České vysoké učení technické v Praze

Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky

Obor: Inženýrská informatika

Zaměření: Softwarové inženýrství

Iterativní lokální dynamické programování pro návrh duálního řízení

Iterative local dynamic programming for dual control

BAKALÁŘSKÁ PRÁCE

Vypracoval: Michal Vahala

Vedoucí práce: Ing. Václav Šmídl, Ph.D.

Rok: 2010

zadání práce

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

V Praze dne ....

Michal Vahala

Poděkování

Děkuji ... za ...

Michal Vahala

Název práce:
Iterativní lokální dynamické programování pro návrh duálního řízení

Autor: Michal Vahala

Obor: Inženýrská informatika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Václav Šmídl, Ph.D.

Konzultant: —

Abstrakt: abstrakt

Klíčová slova: klíčová slova

Title:
Iterative local dynamic programming for dual control

Author: Michal Vahala

Abstract: abstrakt

Key words: klíčová slova

Obsah

Seznam použitého označení

iLDP	iterativní lokální dynamické programování
LQG	lineáně kvadraticky gaussovské řízení (Linear-Quadratic-Gaussian)
iLQG	iterativní LQG
DDP	diferenciální dynamické programování

Úvod

Skutečný svět se nikdy nechová přesně podle matematických rovnic, protože ty jsou vždy jen jakýmsi zjednodušením nebo přiblížením. V reálném světě se vyskytuje mnoho neznámých veličin, poruch, nepředvídatelných vlivů a ani naše měřící přístroje nejsou přesné. Chceme-li efektivně řídit nějaký systém, musíme si být těchto vlivů vědomi a zahrnout je do našich uvažování. Situace se však ještě může zkomplikovat, když jeden nebo více parametrů neznáme. To může nastat z různých důvodů, například příšlušné čidlo nebo měřící přístroj nemůžeme nebo nechceme (například z důvodů vysoké ceny) instalovat a tedy o velikosti příslušné hodnoty můžeme jen usuzovat ze známých dat. Ještě složitější situace nastane, když uvažujeme neznámý parametr proměnný.

Máme tedy dva cíle, musíme systém co nejlépe řídit a současně se snažit o co nejpřesnější určení neznámých parametrů. Tyto dva postupy jsou však obecně v rozporu, protože parametry se nejlépe určují, když je systém vybuzen a nechová se optimálně. Právě tento rozpor a nalezení kompromisu, který povede k jeho řešení, je podstatou duálního řízení.

Pro přiblížení ilustrujme problém na jednoduchém příkladě: Uvažujme elektromotor s možností řídit napětí na vstupu motoru a měřit příslušné proudy. Jedná se tedy o systém se dvěma vstupy a dvěma výstupy. Cílem našeho řízení je dosažení požadovaných otáček rotoru. Ovšem otáčky a ani polohu hřídele měřit nemůžeme. Máme o nich však znalost v podobě středních hodnot a variancí. Naší snahou je co nejpřesněji určit hodnotu otáček a polohy hřídele a současně systém řídit tak, abychom dosáhly požadované hodnoty otáček. Tyto dvě snahy jsou ale v rozporu, protože nejvíce informací o neznámých parametrech získáme, když je motor vybuzen. Tedy například se prudce rozjíždí, brzdí, rychle mění rychlost nebo kmitá, což se projevuje v proudech, které máme možnost měřit. Ale právě vybuzení motoru je v rozporu se snahou o dobré řízení, protože chyba, které se dopustíme je většinou nepřijatelná. Naopak, když se systém snažíme řídit, bez dostatečné znalosti jeho parametrů, s velkou pravděpodobností selžeme.

Námětem této bakalářské práce je algoritmus *iterativního lokálního dynamického programování* (iLDP) jako jedna z metod pro řešení problému duálního řízení. Algoritmus byl navržen a popsán v článku [7]. Jak už prozrazuje název algoritmu, jedná se o iterační metodu. Tedy stručně řečeno, algoritmus vyjde od nějakého počátečního řízení, které je ovšem nutno dodat jako apriorní informaci a v cyklech (iteracích) tuto řídící strategii vylepšuje, za účelem získání řízení optimálního. Dále se jedná o metodu lokální, což v můžeme jednoduše chápat tak, že kandidáti na „vylepšení“ řízení jsou vybírání z jistého, zatím blíže nespecifikovaného, okolí původní řídící strategie. Nakonec algoritmus využívá obecné schéma dynamického programování, které bude blíže popsáno v dalším textu.

Cílem této práce bylo seznámit se s obecnou tématikou duálního řízení a detailněji s konkrétním algoritmem - iterativním lokálním dynamickým programováním. Následně

tento algoritmus implementovat a aplikovat na jednoduchý systém. Otestovat jeho funkčnost a schopnost řídit a to i v porovnání s jinými metodami a algoritmy. *Dále implementovat algoritmus iLDP pro složitější systém blíže praktické aplikaci, konkrétně se jedná o synchronní motor s permanentními magnety. Opět otestovat funkčnost a případně srovnat s dostupnými výsledky jiných řídících strategii.* Na základě získaných výsledků posoudit výhody a nevýhody algoritmu a jeho použitelnost na další úlohy.

Hlavním přínosem práce je otestování vlastností algoritmu iLDP na jiných problémech, než pro které byla vyvinuta autory. Objevení kladů a záporů algoritmu a dále díky srovnání s jinými algoritmy získání přehledu, pro které praktické aplikace je vhodnější respektive méně vhodný než srovnávané metody. Prvotní očekávání pro srovnání algoritmu iLDP a *principu separace* jsou, že iLDP bude rychlejší co do výpočetního času, avšak přesnost získaných výsledků bude nižší. Dále je očekávána nezanedbatelná závislost výsledného řízení na volbě použitých aproximací a apriorní řídící strategie.

1 Teorie duálního řízení

1.1 Základní pojmy

1.1.1 Systém a řízení

Systém

Základním pojmem, se kterým budeme v textu pracovat je *systém*. Obdobně jako základní pojmy zejména v matematických vědách (bod, množina, algoritmus,...), nelze tento pojem exaktně definovat. Systém si můžeme představit jako jistý „objekt“, často bude reprezentovat objekt skutečného světa. Hlavní vlastností systému je, že má zpravidla jeden nebo více vstupů, pomocí kterých mu můžeme předávat informaci – řízení a jeden nebo více výstupu, což jsou hodnyty, které pozorujeme. Co se odehrává uvnitř systému však obecně nevíme. Řízení, které budeme dodávat systému na vstup bude v textu značeno písmenem u. Analogicky bude písmenem y označena pozorovaná hodnota na výstupu.

Chování systému, to je jakým výstupem reaguje na vstup, popisujeme dle [3] obecně diferenciální rovnicí respektive soustavou diferenciálních rovnic vyšších řádů. Jedná se o takzvaný vnější popis. Tento druh popisu, pohlíží na systém „zvenku“ bez skutečné znalosti, co se odehrává uvnitř systému a jaká je jeho podstata. Vnější popis obvykle obdržíme při odvození modelu systému z fyzikálních rovnic. Omezení, která z něj plynou, se snažíme odstranit zavedením vnitřního (stavového) popisu, kdy (soustavu) diferenciálních rovnic vyššího řádu, převedeme vhodnou volbou nových proměnných x na soustavu diferenciálních rovnic prvního řádu. Proměnné x označujeme jako stavové proměnné.

