Technická univerzita v Liberci
Fakulta mechatroniky a mezioborových inženýrských studií

Návrh optimálních trajektorií dynamických systémů s využitím numerických algoritmů optimalizace

Habilitační práce

Jan Cvejn

Únor 2005

Obsah

Tabulka použitých symbolů

Symbol	Význam
$\mathbf{x}$	vektor stavu
$\mathbf{u}$	vektor řízení
$\mathbf{y}$	vektor zobecněných souřadnic
t	čas
$\mathbf{f}$	pravá, popř. levá strana soustavy pohybových rovnic
t_f	doba pohybu
n	řád soustavy pohybových rovnic; řád prostoru spline-funkcí
m	dimenze vektoru $\mathbf{y}$ a $\mathbf{u}$
$\mathbf{0}$	nulový vektor nebo matice
$\mathbf{1}$	jednotková matice
a_i, a_{ij}	bázové koeficienty
$\{b_i\}_{i_1}^{i_2}$	systém bázových funkcí $b_i(t)$, kde $i = i_1,...,i_2$
N	parametr bázového systému (u systémů typu spline dělení intervalu $\left[0,t_f\right]$)
T	parametr bázového systému – u systémů typu spline je $T = t_f / N$
$\langle a \mid b \rangle$	skalární součin
$\|\cdot\|_2$	norma indukovaná skalárním součinem
$\|\cdot\|_{\mathbf{K}}$	zobecněná eukleidovská norma $\|\mathbf{x}\|_{\mathbf{K}}^2 = \mathbf{x}^T\mathbf{K}\mathbf{x}$, $\mathbf{K}$ pozitivně definitní
$\Lambda^k(t)$	bázový prvek vyššího řádu odvozený od $\Lambda^1(t)$
Λ_i^k, Σ_i^k	zkrácené označení $\Lambda^k(t-iT)$, $\Sigma^k(t-iT)$
$S_N^k(0,t_f)$	prostor spline-funkcí řádu k na $\left[0,t_f\right]$
$Z_N^k(0,t_f)$	prostor spline-funkcí s nulovými okrajovými podmínkami do řádu n
$V(\lambda_1,\cdots\lambda_n)$	Vandermondův determinant
$\{\Lambda_i^{n+1}\}_{-n}^{N}$	bázový systém v prostoru $S_N^n(0,t_f)$
$\{\Sigma_i^{n+1}\}_{-n}^{N}$	finitní bázový systém v prostoru $S_N^n(0,t_f)$
$\mathbf{B}(t)$	matice báze

$\mathbf{A}_p$	bod splňující okrajové podmínky
$\mathbf{A}_0$	bod splňující nulové okrajové podmínky
$\mathbf{C}$	matice soustavy okrajových podmínek
$\mathbf{C}_1$	regulární podmatice matice soustavy okrajových podmínek
R	pravá strana soustavy okrajových podmínek
$\mathbf{U}, \mathbf{V}$	matice rozkladu soustavy okrajových podmínek
$\mathbf{X}$	matice koeficientů v redukovaném prostoru parametrů
J	kritérium optimality
$\nabla_{\mathbf{A}} J$	gradient J podle $\mathbf{A}$
J'	kritérium rozšířené o pokutové složky
$\mathbf{Z}$	vektor optimalizovaných parametrů
n_z	počet složek vektoru $\mathbf{z}$
C	prostor spojitých funkcí
C_k	prostor funkcí k-krát spojitě diferencovatelných
C_k^m	prostor vektorových funkcí dimenze m k-krát spojitě diferencovatelných

Předmluva

Cílem této práce je popsání výsledků, kterých autor dosáhl v období mezi roky 2000-2004 v oblasti výpočtu optimálních trajektorií dynamických systémů, avšak v návaznosti na postupy, které byly realizovány již v rámci diplomové a disertační práce v letech 1996-2000. Jedná se tedy o výsledky poměrně dlouhodobého individuálního výzkumu, který však v minulých letech probíhal s různou intenzitou. Diplomová práce byla především realizací metody aproximace trajektorie průmyslových robotů s využitím spline-funkce navržené školitelem autora Doc. Mgr. Václavem Zádou, CSc. [1]. Počínaje disertační prací však byly realizovány postupy, které vychází výhradně z vlastních úvah na základě poznatků získaných v odborné literatuře.

Předmětem tohoto studia je hledání efektivních metod umožňujících při zadaných dynamických vlastnostech systému, počátečních a koncových podmínkách pohybu, a případně i dodatečných reálných omezeních, numericky určit trajektorii optimální dle zadaného kritéria. Přitom jsou uvažována i komplikovaná omezení, jako jsou překážky v pracovním prostoru. Třebaže aplikace výsledků byla původně předpokládána v oblasti robotiky, jejich využitelnost je možná pro mnohem širší třídu technických problémů. Vlastnosti realizovaných metod jsou demonstrovány v konfigurovatelném softwarovém produktu, který je postupně přepracováván a aktualizován. Program umožňuje:

- definovat dynamiku systému
- definovat tvar systému a jeho závislost na pozici členů
- definovat tvar a pozici překážek
- zvolit metodu optimalizace a nastavit parametry výpočtu

Program generuje výsledné časové průběhy a prostorovou scénu v podobě textových souborů, které jsou zpracovatelné do grafické podoby pomocí dalších softwarových nástrojů.

Ačkoliv téma práce se do značné míry shoduje s disertační prací autora, v posledních letech bylo dosaženo výrazného zdokonalení. Došlo zejména k několikanásobnému urychlení výpočtu, jak pro problémy bez omezení, tak i v případě pohybu v prostředí s překážkami. Mnohé z těchto výsledků jsou publikovány poprvé v této práci. Oproti disertační práci bylo rovněž prakticky realizováno několik alternativních přístupů, umožňujících porovnání výsledků, zobecnění a další vývoj směrem k urychlení výpočtu.

Kromě této oblasti se však autor v minulých letech zabýval i jinými okruhy technických problémů, a to dokonce z větší části. Několik publikací bylo věnováno numerickému návrhu zpětnovazebních regulátorů pro řízení lineárních systémů s omezeními. Autor byl inspirován myšlenkou provést návrh kvadraticky optimálního regulátoru čistě numericky s využitím univerzálních algoritmů optimalizace. V případě, že nejsou zadány omezující požadavky na parametry regulátoru a hodnoty stavových a akčních veličin, lze použít klasické metody založené na řešení dvoubodového okrajového problému, který vznikne dosazením do Eulerových-Lagrangeových rovnic. V případě zadaných omezení však tyto postupy nelze aplikovat. Existuje možnost výpočtu optimálního akčního zásahu minimalizací kritéria v každém kroku řízení. Tento přístup je však obvykle vhodný pouze pro systémy s pomalou dynamikou a není použitelný v případě jednoduchých hardwarových řídicích platforem, jako jsou programovatelné logické automaty.

Byla proto navržena odlišná metoda, založená na konstrukci rozšířeného kritéria optimality, které zaručuje kvalitní budoucí průběh regulace v rámci omezení pro všechny stavy ze zadané množiny. Optimální konstantní regulátor pak může být nalezen minimalizací této funkce pomocí běžných algoritmů optimalizace. Navržené postupy byly softwarově realizovány, avšak dosud nebyly prakticky využity.

Značné množství energie bylo v minulých letech věnováno vývoji konkrétního software pro průmysl. Nejvýznamnějším výsledkem v této oblasti, třebaže ne jediným, je komplexní software GIOpt pro výrobu rovinných tvarů z tabulí skla. Program umožňuje na základě vstupních dat, jako jsou rozměry materiálu a výrobků, požadované počty kusů a další technologické parametry, generovat optimální rozmístění výrobků na řezných plánech z hlediska využitého materiálu a program pro řezací CNC stroj. Pro optimalizaci je v principu využíván algoritmus prohledávání s heuristikami; problém je však výrazně komplikován množstvím omezujících technologických požadavků.

Program byl vyvíjen ve spolupráci s libereckou firmou Sklopan, a.s., a třebaže je původně určen především pro řezací stroje vyráběné touto firmou, je použitelný i pro stroje jiných výrobců. Do současné doby bylo prodáno přibližně 15 licencí do České republiky, Slovenské republiky, Polska, Bulharska a Ruska. Program tvoří neoddělitelnou součást výbavy stroje a v případě jeho chybné funkce by byla užitná hodnota celého zařízení nulová.

Dodatečně byl vytvořen jednoduchý CAD software G-Shape pro interaktivní návrh rovinných tvarů a některé další pomocné programy. V současné době jsou programy GIOpt a G-Shape k dispozici v české a anglické verzi, včetně dokumentace, v minulých letech však byla nabízena i polská verze. Ačkoliv jsou tyto programy čistě praktickou záležitostí, při jejich realizaci byla řešena řada zajímavých technických problémů. Hrubý popis základních problémů a principů řešení byl publikován v [A16].

