \qquad t = 1, \ldots, N-1. \tag{1.21}$$

Rovnici (1.21) můžeme podobně jako (1.15) považovat za rovnici systému (1.1) pro stav H_t a vstup u_t s šumem y_{t+1}.

1.4.2 Převod na úlohu s nepřesnými daty

Ztrátová funkce je nyní

$$g(y_{1:N}, u_{0:N-1}) = \sum_{t=0}^{N-1} g_t(y_{t+1}, u_t). \tag{1.22}$$

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

$$J_\pi = \underset{\theta_0, v_{0:N-1}}{\mathrm{E}} \left\{ \sum_{t=0}^{N-1} g_t(y_{t+1}, \mu_t(H_t)) \right\}, \tag{1.23}$$

za apriorní informace $f(\theta|T_0)$, známého rozdělení šumu v_t a podmínek (1.21) a (1.19).

Rovnice (1.21), (1.19) a (1.22) potom představují úlohu stochastického řízení s nepřesnými daty.

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

$$J_N(H_N) = 0 \tag{1.24}$$

$$J_t(H_t) = \min_{u_t \in U_t} \underset{y_{t+1}}{\mathrm{E}} \left\{ g_t(y_{t+1}, u_t) + J_{t+1}(H_{t+1}) | H_t, u_t \right\}, \qquad t = 0, \ldots, N-1, \tag{1.25}$$

kde H_{t+1} se počítá dle (1.21). Střední hodnota vzhledem k y_{t+1} se počítá pomocí (1.19) a $f(\theta|T_t)$ jakožto aktuálního odhadu na parametr θ.

1.4.3 Kalmanův filtr

Pokud v rovnicích (1.19) popisujících výstup systému vystupuje aditivní gaussovký šum a neznámý parametr je separován jako lineání člen, můžeme vypočítat konkrétní tvar rovnice (1.21), tzv. Kalmanův filtr [7].

Dle předpokladu má výstup v čase t tvar

$$y_{t+1} = \tilde{h}_t(I_t, u_t) + A_t(I_t, u_t)\theta + v_{t+1}, , \qquad t = 0, \ldots, N-1. \tag{1.26}$$

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

$$v_{t+1} \sim N(0, Q_{t+1}), \tag{1.27}$$

gaussovské rozložení odhadu neznámého parametru θ_t a jejich nekorelovanost, tedy

$$\theta_t \sim N(\hat{\theta}_t, P_t), \tag{1.28}$$

$$\mathrm{Cov}(v_{t+1}, \theta_t) = 0. \tag{1.29}$$

Dosazením do (1.20) 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

$$K_t = P_t A_t (A_t^T P_t A_t + Q_t)^{-1} \tag{1.30}$$

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t(y_{t+1} - \tilde{h}_t(I_t, u_t) - A_t\hat{\theta}_t), \tag{1.31}$$

$$P_{t+1} = (I - K_t A_t)P_t. \tag{1.32}$$

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

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t(y_{t+1} - \mathop{\mathrm{E}}_{\theta,v_t} y_{t+1}), \tag{1.33}$$

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

$$P_{t+1}(K_t) = \mathrm{E}[(\theta - \hat{\theta}_{t+1})(\theta - \hat{\theta}_{t+1})^T]. \tag{1.34}$$

Dosazením za $\hat{\theta}_{t+1}$ z (1.33) a za y_t ze (1.26) a úpravou dostaneme (pro libovolnou matici B budeme pro lepší čitelnost namísto BB^T psát zkráceně B^2)

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

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

$$P_{t+1}(K_t) = (I - K_tA_t)P_t(I - K_tA_t)^T + K_tQ_tK_t^T. \tag{1.35}$$

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

$$\frac{\partial\,\mathrm{tr}(P_t)}{\partial K_t} = 0. \tag{1.36}$$

K provedením derivace použijeme vzorce

$$\frac{\partial\,\mathrm{tr}(MXN)}{\partial X} = M^TN^T, \tag{1.37}$$

$$\frac{\partial\,\mathrm{tr}(MXNX^TO)}{\partial X} = M^TO^TXN + OMXN, \tag{1.38}$$

kde M, N a O jsou konstantní matice.

Tím získáme lineární rovnici pro K_t tvaru

$$-P_t^T A_t - P_t A_t + K_t A_t P_t K_t + K_t A_t^T P_t K_t + 2Q_t K_t = 0, \tag{1.39}$$

která má řešení

$$K_t = P_t A_t (A_t^T P_t A_t + Q_t)^{-1} \tag{1.40}$$

Dosazením (1.40) do (1.35) po úpravě dostaneme

$$P_{t+1} = (I - K_t A_t) P_t \tag{1.41}$$

Rovnice (1.33), (1.40) a (1.41) představují rovnice Kalmanova filtru.

Kapitola 2

Suboptimální přístupy k návrhu řídící strategie

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 problémemy: 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 řídící strategie $\pi = \mu_{0:N-1}$, která by minimalizovala očekávanou ztrátu

$$J_\pi = \mathop{\mathrm{E}}_{\theta_0, v_{0:N-1}} \left\{ \sum_{t=0}^{N-1} g_t(y_{t+1}, \mu_t(H_t)) \right\}, \tag{2.1}$$

za apriorní informace θ_0 a podmínek

$$H_{t+1} = f_t(H_t, u_t, y_{t+1}), \tag{2.2}$$

$$y_0 = h_0(\theta, v_0), \qquad y_{t+1} = h_t(I_t^{(d)}, \theta, u_t, v_{t+1}), \qquad t = 0, \ldots, N-1, \tag{2.3}$$

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 řídícího horizontu

$$J_t(H_t) = \min_{u_t \in U_t} \mathop{\mathrm{E}}_{y_{t+1}} \left\{ g_t(y_{t+1}, u_t) + J_{t+1}(H_{t+1}) | H_t, u_t \right\}, \qquad t = 0, \ldots, N-1, \tag{2.4}$$

kde T_{t+1} a y_{t+1} se počítá dle (2.2) a (2.3).

2.1 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 obvykle v 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 řídícího $\mu_t(H_t)$, aby odpovídal principům duálního řízení, je obtížná úloha. Často (například [12]) se tak volí aproximace tvaru

$$\mu_t(H_t) = \mu_t^{(1)}(H_t) + \mu_t^{(2)}(H_t), \tag{2.5}$$

kde $\mu_t^{(1)}$ je řídící 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 (2.5) vhodně určit funkce $\mu_t^{(1)}$ a $\mu_t^{(2)}$. Vhodným výběrem na řídící 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.