Řízení

Naším úkolem je pro zadaný systém nalézt regulátor, tedy obecně řízení u takové, které dodané na vstup způsobí, že systém se bude „chovat podle našich požadavků“. To zpravidla znamená, že hodnoty výstupní veličiny y dosáhnou (nebo se přiblíží s danou přesností) požadované hodnotě v podobě referenčního signálu, který regulátor dostává z vnějšku a současně dodrží předem stanovená omezení. Práce je ovšem zaměřena na řízení složitějších systémů, u kterých jeden nebo více parametrů neznáme přesně. Tedy některý (více) z koeficientů v rovnicích popisujících systém není znám. Máme však o něm jistou statistickou informaci v podobě jeho očekávané hodnoty a variance. Dále je-li systém nelineární, jsou výsledné rovnice příliš složité a tedy analyticky neřešitelné. Pro numerické řešení, jsou rovnice systému zpravidla převáděny do diskrétního tvaru.

Řízení obecně dělíme podle [3] na dva typy: *Přímovazební řízení* užíváme v případě, kde je k dispozici přesný matematický model systému a je vyloučen výskyt neurčitostí.

Toto řízení nevyužívá žádné zpětné informace od systému a regulátor pracuje pouze s referenčním signálem. Naproti tomu *zpětnovazební řízení* využívá i informace o skutečném výstupu systému a snaží se tak eliminovat chyby v důsledku neurčitostí a chyb způsobenych nepřesností modelu.

Duální řízení

Chceme navrhnout regulátor pro zadaný systém s neznámými parametry. Úkoly jsou tedy dva: 1. *opatrnost* - efektivně systém řídit a 2. *testování* - určit neznáme parametry. Tyto dva přístupy jsou ale obecně v rozporu. Abychom mohli systém dobře řídit, potřebujeme znát parametry co nejpřesněji. Nejvíce informací o parametrech však získáme, když je systém vybuzen a nechová optimálně. Tyto pojmy není snadné kvantifikovat, ale velmi často se projevují v konkrétních řídících schématech. Naším úkolem je pokusit nalézt nějaký kompromis mezi oběma úkoly. Právě tento přístup je označován jako *duální řízení* [2].

1.1.2 Formulace problému

V textu budeme pracovat zpravidla s diskrétním systémem, ve smyslu systému s diskrétním časem, protože výpočty jsou prováděny ve většině případů problematiky duálního řízení numericky. Rovnice popisující systém jsou však zpravidla ve spojitém tvaru, (model často vychází ze skutečnosti, popřípadě fyzikálních zákonů). V tomto případě provádíme diskretizaci.

Základní problém je formulován podle [2] následovně:

Uvažujme stavový popis diskrétního dynamického systému

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, \ldots, N-1 \ , \tag{1.1}$$

kde x_k je stavová proměná z prostoru S_k, u_k řízení z prostoru C_k a w_k náhodná porucha z prostoru D_k, vše v čase k při celkovém časovém horizontu N. Na řízení u_k klademe omezení, že může nabývat pouze hodnot z neprázdné podmonožiny $U_k(x_k) \subset C_k$ závislé na stavu x_k. Náhodná porucha w_k je charakterizována rozdělením pravděpodobnosti P_k, které může explicitně záviset na x_k a u_k, ne však na předchozích poruchách $w_{k-1}, \ldots, w_0$.

Dále uvažujme množinu řízení, jedná se o posloupnost funkcí

$$\pi = \{\mu_0, \ldots, \mu_{N-1}\},$$

kde μ_k přiřazuje stavu x_k přípustné řízení $u_k = \mu_k(x_k)$, to je takové, že $\mu_k(x_k) \in U_k(x_k)$, množinu přípustných řešení označme Π. Máme-li dány počáteční stav x_0 a přípustné řešení π můžeme stavy x_k a poruchy w_k považovat za náhodné veličiny s rozdělemím definovaným systémem rovnic 1.1, kde za řízení u_k dosadíme hodnotu funkce μ_k v x_k.

Pro dané ztráty v jednotlivých časech – funkce g_k, pak definujeme očekávanou ztrátu π v x_0 jako

$$J_\pi(x_0) = \mathbf{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\left(x_k, \mu_k(x_k), w_k\right) \right\}$$

kde je očekávaná hodnota počítána přes náhodné veličiny w_k a x_k. Optimální řízení π^*je právě to, které minimalizuje ztrátu

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

Optimální ztrátu označme $J^*(x_0)$.

1.1.3 Dynamické programování

Dynamické programovní dle [9] je jedním ze způsobů návrhu algoritmů pro řešení jistých typu optimalizačních problémů. Konkrétně se uplatňuje v případě, že jde o diskrétní optimalizační úlohu, na řešení daného problému můžeme nahlížet jako na konečnou posloupnost rozhodnutí a platí *princip optimality*.

Princip optimality

říká, že optimální posloupnost rozhodnutí musí mít následující vlastnost: *Jestliže jsme už udělali* k *rozhodnutí, musí být všechna následující rozhodnutí optimální vzhledem k výsledkům rozhodnutí předchozích, jinak nemůžeme dostat optimální řešení* [9].

Princip optimality v teorii řízení

Nechť $\pi^* = \left\{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\right\}$ je optimální řídící strategie pro základní problém a předpokládejme, že když aplikujeme řízení π^*, daný stav x_i se vyskytne v čase i s pozitivní pravděpodobností. Uvažujme podproblém, kdy ve stavu x_i a čase i chceme minimalizovat *náklady na pokračování* (v anglické literatuře označováno jako „cost-to-go“) od času i do N

$$\mathbf{E}\left\{g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k)\right\}$$

Potom úsek strategie $\left\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\right\}$ je optimální pro tento podproblém.

Intuitivně je princip optimality velmi jednoduchý. Jestliže úsek strategie $\left\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\right\}$ nebude optimální, budeme schopni dále zredukovat cenu přechodem k optimální strategii pro podproblém.

Princip optimality umožňuje optimální strategii konstruovat postupně. Nejdříve nalezneme optimální strategii pro koncový podproblém zahrnující poslední krok. Poté rozšiřujeme podproblém od konce přidáním předposledního kroku a tak dále. Takto může být vytvořena optimální strategie pro celý problém.

Algoritmus dynamického programování je tedy založen na následující myšlence: Algoritmus pracuje iterativně a řeší koncové podproblémy pro daný časový úsek, při tom využívá řešení předchozích koncových podproblémů pro kratší časové úseky. Převzato z [2].

Formulace algoritmu dynamického programování

Podle [2], pro každý počáteční stav x_0, je optimální cena $J^*(x_0)$ základního problému rovna $J_0(x_0)$, získané z posledního kroku následujícího algoritmu, který prochází zpět časy od $N-1$ do 0:

$$J_N(x_N) = g_N(x_N)$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k) w_k} \mathbf{E}\left\{g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\right\} \tag{1.2}$$

$$k = 0, 1, \ldots, N-1$$

kde je očekávaná hodnota počítána podle náhodné veličiny w_k, která obecně závisí na x_k a u_k. Dále, když $u_k^* = \mu_k^*(x_k)$ minimalizuje pravou stranu rovnice (1.2) pro každé x_k a k, stretegie $\pi* = \left\{\mu_1^*, \ldots, \mu_{N-1}^*\right\}$ je optimální.

