Ústředním pojmem v teorii řízení je systém. Úlohou řízení systému je prostřednictvím vstupů udržet výstupy systému na požadované hodnotě. Výstup systému je popsán rovnicí (1), kde napravo je známá funkce předchozích vstupů a výstupů, aktuálního vstupu, realizace přítomného šumu a neznámého parametru, který nemůžeme přímo pozorovat. O parametru \theta máme apriorní informaci v podobě postačující statistiky T\_0.

Ačkoliv není úkolem co nejlépe zidentifikovat neznámý parametr systému, nýbrž udržet vstupy na požadované hodnotě, pro kvalitní řízení je dobré systém znát co nejpřesněji. Na základě odezvy systému můžeme pomocí Bayesova vzorce získat aposteriorní hustotu pravděpodobnosti.

Na základě (1) a (2) se zavádí složená veličina zvaná hyperstav, pro kterou je pomocí (1) a (2) definován vývoj v čase.

Cílem řízení je držet výstupy systému na požadované hodnotě. Kvalitu řízení určuje ztrátová funkce – platí, že čím menší ztráta, tím lepší. Ztrátovou funkce je aditivní v čase.

Řídící strategie je funkce, která každému bodu hyperstavu přiřadí nějaký vstup.

Hledáme posloupnost řídících strategií, která minimalizuje očekávanou ztrátu. Je potřeba počítat střední hodnotu, protože kvůli přítomnosti šumu a neznámého parametru je ztráta náhodnou veličinou

Protože ztrátová funkce je aditivní v čase, platí takzvaný princip optimality. Ten umožňuje hledat optimální řídící strategie postupně od konce řídícího horizontu. Pro každé H\_t tedy minimalizuji pouze vzhledem k u\_t – to je jednodušší úloha! Při počítání se používá rovnice pro výstup systému a rovnice hyperstavu.

Tento postup má ale stále vady na kráse – počítám střední hodnotu veličiny, jejíž rozdělení nemám explicitně vyjádřené a minimalizuji funkci, pro kterou nemám předpis. Proto je vhodné uchýlit se k použití aproximačních metod.

Zabýval jsem se aproximační metodou stochastických iterativních aproximací dynamického programování. To spočívá v dvojím.

Za prvé v použití iterativního dynamického programování. Při tom se namísto přímého nalezení řešení konstruuje posloupnost, která za určitých předpokladů konverguje k řešení. To usnadňuje problém minimalizace – v každé iteraci například můžeme minimalizovat pouze přes nějakou konečnou množinu, kde minimum nalezneme snadno

Druhý pilíř algoritmu spočívá v použití metody Monte Carlo. Ta je použita k odhadu očekávané ztráty pomocí průměru přes n realizací. Realizace ztráty pro konkrétní bod hyperstavu se pak generuje takto…

Při počítání optimálního řízení se tedy postupuje od konce řídícího horizontu, metodou Monte Carlo se napočítává očekávaná ztráta, ta se minimalizuje v okolí řízení spočteného v předchozí iteraci algoritmu. Tím napočítám nové optimální řízení, které mohu dále iterovat.

Systém, na kterém jsem uvedený algoritmus zkoušel, je popsán takto. \theta je neznámý parameter.

Optimální řízení dává nulové výstupy a ztrátová funkce je kvadratická, jak popisuje (8)

Předpokládá se, že odhad neznámého parametru je nekorelovaný s šumem a apriorní informace je ve formě střední hodnoty a rozptylu. Za těchto předpokladů lze odvodit konkrétní tvar rovnic pro vývoj hyperstavu.

Hyperstav je tvořen vektorem obsahujícím … vhodnou transformací souřadnic lze dimenzi úlohy ještě zredukovat.

Pro posouzení kvality navrženého řízení jsem pro tento systému implementoval ještě tři další metody.

První se v literatuře označuje jako certainty equivalence. Spočívá v tom, že pro návrh řídící strategie se namísto náhodných veličin uvažují jejich bodové odhady. Díky tomu je možné nalézt analytické vyjádření pro řídící zásah.

Druhá metoda se jmenuje opatrné řízení. Pomocí té se určuje řízení minimalizací očekávané ztráty na horizont délky 1. Opět je možné nalézt analytické vyjádření pro řídící zásah.

Poslední porovnávaná metoda je řízení získané klasickým numerickým přístupem k dynamickému programování. Pro výpočet střední hodnoty je použitá Simpsonova metoda a pro minimalizaci ztráty pak jednoduchá interpolační metoda.

Nejdříve jsem zkoumal průměrnou ztrátu, které jednotlivé řídící algoritmy dosáhly. Výsledky zachycuje obrázek. Tady je dosažená ztráta (průměr přes 1000 simulací) pro konkrétní algoritmy a různé počáteční odhady na neznámý parametr. *Okomentovat sloupečky*

Z kvalitativního hlediska se řízení získané pomocí SIDP a DP liší. V případě SIDP nedochází k případům ve kterých je řízení výrazně špatné, zároveň ale často dosahuje mírně vyšší ztráty než DP. Celkově se dá říci, že řízení navržené pomocí SIDP navrhuje opatrnější záshy.

Poslední testovanou vlastností byla robustnost řízení vůči přesnosti apriorní informace. Při tom se prokázalo, že při výrazně špatné apriorní informaci (\hat{\theta}\_0=-10 a \theta=1) je nejlepší řízení navržené metodou SIDP. Toto řízení mělo jen mírně horší ztrátu oproti případu přesné apriorní informace. Zhoršení u ostatních algoritmů bylo znatelnější.

Shrnutí + komentář k robustnosti