2.2 Certainty equivalent control

Při použití metody Certainty equivalent (CE, například [4]) se v rovnici pro očekávanou ztrátu (2.4) náhodná veličina y_{t+1} nahradí střední hodnotou $\hat{y}_{t+1}$. Ta se vypočítá z (2.3) pomocí známých rozdělení na v_t a postačující statistiky T_t pro parametr θ. Očekávaná ztráta (2.4) tak přejde v

$$J_t(H_t) = \min_{u_t \in U_t} \left\{ g_t(\hat{y}_t, u_t) + J_{t+1}(\hat{H}_{t+1}) | H_t, u_t \right\}, \qquad t = 0, \ldots, N-1, \tag{2.6}$$

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 touto metodou nemusí být optimální pro původní úlohu. Některé vlastnosti CE jsou probírány dále při aplikaci na řízení jednoduchého systému. Podrobnější pojednání lze nalézt v [4].

2.3 Opatrné řízení

Metoda opatrného řízení (anglicky cautious control, například [12]) spočívá v optimalizaci očekávané ztráty (2.4) na horizontu délky $N = 1$. Minimalizujeme tedy

$$J_0(H_0) = \min_{u_0 \in U_0} \mathrm{E}_{y_1} \left\{ g_0(y_1, u_0) | H_0, u_0 \right\}. \tag{2.7}$$

Stačí proto spočíst střední hodnotu $g_0(y_1, u_0)$ vzhledem k y_1 a výsledek minimalizovat vyhledem 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 vybuzení systému mimo požadovaný stav.

2.4 Iterativní dynamické programování

Iterativní dynamické programování [8] je jednou z variant klasického přístupu k nalezení optimální strategie, která minimalizuje očekávanou ztrátu (2.4). Standardní numerický přístup k dynamickému progamování 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 řídícího horizontu.

Tento postup je přímočarou aplikací principu dynamického programování. Bohužel je velmi citlivý na dimenzi stavového prostoru H_t. Při diskretizaci daného protoru totiž počet použitých bodů roste exponenciálně s jeho dimenzí. Tato skutečnost, v anglické literatuře označovaná jako "curse of dimenzionality", pak prakticky znemožňuje řešení komplexnějších úloh.

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. Jak bude ještě diskutováno, výhodnost iterativního přístupu spočívá v jeho nížší citlivosti na dimenzi úlohy.

2.4.1 Diskretizace prostoru

Při hledání optimální strategie $\mu_t(H_t)$ je pro přesné vyčíslení očekávané ztráty (2.8) na úseku řídícího 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 (2.8) 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.

2.4.2 Konvergence metody

Lze ukázat, že za určitých, relativně obecných, předpokladů, algoritmus iterativního dynamického programování konverguje k optimálnímu řešení. Předpoklady a hlavní teorémy zaručující konvergenci metody jsou shrnuty v tomto oddíle. Jejich znění je převzato z [11]. Důkazy je možno nalézt tamtéž.

Označme $U = \{U_1, U_2, \ldots, U_{N-1}\}$ množinu přípustných řídících zásahů. Dále buď $J_0(u)$ hodnota očekávané ztráty v závislosti na zvolené posloupnosti řídících zásahů $u = \{u_1, u_2, \ldots, u_{N-1}\}$ pro konkrétní hodnotu počátečního stavu. Počáteční rozsah pro hledání optimálního u^* pomocí perturbací dosavadního napočteného řešení označíme β^{in}, faktor redukující počáteční rozsah v průběhu iterací IDP pak γ ($\gamma < 1$). Řešení napočtené v j-té iteraci algoritmu budeme značit $u^{*[j]}$.

Předpoklady 1 *Základní předpoklady pro použití metody IDP jsou*

- $U \in \mathbb{R}^N$ *je kompaktní,*
- $J_0(u)$ *je ostře konvexní na* U*,*
- $J_0(u)$ *je zdola omezená na* U*,*
- *množina* $L(u^{*[j]}) = \{u \in U | J_0(u) \leq J_0(u^{*[j]})\}$ *je kompaktní* $\forall u^{*[j]}$

Věta 1 *Za předpokladů 1 existuje* u^* *splňující* $u^* = \operatorname{argmin}_{u \in U} J_0(u)$*. Navíc* u^* *je konečné a právě jedno.*

Věta 2 *Množina bodů* $u^* \in \{\operatorname{argmin}_{u \in U} J_0(u)\}$ *jsou pevné body iterační posloupnosti* $\{u^{*[j]}\}_{j=1}^{\infty}$ *generované pomocí IDP.*

Věta 3 *Posloupnost iterací* $\{u^{*[j]}\}_{j=1}^{\infty}$ *generovaná pomocí IDP je vždy konvergentní. Navíc pro dostatečně velké* γ *(tj. pro* $\gamma \to 1$*) konverguje iterační posloupnost pro libovolné počáteční* $u^{*[0]}$ *k* u^* *s pravděpodobností rovnou jedné.*

2.5 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 (2.4).

2.5.1 Použití metody Monte Carlo k výpočtu očekávané ztráty

Pří 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 z n realizací náhodných veličin přes které je prováděna střední hodnota (jsou to $\theta_{t:N-1}$ a $v_{t:N}$), tedy

$$\frac{1}{n}\sum_{i=1}^{n}\left(g_j(y^i_{j+1},\mu_j(H_j)+\sum_{j=t+1}^{N-1}g_j(y^i_{j+1},\mu_j(H^i_j)\right), \tag{2.8}$$

kde y^i_{j+1} se počítá podle (2.3) jako

$$y^i_{j+1}=h_j(I^i_j,\theta^i_j,\mu(H^i_j),v^i_{j+1}), \qquad j=t,\ldots,N-1, \qquad i=1\ldots,n, \tag{2.9}$$

a index i označuje i-tou realizaci dané veličiny. Realizace $\theta_{t:N-1}$ se generují podél trajektorie (2.3). To znamená, že θ^i_{j+1} se generuje až ve chvíli, kdy je známé I^i_j, u^i_j, postačující statistika T^i_j a y^i_{j+1} a tedy přes (2.2) i postačující statistika T^i_{j+1}.