Hodnotu $J_k(x_k)$ je možno interpretovat jako optimální cenu pro $(N-k)$-tý krok problému začínajícího ve stavu x_k a čase k, a končícího v čase N. Následně označujeme $J_k(x_k)$ náklady na pokračování („cost-to-go“) ve stavu x_k a čase k, a J_k označujeme jako funkci nákladů na pokračování („cost-to-go function“) v čase k.

Ideálně bychom chtěli využít algoritmus dynamického programování k získání J_k vyjádřené v uzavřeném tvaru nebo k získání optimální strategie. Existuje mnoho případů, kdy je daná úloha řešitelná analyticky, obzvláště za zjednodušujících předpokladů. To je velmi užitečné zejména pro lepší náhled do problematiky a jako vodítko pro složitější modely. Avšak ve většině případů není analytické řešení možné, pak je třeba použít numerické řešení pomocí algoritmu dynamického programování. Tento přístup může být časově velmi náročný, zejména minimalizaci v rovnici (1.2) je třeba provést pro každou hodnotu x_k. Stavový prostor musí být diskretizován, nejedná-li se o konečnou množinu a výpočetní nároky pak narůstají proporcionálně k počtu možných hodnot x_k. Nicméně dynamické programování je pouze obecný přístup pro iterativní optimalizaci při uvažování nejistoty v systému.

1.1.4 Úplná a neúplná stavová informace

V optimálním případě by bylo možno měřit všechny stavové veličiny systému a na jejich základě libovolným způsobem upravovat jeho dynamické vlastnosti. Ve skutečnosti ale zpravidla není možné všechny stavy změřit a musíme se rozhodovat pouze na základě informací, které máme k dispozici, pak mluvíme o *neúplné informaci o stavu systému* [5, 2]. Může to být způsobeno například nedostupností hodnot některých stavů, použité měřící přístroje mohou být nepřesné nebo náklady na získání přesné hodnoty stavu mohou být příliš omezující. Případy tohoto typu modelujeme zpravidla tak, že v každém kroku regulátor obdrží jisté pozorování skutečné hodnoty stavu, které ovšem může být ovlivněno a narušeno stochastickou nejistotou. Teoreticky se však problém s neúplnou informací o stavu neodlišuje od úloh s úplnou stavovou informací, protože existují způsoby, jak převést (redukovat) systém s neúplnou informací na systém s úplnou. Tyto postupy

obecně vedou na algoritmy využívající dynamické programování, ale jsou výpočetně mnohem náročnější, než v případě úplné informace. Dva možné postupy redukce převzaté z [2] budou následovat po formulaci problému:

Formulace problému s neúplnou informací o stavu

Nejdříve formulujme základní problém s neúplnou stavovou informací, který následně redukujeme na systém s informací úplnou. Uvažujme rozšíření základního problému 1.1, kde ale regulátor, namísto přístupu ke stavu systému, získává pouze pozorování z_k ve tvaru

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k = 1, 2, \ldots, N-1, \tag{1.3}$$

kde v_k reprezentuje náhodnou poruchu pozorování charakterizovanou rozdělením pravděpodobnosti P_{v_k}, která závisí na současném stavu a všech předchozích stavech, řízeních a poruchách. Dále také počáteční stav x_0 považujeme za náhodnou veličinu s rozdělením P_{x_0}.

Soubor informací dostupných regulátoru v čase k označme I_k informačním vektorem. Tedy

$$\begin{aligned} I_k &= (z_0, \ldots, z_k, u_0, \ldots, u_{k-1}), \quad k = 1, \ldots, N-1, \\ I_0 &= z_0. \end{aligned}$$

Uvažujme množinu přípustných řízení jako posloupnost funkcí $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, kde každá funkce μ_k přiřazuje informačnímu vektoru I_k řízení $\mu_k(I_k) \in U_k$, pro všechna I_k, kde $k = 0, \ldots, N-1$. Chceme najít přípustnou řídící strategii, to jest posloupnost π, která minimalizuje očekávanou ztrátu

$$J_\pi = \mathbf{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\left(x_k, \mu_k(I_k), w_k\right) \right\},$$

kde je očekávaná hodnota počítána přes náhodné veličiny x_0 a w_k, v_k pro $k = 0, \ldots, N-1$. Veličiny x_k a z_k se vypočítají z rovnic 1.1 respektive 1.3, přičemž v nich položíme $u_k = \mu_k(I_k)$.

Redukce na systém s úplnou stavovou informací

Tento postup je založen na myšlence definovat nový systém, jehož stav v čase k je množina všech hodnot, kterých může využít regulátor při tvorbě řízení. Jako stav nového systému tedy volíme informační vektor I_k a získáme systém

$$I_{k+1} = (I_k, z_{k+1}, u_k), \quad I_0 = z_0, \quad k = 0, \ldots, N-2. \tag{1.4}$$

Na tento systém povahy základního problému s úplnou informací můžeme pohlížet tak, že I_k je stav. Řízení u_k a pozorování z_k lze pak chápat jako náhodné poruchy. Dále rozdělení pravděpodobnosti z_{k+1} závisí explicitně pouze na stavu I_k a řízení u_k. Ztrátovou funkci vyjádřenou pro nový systém je možno zapsat jako

$$\mathbf{E}\left\{ g_k\left(x_k, u_k, w_k\right) \right\} = \mathbf{E}\left\{ \mathbf{E}_{x_k, w_k}\left\{ g_k\left(x_k, u_k, w_k\right) \mid I_k, u_k \right\} \right\}.$$

Tedy ztráta během jednoho kroku vyjádřená jako funkce nového stavu I_k a řízení u_k je

$$\tilde{g}_k(I_k,u_k) = \mathbf{E}_{x_k,w_k}\left\{g_k\left(x_k,u_k,w_k\right) \mid I_k,u_k\right\}. \tag{1.5}$$

Původní základní problém s neúplnou stavovou informací byl tedy převeden na úlohu s úplnou stavovou informací s rovnicí popisující systém 1.4 a ztrátou během jednoho kroku 1.5. Nyní je pro něj možno napsat algoritmus dynamického programování. (možná sem dát i rovnice DP)

Postačující statistika

Při užití algoritmu dynamického programování za neúplné stavové informace je hlavní problém v jeho vyhodnocování ve stavovém prostoru, jehož dimenze neustále roste. S každým dalším měřením dimenze stavu a tedy informační vektor I_k narůstá, proto se snažíme redukovat množství dat skutečně potřebných pro účely řízení. Hledáme tedy popis známý jako *postačující statistika*, který bude mít menší dimenzi než I_k ale současně zahrne veškerý důležitý obsah I_k potřebný pro řízení. Jako postačující statistiku označme funkci S_k informačního vektoru I_k, tedy $S_k(I_k)$ takovou, že minimalizuje ztrátu v algoritmu dynamického programování přes všechna přípustná řízení. Což můžeme zapsat pro vhodnou funkci H_k jako

$$J_k(I_k) \quad = \quad \min_{u_k \in U_k} H_k(S_k(I_k),u_k).$$

Po funkci S_k samozřejmě chceme, aby byla charakterizována menší množinou čísel, než informační vektor I_k, abychom získaly výhody z jejího použití. Obecně existuje mnoho funkcí, které mohou sloužit jako postačující statistika. Triviálním příkladem může být identita $S_k(I_k) = I_k$.