Třebaže autor si osobně považuje především výsledků své výzkumné a programátorské práce, jeho hlavní pracovní náplní v letech 2000-2004 byla pedagogická činnost na Fakultě mechatroniky a mezioborových inženýrských studií Technické univerzity v Liberci. Zde autor garantuje výuku předmětů Programovací jazyky a operační systémy I a II a Programovací techniky. Obsahem prvních dvou předmětů jsou programovací jazyky C a C++, úvod do problematiky operačních systémů a rysy architektury operačních systémů MS Windows a UNIX. Předmět Programovací techniky se zabývá zejména popisem datových struktur a algoritmů pro optimální organizaci dat a efektivní přístup k nim. Ačkoliv tyto předměty autor přímo nezavedl do výuky, zejména poslední dva předměty získaly pod jeho vedením novou náplň. K přednášené látce byly vytvořeny elektronické studijní materiály v celkovém rozsahu skoro 200 stran. Uvedené texty však nedosahují úrovně skript; jsou spíše široce rozvětvenou osnovou přednášek.

Pro porovnání množství programátorské práce vykonané v minulých letech je dále uveden přehled vytvořených softwarových produktů a jejich rozsah měřený přibližným počtem řádků. Započítány jsou pouze zdrojové texty vytvořené autorem, nikoliv standardní knihovny a hlavičkové soubory. Programy GECO a SMDMonitor jsou praktické aplikace menšího rozsahu, v obou případech pro monitorování a archivaci technologických veličin. Programy DHOpt a DUOpt jsou realizací aktuálních výsledků prezentovaných v této práci (metoda nahrazení trajektorie a nahrazení řízení včetně návrhu trajektorie v prostoru s překážkami). Nejrozsáhlejším dílem je dvojice programů GIOpt a G-Shape, jejíž kompletní výpis by vyžadoval přibližně 2000 stran. Pro vývoj byl ve všech případech použit programovací jazyk C++ a (kromě disertační práce) vývojové prostředí Microsoft Visual C++.

Program	**Období vývoje**	**Počet řádků (x1000)**
Disertační práce	1998-2000	20
GECO	2001	8
SMDMonitor	2002	17
DHOpt, DUOpt	2003-2004	30
GIOpt + G-Shape	1999-2003	100

K sepsání této práce významně přispěl pobyt autora na Technické univerzitě v Drážďanech od října 2004 do konce ledna 2005. Proto patří poděkování nadaci Herberta Quandta při TU Dresden, která na tento výzkum poskytla stipendium, i vedení Fakulty mechatroniky Technické univerzity v Liberci a Katedry softwarového inženýrství na této fakultě pro uvolnění autora z výuky ve jmenovaném období.

Část 1.

Úvod

1.1. Problém optimálního řízení

Teorie optimálních procesů je matematickou disciplínou, která nachází uplatnění v mnoha aplikacích technické nebo ekonomické povahy. Tento trend je na jedné straně dán stále většími požadavky na maximální využití omezených zdrojů, minimalizaci nákladů a co největší efektivnost, a na straně druhé rozvojem výpočetní techniky.

Úkolem teorie optimálních procesů je určení funkce řízení, která přemístí daný dynamický systém z počátečního stavu do koncového stavu optimálním způsobem. V této práci jsou uvažovány pouze spojité systémy popsané soustavou obyčejných diferenciálních rovnic

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},t) \tag{1.1}$$

kde $\mathbf{x}$ je vektor stavu, $\mathbf{u}$ vektor řízení a t čas, s okrajovými podmínkami ve tvaru

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{1.2}$$

$$\boldsymbol{\psi}(\mathbf{x}(t_f)) = \mathbf{0}\,. \tag{1.3}$$

Jako kritérium optimality je uvažován integrální funkcionál podél trajektorie, který pro optimální průběh řízení dosahuje svého minima:

$$J = \int_0^{t_f} f_0(\mathbf{x},\mathbf{u},t)\,dt \to \min\,. \tag{1.4}$$

U reálných systémů je často nutné uvažovat dodatečné omezující požadavky. Čas t_f může, ale nemusí být zadán. Funkcionál (1.4) je vyjádřením kvality pohybu podél trajektrorie, popř. splnění dodatečných požadavků. Často je voleno např. kritérium ve tvaru $f_0 \equiv \mathbf{u}^T\mathbf{K}\mathbf{u}$, kde $\mathbf{K}$ je zvolená diagonální matice s kladnými prvky.

V případě, že funkce $\mathbf{f}, f_0, \boldsymbol{\psi}$ jsou spojitě diferencovatelné podle svých parametrů $\mathbf{x}$, $\mathbf{u}$, resp. $\mathbf{x}(t_f)$, a po částech spojité podle času, teoretický rozbor pro úlohy bez dodatečných omezení vede na soustavu Eulerových-Lagrangeových diferenciálních rovnic které jsou splněny podél optimální trajektorie a společně s pohybovými rovnicemi systému tvoří dvoubodový okrajový problém [4]. Tento problém lze bohužel řešit analyticky pouze v jednoduchých případech; pro reálné situace je zpravidla nutné použít numerické metody.

Pro numerické řešení úloh optimálního řízení byly v minulosti vyvinuty algoritmy, které jsou buďto založeny na iteračním řešení dvoubodového okrajového problému, nebo na nebo na přímé minimalizaci kritéria v prostoru funkcí řízení [4], [5]. V prvním případě jsou některé z okrajových podmínek splněny přesně a ostatní s určitou chybou, která se v posloupnosti následujících kroků zmenšuje k nule (Goodmanova-Lancova metoda a kvazilinearizace). Metody z druhé skupiny jsou obdobou algoritmů optimalizace funkcí konečného počtu proměnných. Využívají gradientu, popř.

druhé derivace funkcionálu v prostoru funkcí řízení pro výpočet korekce řízení. Koncová podmínka představuje omezení typu rovností.

Konvergence těchto metod k řešení však většinou není zaručena a velmi záleží na kvalitním počátečním odhadu. Nevýhodou je rovněž skutečnost, že jsou schopné nalézt pouze lokální minimum kritéria, což je nedostatkem, neboť především u úloh s omezeními může být těchto lokálních extrémů více a není předem znám jejich počet. Často však není teoreticky zaručeno dokonce ani nalezení lokálního minima (tzv. globální konvergence), na rozdíl od obdobných algoritmů pro minimalizaci funkcí konečného počtu parametrů. Některé z těchto metod jsou rovněž značně náchylné k numerickým chybám vznikajícím integrací Eulerových-Lagrangeových a pohybových rovnic.

Z uvedených důvodů je v literatuře [5] doporučeno soustavu převést do diskrétního tvaru a problém (1.4) nahradit úlohou optimalizace účelové funkce konečného počtu parametrů:

$$\begin{aligned} \mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{g}_k(\mathbf{x}_k, \mathbf{u}_k) \\ J &= \mu(\mathbf{x}_k) + \sum_{k=0}^{N-1} h_k(\mathbf{x}_k, \mathbf{u}_k) \to \min \end{aligned} \tag{1.5}$$

kde $\mathbf{x}_k = \mathbf{x}(kT)$, $\mathbf{u}_k = \mathbf{u}(kT)$, $T = t_f / N$ a N je zvolený počet kroků vzorkování.

Ačkoliv diskrétní tvar (1.5) v případě nelineárních systémů obvykle není znám v explicitním tvaru, vždy je možné provést transformaci do diskrétního tvaru integrací rovnice (1.3) v každém kroku, neboť platí:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \int_{kT}^{(k+1)T} \mathbf{f}(\mathbf{x}(t), \mathbf{u}_k, t)\, dt\,. \tag{1.6}$$

V diskrétním tvaru (1.5) je kritérium J funkcí parametrů $\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}$ a problém optimálního řízení přechází na běžnou optimalizační úlohu konečného počtu parametrů. Pro hledání minima lze pak použít univerzální metody optimalizace.

Metody optimalizace představují zvláštní oblast numerické matematiky, která nachází uplatnění v mnoha technických problémech. Pro minimalizaci hladkých funkcí jsou v současné době k dispozici efektivní algoritmy, jejichž vlastnosti jsou dobře známy po teoretické i praktické stránce. To lze částečně říci i v případě úloh s omezeními, třebaže zde je situace složitější a existuje mnohem méně experimentálních studií.

V komplikovanějších případech úlohy (1.4), popř. (1.5), bohužel vedou na hledání globálního minima. Více lokálních extrémů kritéria vzniká u silně nelineárních systémů a v případě nekonvexních omezení. Třebaže problém globální optimalizace není obecně řešitelný ani v konečné dimenzi, existují algoritmy umožňující globální minimum nalézt v mnoha praktických případech, zpravidla však za cenu velkého počtu vyhodnocení účelové funkce. Úspěšnost nebo neúspěšnost těchto postupů často není podložena teoretickými výsledky, ale spíše praktickými zkušenostmi. Výhodou algoritmů globální optimalizace je schopnost přímo řešit úlohy s omezeními ve velmi obecném tvaru a nalézt minimum nehladkých, popř. i nespojitých funkcí.