Výpočet je při uchovávání $\mu_t(H_t)$ namísto $J_t(H_t)$ časově náročnější. Namísto přečtení $J_t(H_t)$ je totiž nutné přečíst hodnotu $\mu_t(H_t)$ a následně vygenerovat trajektorii od času t do konce horizontu. To obnáší generovat náhodnou realizaci šumu v_t a neznámého parametru θ (pomocí T_t), aplikovat řídící zásah, tedy dle (2.3) vypočítat y_{t+1} a následně z (2.2) 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 řídícího horizontu. Při výpočtu postupně napočítáváme hodnotu ztrátové funkce.

2.6 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 optimálního řízení.

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

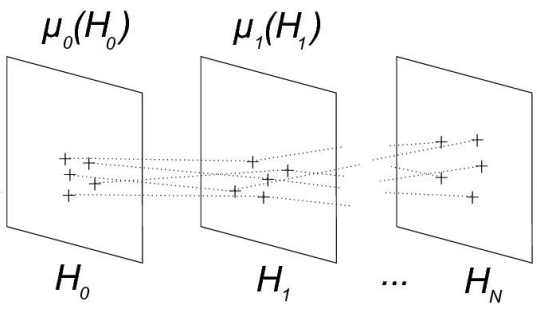
Tabulka 2.1: Parametry algoritmu SIDP

2.6.1 Algoritmus SIDP

V tomto odílu je popsán algoritmus SIDP, tak jak byl navržen v [11]. Parametry algoritmu jsou uvedeny v tabulce 2.1. Jak plyne z uvedeného schématu, časová složitost algoritmu SIDP vzhledem k jeho parametrům a délce řídícího horizontu 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).

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 π^*, její interpolace a extrapolace a realizace neznámého parametru $\theta_0, \ldots, \theta_{N-1}$ 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 $m = -\left[\frac{m-1}{2}\right]$ to $\left[\frac{m}{2}\right]$ **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



Obrázek 2.1: Trajektorie v hyperstavu – jednotlivá realizace trajektorie je napočítávána pomocí realizací šumu a neznámého parametru

2.6.2 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 2.1. Jak danou část prostoru diskretizovat je otázka implementace algoritmu na konkrétní systém, bude tedy probrána později.

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.

Pro každý z bodů (nejprve pro ty na konci řídícího horizontu) se optimální řídící 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 (2.8) spočte průměr. Řídící zásah, pro který bude průměr přes realizace ztráty nejnižší, se ponechá do další iterace.

Namísto jednoduchého porovnání pomocí průměru lze kandidáty na optimální řídící zásah porovnat nějakým sofistikovanějším víceúrovňovým algoritmem. Jedno z možných vylepšení je použito i v [11]. Konkrétně se jedná o dvouúrovňový algoritmus poposaný v článku [9]. V první úrovni tohoto algoritmu se pro každého kandidáta u_t vygeneruje n_0 realizací. Na jejich základě se vyberou ti, na kterých 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 tabulka optimálních řídících zásahů bodech diskretizace. Samotné řízení systému je pak prováděno aplikací předpočtených optimálních zásahů.

Kapitola 3

Srovnání suboptimální přístupů při řízení jednoduchého systému

V této kapitole je popsán jednoduchý systém, na kterém jsou porovnány řídící algoritmy založené na principech uvedených v předešlé kapitole. Řídící algoritmy byly implementovány v prostředí *Matlab*.

3.1 Integrátor s neznámým ziskem

Systém tvaný integrátor s neznámým ziskem, byl podrobně zkoumán v [1]. Pro srovnání uvádíme tamější výsledky.

3.1.1 Popis systému

Výstup systému je popsán jako

$$y_{t+1} = y_t + \theta u_t + v_{t+1} \qquad t = 0, \ldots, N-1, \tag{3.1}$$

$$v_{t+1} \sim N(0, \sigma^2), \tag{3.2}$$

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 odhadu θ_t s šumem, tedy že

$$\mathrm{Cov}(v_{t+1}, \theta_t) = 0. \tag{3.3}$$

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

$$g(y_{1:N}, u_{0:N-1}) = \sum_{t=0}^{N-1} y_{t+1}^2. \tag{3.4}$$

Odhadovací procedurou pro parametr θ je Kalmanův filtr. Pro systém (3.1) má tvar

$$K_t = \frac{u_t P_t}{u_t^2 P_t + \sigma^2} \tag{3.5}$$

$$\hat{\theta}_{t+1} = \hat{\theta}_t + K_t(y_{t+1} - u_t\hat{\theta}_t), \tag{3.6}$$

$$P_{t+1} = (1 - K_t u_t)P_t. \tag{3.7}$$

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

$$J_t(H_t) = \min_{u_t \in U_t} \mathrm{E}_{y_{t+1}} \left\{ y_{t+1}^2 + J_{t+1}(H_{t+1}) | H_t, u_t \right\}, \qquad t = 0, \ldots, N-1. \tag{3.8}$$

Ta po dosazení z (3.1) a částečném provedení střední hodnoty přejde na tvar

$$J_t(H_t) = \min_{u_t \in U_t} \left\{ (y_t + \hat{\theta}_t u_t)^2 + u_t^2 P_t + \sigma^2 + \mathrm{E}_{y_{t+1}} \left(J_{t+1}(H_{t+1}) \right) | y_t, \hat{\theta}_t, P_t, u_t \right\}. \tag{3.9}$$

3.1.2 Transformace rovnic systému

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

$$\eta_t = \frac{y_t}{\sigma}, \tag{3.10}$$

$$\beta_t = \frac{\hat{\theta}_t}{\sqrt{P_t}}, \tag{3.11}$$

$$\zeta_t = \frac{1}{\sqrt{P_t}}, \tag{3.12}$$

$$\nu_t = \frac{u_t\sqrt{P_t}}{\sigma}. \tag{3.13}$$

Současně můžeme neurčitost ve výstupu (3.1) reprezentovat jedinou normalizovanou náhodnou veličinou podle

$$s_t = \frac{y_{t+1} - y_t + \hat{\theta}_t u_t}{\sqrt{u_t^2 P_t + \sigma^2}} \sim N(0, 1). \tag{3.14}$$