Závisí-li rozdělení pravděpodobnosti poruchy pozorování v_k explicitně pouze na bezprostředně předcházejícím stavu, řízení a poruše systému, tedy na x_k, u_k, w_k a nezávisí na předchozích hodnotách $x_{k-1},\ldots,x_0,u_{k-1},\ldots,u_0,w_{k-1},\ldots,w_0,v_{k-1},\ldots,v_0$ můžeme za postačující statistiku volit podmíněné rozdělení pravděpodobnosti $P_{x_k|I_k}$, o kterém lze ukázat (viz [2]), že

$$J_k(I_k) = \min_{u_k \in U_k} H_k(P_{x_k|I_k},u_k) = \overline{J}_k(P_{x_k|I_k}),$$

kde H_k a $\overline{J}_k$ jsou vhodné funkce. Optimální řízení pak získáme ve tvaru funkcí podmíněného rozdělení pravděpodobnosti $\mu_k(I_k) = \overline{\mu}_k(P_{x_k|I_k})$ pro $k = 0,\ldots,N-1$. Tato reprezentace může být velmi užitečná, protože nám umožňuje rozložit optimální řízení na dvě nezávislé časti:

1. *pozorovatel* (estimátor), který v čase k použije měření z_k a řízení u_{k-1} k vygenerování rozdělení pravděpodobnosti $P_{x_k|I_k}$
2. *ovladač* (regulátor), který generuje vstupy (řízení) pro systém jako funkci rozdělení pravděpodobnosti $P_{x_k|I_k}$

Tento rozklad pak umožňuje navrhovat každou z částí samostatně podle charakteru konkrétní úlohy.

1.1.5 Kalmanův filtr

Chceme řešit následující problém, viz [5]: Máme lineární systém s neúplnou stavovou informací a snažíme se odhadnout (rekonstruovat, estimovat) stav systému z měřitelných vstupních a výstupních veličin. Dále předpokládejme, že měření výstupu a popřípadě i vstupu je zatíženo chybou měření. Tyto nepřesnosti měření můžeme modelovat jako aditivní šum. Odhadování (rekonstrukci, estimaci) potom navrhujeme pomocí stochastických metod. Řešení vede na takzvaný *Kalmanův filtr*.

Následující formulace problému a popis algoritmu Kalmanova filtru je převzat z [2], kde lze také nalézt odvození příslušných rovnic: Máme dva náhodné vektory x a y, které jsou svázány společným rozdělením pravděpodobnosti („joint probability distribution“) tak, že hodnota jednoho poskytuje informaci o hodnotě druhého. Známe hodnotu y a chceme určit (odhadnout) hodnotu x tak, aby střední kvadratická odchylka mezi x a jeho odhadem byla minimální.

Takový odhad můžeme zístat v nejjednodušším případě metodou nejmenších čtverců, ale pro tento způsob je třeba velkého počtu měření. Jako lepší způsob se ale jeví využít sekvenční struktury problému a iterativně použít Kalmanův filtr, kdy odhad v čase $k+1$ získáme na základě jednoduchých rovnic pouze z předchozího odhadu a nového měření v čase k, žádná předchozí měření nejsou explicitně zahrnuta.

V dalším textu označme $\hat{x}_{k|k-1}$ apriorní odhad stavu, tedy odhad stavu v čase k na základě informací až do času $k-1$. Analogicky $\Sigma_{k|k-1}$ označuje apriorní kovarianční matici. Aposteriorní odhad stavu označme $\hat{x}_{k|k}$, to jest odhad v čase k na základě informačí až do času k. Aposteriorní kovarianční matice je pak označena $\Sigma_{k|k}$.

System

Uvažujme lineární dynamický systém bez řízení ($u_k \equiv 0$) ve tvaru

$$x_{k+1} = A_k x_k + w_k,\ k = 0, 1, \ldots, N-1,$$

kde x_k je vektor stavu, w_k vektor náhodné poruchy a matice A_k předpokládáme známé. Dále rovnice měření je

$$z_k = C_k x_k + v_k,\ k = 0, 1, \ldots, N-1,$$

kde z_k je vektor pozorování (měřených veličin) a v_k vektor šumu. Nechť $x_0, w_0, \ldots, w_{N-1}, v_0, \ldots, v_{N-1}$ jsou vektory nezávislých náhodných veličin s daným rozdělením pravděpodobnosti, takovým, že

$$\mathrm{E}\{w_k\} = \mathrm{E}\{v_k\} = 0,\ k = 0, 1, \ldots, N-1.$$

Označme

$$S = \mathrm{E}\left\{(x_0 - \mathrm{E}\{x_0\})(x_0 - \mathrm{E}\{x_0\})^T\right\},\ M_k = \mathrm{E}\{w_k w_k^T\},\ N_k = \mathrm{E}\{v_k v_k^T\},$$

a nechť matice N_k pozitivně definitní pro všechny časy k.

Algoritmus Kalmanova filtru

Předpokládejme, že máme spočítaný odhad $\hat{x}_{k|k-1}$ společně s kovarianční maticí $\Sigma_{k|k-1} = \mathrm{E}\left\{\left(x_k - \hat{x}_{k|k-1}\right)\left(x_k - \hat{x}_{k|k-1}\right)^T\right\}$. V čase k získáme další měření $z_k = C_k x_k + v_k$. Nyní můžeme získat aposteriorní odhad stavu $\hat{x}_{k|k}$ v čase k jako

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \Sigma_{k|k-1} C_k^T \left(C_k \Sigma_{k|k-1} C_k^T + N_k\right)^{-1} \left(z_k - C_k \hat{x}_{k|k-1}\right), \tag{1.6}$$

dále pak apriorní odhad stavu $\hat{x}_{k+1|k}$ v čase $k+1$, tedy $\hat{x}_{k+1|k} = A_k \hat{x}_{k|k}$. Apriorní kovarianční matici v čase $k+1$ vypočítáme z

$$\Sigma_{k+1|k} = A_k \Sigma_{k|k} A_k^T + M_k,$$

kde aposteriorní kovarianční matici $\Sigma_{k|k} = \mathrm{E}\left\{\left(x_k - \hat{x}_{k|k}\right)\left(x_k - \hat{x}_{k|k}\right)^T\right\}$ můžeme získat z rovnice

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \Sigma_{k|k-1} C_k^T \left(C_k \Sigma_{k|k-1} C_k^T + N_k\right)^{-1} C_k \Sigma_{k|k-1}.$$

Přidáním počátečních podmínek $\hat{x}_{0|-1} = \mathrm{E}\{x_0\}$ a $\Sigma_{0|-1} = S$ získáme *algoritmus Kalmanova filtru*, který ve své podstatě rekurzivně generuje posloupnost lineárních odhadů založených na metodě nejmenších čtverců.