1.2. Využití systémů bázových funkcí

Nevýhodou transformace do diskrétního tvaru je skutečnost, že pro kvalitní aproximaci hladkých funkcí je třeba jemného vzorkování, což může mít za následek příliš vysoký počet parametrů pro optimalizaci (např. stovky). Třebaže současné metody hledání lokálního minima hladkých funkcí jsou schopny řešit i takovéto problémy v akceptovatelné době, nelze očekávat úspěšnost metod globální optimalizace, neboť náročnost prohledání množiny parametrů roste exponenciálně. Rozdělíme-li např. daný interval hodnot každého parametru na k dílů, získáme k^p p-rozměrných

intervalů, kde p je počet parametrů. Zkušenosti autora ukazují, že praktickým limitem u technických problémů se zdá být u nejlepších metod zhruba 100 parametrů, avšak v tom případě je často vyžadován enormně vysoký počet vyhodnocení účelové funkce.
Z výše uvedených důvodů autor považuje za výhodný jiný způsob transformace problému do konečné dimenze, který lze považovat za určité zobecnění diskretizace. Složky funkce řízení, popř. trajektorie, mohou být vyjádřeny konečným váženým součtem zvoleného systému nezávislých bázových funkcí $\{b_0(t),...,b_N(t)\}$. Jsou pak hledány optimální hodnoty váhových koeficientů. V rámci této práce byly realizovány dvě varianty tohoto přístupu:

1. nahrazení trajektorie
2. nahrazení funkce řízení

První způsob je výrazně efektivnější, neboť nevyžaduje řešení diferenciálních rovnic (1.1). Je však aplikovatelný pouze v případě, že systém umožňuje z pohybových rovnic explicitně vyjádřit funkci řízení pomocí derivací trajektorie. Tuto vlastnost mají zejména určité typy mechanických systémů. Obzvlášť výhodný je tento postup v případě speciálního tvaru koncových podmínek, který se však v praktických situacích vyskytuje velmi často. V tomto směru autor navazuje zejména na svou disertační práci, kde se zabýval konstrukcí prostorů spline-funkcí pro nahrazení optimální trajektorie. Původní výsledky se však podařilo v posledních letech v řadě směrů překonat.

Druhý způsob je obecnější a nevyžaduje žádné dodatečné požadavky na tvar rovnic (1.1). Tento postup byl nově implementován do jedné z variant programu pro výpočet optimálních trajektorií. Nevýhodou metody je nutnost integrace pohybových rovnic jako součást výpočtu hodnoty kritéria v každém iteračním kroku a nepřesné splnění koncových podmínek. Tím je tento způsob ekvivalentní gradientním metodám výpočtu optimálního řízení ve funkcionálním prostoru.

1.3. Návrh optimálních trajektorií v prostoru s překážkami

Popsaný způsob určení optimální trajektorie je možné využít i v případě složitých omezení, jako jsou překážky v pracovním prostoru mechanických systémů. Praktická realizace tohoto výpočtu je z hlediska složitosti vrcholem této práce. Jedná se však o problém globální optimalizace, kdy omezení jsou ve velmi obecném tvaru.

Pro reprezentaci tvaru systému a překážek je třeba použít určitý typ modelů, který musí být zvolen se zřetelem na maximální efektivitu detekce kolize, popř. zjištění míry kolize. Pro první přiblížení je možné např. nahradit systém určitým počtem bodů na jeho povrchu a testovat zda některý z těchto bodů se v nějakém čase nachází uvnitř překážky. Pro praktické využití je však třeba často definovat velký počet těchto bodů a určovat jejich pozici v každém čase, což mimořádně prodlouží dobu výpočtu. Proto další postup směřoval k plnému nahrazení tvaru.

V tomto případě je v každém časovém bodě testována kolize mezi tělesy reprezentujícími systém a překážky, popř. je vyhodnocena míra kolize mezi modely, která může být výhodnější z hlediska efektivity optimalizace. Vyšetření kolize lze rozdělit na dva podproblémy:

- efektivní vyšetření kolize v konkrétním čase t, které souvisí s volbou typu modelu pro aproximaci tvaru systému a překážek
- efektivní vyšetření kolize podél trajektorie, které souvisí s určením optimálních diskrétních časových bodů testování kolize, které nemusí být ekvidistantní

Jiným faktorem, který se v tomto případě výrazně podílí na celkové rychlosti výpočtu a kvalitě výsledků, je volba algoritmu globální optimalizace. Bylo testováno několik metod, jak na množině vhodných testovacích funkcí, tak i přímo pro řešení popsaného problému. Existují sice určité teoretické výsledky v této oblasti, např. [21], ale v praxi se ukazuje, že často nejsou směrodatné pro posouzení kvality jednotlivých algoritmů, ani pro jejich návrh. Většina metod bohužel selhává, neboť potřebný počet vyhodnocení kritéria je příliš vysoký a samotný výpočet kritéria, který

zahrnuje rovněž vyšetření kolize ve všech bodech trajektorie, je časově náročnou operací. Přesto se však podařilo nalézt algoritmy, které přináší uspokojivé výsledky v akceptovatelné době.

Práce je dále rozdělena do několika částí, které jsou do značné míry nezávislé:

- 2. Systémy bázových funkcí
- 3. Výpočet metodou nahrazení trajektorie
- 4. Výpočet metodou nahrazení řízení
- 5. Optimální pohyb v prostoru s překážkami
- 6. Algoritmy optimalizace
- 7. Příloha – matematické modely mechanických systémů

Každá část (kromě poslední) má svůj úvod a závěr. Části 3 - 5 rovněž obsahují řešené příklady.

Část 2.

Systémy bázových funkcí

Základním principem, který je v této práci využíván, je nahrazení trajektorie nebo funkce řízení váženým součtem zvolených bázových funkcí. V následujících kapitolách jsou nejprve definovány některé základní pojmy a tvrzení z oblasti prostorů funkcí. Následně jsou popsány výsledky týkající se konstrukce prostorů spline-funkcí pro aproximaci trajektorie, které jsou využívány v následujících částech práce.

2.1. Lineární prostory funkcí

Lineárním prostorem funkcí na intervalu $\left[0,t_f\right]$ nazýváme množinu M takovou, že jestliže $f_1 \in M$ a $f_2 \in M$, pak v $\left[0,t_f\right]$ platí také $a_1 f_1 + a_2 f_2 \in M$, kde a_1, a_2 jsou libovolná reálná čísla.

Uvažujme soustavu funkcí $B \equiv \{b_0(t),...,b_N(t)\}$ v intervalu $\left[0,t_f\right]$. Tyto funkce se nazývají *nezávislé*, jestliže soustava

$$a_0 b_0(t) + a_1 b_1(t) + ... + a_N b_N(t) = 0 \tag{2.1}$$

má v $\left[0,t_f\right]$ pouze nulové řešení $a_0 = a_1 = ... = a_N = 0$. Přitom rovností rozumíme rovnost v každém bodě a „0" v prostoru funkcí se chápe jako funkce s hodnotou 0.

Tvrzení: Je-li možné funkci $f \in M$ vyjádřit v systému nezávislých funkcí B jako součet

$$f(t) = \sum_{i=0}^{N} a_i b_i(t) \tag{2.2}$$

pak je toto vyjádření jednoznačné.

Důkaz: V opačném případě totiž existují různé dva vektory koeficientů $(a_i),(\tilde{a}_i)$ takové, že platí

$$\sum_{i=0}^{N} (a_i - \tilde{a}_i) b_i(t) = 0 . \tag{2.3}$$

Vzhledem k tomu, že systém $B \equiv \{b_0(t),...,b_N(t)\}$ je nezávislý, má soustava (2.3) pouze řešení $a_i = \tilde{a}_i$ pro všechna i, což je spor. ♦

Je-li dán nějaký systém nezávislých funkcí B, lze zřejmě zkonstruovat lineární prostor funkcí jako množinu všech lineárních kombinací (2.2).

Uvažujme lineární prostor funkcí M na $\left[0,t_f\right]$. Systém $B \equiv \{b_0(t),...,b_N(t)\}$ nazvěme *úplným* na M, jestliže libovolná funkce $f \in M$ lze v $\left[0,t_f\right]$ vyjádřit součtem (2.2).

Systém $B \equiv \{b_0(t),...,b_N(t)\}$ nazvěme *bázovým systémem* v M, jestliže je nezávislý a úplný na M. Pokud lze takový systém sestrojit pro konečné N, pak říkáme, že M má dimenzi $N+1$.

Pomocí bázového systému lze nahradit libovolnou funkci z M váženým součtem (2.2), čehož se často využívá v teoretických úvahách i v praktických aplikacích. V mnoha praktických případech je např. reprezentace funkce pomocí vektoru bázových koeficientů výrazně úspornější než reprezentace pomocí hodnot v čase. Pro soustavy funkcí je dále často použito zkrácené označení $\{b_j\}_i^k \equiv \{b_i,...,b_k\}$.

Tvrzení: Je-li $\{b_i\}_0^N$ báze v prostoru M a $\{h_i\}_0^N$ systém funkcí takový, že platí

$$\begin{pmatrix} h_0 \\ h_1 \\ \cdots \\ h_N \end{pmatrix} = \mathbf{H} \begin{pmatrix} b_0 \\ b_1 \\ \cdots \\ b_N \end{pmatrix} \tag{2.4}$$

kde $\mathbf{H}$ je regulární matice rozměru $(N+1, N+1)$, je systém $\{h_i\}_0^N$ báze v prostoru M.