Rovnice pro výstup (3.1) a následující odhad neznámého parametru (3.6) tak přejde v

$$\eta_{t+1} = \eta_t + \beta_t\nu_t + \sqrt{1 + \nu_t^2}\, s_t \tag{3.15}$$

$$\beta_{t+1} = \sqrt{1 + \nu_t^2}\,\beta_t + \nu_t s_t \tag{3.16}$$

Přejdeme-li k vhodně upravené očekávané ztrátě, dostaneme

$$V_t(\eta_t, \beta_t, \zeta_t) = \frac{J_t(y_t, \hat{\theta}_t, P_t)}{\sigma^2} \tag{3.17}$$

$$= \min_{\nu_t} \left\{ (\eta_t + \beta_t\nu_t)^2 + \nu_t^2 + 1 + \mathrm{E}_{s_t}(V_{t+1}(\eta_{t+1}, \beta_{t+1}, \zeta_{t+1})) \right\}. \tag{3.18}$$

Nyní spočteme očekávanou ztrátu pro $N-1$ jako

$$V_{N-1}(\eta_{N-1}, \beta_{N-1}, \zeta_{N-1}) = \min_{\nu_{N-1}} \left\{ (\eta_{N-1} + \beta_{N-1}\nu_{N-1})^2 + \nu_{N-1}^2 + 1 \right\}. \tag{3.19}$$

Pomocí diferenciálního počtu pak získáme optimální zásah ve tvaru

$$\nu_{N-1} = -\frac{\eta_{N-1}\beta_{N-1}}{1+\beta_{N-1}^2} \tag{3.20}$$

a očekávanou ztrátu rovnu

$$V_{N-1}(\eta_{N-1}, \beta_{N-1}, \zeta_{N-1}) = \frac{\eta_{N-1}^2 + 1}{\beta_{N-1}^2 + 1}. \tag{3.21}$$

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 (3.1) dostaneme požadovaný řídící zásah vhodnou volbou znaménka napočteného zásahu.

3.2 Použité řídící algoritmy

V tomto oddílu jsou popsány řídící algoritmy, které budou posléze porovnány při řízení systému (3.1). Algoritmy jsou založené na principech uvedených v předešlé kapitole.

3.2.1 Klasický přístup k dynamickému programování

V článku [1] je problém řízení systému (3.1) řešen pomocí přímé aplikace numerických metod na řešení úlohy dynamického programovaní. Jde o schéma popsané v sekci 2.4. 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, nalezlo její minimum a to se pak testovalo, zda-li je globálním minimem očekávané ztráty.

Výslednou aproximaci optimálního řízení pak autoři uvádějí jako

$$\nu_t = -\frac{0,56 + \beta_t}{2,2 + 0,08\beta_t + \beta_t^2}\eta_t - \frac{1,9}{1,7 + \beta_t^4} \tag{3.22}$$

První člen v (3.22) můžeme interpretovat jako modifikované opatrné řízení, druhý člen pak jako budící složku řízení.

3.2.2 Certainty equivalent control

Aplikací metody certainty equivalent (CE) přejde očekávaná ztráta (3.17) v

$$V_t(\eta_t, \beta_t) = \min_{\nu_t} \left\{ \hat{\eta}_{t+1}^2 + V_{t+1}(\eta_{t+1}, \beta_{t+1}) \right\} \tag{3.23}$$

Střední hodnota výstupu je dle (3.15) rovna

$$\hat{\eta}_{t+1} = \eta_t + \beta_t \nu_t, \tag{3.24}$$

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

$$\nu_t = -\frac{\eta_t}{\beta_t}. \tag{3.25}$$

Pokud $\beta_t = 0$, pak to dle (3.11) znamená, že i $\hat{\theta}_t = 0$ (to se může stát ačkoliv $\theta \neq 0$). Aktuální očekávaná ztráta pak nezávisí na ν_t a můžeme tedy volit libovolný řídící zásah bez vlivu na hodnotu očekávané ztráty. V takovém případě volíme za řídící zásah realizaci bílého šumu $N(0, 1)$. Jedná se vlastně o jednoduchou aplikaci principu duálního řízení popsaného rovnicí (2.5).

Z tvaru optimálního řídícího zásahu (3.25) snadno zjistíme, že CE je metodou neduální. Pokud je totiž výstup systému na požadované hodnotě, řídící zásah nebude systém vychylovat za účelem lepší identifikace parametru θ. Nicméně díky tomu, že máme k dispozici analitické vyjádření ν_t, může být řízení pomocí metody CE prováděno s minimálními výpočetními nároky. To může být v některých aplikacích rozhodující výhodou.

Aplikace metody CE v podobě řídícího zásahu (3.25) je zjevně nevhodná pro malé hodnoty β_t. Metoda bude totiž generovat, příliš velké řídící zásahy bez ohledu na možnou přítomnost neurčitosti. Konkrétní důsledky budou prezentovány dále.

3.2.3 Metoda opatrného řízení

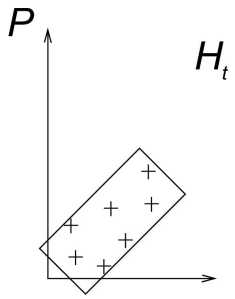
Optimální řídící zásah je pro metodu opatrného řízení (caution control, CC) dán dle (3.20) jako

$$\nu_t = -\frac{\eta_t \beta_t}{1 + \beta_t^2}. \tag{3.26}$$

Všimněme se, že pro velké hodnoty β_t přejde optimální zásah pro metodu opatrného řízení (3.26) v optimální zásah metody CE (3.25). Naproti tomu pro malé hodnoty β_t (tedy pro velké hodnoty neurčitosti v identifikaci parametru θ, viz (3.11)) bude řízení podstatně konzervativnější. Opět se podařilo získat analitické vyjádření ν_t a výpočet optimálního řízení pomocí metody CC může být prováděn velmi efektivně.

3.2.4 SIDP

Dle popisu (a následné transformace) systému (3.1) je pro výpočet optimální strategie pomocí algoritmu SIDP nutné diskretizovat část dvoudimenzionálního prostoru



Obrázek 3.1: Oblast určená k diskretizaci H_t pomocí nejmenšího obdélníku – body uvnitř obecně orientovaného obdélníka nemusí splňovat požadavek na nezápornost

nezávisle proměnných $H_t = (\eta_t, \beta_t)$. Jak bylo výše zdůvodněno, optimální řídící zásahy stačí napočíst pro kladné hodnoty (η_t, β_t).

V práci [11], kde je metoda SIDP navržena, je pro diskretizaci prostoru použita oblast obdélníkového tvaru. Kolem vygenerovaných bodů se vytvoří nejmenší obdélník, který odsahuje všechny vygenerované body. Metodu k jeho určení převzali autoři z [2]. Nicméně při použití této metody není zaručeno, že část takto vygenerovaného obdélníku nebude obsahovat i záporné hodnoty (η_t, β_t), viz obrázek 3.1.