Dále je možno vyjádřit rovnici 1.6 ve tvaru

$$\hat{x}_{k|k} = A_{k-1} \hat{x}_{k-1|k-1} + \Sigma_{k|k} C_k^T N_k^{-1} \left(z_k - C_k A_{k-1} \hat{x}_{k-1|k-1}\right),$$

který při uvažování systému se vstupem

$$x_{k+1} = A_k x_k + B_k u_k + w_k,\ k = 0, 1, \ldots, N-1,$$

umožňuje vypočítat rekurzivně aposteriorní odhady stavů $\hat{x}_{k|k}$ v časech k z rovnice

$$\hat{x}_{k|k} = A_{k-1} \hat{x}_{k-1|k-1} + B_{k-1} u_{k-1} + \Sigma_{k|k} C_k^T N_k^{-1} \left(z_k - C_k A_{k-1} \hat{x}_{k-1|k-1}\right),$$

přičemž rovnice pro výpočet aposteriorní kovarianční matice $\Sigma_{k|k}$ zůstávají nezměněny.

1.1.6 Deterministické systémy se spojitým časem

I když zpravidla pracujeme s diskrétními systémy, zejména z důvodů výpočtů na počítači, teorie optimálního řízení spojitých systémů může být velmi užitečná. Poskytuje totiž důležité principy, které jsou velmi často používány při návrhu algoritmů pro duální řízení. Konkrétně se jedná o Hamilton-Jacobi-Bellmanovu rovnost a Pontryaginův princip minima.

Spojitý systém

Dynamický systém se spojitým časem uvažujeme dle [2] ve tvaru

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)),\ 0 \leq t \leq T, \\ x(0) &= x_0, \end{aligned} \tag{1.7}$$

kde $x(t)$ je stavový vektor v čase t, $\dot{x}(t)$ je vektor prvních derivací podle času v čase t, $u(t) \in U$ je řídící vektor v čase t, U je množina omezení řízení a T je časový horizont. O funkci f předpokládáme, že je spojitě diferencovatelná vzhledem k x a spojitá vzhledem k u. Rovnice 1.7 představuje soustavu n diferenciálních rovnic prvního řádu. Naším cílem je nalézení přípustné řídící trajektorie $\{u(t) \mid t \in [0,T]\}$ a odpovídající stavové trajektorie $\{x(t) \mid t \in [0,T]\}$ takové, že minimalizují ztrátovou funkci ve tvaru

$$h(x(T)) + \int_0^T g\left(x(t), u(t)\right) dt,$$

o funkcích g a h předpokládáme, že jsou spojitě diferencovatelné vzhledem k x a g je spojitá vzhledem k u.

Hamilton-Jacobi-Bellmanova rovnost

Hamilton-Jacobi-Bellmanova rovnost je parciální diferenciální rovnicí, která je splněna optimální funkcí nákladů na pokračování $J^*(t,x)$. Tato rovnice je analogií algoritmu dynamického programování ve spojitém čase. Rovnici lze psát podle [2] ve tvaru

$$\begin{aligned} 0 &= \min_{u \in U} \left[g(x,u) + \nabla_t J^*(t,x) + \nabla_x J^*(t,x)^T f(x,u) \right], \quad \forall t, x, \\ J^*(T,x) &= h(x). \end{aligned} \tag{1.8}$$

Jedná se tedy o parciální diferenciální rovnici s okrajovou podmínkou. O funkci $J^*(t,x)$ jsme předpokládali diferencovatelnost, apriorně ale její diferencovatelnost neznáme a tedy nevíme, jestli $J^*(t,x)$ řeší rovnici 1.8. Můžeme však použít následující tvrzení, jehož formulaci i důkaz lze nalézt v [2]:

Věta o dostatečnosti:

Nechť je funkce $V(t,x)$ spojitě diferencovatelná vzhledem k t a x a nechť je řešením Hamilton-Jacobi-Bellmanovy rovnosti:

$$\begin{aligned} 0 &= \min_{u \in U} \left[g(x,u) + \nabla_t V(t,x) + \nabla_x V(t,x)^T f(x,u) \right], \quad \forall t, x, \\ V(T,x) &= h(x), \quad \forall x. \end{aligned} \tag{1.9}$$

Předpokládejme dále, že $\mu^*(t,x)$ dosáhne minima v rovnosti 1.9 pro všechna t a x. Nechť $\{x^*(t) \mid t \in [0,T]\}$ označuje stavovou trajektorii získanou při dané počáteční podmínce $x^*(0) = x_0$ a řídící trajektorii $u^*(t) = \mu^*(t, x^*(t)),\ t \in [0,T]$. Pak V je rovno optimální funkci nákladů na pokračování, tedy

$$V(t,x) = J^*(t,x), \quad \forall t, x.$$

Navíc řídící trajektorie $\{u^*(t) \mid t \in [0,T]\}$ je optimální.

Pontryaginův princip minima

Pontryaginův princip minima je důležitým teorémem optimálního řízení. Poskytuje nutnou (ne však postačující) podmínku pro optimální trajektorii, je úzce spřízněn s Hamilton-Jacobi-Bellmanovou rovností a lze ho z ní podle [2] také odvodit. Princip minima je výhodné formulovat pomocí Hamiltoniánu. Označme p gradient optimální funkce nákladů na pokračování pro optimální stavovou trajektorii $p(t) = \nabla_x J^* (t, x^*(t))$ a definujme Hamiltonián jako funkci zobrazující trojice vektorů (x, u, p) do reálných čísel

$$H(x, u, p) = g(x, u) + p^T f(x, u).$$

Rovnice pro systém pak může být zapsána v kompaktním tvaru

$$\dot{x}^*(t) = \nabla_p H \left(x^*(t), u^*(t), p(t)\right).$$

Obdobně může být zapsána pro p takzvaná *adjungovaná rovnice*

$$\dot{p}(t) = -\nabla_x H \left(x^*(t), u^*(t), p(t)\right).$$

Pontryaginův princip minima je podle [2] formulován následovně:

Princip minima:

Nechť $\{u^*(t) \mid t \in [0, T]\}$ je optimální řídící trajektorie a nechť $\{x^*(t) \mid t \in [0, T]\}$ je odpovídající stavová trajektorie, to jest

$$\dot{x}^*(t) = f \left(x^*(t), u^*(t)\right), \quad x^*(0) = x_0.$$

Nechť dále $p(t)$ je řešením adjungované rovnice

$$\dot{p}(t) = -\nabla_x H \left(x^*(t), u^*(t), p(t)\right),$$

s okrajovou podmínkou $p(T) = \nabla h \left(x^*(T)\right)$. Pak pro všechna $t \in [0, T]$

$$u^*(t) = \arg\min_{u \in U} H \left(x^*(t), u, p(t)\right).$$

Navíc existuje konstanta C taková, že

$$H \left(x^*(t), u^*(t), p(t)\right) = C, \quad \forall t \in [0, T].$$

1.1.7 Algoritmy pro duální řízení

Metody pro nalezení optimálního řízení lze obecně rozdělit do dvou základních kategorií na *globální* a *lokální* viz [8, 7].

Globální metody, používané zejména v posilovaném učení („Reinforcement Learning“), jsou založeny na na Bellmanově principu optimality, Hamilton-Jacobi-Bellmanově rovnosti a dynamickém programování. Tyto algoritmy hledají globálně optimální zpětnovazební řízení pro všechny stavy obecného stochastického systému a proto podléhají nebezpečí „problému dimenzionality“ nebo také rozměrnosti (z anglického “curse of dimensionality”

doslovně - *kletba rozměrnosti*). Jednoduše můžeme tento problém chápat tak, že při numerickém řešení úlohy jsou počítačem procházeny všechny body diskretizovaného stavového a řídícího prostoru jejichž počet s rostoucím počtem dimenzí extrémně (exponenciálně) rychle roste. Výpočet pro mnohadimenzionální úlohy se pak stává co do paměťových nároků, ale hlavně z hlediska výpočetního času prakticky nerealizovatelným.