Důkaz: Vyjádřeme v systému $\{b_i\}$ součet

$$f(t) = \sum_{i=0}^{N} a_i h_i(t) \ . \tag{2.5}$$

Dosadíme-li (2.4):

$$f(t) = (a_0,...,a_N) \begin{pmatrix} h_0 \\ \cdots \\ h_N \end{pmatrix} = (\tilde{a}_0,...,\tilde{a}_N) \begin{pmatrix} b_0 \\ \cdots \\ b_N \end{pmatrix} \tag{2.6}$$

kde $(\tilde{a}_0,...,\tilde{a}_N) = (a_0,...,a_N)\mathbf{H}$ je nějaký vektor koeficientů. To znamená, že pro každý vektor koeficientů $(a_0,...,a_N)$ lze nalézt vektor $(\tilde{a}_0,...,\tilde{a}_N)$ a naopak,

$$(a_0,...,a_N) = (\tilde{a}_0,...,\tilde{a}_N)\mathbf{H}^{-1} \tag{2.7}$$

neboť $\mathbf{H}^{-1}$ existuje. Odtud okamžitě vyplývá, že každý prvek $f \in M$ lze vyjádřit jako součet (2.5), protože systém $\{b_i\}$ je bázový. Nezávislost vyplývá přímo z (2.6). Je-li levá strana (2.6) rovna nule, musí totéž platit i pro pravou stranu. To je však možné jedině, pokud $(\tilde{a}_0,...,\tilde{a}_N)^T = \mathbf{0}$. Z (2.7) pak plyne, že $(a_0,...,a_N)^T = \mathbf{0}$. ♦

Tvrzení: Jsou-li $\{b_i\}_0^N$ a $\{h_i\}_0^N$ dva bázové systémy v prostoru M, existuje regulární matice $\mathbf{H}$ taková, že platí (2.4).

Důkaz: Platí

$$\begin{pmatrix} h_0 \\ \cdots \\ h_N \end{pmatrix} = \mathbf{H}_1 \begin{pmatrix} b_0 \\ \cdots \\ b_N \end{pmatrix} \tag{2.8}$$

$$\begin{pmatrix} b_0 \\ \cdots \\ b_N \end{pmatrix} = \mathbf{H}_2 \begin{pmatrix} h_0 \\ \cdots \\ h_N \end{pmatrix} \tag{2.9}$$

kde $\mathbf{H}_1$, $\mathbf{H}_2$ jsou nějaké čtvercové matice. Po dosazení (2.9) do (2.8) dostaneme

$$\begin{pmatrix} h_0 \\ \cdots \\ h_N \end{pmatrix} = \mathbf{H}_1 \mathbf{H}_2 \begin{pmatrix} h_0 \\ \cdots \\ h_N \end{pmatrix} \tag{2.10}$$

Jelikož $\{h_i\}_0^N$ je bázový systém, je vyjádření každé funkce jednoznačné. Musí tedy platit

$$\mathbf{H}_1 \mathbf{H}_2 = \mathbf{1}. \tag{2.11}$$

Dosadíme-li naopak (2.8) do (2.9), dostáváme obdobně

$$\mathbf{H}_2 \mathbf{H}_1 = \mathbf{1}. \tag{2.12}$$

To je však možné pouze, jestliže $\mathbf{H}_1$ je regulární a $\mathbf{H}_2 = \mathbf{H}_1^{-1}$. ♦

Tvrzení: Libovolný systém N nezávislých funkcí $\{h_0,...,h_N\}$, $h_i \in M$, tvoří v prostoru M konečné dimenze $N+1$ bázi.

Důkaz: Vyjádřeme funkce $\{h_i\}$ v systému $\{h_i\}$.

$$\begin{pmatrix} h_0 \\ h_1 \\ \cdots \\ h_N \end{pmatrix} = \mathbf{H} \begin{pmatrix} b_0 \\ b_1 \\ \cdots \\ b_N \end{pmatrix} \tag{2.13}$$

Stačí dokázat, že matice $\mathbf{H}$ je regulární. Vynásobme (2.13) zleva vektorem $(a_0,...,a_N)$. Z nezávislosti vyplývá, že levá strana (2.13) je nulová jedině, jestliže $(a_0,...,a_N) = \mathbf{0}$. Pravá strana je nulová jedině, jestliže

$$\mathbf{H}^T \begin{pmatrix} a_0 \\ ... \\ a_N \end{pmatrix} = \mathbf{0}. \tag{2.14}$$

Jestliže matice $\mathbf{H}$ není regulární, pak soustava (2.14) má nenulové řešení, což je spor. ♦

2.2. Aproximace funkce

Označme $L_2(0,t_f)$ prostor všech funkcí integrovatelných v Lebesgueově smyslu s kvadrátem na intervalu $\left[0,t_f\right]$, tj. takových, že platí

$$\int_0^{t_f} f(t)^2\,dt < \infty\,. \tag{2.15}$$

V prostoru $L_2(0,t_f)$ je definován skalární součin vztahem

$$\langle f \mid g \rangle = \int_0^{t_f} f(t)g(t)\,dt \tag{2.16}$$

a norma je zavedena pomocí skalárního součinu:

$$\|f\|_2 = \sqrt{\langle f \mid f \rangle}\,. \tag{2.17}$$

Prostor $L_2(0,t_f)$ se zpravidla využívá pro práci s se spojitými a po částech spojitými omezenými funkcemi.

Bází v nekonečném prostoru $L_2(0,t_f)$ [6] většinou rozumíme posloupnost nezávislých funkcí $\{b_i\}_0^\infty \in L_2$, která je *úplná*, tj. ke každé funkci $f \in L_2$ a každému $\varepsilon > 0$ lze nalézt k a reálné koeficienty $\{a_i\}_0^k$ takové, že platí $\left\| f - \sum_{i=0}^{k} a_i b_i \right\|_2 < \varepsilon$.

V souvislosti s aproximací funkce je třeba zmínit speciální třídu ortogonálních bázových systémů, třebaže v této práci mají tyto bázové systémy pouze okrajový význam.

Ortogonálním systémem funkcí nazýváme takový systém B v $L_2(0,t_f)$, kde pro každé dvě funkce platí:

$$\langle b_i \mid b_j \rangle \begin{cases} \neq 0 \text{ pro } i = j \\ = 0 \text{ pro } i \neq j \end{cases} \tag{2.18}$$

Normovaným systémem funkcí nazýváme takový systém B, že pro každou funkci $b_i \in B$ platí:

$$\|b_i\|_2 = 1\,. \tag{2.19}$$

Ortonornálním systémem se nazývá systém ortogonální a současně normovaný.

Lze snadno dokázat, že každý ortogonální systém je nezávislý a tvoří tedy bázi v nějakém prostoru M. Ortogonální systémy mají pro aproximaci funkce některé podstatné výhody. Zejména pro danou funkci f lze snadno určit bázové (tzv. Fourierovy) koeficienty a_i, neboť platí

$$\langle f \mid b_i \rangle = \sum_{j=0}^{N} a_j \langle b_j \mid b_i \rangle = a_i \|b_i\|_2^2\,. \tag{2.20}$$

Tyto koeficienty pak rovněž dávají nejlepší aproximaci libovolné funkce $f \in L_2$ konečným součtem (2.2) ve smyslu minimální kvadratické odchylky

$$\left\| f - \sum_{i=0}^{N} a_i b_i \right\|_2^2 . \tag{2.21}$$

Klasickými příklady ortonormálních bází v $L_2(0,\pi)$ jsou funkce

$$\left\{ \sqrt{\frac{2}{\pi}} \sin(t), \sqrt{\frac{2}{\pi}} \sin(2t), \ldots \right\} \tag{2.22}$$

$$\left\{ \frac{1}{\sqrt{\pi}}, \sqrt{\frac{2}{\pi}} \cos(t), \sqrt{\frac{2}{\pi}} \cos(2t), \ldots \right\} . \tag{2.23}$$

Chceme-li aproximovat průběh libovolné funkce z $L_2(0,t_f)$ konečným součtem (2.2), jsou možné dva přístupy:

a) Zvolit (zpravidla ortonormální) bázi B v $L_2(0,t_f)$ a pro nahrazení funkce se omezit pouze na konečný počet bázových funkcí z B. Tím je vlastně vytvořen konečněrozměrný prostor M. Tento postup je teoreticky podložen skutečností, že posloupnost Fourierových koeficientů v nekonečné ortonormální bázi konverguje k nule, neboť:

$$\|f\|_2^2 = \left\langle \sum_{i=0}^{\infty} a_i b_i \,|\, \sum_{i=0}^{\infty} a_i b_i \right\rangle = \sum_{i=0}^{\infty} a_i^2 . \tag{2.24}$$

Aby byl součet (2.24) konečný, musí posloupnost $\{a_i\}_0^{\infty}$ konvergovat k nule.

b) Zvolit konečněrozměrný prostor funkcí $M \in L_2$ se známými vlastnostmi, které odpovídají praktickým požadavkům, a v něm sestrojit bázi. Tento přístup je v této práci dále využíván.