V této práci se proto volí odlišná metoda. K diskretizaci zasažené části prostoru se volí opět oblast obdélníkového tvaru. Ta se určí následující jednoduchou metodou (matici obsahující všechny vygeneované body označme $A = (x_1, x_2, \ldots, x_N) \in \mathbb{R}_+^{2,N}$)

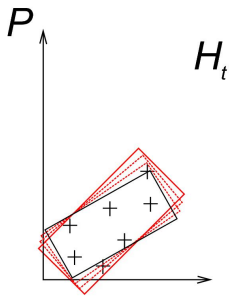
- spočtou se vlastní čísla a vektory kovarianční matice AA'
- vlastní vektory určují směr hran kvádru, jejich délka je $4\sqrt{\lambda_i}/N$
- za střed obdélníka se zvolí těžiště množiny $\{x_1, x_2, \ldots, x_N\}$.

Pokud část obdélníka obsahuje záporné hodnoty (η_t, β_t), vezme se vhodné natočení a zmenšení původního. To se provede iterativně, kdy v každé iteraci se původní obdélník natočí o 5° a zmenší o 5 %. V iteracích se pokračuje dokud není splněna podmínka na nezáporné hodnoty (η_t, β_t) uvnitř obdélníka. Možnou situaci ilustruje obrázek 3.2

Implementace dalších částí algoritmu byla provedena v souladu s oddílem 2.6.2. Konkrétní nastavení parametrů algoritmu zachycuje tabulka 3.1. Výpočet za daných parametrů trval v řadech minut.

3.3 Srovnání jednotlivých přístupů

V této sekci jsou porovnány popsané algoritmy při řízení systému (3.1). Systém je možné v případě potřeby vhodně posunou či přeškálovat, není tedy potřeba uvažovat



Obrázek 3.2: Oblast určená k diskretizaci H_t pomocí zde použité metody – v případě potřeby se napočtený obdélník vhodně zmenší a natočí

počet opakování algoritmu	n_{pass}	3
počet iterací algoritmu	n_{iter}	8
počet bodů v diskretizaci každé dimenze H_t	n_g	10
apriorní řídící strategie	$\mu_{0:N-1}$	0
počet kadnidátů na změnu řídícího zásahu	m	7
počáteční rozsah pro hledání optimálního řídícího zásahu	β^{in}	1
parametr pro redukci β^{in} při opakování algoritmu	γ	0,9
parametr pro redukci β^{in} při iterování algoritmu	λ	0,5
počet realizací pro odhad metodou Monte Carlo	n	20

Tabulka 3.1: Konkrétní volba parametrů algoritmu SIDP

jiné hodnoty referenčního signálu a počátečního výstupu. Kvalitu výsledného řízení posuzujeme z hlediska celkové ztráty vygenerované podél horizontu délky N.

Očekávaným výsledek bylo, že v případě velké počáteční neznalosti systému bude duální řízení získané pomocí SIDP výhodnější oproti neduálním metodám. Ty zde zastupuje metoda certainty equivalence (CE) a metoda caution control (CC). Pro srovnání je uveden výsledek získaný klasickým numerickým přístupem k dynamickému programování (DP) převzatý z [1]. Jak vyplývá z tvaru (3.22), jedná se 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 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 nezbytná v další iteraci algoritmu. Pro efektivní diskretizaci tedy stačí výrazně méně bodů (ve výsledcích publikovaných v práci [11] to bylo 64x64 bodů pro metodu DP, zatímco pro SIDP jen 20x20).

3.3.1 Kvantitativní srovnání

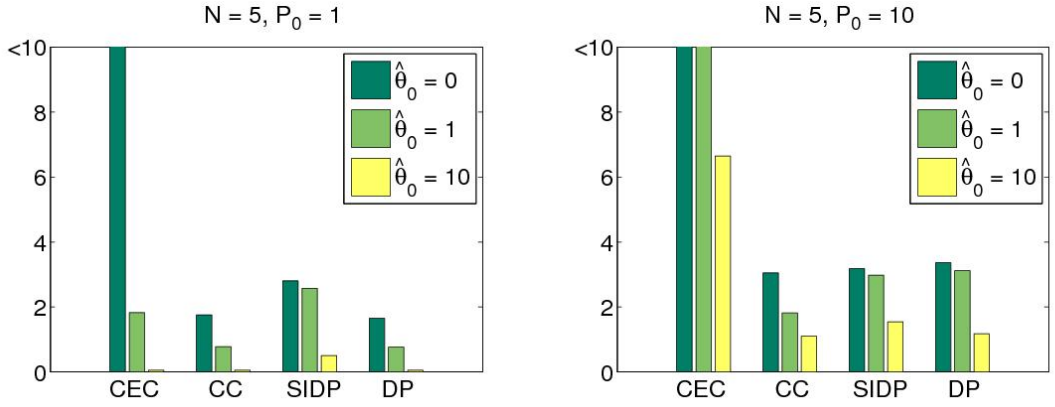
Řízení bylo testováno vzhledem k různým hodnotám počátečního odhadu θ, tedy k různým hodnotám $(\hat{\theta}_0, P_0)$. Ty byly postupně voleny z množiny $\{10; 1; 0\} \times \{10; 1; 0, 1\}$. Řídící horizont byl v prní sérii experimentů zvolen jako $N_1 = 5$ v druhé pak $N_2 = 10$. Každá simulace byla 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 řídící 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.

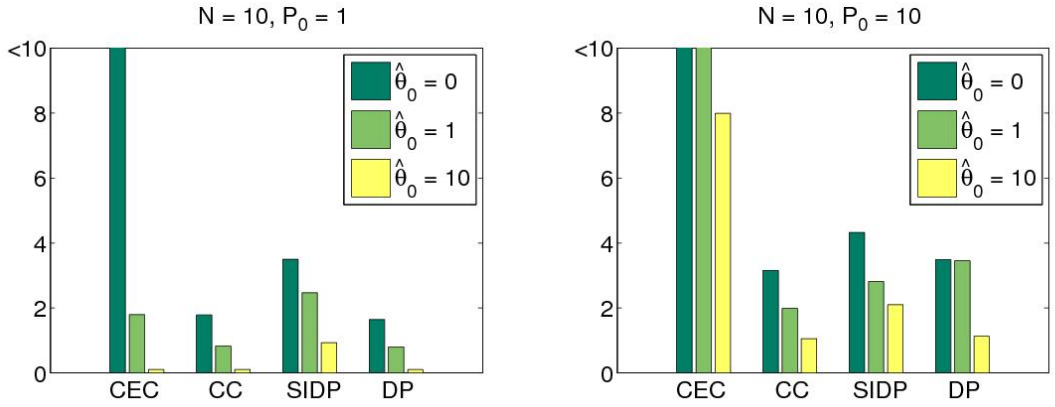
Na obrázku 3.3 jsou zachyceny výsledek pro délku řídícího horizontu $N = 5$, počáteční hodnoty royptylu $P_0 = 1$ a $P_0 = 10$ a různé hodnoty $\hat{\theta}_0$, konkrétně pro hodnoty $\{0, 1, 10\}$.