Lokální metody, častěji studované v teorii řízení, souvisí s Pontryaginovým principem maxima. Jejich podstatou je nalezení řízení, které je pouze lokálně optimální v okolí nějaké „extremalní" trajektorie. Většinou je užito deterministických prostředků jako řešení soustavy obyčejných diferenciálních rovnic (střelbou, relaxací, uspořádáním - collocation nebo spádem gradientu - gradient descent). Tento přístup ale vede na přímovazební řízení a nezle užít pro stochastické úlohy, vyhýbá se ale problému dimenzionality, což umožňuje řešit i komplexnější problémy.

V poslední době je snaha vyvíjet nové algoritmy, které kombinují výhody obou výše zmíněných přístupů. Příkladem může být *diferenciální dynamické programování* (DDP). Tento algoritmus zůstává lokální metodou ve smyslu, že uchovává pouzve jedinou trajektorii, která je lokálně vylepšována. Vylepšení však není založeno na řešení soustavy obyčejných diferenciálních rovnic, ale na dynamickém programování aplikovaném na okolí - "trubici" podél současné trajektorie. Jedná se o algoritmus s konvergencí druhého řádu. Ještě efektivnější je metoda podobná DDP, *iterativní LQG* (iLQG). Tento algoritmus je založen na linearizaci nelineární úlohy v každém bodě reprezentativní trajektorie a následném řešení modifikované Riccatiho rovnice. Výhodou DDP i iLQG je, že jejich výsledkem je zpětnovazební řízení. Obě metody jsou ale stále deterministické a nedokáží se vypořádat s nekvadratickými ztrátovými funkcemi a požadavky na omezené řízení.

S výše zmíněnými problémy se snaží vypořádat modifikovaná iLQG, která bude možná použita pro srovnání s ústřední metodou této práce iLDP. Dále pak do kategorie smíšených metod spadá právě i metoda iLDP, která bude podrobně popsána dále.

1.2 Výběr konkrétních algoritmů pro srovnání

1.2.1 LQG

(iLQG)

1.2.2 Princip separace

1.3 Algoritmus iterativního lokálního dynamického programování

Algoritmus iLDP byl vytvořen pro účely nalezení stochastického optimálního řízení v mnohadimenzionálních stavových a řídících prostorech. Tento případ je častý zejména při řízení biologických pohybů. Metoda je popsána autory v článku [7] a z tohoto zdroje je také převzata.

Základní popis algoritmu, tak jak ho autoři podali, je však pouze šablonou a mnoho detailů a dílčích částí je ponecháno na vyřešení při konkrétní realizaci. To se týká zejména

použitých aproximací pro jednotlivé funkce, zejména aproximace Bellmanovy funkce a aproximace hledaného regulátoru. Dále, protože algoritmus využívá hledání minima, není v základním popisu algoritmu vyřešen konkrétní typ minimalizace. Použitý minimalizační algoritmus se samozřejmě liší podle konkrétního problému, zejména jedná-li se o minimalizaci omezenou nebo neomezenou. Ještě je třeba zmínil, že pro algoritmus je nutno zvolit parametr „velikosti“ okolí, protože se jedná o lokální metodu.

1.3.1 Formulace problému

Naším úkolem je nalézt řízení $\mathbf{u} = \pi(t, \mathbf{x})$, které minimalizuje očekávanou ztrátu

$$J(\pi) = E_\omega \left(h(\mathbf{x}, \pi(t, \mathbf{x})) + \int_0^T l(\mathbf{x}, \pi(t, \mathbf{x}))dt \right)$$

obecně pro spojitý systém:

$$\begin{aligned} d\mathbf{x} &= \mathbf{f}(\mathbf{x}, \mathbf{u})dt + F(\mathbf{x}, \mathbf{u})d\omega \\ \mathbf{x}(0) &= \mathbf{x}_0 \\ t &\in [0, T] \end{aligned} \tag{1.10}$$

v diskrétním tvaru:

$$\begin{aligned} \mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \cdot \Delta k + F(\mathbf{x}, \mathbf{u})e_k \\ \mathbf{x}_{(k=0)} &= \mathbf{x}_0 \\ k &\in \{0, 1, \ldots, N\} \\ \Delta k &= (k+1) - (k) \end{aligned} \tag{1.11}$$

kde hledáme řízení $\mathbf{u} = \pi(k, \mathbf{x})$, které minimalizuje očekávanou ztrátu ***asi***

$$J(\pi) = E \left(h(\mathbf{x}, \pi(N, \mathbf{x})) + \sum_{k=0}^{N-1} l_k(\mathbf{x}, \pi(k, \mathbf{x}))\Delta k \right)$$

1.3.2 Osnova algoritmu

Algoritmus pracuje iteračně, každá iterace začne s řízením π a vytvoří zlepšení π'. Přičemž prvotní řešení π_0 musíme algoritmu dodat jako apriorní informaci. Pro zajištění globální konvergence je možno nové řešení hledat jako konvexní kombinaci starého a algoritmem nalezeného řešení

$$\pi^{nové} = \alpha\pi' + (1 - \alpha)\pi;\ 0 < \alpha \leq 1;\ J(\pi^{nové}) < J(\pi)$$

V každé iteraci proběhne nejprve přípravná fáze, kdy z řízení $\pi(k, \mathbf{x})$ generuje průměrnou trajektorii $\bar{x}(k)$ řešením rovnice 1.10 respektive 1.11. Následně se počítá aproximace $\tilde{V}(k, \mathbf{x})$ Bellmanovy funkce $V(k, \mathbf{x})$ v čase odzadu, tj. od N k 1. Současně počítáme i aproximaci řízení $\pi'(k, \mathbf{x}) \dots \pi'(N-1, \mathbf{x})$. Tedy pro každý čas k takový, že $k = N-1 \dots 1$ jdeme zpět, přičemž pokládáme v koncovém čase N hodnotu aproximace Bellmanovy funkce $\tilde{V}(N, \mathbf{x}) = h(\mathbf{x})$ a provádíme následující čtyři kroky:

1. Generujeme množinu stavů $\left\{\mathbf{x}^{(n)}\right\}_{n=1\dots M}$ shromážděných kolem průměrného stavu $\bar{\mathbf{x}}(k)$.

   Zde se projevuje lokálnost metody. Množina stavů $\left\{\mathbf{x}^{(n)}\right\}$ je vybrána z okolí průměrného stavu $\bar{\mathbf{x}}(k)$. Toto okolí a způsob výběru množiny je třeba konkrétně specifikovat. Pro účely implementace tohoto algoritmu bylo okolí specifikováno parametrem ρ^2. Množina stavů $\left\{\mathbf{x}^{(n)}\right\}$ pak byla generována náhodně jako náhodná veličina s normálním rozdělením se střední hodnotou rovnou průměrnému stavu $\bar{\mathbf{x}}(k)$ a rozptylem specifikovaným parametrem ρ^2.
   Počet vzorků M je nutno zvolit při implementaci algoritmu. Obecně je nejlepší volit maximální možné číslo, ovšem s rostoucím počtem vzorků rostou i paměťové nároky a výpočetní čas algoritmu.