Je známo [6], že systém funkcí

$$\{t^i\}_0^{\infty} \tag{2.25}$$

tvoří bázi v $L_2(0,t_f)$. Omezením se na prvních $N+1$ prvků dostáváme systém

$$\{1, t, \ldots, t^N\} . \tag{2.26}$$

který je současně bází v prostoru $P_N(0,t_f)$ všech polynomů do stupně N na intervalu $[0,t_f]$.

Pro praktické využití bázového systému (2.26) je zpravidla nutné provést transformaci času do intervalu $[0,1]$, neboť v opačném případě mohou být příliš velké rozdíly mezi hodnotami bázových prvků a tedy i bázových koeficientů, což má negativní vliv na většinu numerických algoritmů.

Systém (2.26) není ortogonální. Příkladem polynomiálního bázového systému, který je současně ortogonální v intervalu $[-1,1]$, je posloupnost Legendrových polynomů [14]. V numerické matematice však mají větší význam Čebyševovy polynomy, které jsou na intervalu $[-1,1]$ definovány vztahem

$$T_n(t) = \cos(n \arccos t) . \tag{2.27}$$

Čebyševovy polynomy jsou ortogonální s váhou $\frac{1}{\sqrt{1-t^2}}$, tj. platí

$$\int_{-1}^{1} \frac{T_m(t) T_n(t)}{\sqrt{1-t^2}} dt = \begin{cases} 0 & \text{pro } m \neq n \\ \neq 0 & \text{pro } m = n \end{cases} \tag{2.28}$$

a jejich koeficienty lze určit z rekurentního vztahu

$$\begin{aligned} &T_{n+1}(t) - 2t T_n(t) + T_{n-1}(t) = 0 \\ &T_0(t) = 1,\ T_1(t) = t . \end{aligned} \tag{2.29}$$

Transformací

$$T_n^*(t) = T_n(2t-1) \tag{2.30}$$

dostáváme systém Čebyševových polynomů na intervalu $[0,1]$.

Je známým faktem [14], že rozvoj funkce v bázi $\{T_n\}$ při stejném součtu koeficientů konverguje rychleji než Taylorův rozvoj. Rovněž platí, že ze všech polynomů řádu n, které mají počáteční koeficient roven 1, má polynom $\frac{T_n(t)}{2^{n-1}}$ na intervalu $[-1,1]$ nejmenší rozsah hodnot. Z těchto důvodů jsou Čebyševovy polynomy považovány za jeden z nejvhodnějších bázových systémů pro aproximaci hladkých funkcí.

Je zbytečné na tomto místě uvádět další známá fakta z teorie ortogonálních polynomů, neboť pro nahrazení funkce řízení a trajektorie byly především využity prostory spline-funkcí, popsané dále. Pro porovnání byla v poslední době prakticky realizován výpočet optimální trajektorie s využitím bázových systémů (2.23), (2.26) a (2.30). Poměrně uspokojivých výsledků bylo však dosaženo pouze u nejjednoduššího polynomiálního systému (2.26). I v tomto případě však jsou výsledky podstatně horší než u bázových systémů typu spline, zejména z následujících hledisek:

- doba výpočtu
- kvalita získaného řešení
- maximální použitelný počet bázových prvků

2.3. Bázové systémy s generující funkcí

V dalším textu mají zvláštní význam následující třídy bázových systémů.

Systémem s generující funkcí c nazvěme systém funkcí $B \equiv \{b_0(t), ..., b_N(t)\}$ tvaru

$$b_i = c(t - iT) \tag{2.31}$$

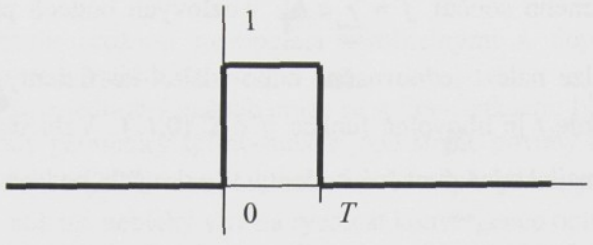
kde $c(t)$ je generující funkce a T parametr.

Finitní funkci v L_2 definujme jako funkci, která je rovna nule mimo uzavřený interval Ω. Interval Ω nazvěme *nosičem*.

Finitním systémem nazvěme systém generovaný finitní funkcí.

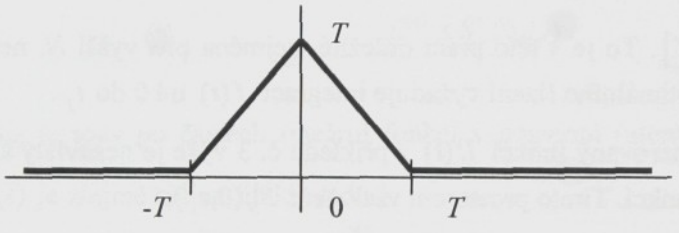
Následuje několik příkladů generujících finitních funkcí:

1. $c(t)=1$ pro $t\in[0,T)$, jinak je $c(t)=0$ (Obr. 2.1). Tento systém je ortogonální bází v prostoru funkcí konstantních v intervalech $t\in[iT,(i+1)T)$, $i=0,...,N$.



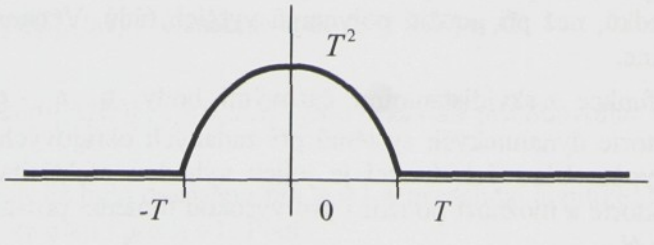
Obr. 2.1: Generující funkce z příkladu 1

2. $c(t)=t+T$ pro $t\in[-T,0)$, $c(t)=T-t$ pro $t\in[0,T)$, jinak je $c(t)=0$ (Obr. 2.2). Tuto generující funkci označme $\Lambda^1(t)$.



Obr. 2.2: Funkce $\Lambda^1(t)$

3. $c(t)=\left[(nT+t)(nT-t)\right]^n$ pro $t\in[-nT,nT]$, jinak $c(t)=0$ (viz Obr. 2.3 pro případ $n=1$). Tuto generující funkci označme $L^n(t)$. Funkce $L^n(t)\in C_{n-1}$, tj. je $n-1$ krát spojitě diferencovatelná.



Obr. 2.3: Funkce $L^1(t)$

V této práci jsou dále využívány zejména systémy generované funkcí $\Lambda^1(t)$.

Pro jednoduchost označme dále $\Lambda^1(t-iT)\equiv\Lambda^1_i(t)$.

Označme $S_N^1(0,t_f)$ prostor spojitých funkcí v $\left[0,t_f\right]$, které jsou po částech lineární v intervalech $t\in\left[iT,(i+1)T\right)$.

Tvrzení: Systém generovaný funkcí $\Lambda^1(t)$ je bází v prostoru $S_N^1(0,t_f)$.

Důkaz: Tvrzení je zřejmé. Funkce $\Lambda_i^1\in S_N^1(0,t_f)$ a jsou nezávislé, neboť v bodě iT je nenulová pouze funkce Λ_i^1. Pro hodnoty váženého součtu $\tilde{f}=\sum_{i=0}^{N}a_i\Lambda_i^1$ v uzlových bodech platí $\tilde{f}_i=a_i.T$. Naopak, pro libovolné hodnoty $\tilde{f}_i$ lze nalézt jednoznačně odpovídající koeficienty a_i ze vztahu $a_i=\tilde{f}_i/T$. Nechť dále $\tilde{f}_i=f(iT)$, kde f je libovolná funkce $f\in S_N^1(0,t_f)$. Vzhledem k tomu, že dvě funkce $\tilde{f},f\in S_N^1(0,t_f)$, které mají stejné funkční hodnoty v uzlových bodech iT musí být stejné, je též $f=\sum_{i=0}^{N}a_i\Lambda_i^1$. ♦

Báze $\left\{\Lambda_j^1(t)\right\}_0^N$ není ortogonální, avšak pro účely aproximace funkce je stále ještě poměrně výhodná, neboť soustavy rovnic pro výpočet Fourierových koeficientů jsou třídiagonální, tedy snadno řešitelné. Na druhé straně, tato báze umožňuje velmi efektivní výpočet hodnot funkce $f(t)=\sum_{i=0}^{N}a_i\Lambda_i^1$, neboť předchozí součet se redukuje na lineární kombinaci hodnot $f(iT)$ a $f((i+1)T)$, kde $t\in\left[iT,(i+1)T\right]$. To je v této práci důležité, zejména pro vyšší N, neboť výpočet hodnoty kritéria pro výpočet optimálního řízení vyžaduje integraci $f(t)$ od 0 do t_f.

Obdobně např. systém generovaný funkcí $L^1(t)$ z příkladu č. 3 výše je nezávislý a je tedy bází v nějakém prostoru spojitých funkcí. Tímto prostorem však není $S_N^1(0,t_f)$.

2.4. Prostory spline-funkcí

Spline-funkcí se zpravidla rozumí funkce, jejíž k-tá derivace je konstantní v intervalech $\left[t_i,t_{i+1}\right)$ a která splňuje podmínky spojitosti derivací řádu $0,\ldots,k-1$ v uzlových bodech. Spline-funkce se kromě aproximace používají s výhodou i pro interpolaci danými body, kdy je v mnohých případech dosaženo výrazně lepších výsledků, než při použití polynomů vyšších řádů. Většinou se používá spline 3. řádu – tzv. kubická spline.