První pozorovatelný výsledek pro případ $P_0 = 1$ je, že metoda CE poskytuje v průměru použitelné výsledky pouze pokud je neurčitost v identifikaci parametru θ malá relativně k jeho velikosti. V ostatních případech je řízení silně závislé na apriorní informaci a náhodných realizacích šumu a v průměru je takové řešení nepoužitelné. Ostatní metody poskytují dobré řízení pro všechny testované kombinace $(\hat{\theta}_0, P_0)$.

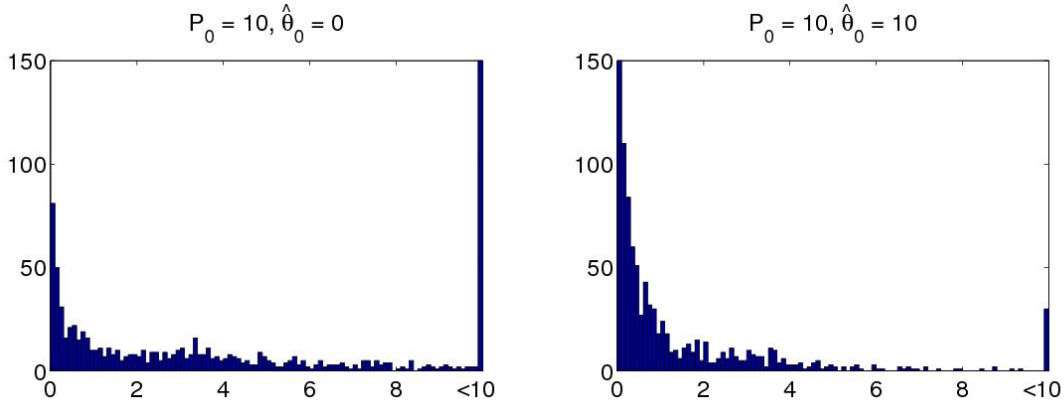
Celkově si ve všech simulacích v průměru nejlépe vedla neduální metoda CC. Ve srovnání duálních metod si vedla většinou lépe metoda DP, srovnatelné výsledky lze pozorovat v případě velké neurčitosti, tedy pro $P_0 = 10$. Pro delší horizont $N = 10$ a přesnější apriorní informaci $P_0 = 1$ dosahovala metoda SIDP horších výsledků kvůli nepřesnostem způsobeným diskretizací.



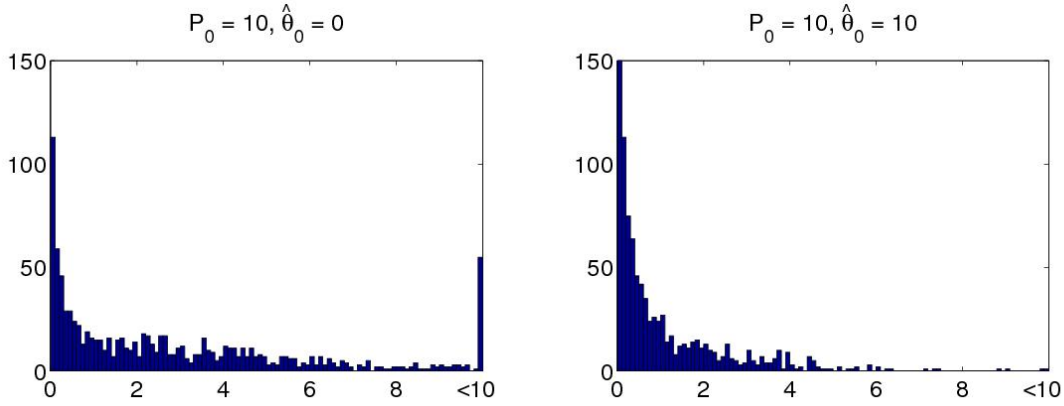
Obrázek 3.3: Výsledky simulace pro délku horizontu $N = 5$, rozptyl $P_0 = 1$ a $P_0 = 10$ a různé střední hodnoty $\hat{\theta}_0$



Obrázek 3.4: Výsledky simulace pro délku horizontu $N = 10$, rozptyl $P_0 = 1$ a $P_0 = 10$ a různé střední hodnoty $\hat{\theta}_0$



Obrázek 3.5: Výsledky jednotlivých simulací při řízení metodou certainty equivalence



Obrázek 3.6: Výsledky jednotlivých simulací při řízení metodou cautious control

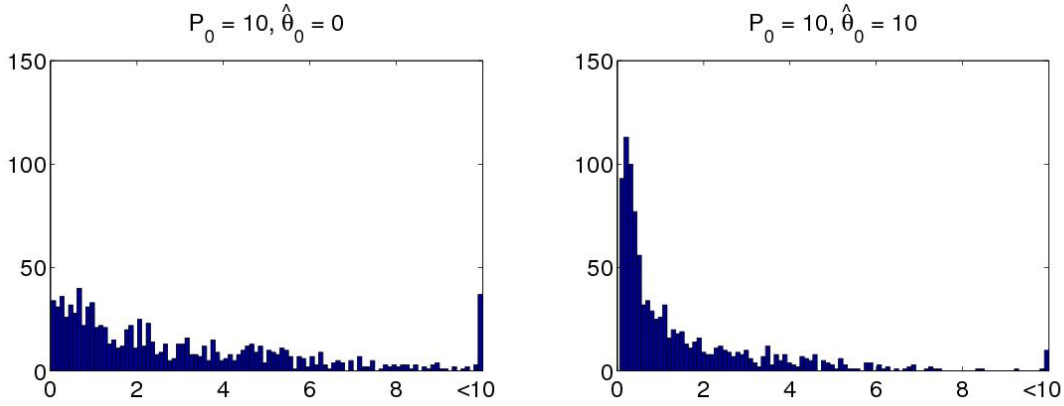
3.3.2 Kvalitativní srovnání

Nastavení simulace je stejné jako v případě kvantitativního srovnání. Horizont je ve všech případech volen $N = 5$, pro hodnotu $N = 10$ byl výsledek obdobný Hodnoty apriorní informace $(\hat{\theta}_0, P_0)$. byly $(0, 10)$, resp. $(10, 10)$.