2. Pro každé $\mathbf{x}^{(n)}$ vypočítáme optimální řízení $\mathbf{u}^{(n)}$ minimalizací Hamiltoniánu

   $$H(k, \mathbf{x}, \mathbf{u}) = \mathbf{l}(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^T \tilde{V}_x(k+1, \mathbf{x}) + \frac{1}{2}\mathbf{tr}\left(\sum(\mathbf{x}, \mathbf{u})\tilde{V}_{xx}(k+1, \mathbf{x})\right)$$

   s inicializačním bodem $\pi(k, \mathbf{x}^{(n)})$. Kde $\Sigma(\mathbf{x}, \mathbf{u}) = \mathbf{F}(\mathbf{x}, \mathbf{u})\mathbf{F}(\mathbf{x}, \mathbf{u})^T$. Tedy optimální řízení v čase k pro stav n hledáme jako $\mathbf{u}^{(n)} = \arg\min_{\mathbf{u}} H(k, \mathbf{x}, \mathbf{u})$.

   Pro minimalizaci lze použít například minimalizační funkce programu *Matlab* z balíku *Optimization Toolbox*, konkrétně se jedná o funkce `fminunc` respektive `fmincon` pro neomezenou respektive omezenou minimalizaci. V případě, že by bylo možno spočítat minimalizaci analyticky, jedná se samozřejmě o nejlepší způsob.

3. Pro každé $\mathbf{x}(k)$ aproximovat $v^{(n)} = V(k, \mathbf{x}^{(n)})$ použitím Hamolton-Jacobi-Bellmanovi rovnosti

   $$V(k, \mathbf{x}^{(n)}) \approx \Delta k \cdot H(k, \mathbf{x}^{(n)}, \mathbf{u}^{(n)}) + \tilde{V}(k+1, \mathbf{x}^{(n)})$$

4. Vypočítat novou aporximaci funkce $\tilde{V}(k, \mathbf{x})$ z množiny bodů $\left\{\mathbf{x}^{(n)}, v^{(n)}\right\}$ a aproximaci řízení $\pi'(k, \mathbf{x}^{(n)})$ definované pro všechna x jako z množiny bodů $\left\{\mathbf{x}^{(n)}, \mathbf{u}^{(n)}\right\}$.

1.3.3 Detaily implementace

1.3.4 Konkrétní použité aproximace

Výpočet hodnot a aproximace $\tilde{V}$ $(\tilde{V}_x, \tilde{V}_{xx})$ je opakovaný. Je tedy třeba vysoké optimalizace, proto je použita lineární aproximace ve tvaru lineární kombinace dvakrát diferencovatelných základních funkcí $\phi(x) \in \mathbf{R}^P$ kde $P < N$. Jako základní funkce

jsou voleny funkce 1, x_i, $x_i x_j$, $x_i^2 x_j$. Aproximace je volena jako časově proměnná, kdy $\tilde{V}(k, \mathbf{x}) = \phi(\mathbf{x} - \bar{\mathbf{x}}(k))^T \mathbf{w}(k)$, kde $\mathbf{w}(k)$ je parametrický vektor závislý na čase k.

Označme $\tilde{V}_x = \phi_x^T \mathbf{w}$ a $\tilde{V}_{xx} = \phi_{xx}^T \mathbf{w}$ první a druhou derivaci aproximace Bellmanovy funkce podle proměnné $\mathbf{x}$ respektive *vektor* a *matici* parciálních derivací podle složek vektoru $\mathbf{x}$. Parametry aproximace pro jednotlivé časy $\mathbf{w}$ se určí lineární regresí. Pro $\mathbf{v} = \left[v^{(1)} \dots v^{(M)}\right]$ vektor cílových hodnot a matici $\mathbf{\Phi} = \left[\phi(\mathbf{x}^{(1)} - \bar{\mathbf{x}}(k)) \dots \phi(\mathbf{x}^{(M)} - \bar{\mathbf{x}}(k))\right]$ je minimální kvadratická odchylka $\| \mathbf{v} - \mathbf{\Phi}^T \mathbf{w} \|^2$ pro volbu parametru $\mathbf{w} = \left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^{-1} \mathbf{\Phi}\mathbf{v}$.

Protože je průměrná trajektorie $\bar{\mathbf{x}}(k)$ konstantní v iteraci algoritmu, je z důvodu urychlení výpočtu aproximace vycentrována v tomto bodě. Množina $\left\{\mathbf{x}^{(n)}\right\}$ je časově proměnná, abychom nemuseli v každém kroku počítat $\left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^{-1} \mathbf{\Phi}$, položíme $\mathbf{x}^{(n)} = \bar{\mathbf{x}}(k) + \varepsilon^{(n)}$, kde $\left\{\varepsilon^{(n)}\right\}$ je stejná pro všechny časy k. Množina $\left\{\mathbf{x}^{(n)}\right\}$ se pak jakoby pohybuje podél trajektorie $\bar{\mathbf{x}}(k)$. Tedy $\tilde{V}(k, \mathbf{x}^{(n)}) = \phi(\varepsilon^{(n)})^T \mathbf{w}(k)$ a Φ je konstantní v nejen čase, ale i v iteracích algoritmu a matici $\left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^{-1} \mathbf{\Phi}$ je možno předpočítat (což by nešlo při závislosti na stavech).

1.3.5 Předběžný odhad vlatností algoritmu

2 Systémy pro testování

2.1 Jednoduchý systém

2.1.1 Popis problému

Tato úloha byla převzata z článku [6] zejména z důvodu, aby mohla být porovnána s algoritmem navrženým ve zmíněném zdroji. Sami autoři [6] pak přejali tento problém z [1].

Jedná se o integrátor s neznámým ziskem, tedy lineární časově invariantní systém s jedním vstupem a jedním výstupem.

$$\begin{aligned} y_{k+1} &= y_k + b_k u_k + e_{k+1}, \\ b_k &\sim N(\hat{b}_k, P_k), \\ e_k &\sim N(0, \sigma^2), \\ \mathrm{cov}(e_k, b_k) &= 0, \ \forall k. \end{aligned} \tag{2.1}$$

kde y_k je výstup nebo také stav procesu v čase k, u_k je řízení v čase k. Varianci šumu σ^2 předpokládáme známou, stejně jako počáteční hodnoty systému y_0, $\hat{b}_0$ a P_0. Úkolem je nalézt zpětnovazební řízení

$$u_k^* = u_k^*(y_k, y_{k-1}, \ldots, y_0, u_{k-1}, u_{k-2}, \ldots, u_0), \ 0 \leq k \leq N-1$$

minimalizující očekávanou ztrátu

$$\begin{aligned} J_0 &= \left\{ \sum_{k=0}^{N-1} g_k \right\}, \\ g_k &= (y_{k+1} - r_{k+1})^2, \end{aligned}$$

pro daný časový horizont N a referenční signál, tj. požadovanou hodnotu výstupu, ve formě posloupnosti $\{r_k\}_{k=1}^N$.