V této práci jsou spline-funkce s ekvidistantními časovými body, tj. $t_{i+1}-t_i=T=konst.$, využívány pro nahrazení trajektorie dynamických systémů při zadaných okrajových podmínkách. Ukazuje se, že oproti jiným typům bázových funkcí je jejich výhodou efektivita vyhodnocení integrálního kritéria podél trajektorie a možnost použití i pro vysokou dimenzi prostoru parametrů, která odpovídá dělení trajektorie N.

V případě zadaného počátečního bodu a derivací v tomto bodě do řádu $n-1$ je spline-funkce určena hodnotami n-tých derivací v intervalech $\left[t_i,t_{i+1}\right)$, tj. pro kubickou spline

$$y(t)\equiv y(y_0,\dot{y}_0,\ddot{y}_0,y_0^{(3)},y_1^{(3)},\ldots,y_{N-1}^{(3)};t) \tag{2.32}$$

kde $y_i^{(k)}\equiv y^{(k)}(iT)$.

V případě zadaných okrajových podmínek jsou některé z parametrů $(y_0, \dot{y}_0, \ddot{y}_0, y_0^{(3)}, y_1^{(3)}, ..., y_{N-1}^{(3)})$ dopočítány na základě těchto podmínek a hodnot ostatních parametrů. Tento způsob vyjádření, kdy jako dopočítávané byly zvoleny kinematické veličiny na konci trajektorie, autor prakticky ověřil již ve své diplomové práci. V zprávě [1], která byla pro ni podkladem, byly pro splnění koncových podmínek použity, dle názoru autora nevhodně, polynomy vyšších řádů na konci trajektorie (tzv. spline typu 3-4-5).

V rámci disertační práce však byl navržen a prakticky realizován elegantnější aparát pro parametrizaci spline-funkcí, který je popsán dále a má zejména následující výhody:

- umožňuje pružnou manipulaci s volitelnými a dopočítávanými parametry pro splnění okrajových podmínek
- umožňuje jednotný způsob práce se spline-funkcemi libovolného řádu
- všechny parametry spline-funkce jsou stejné povahy a tedy i přibližně stejné velikosti – u konstrukce (2.32) tomu tak není, neboť např. hodnoty y_0 se od $y_{N-1}^{(n)}$ mohou lišit o několik řádů, což má neblahý vliv na rychlost konvergence optimalizačních algoritmů.
- sjednocuje způsob práce se spline-funkcí s jinými způsoby aproximace trajektorie

V podstatě je možné tento postup charakterizovat jako definování lineárního prostoru spline-funkcí v daném intervalu a konstrukce báze v tomto prostoru.

Spline-funkcí f řádu $n+1$ označme funkci, která je n-krát spojitě diferencovatelná, tj. $f \in C^n$, a pro jejíž n-tou derivaci platí:

$$f^{(n)} \in S_N^1(0,t_f). \tag{2.33}$$

n-tá derivace je tedy po částech lineární funkcí s uzlovými (zlomovými) body iT, kde $T = \frac{t_f}{N}$.

Funkce $\Lambda^1(t)$ je zřejmě spline-funkcí řádu 1.

Prostor všech spline-funkcí řádu n na intervalu $\left[0,t_f\right]$ označme $S_N^n(0,t_f)$.

Označme dále

$$\Lambda^n(t) = \int_{-\infty}^{t} \Lambda^{n-1}(t)\,dt \;,\; n>1. \tag{2.34}$$

Zřejmě $\Lambda^n(t) \in S_N^n(0,t_f)$. Označme opět $\Lambda^n(t-iT) \equiv \Lambda_i^n$.

Tvrzení: Funkce $\left\{\Lambda_i^n(t)\right\}$, $i = 0,...,N$, jsou nezávislé pro libovolné $N>0$.

Důkaz: Tvrzení již bylo dokázáno pro $n=1$ v předchozí kapitole. Předpokládejme dále, že platí pro n a dokažme, že platí i pro $n+1$. Platí

$$\sum_{i=0}^{N} a_i \Lambda_i^{n+1}\,dt = \sum_{i=0}^{N} a_i \int_{-\infty}^{t} \Lambda_i^n\,dt = \int_{-\infty}^{t} \sum_{i=0}^{N} a_i \Lambda_i^n\,dt. \tag{2.35}$$

Pravá strana (2.35) je rovna nule buď jestliže je integrand identicky roven nule, což je možné na základě předpokladů jedině jestliže $a_0 = ... = a_N = 0$, nebo jestliže pro každé t existuje čas $\tau \in \left[0,t\right)$ takový, že platí

$$\int_{-\infty}^{\tau}\sum_{i=0}^{N} a_i\Lambda_i^n\,dt = -\int_{\tau}^{t}\sum_{i=0}^{N} a_i\Lambda_i^n\,dt \neq 0\,. \tag{2.36}$$

Avšak platí-li $\sum_{i=0}^{N} a_i\Lambda_i^{n+1}\,dt = 0$ pro všechna t, musí platit i pro libovolné $\tau \in [0,t)$, tedy $\int_{-\infty}^{\tau}\sum_{i=0}^{N} a_i\Lambda_i^n\,dt = 0$, což je spor s (2.36). ♦

Tvrzení: Funkce $\{\Lambda_i^{n+1}(t)\}$, kde $i=-n,\ldots,N$, tvoří bázi v $S_N^{n+1}(0,t_f)$.

Důkaz: Tvrzení již bylo dokázáno pro $n=0$. Předpokládejme, že platí pro $n-1$ a dokažme, že platí i pro n. Stačí dokázat, že libovolnou funkci $f \in S_N^{n+1}(0,t_f)$ lze v intervalu $[0,t_f]$ vyjádřit jako nějakou kombinaci $f = \sum_{i=-n}^{N} a_i\Lambda_i^{n+1}\,dt$, neboť nezávislost již byla dokázána. Jelikož derivace $\dot{f} \in S_N^n(0,t_f)$, lze ji vyjádřit v $[0,t_f]$ jako $\dot{f} = \sum_{i=-n+1}^{N} a_i\Lambda_i^n\,dt$. Po integraci

$$f(t) = f_{-n} + \int_{-nT}^{t}\sum_{i=-n+1}^{N} a_i\Lambda_i^n\,dt = f_{-n} + \int_{-\infty}^{t}\sum_{i=-n+1}^{N} a_i\Lambda_i^n\,dt = f_{-n} + \sum_{i=-n+1}^{N} a_i\Lambda_i^{n+1} \tag{2.37}$$

kde $f_{-n} = f(-nT)$, neboť hodnoty funkcí ze systému $\{\Lambda_i^n(t)\}_{-n+1}^{N}$ jsou nulové pro $t \le -nT$. Předchozí vztah lze vyjádřit jako

$$f(t) = \sum_{i=-n}^{N} a_i'\Lambda_i^{n+1} \tag{2.38}$$

kde koeficient a_{-n}' lze jednoznačně určit tak, aby $f_{-n} = a_{-n}'\Lambda_{-n}^{n+1}(-nT)$. Ostatní koeficienty a_i', $i > -n$ lze pak nalézt zderivováním předchozího vztahu a přemístěním prvního členu sumy na levou stranu

$$\dot{f}(t) - a_{-n}'\Lambda_{-n}^n = \sum_{i=-n+1}^{N} a_i'\Lambda_i^n \tag{2.39}$$

na základě indukčního předpokladu, protože funkce $\dot{f}(t) - a_{-n}'\Lambda_{-n}^n \in S_N^n(t_f)$ a $\{\Lambda_i^n(t)\}_{-n+1}^{N}$ je báze v $S_N^n(t_f)$. Z nezávislosti funkcí $\{\Lambda_i^{n+1}(t)\}_{-n}^{N}$ pak vyplývá, že vyjádření (2.38) je jediné. ♦

Pozn.: Prvních $(n+1)$ bázových koeficientů v systému $\{\Lambda_i^{n+1}(t)\}_{-n}^{N}$ zřejmě nahrazuje parametry $y_0,\ldots,y_0^{(n)}$ klasické spline-funkce (2.32).