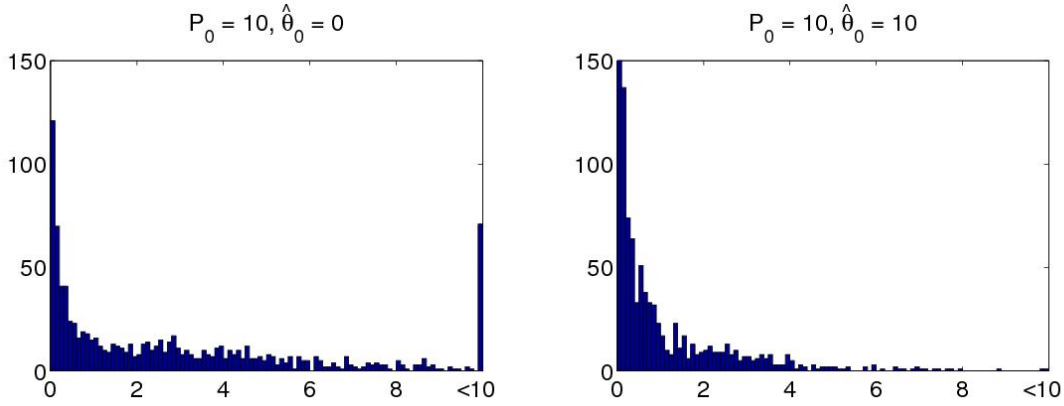
Série obrázků 3.5 zachycuje výsledky řízení metodou certainty equivalence při jednotlivých simulacích. Uvedené histogramy ukazují, že ač se v mnoha případech podařilo pomocí metody CE navrhnout řídící strategii s celkově nízkou ztrátou, ve většině případů bylo řízení neúspěšné a vedlo k vysoké ztrátě. Přesto i tato metoda v některých případech navrhla kvalitní řízení.

Na obrázcích 3.6 jsou výsledky pro metodou cautious control. Oproti metodě CE můžeme pozorovat výrazný pokles špatného řízení. Obzvláště pro hodnoty $(\hat{\theta}_0, P_0) = (0, 10)$ je pak možné si všimnout širokého spektra realizací ztráty. To je způsobeno opatrností při návrhu řídících zásahů a tedy i pomalejší identifikací parametru θ.

Výsledky pro jednotlivé simulace při řízení metodou SIDP jsou na obrázku 3.7. Charakteristickým rysem histogramů je šířka spektra dosažených ztrát. Hlavní rozdíl oproti metodě CE je, stejně jako v případě CC, znatelný pokles v počtu totálního selhání řízení. To je v příadě $\hat{\theta}_0 = 0$ nejnižší ze všech testovaných metod. Naproti



Obrázek 3.7: Výsledky jednotlivých simulací při řízení metodou SIDP



Obrázek 3.8: Výsledky jednotlivých simulací při řízení metodou metodou DP

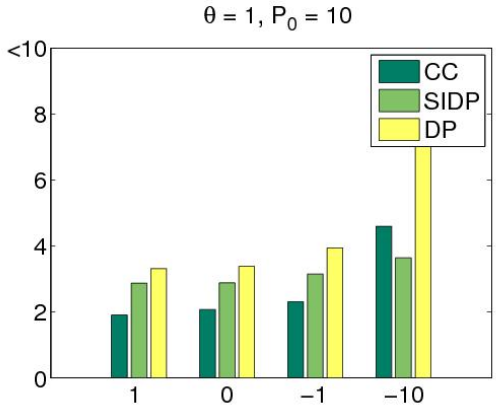
tomu je v mnoha případech dosaženo ztráty, která je vyšší než při použití řízení pomocí ostatních metod. Algoritmus SIDP tedy dává v průměru dobré výsledky díky robustnosti vůči výrazně špatnému řízení.

Histogramy pro metodu klasického numerického přístupu k řešení DP zobrazuje 3.8. 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.

3.3.3 Porovnání robustnosti

Kvalitu výsledného řízení ovlivňuje přesnost apriorní informace o neznámém parametru θ, která je pro systém (3.1) reprezentována veličinou $(\hat{\theta}_0, P_0)$. Následující simulace dokumentuje úspěšnost řídících algoritmů v případě, že apriorní informace příliš neodpovídá skutečnosti.

Pro tyto účely byl parametr θ volen jako první nenulová realizace náhodné veličiny s rozdělením $N(1, 10)$. Robustnost pak byla testována vzhledem k různé počáteční informaci o střední hodnotě $\hat{\theta}_0$ neznámého parametru θ. Rozptyl byl ve všech sim-



Obrázek 3.9: Porovnání robustnosti navržené řídící strategie vzhledem k nepřesnosti apriorní informace $\hat{\theta}_0$

ulacích položen $P_0 = 10$. Výsledky zachycuje obrázek 3.9. Každá simulace byla opakována 1000x, uveden je průměr přes jednotlivé realizace. Metoda CE není s ohledem k předchozím výsledkům do srovnání zahrnuta.

Pozorovaným výsledkem je především dobrá odolnost metody SIDP vůči špatné apriorní informaci $\hat{\theta}_0$. Výsledné řízení získané touto metodou dosahovalo dobré ztráty i v případě, kdy byla apriorní informace naprosto nepoužitelná ($\hat{\theta}_0 = -10$). Ostatní řídící algoritmy (zejména pak DP) dosáhly v totmo případě v průměru vyšší ztraty.

3.3.4 Časová náročnost SIDP

Pro neduální metody CE a CC je k dispozici analytický tvar řídící strategie, výpočet může být tedy prováděn prakticky v reálném čase. Strategie získaná algortitmem DP (převzata z [1]) má rovněž analytický tvar. Její výpočet trval (dle výsledků [11]) přibližně desetkrát déle, oproti použití SIDP za obdržení srovnatelných výsledků.