Při řešení tohoto problému je výhodné nahlížet na systému jako úlohu s hyperstavem $H_k = [y_k, \hat{b}_k, P_k]$. Pak první rovnici v 2.1 doplníme rovnicemi, ze kterých mohou být rekurzivně napočítány parametry $\hat{b}_k$ a P_k

$$\begin{aligned} \hat{b}_{k+1} &= \hat{b}_k + K_k(y_{k+1} - y_k - \hat{b}_k u_k), \\ P_{k+1} &= (1 - K_k u_k) P_k, \\ K_k &= \frac{u_k P_k}{u_k^2 P_k + \sigma^2}. \end{aligned}$$

Přičemž ztráta v čase k se změní na

$$g_k = (y_{k+1} - r_{k+1})^2 + P_k u_k^2.$$

2.1.2 Úpravy rovnic

2.1.3 Konkrétní užití

2.1.4 Pozorované výsledky

2.2 Synchronní motor s permanentními magnety

2.2.1 Popis systému

Následující model popisuje synchronní elektromotormotor s rotorem tvořeným permanentními magnety. Systém je popsán standartními rovnicemi synchronního stroje s permanentními magnety ve stacionárním tvaru

$$\begin{aligned}
\frac{di_\alpha}{dt} &= -\frac{R_s}{L_s} i_\alpha + \frac{\Psi_{PM}}{L_s}\omega \sin\vartheta + \frac{u_\alpha}{L_s}, \\
\frac{di_\beta}{dt} &= -\frac{R_s}{L_s} i_\beta - \frac{\Psi_{PM}}{L_s}\omega \cos\vartheta + \frac{u_\beta}{L_s}, \\
\frac{d\omega}{dt} &= \frac{k_p p_p^2 \Psi_{PM}}{J}\left(i_\beta \cos\vartheta - i_\alpha \sin\vartheta\right) - \frac{B}{J}\omega - \frac{p_p}{J} T_L, \\
\frac{d\vartheta}{dt} &= \omega.
\end{aligned} \tag{2.2}$$

Zde $i_{\alpha,\beta}$ reprezentují proudy a $u_{\alpha,\beta}$ napětí na statoru. Poloha (úhel otočení) rotoru je označen ϑ a ω je pak rychlost otáčení. Dále R_s je rezistance a L_s induktance statoru. Ψ_{PM} má význam magnetického toku permanentních magnetů rotoru, B tření a T_L je zatěžovací moment. Konstanta p_p označuje počet párů polů a k_p Parkovu konstantu.

Cílem je návrh řízení bez senzorů, kdy čidla pro měření polohy a otáček nejsou (z různých důvodů) přítomna. Tedy jediné měřitelné veličiny jsou:

$$y_t = \left[i_\alpha(t), i_\beta(t), u_\alpha(t), u_\beta(t)\right].$$

Které samozřejmě můžeme měřit jen s určitou přesností.

Diskretizace modelu 2.2 pomocí Eulerovy metody vede na následující diskrétní popis:

$$\begin{aligned}
i_{\alpha,k+1} &= \left(1 - \frac{R_s}{L_s}\Delta k\right) i_{\alpha,k} + \frac{\Psi_{PM}}{L_s}\Delta k \omega_k \sin\vartheta_k + \frac{\Delta k}{L_s} u_{\alpha,k}, \\
i_{\beta,k+1} &= \left(1 - \frac{R_s}{L_s}\Delta k\right) i_{\beta,k} - \frac{\Psi_{PM}}{L_s}\Delta k \omega_k \cos\vartheta_k + \frac{\Delta k}{L_s} u_{\beta,k}, \\
\omega_{k+1} &= \left(1 - \frac{B}{J}\Delta k\right)\omega_k + \frac{k_p p_p^2 \Psi_{PM}}{J}\Delta k \left(i_{\beta,k}\cos\vartheta_k - i_{\alpha,k}\sin\vartheta_k\right) - \frac{p_p}{J} T_L \Delta k, \\
\vartheta_{k+1} &= \vartheta_k + \omega_k \Delta k.
\end{aligned}$$

Kde Δk označuje diskrétní časový okamžik. Předpokládáme, že paremetry modelu známe, můžeme tedy provést následující substituci za účelem zjednodušení: $a = 1 - \frac{R_s}{L_s}\Delta k$, $b = \frac{\Psi_{PM}}{L_s}\Delta k$, $c = \frac{\Delta k}{L_s}$, $d = 1 - \frac{B}{J}\Delta k$, $e = \frac{k_p p_p^2 \Psi_{PM}}{J}\Delta k$. Pro jednoduchost uvažujme model bez zatížení, tedy zatěžovací moment T_L je nulovy a zjednodušený model je:

$$\begin{aligned} i_{\alpha,k+1} &= a i_{\alpha,k} + b\omega_k \sin\vartheta_k + c u_{\alpha,k}, \\ i_{\beta,k+1} &= a i_{\beta,k} - b\omega_k \cos\vartheta_k + c u_{\beta,k}, \\ \omega_{k+1} &= d\omega_k + e\left(i_{\beta,k}\cos\vartheta_k - i_{\alpha,k}\sin\vartheta_k\right), \\ \vartheta_{k+1} &= \vartheta_k + \omega_k \Delta k. \end{aligned} \tag{2.3}$$

Tyto rovnice můžeme chápat jako popis systému se stavem $x_k = [i_{\alpha,k}, i_{\beta,k}, \omega_k, \vartheta_k]$.

2.2.2 Úprava rovnic

2.2.3 Aplikace iLDP

2.2.4 Výsledky jiných metod

3 Výsledky

3.1 Výsledky algoritmu iLDP

3.1.1 Různá počáteční nastavení

3.2 Výsledky ostatních použitých metod

3.2.1 LQG

3.2.2 Princip separace

3.3 Srovnání

3.3.1 Získané výsledky

3.3.2 Porovnání algoritmů

3.4 Diskuze pro metodu iLDP

Závěr

Literatura

[1] Aström K. J.; Helmersson A.: Dual control of an integrator with unknown gain. *Computers and Mathematics with Applications*, 1986: s. 12A, 653–662.

[2] Bertsekas D. P.: *Dynamic Programming and Optimal Control*, ročník I. Belmont, Massachusetts: Athena Scientific, třetí vydání, 2005.

[3] Melichar J.: Lineární systémy 1: Učební texty. [online], 2010, [cit. 2010-04-03] Dostupné z: <http://www.kky.zcu.cz/uploads/courses/ls1/LS1-Ucebni-texty-2010.pdf>.

[4] Nagy I.; Pavelková L.; Suzdaleva E.; aj.: *Bayesian decision making: Theory and examples*. Prague: ÚTIA AVČR, 2005.

[5] Štecha J.: *Teorie dynamických systémů: transparenty a přednášky*. Praha: ČVUT, Elektrotechnická fakulta, 2003, ISBN 80-01-02744-9.

[6] Thompson A. M.; Cluett W. R.: Stochastic iterative dynamic programming: a Monte Carlo approach to dual control. *Automatica*, 2005: s. 41:767–778.

[7] Todorov E.; Tassa Y.: Iterative local dynamic programming. *Proceedings of the 2nd IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009: s. 90–95.

[8] Todorov E.; Weiwei L.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *In proceedings of the American Control Conference*, 2005.

[9] Virius M.: *Základy algoritmizace*. Praha: ČVUT, Fakulta jaderná a fyzikálně inženýrská, 2008, ISBN 978-80-01-04003-4.