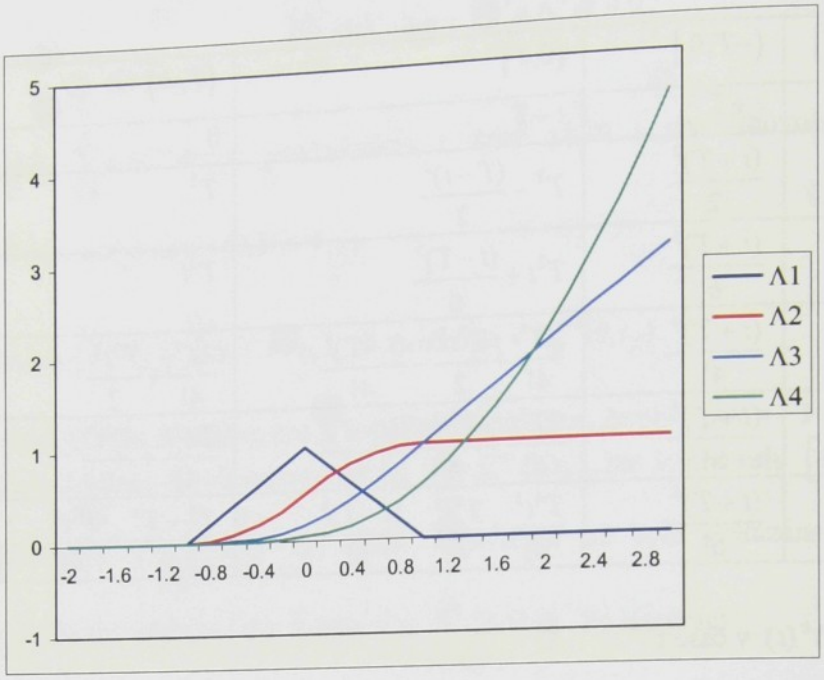
Tab. 2.1 uvádí vztahy pro výpočet funkcí $\Lambda^i(t)$, $i=1,\ldots,6$, získané s využitím rekurzivních vztahů (2.34). Tab. 2.2 udává hodnoty funkcí $\Lambda^i(t)$ v bodech $t=iT$. Obr. 2.4 znázorňuje průběhy funkcí $\Lambda^k(t)$, $k=1,\ldots,4$.

T	$(-\infty,-T]$	$(-T,0]$	$(0,T]$	(T,∞)
$\Lambda^1(t)$	0	$t+T$	$t-T$	0
$\Lambda^2(t)$	0	$\frac{(t+T)^2}{2}$	$T^2-\frac{(T-t)^2}{2}$	T^2
$\Lambda^3(t)$	0	$\frac{(t+T)^3}{6}$	$T^2t+\frac{(t-T)^3}{6}$	T^2t
$\Lambda^4(t)$	0	$\frac{(t+T)^4}{4!}$	$\frac{2T^4}{4!}+\frac{T^2t^2}{2}-\frac{(t-T)^4}{4!}$	$\frac{2T^2}{4!}+\frac{T^2t^2}{2}$
$\Lambda^5(t)$	0	$\frac{(t+T)^5}{5!}$	$\frac{2T^4t}{4!}+\frac{T^2t^3}{6}+\frac{(t-T)^5}{5!}$	$\frac{2T^2t}{4!}+\frac{T^2t^3}{6}$
$\Lambda^6(t)$	0	$\frac{(t+T)^6}{6!}$	$\frac{T^4t^2}{4!}+\frac{T^2t^4}{4!}+\frac{(t-T)^6}{6!}$	$\frac{T^6-T^4}{4!}+\frac{T^2t^2}{4!}+\frac{T^2t^4}{4!}$

Tab. 2.1: Funkce $\Lambda^k(t)$ v čase t

i	-1	0	≥ 1
Λ_i^1	0	T	0
Λ_i^2	0	$\frac{T^2}{2}$	T^2
Λ_3^i	0	$\frac{T^3}{6}$	iT^3
Λ_i^4	0	$\frac{T^4}{4!}$	$\frac{2T^2}{4!}+i^2\frac{T^4}{2}$
Λ_i^5	0	$\frac{T^5}{5!}$	$i\frac{2T^3}{4!}+i^3\frac{T^5}{6}$
Λ_i^6	0	$\frac{T^6}{6!}$	$\frac{T^6-T^4}{4!}+i^2\frac{T^4}{4!}+i^4\frac{T^6}{4!}$

Tab. 2.2: Funkce $\Lambda^k(t)$ v uzlových bodech



Obr. 2.4: průběhy funkcí $\Lambda^k(t)$

Na závěr si všimněme hodnot funkce Λ_i^j v uzlových bodech pro $i \geq 1$ (viz Tab. 2.2). Obecně lze konstatovat, že pro $i \geq 1$ platí

$$\begin{aligned} \Lambda_i^1 &= 0 \\ \Lambda_i^2 &= c_{20} \\ \Lambda_i^3 &= c_{30} + c_{31} i \\ \Lambda_i^4 &= c_{40} + c_{41} i + c_{42} i^2 \\ &\cdots \\ \Lambda_i^k &= c_{k0} + c_{k1} i + \ldots + c_{k,k-2} i^{k-2} \end{aligned} \tag{2.40}$$

kde c_{rs} jsou konstanty, přičemž $c_{rs} \neq 0$ pro $s = r - 2$.

Na základě vztahů (2.40) lze dokázat tvrzení, které je několikrát využito dále v textu:

Tvrzení: Nechť jsou dána různá celá čísla $k_i > 0$, $i = 1,...,n$. Potom matice

$$\mathbf{D} = \begin{pmatrix} \Lambda^{n+1}(k_1 T), \cdots, \Lambda^{n+1}(k_n T) \\ \cdots \\ \Lambda^2(k_1 T), \cdots, \Lambda^2(k_n T) \end{pmatrix} \tag{2.41}$$

je regulární.

Důkaz: S využitím vztahů (2.40) má matice (2.41) tvar

$$\mathbf{D}=\begin{pmatrix} \sum_{j=0}^{n-1} c_{n+1,j} h_1^j, & \sum_{j=0}^{n-1} c_{n+1,j} h_2^j, & \cdots, & \sum_{j=0}^{n-1} c_{n+1,j} h_n^j \\ & & \cdots & \\ c_{30}+c_{31}h_1, & c_{30}+c_{31}h_2, & \cdots, & c_{30}+c_{31}h_n \\ c_{20}, & c_{20} & , \cdots, & c_{20} \end{pmatrix} \tag{2.42}$$

kde $h_1,...,h_n \neq 0$ jsou různá čísla a c_{ij} jsou konstanty, $c_{j+2,j} \neq 0$ pro $j=0,...,n-1$. Dokažme nejprve, že **D** je ekvivalentní matici

$$\mathbf{D}'=\begin{pmatrix} c_{n+1,n-1} h_1^{n-1}, & c_{n+1,n-1} h_2^{n-1}, \ldots, & c_{n+1,n-1} h_n^{n-1} \\ & \cdots & \\ c_{31}h_1 & c_{31}h_2 \quad \cdots & c_{31}h_n \\ c_{20} & c_{20} \quad \ldots & c_{20} \end{pmatrix}. \tag{2.43}$$

Pro $n=1$ tvrzení platí. Předpokládejme, že platí pro $n-1$ a dokažme, že pak platí i n. Matice (2.42) je pak ekvivalentní matici

$$\begin{pmatrix} \sum_{j=0}^{n-1} c_{n+1,j} h_1^j, & \sum_{j=0}^{n-1} c_{n+1,j} h_2^j, & \cdots, & \sum_{j=0}^{n-1} c_{n+1,j} h_n^j \\ c_{n,n-2} h_1^{n-2} & c_{n,n-2} h_2^{n-2} & \cdots & c_{n,n-2} h_n^{n-2} \\ & & \cdots & \\ c_{20} & c_{20} & \cdots & c_{20} \end{pmatrix}. \tag{2.44}$$

neboť tvrzení musí platit pro jakoukoliv čtvercovou matici $(n-1, n-1)$ vybranou z spodních $n-1$ řádků matice **D**. Odečteme-li od horního řádku součet spodních řádků násobených $\frac{c_{n+1,j}}{c_{j+2,j}}$, tj.

$$a_{1i} \leftarrow a_{1i} - \sum_{j=0}^{n-2} \frac{c_{n+1,j}}{c_{j+2,j}} c_{j+2,j} h_i^j = c_{n+1,n-1} h_i^{n-1} \tag{2.45}$$

dostaneme (2.43).

Determinanty matic **D** a **D′** jsou díky ekvivalenci shodné. Obrátíme-li pořadí řádků i sloupců matice **D′** a vytkneme-li konstanty $c_{20},...,c_{n+1,n-1}$, je

$$\det \mathbf{D}' = c_{20}c_{31}...c_{n+1,n-1} V(h_n,...,h_1) \tag{2.46}$$

kde

$$V(\lambda_1,\ldots,\lambda_n)=\begin{vmatrix} 1, & 1, & \ldots, & 1 \\ \lambda_1, & \lambda_2, & \ldots, & \lambda_n \\ & & \cdots & \\ \lambda_1^{n-1}, & \lambda_2^{n-1}, & \ldots, & \lambda_n^{n-1} \end{vmatrix} \tag{2.47}$$

je Vandermondův determinant [14], pro který platí

$$V(\lambda_1,\cdots\lambda_n)=(\lambda_n-\lambda_1)(\lambda_{n-1}-\lambda_1)(\lambda_2-\lambda_1)\cdots(\lambda_n-\lambda_{n-1})=\prod_{i>j}(\lambda_i-\lambda_j)\,. \tag{2.48}$$

$\det \mathbf{D}'$ tedy musí být nenulový, neboť čísla h_i jsou různá a nenulová. Proto matice (2.41) je regulární. ♦