Tento oddíl ilustruje náročnost výpočtu řídící strategie pomocí algoritmu SIDP v závislosti na délce řídícího horizontu a jemnosti diskretizace. Výstupem algoritmu je tabulka vypočtených řídících zásahů v jednotlivých bodech diskretizace. Samotné řízení se provádí vyhledáváním v této struktuře. Pro konkrétní bod hyprstavu H_t se v tabulce vyhledá nejbližší napočtená hodnota a přečte se řídící zásah, který se následně použije. Jsou-li tedy předpočtené tabulky k dispozici, řízení může probíhat prakticky v reálném čase, závisí pouze na efektivitě vyhledávání v předpočtené struktuře. Při použití metody SIDP je časově náročná fáze přípravy tabulek, pomocí nichž se posléze řízení provádí. Diskutována je tedy tato část algoritmu.

V jednotlivých simulacích, které měli prověřit výpočetní nároky metody SIDP, byl parametr θ volen jako první nenulová realizace náhodné veličiny s rozdělením $N(0, 10)$. Tomu odpovídala rovněž apriorní informace $(\hat{\theta}_0, P_0) = (0, 10)$. Doba výpočtu byla zkoumána v závislosti na délce řídícího horizontu N a jemnosti diskretizace n_g (tj. počtu bodů v diskretizaci každé dimenze hyperstavu H_t). Ostatní parametry zůstaly shodné s nastavením v tabulce 3.1.

N	n_g	$\bar{J}$	t $[s]$
5	5	3,11	23
5	10	3,06	76
5	15	3,01	180
10	5	4,67	124
10	10	4,05	432
10	15	3,91	1032
15	5	5,60	294
15	10	5,12	1284
15	15	5,06	2808

Tabulka 3.2: Výpočetní čas t algoritmu SIDP v závislosti na délce řídícího horizontu N a jemnosti diskretizace n_g, $\bar{J}$ je průměrná ztráta

Obdržené výsledky zachycuje tabulka 3.2. Pro srovnání je rovněž uvedena ztráta, které takto navržené řízení dosáhlo. Jedná se opět o průměr z tisíce opakování.

Dosažené výsledky ukazují, že i pro delší řídící horizonty lze k návrhu použít metodu SIDP. Výpočetní doba však metodu zřejmě limituje pro výrazně delší časové horizonty. Jemnost diskretizace ovlivňovala průměrnou ztrátu vzhledem k ušetřenému výpočetnímu času relativně málo. Obzvláště pro krátký časový horizont ($N = 5$) byla kvalita výsledného řízení srovnatelná i pro velmi hrubou diskretizaci (stačilo pouze 5x5 bodů).

3.3.5 Shrnutí výsledků simulace

Dle provedených simulací vychází pro řízení systému (3.1) neduální metoda cautious control (CC). Dle kvantitativního srovnání dosahovala průměrně nejnižší ztráty. Kvalitativní porovnání pak prokázalo robustnost tohoto řízení.

Rovněž 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 v případě velké počáteční neznalosti ($P_0 = 10$) kvantitativně srovnatelných výsledků. Výraznější rozdíl byl 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.

Metoda certainty equivalence (CE) se ukázala použitelná pouze pokud je neurčitost v identifikaci parametru θ malá relativně k jeho velikosti. Podle kvalitativního zkoumání to bylo způsobeno tím, že v mnoha případech, ztráta výrazně překročila rozumnou mez. Přesto se občas i touto metodou podařilo navrhnout kvalitní řízení.

Analýza výpočetní náročnosti algoritmu SIDP ukázala, že pro komplexnější úlohy by připadala v úvahu pouze v případě krátkých časových horizontů a nepříliš jemné

diskretizace. Jako vhodné se tedy jeví použití metody SIDP v kombinaci s nějakou výpočetně efektivní metodou, například CE. V případě špatné identifikace systému by se použilo řízení navržené pomocí SIDP a jakmile by znalost systému překročila jistou zvolenou mez, k řízení by se použia neduální metoda. V takovém případě by totiž stačilo metodou SIDP řídit jen na krátkém časovém horizontu, konkrétně než by bylo dosaženo dostatečné znalosti systému. Dále by se již řídilo výpočetně efektivní metodou.

Závěr

Tato práce se zabývala suboptimálními přístupy k otázce řízení za neurčitosti, zejména pak metodou stochastického itarativního dynamického programování (SIDP).

V úvodu práce byla formulována úloha stochastického řízení s aditivní ztrátou, která byla řešena pomocí dynamického programování. Úloha stochastického řízení byla následně modifikována na úlohu řízení systému s neúplným pozorováním a neznámými parametry.

V další části práce byly představeny některé principy a metody, kterými lze nalézt suboptimální řídící strategii. Jednalo se zejména o návrh pomocí duálního řízení, certainty equivalence (CE), cautionus control (CC), použití metody Monte Carlo a iterativního dynamického programování.

Navržené principy byly posléze použity k návrhu řízení jednoduchého systému. Při provedených simulacích se ukázalo jako nejlepší řízení to, které poskytla neduální metoda cautious control. Bylo to způsobeno zejména robustností tohoto řízení. Duální metody pak poskytovaly srovnatelné výsledky pro velké hodnoty počáteční neznalosti.

Řídící strategie navržená metodou SIDP byla úspěšná zejména při špatné identifikaci systému. Řízení navržené touto metodou bylo velmi robustní vzhledem k špatné apriorní informaci. V případě značně odlišné apriorní informace od skutečné hodnoty parametru pak bylo řízení pomocí SIDP nejúspěšnější. Algoritmus SIDP se proto ukázal jako vhodný pro navržení řídící strategie pro systém s velkou počáteční neznalostí, kdy apriorní informace není příliš spolehlivá.

Dle simulace zaměřené na výpočetní nároky algoritmu SIDP, je použití této metody na komplexnější úlohy v uvedené podobě možné pouze na krátkých řídících horizontech a pro nepříliš jemnou diskretizaci. V ostatních situacích by bylo vhodné použít metodu SIDP v kombinaci s nějakou výpočetně efektivní metodou. Tato kombinovaná metoda by pak aplikovala řídící zásahy generované pomocí SIDP pouze v případě, že by neznalost v identifikaci systému přesáhla jistou zvolenou mez. Tak by se využilo robustnosti metody SIDP a zároveň minimalizovaly její výpočetní nároky, protože když by se systém opět dobře zidentifikoval, řízení by pokračovalo výpočetně efektivní metodou.

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.