Pozn.: Vzorec (2.48) lze odvodit odečtením λ_1-násobku i-tého řádku od $i+1$-tého řádku pro $i=1,...,n-1$. Dostaneme

$$V(\lambda_1,\cdots\lambda_n)=\begin{vmatrix} 1, & 1, & \cdots, & 1 \\ 0, & \lambda_2-\lambda_1, & \cdots, & \lambda_n-\lambda_1 \\ & & \cdots & \\ 0, & (\lambda_2-\lambda_1)\lambda_2^{n-2}, & \cdots, & (\lambda_n-\lambda_1)\lambda_n^{n-2} \end{vmatrix}=\begin{vmatrix} \lambda_2-\lambda_1, & \cdots, & \lambda_n-\lambda_1 \\ & \cdots & \\ (\lambda_2-\lambda_1)\lambda_2^{n-2}, & \cdots, & (\lambda_n-\lambda_1)\lambda_n^{n-2} \end{vmatrix} \tag{2.49}$$

Vytkneme-li členy $(\lambda_i-\lambda_j)$ před determinant

$$V(\lambda_1,\cdots\lambda_n)=(\lambda_2-\lambda_1)\cdots(\lambda_n-\lambda_1)\begin{vmatrix} 1 & \cdots & 1 \\ & \cdots & \\ \lambda_2^{n-2} & \cdots & \lambda_n^{n-2} \end{vmatrix}=(\lambda_2-\lambda_1)\cdots(\lambda_n-\lambda_1)\;V(\lambda_2,\cdots\lambda_n). \tag{2.50}$$

Rekurzivním postupem získáme (2.48).

2.5. Finitní spline-báze

V následujících částech práce je ukázáno, že finitní báze mohou být určitým způsobem využity pro zefektivnění výpočtu optimální trajektorie. Tyto postupy autor považuje za jeden z vrcholů své dosavadní práce.

Původní motivací pro definování finitních spline-funkcí, které spojují vlastnosti finitních funkcí a spline-funkcí z minulých kapitol, však bylo oddělit problém splnění okrajových podmínek spline-funkce a nalezení optimálního průběhu. Tato konstrukce, která byla popsána jako alternativní způsob výpočtu již v disertační práci, se však nakonec ukázala jako poměrně nevýhodná.

Definujme funkci Σ^p, $p\geq 1$ následovně :

$$\begin{aligned} &\Sigma^p \in S_N^p(0,t_f) \\ &(\Sigma^p)^{(i)}(t)=0 \text{ pro } t\leq -T \text{ a } t\geq kT,\; i=0,...,(p-1) \end{aligned} \tag{2.51}$$

tj. všechny derivace funkce Σ^p do řádu $(p-1)$ jsou nulové mimo interval $(-T,kT)$, kde $k>0$ je nějaké přirozené číslo. Vyjádřeme derivace funkce Σ^p jako vážený součet funkcí Λ_j^p:

$$(\Sigma^p)^{(i)}(t)=\sum_{j=0}^{k-1}\alpha_j\Lambda^{p-i}(t-jT) \tag{2.52}$$

kde $i=0,...,(p-1)$, α_j jsou nějaké koeficienty. Podmínka $(\Sigma^p)^{(i)}(t)=0$ pro $t\leq -T$ je splněna tím, že součet (2.52) jde od 0 a všechny funkce Λ^i jsou nulové pro $t\leq -T$.

Tvrzení. Platí-li $(\Sigma^p)^{(i)}(kT)=0$, $i=0,...,p-1$, platí totéž zároveň pro všechna $t>kT$.

Důkaz: Jelikož v (2.52) chybí bázové prvky pro $j=k,...,N$, funkce $(\Sigma^p)^{(p-1)}(t)$ musí být nulová pro $t>kT$. Jelikož všechny nižší derivace v bodě kT jsou dle předpokladu nulové, integrací získáme, že pro libovolné $t>kT$ je rovna nule i každá funkce $(\Sigma^p)^{(i)}(t)$, $i=0,...,p-2$. ♦

Dosadíme-li druhou z podmínek (2.51) pro $t=kT$ do soustavy rovnic (2.52), dostáváme

$$\begin{aligned}\Sigma^p(kT)&=\sum_{j=0}^{k-1}\alpha_j\Lambda^p((k-j)T)=0\\&\cdots\\(\Sigma^p)^{(p-1)}(kT)&=\sum_{j=0}^{k-1}\alpha_j\Lambda^1((k-j)T)=0\end{aligned}\qquad(2.53)$$

neboli

$$\begin{pmatrix}\Lambda^p(kT),&\cdots,\Lambda^p(T)\\&\cdots\\\Lambda^1(kT),&\cdots,\Lambda^1(T)\end{pmatrix}\begin{pmatrix}\alpha_0\\\cdots\\\alpha_{k-1}\end{pmatrix}=\mathbf{0}.\qquad(2.54)$$

Poslední řádek lze vypustit, neboť všechny jeho prvky jsou rovny 0. Matice soustavy má v případě $k=p-1$ tvar (2.41) a je tedy regulární. Aby homogenní soustava (2.54) byla řešitelná (kromě nulového řešení), musí tedy být $k\geq p$. Zvolme $k=p$ a hodnotu koeficientu $\alpha_0=1$. Soustava (2.54) pak přechází s využitím postupu (2.42) - (2.45) na nehomogenní soustavu

$$\begin{pmatrix}(k-1)^{k-2},&\cdots,1^{k-2}\\&\cdots\\(k-1),&\cdots,1\\1,&\ldots,1\end{pmatrix}\begin{pmatrix}\alpha_1\\\alpha_2\\\cdots\\\alpha_{k-1}\end{pmatrix}=-\begin{pmatrix}k^{k-2}\\\cdots\\k\\1\end{pmatrix}.\qquad(2.55)$$

Determinant matice soustavy (2.55) je po obrácení pořadí řádků i sloupců Vandermondův determinant (2.47) $V(1,\ldots,k-1)\neq 0$. Řešení lze nalézt pomocí Cramerova pravidla

$$\alpha_j=\frac{D_j}{V(1,...,k-1)}\qquad(2.56)$$

kde $j=1,...,(k-1)$ a

$$D_j=\begin{vmatrix}(k-1)^{k-2},&\ldots,(k-j+1)^{k-2},&-k^{k-2},&(k-j-1)^{k-2},&\ldots,1^{k-2}\\&&\cdots&&\\(k-1),&\ldots,(k-j+1),&-k,&(k-j-1),&\ldots,1\\1,&\ldots,&-1,&&\ldots,1\end{vmatrix}.\qquad(2.57)$$

Zřejmě opět platí:

$$D_j = V(1,2,...,(k-j-1),-k,(k-j+1),...,(k-1)). \tag{2.58}$$

Tento výsledek je možno dosadit do vztahu pro α_j, přičemž v součinech lze uvažovat pouze členy obsahující k (v čitateli) a $k-j$ (ve jmenovateli), protože ostatní se vykrátí:

$$\begin{aligned}\alpha_j &= -\frac{(k-1)(k-2)\ldots(k-(k-j-1))(k-1-k)\ldots(k-j+1-k)}{(k-j-1)\ldots(k-j-(k-j-1))(k-1-(k-j))\ldots(k-j+1-(k-j))} = \\ &= -\frac{(k-1)(k-2)\ldots(j+1)(-1)\ldots(-j+1)}{(k-j-1)\ldots1(j-1)\ldots1} = -(-1)^{(j-1)}\frac{(k-1)!}{j!(k-j-1)!}\end{aligned} \tag{2.59}$$

Konečně dostáváme elegantní vztah

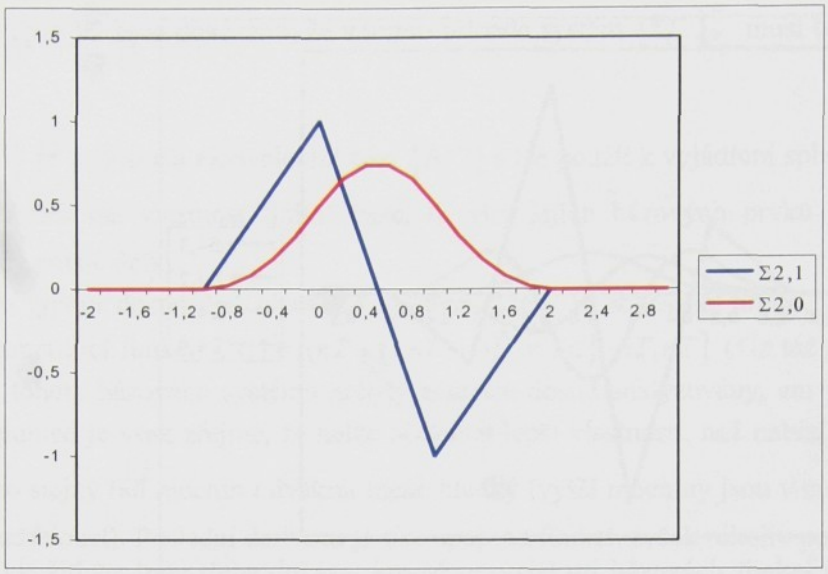
$$\alpha_j = (-1)^j C(k-1,j) \tag{2.60}$$

kde $C(k-1,j)$ je kombinační číslo. Hodnoty koeficientů α_j pro rostoucí k tvoří tedy až na znaménko Pascalův trojúhelník

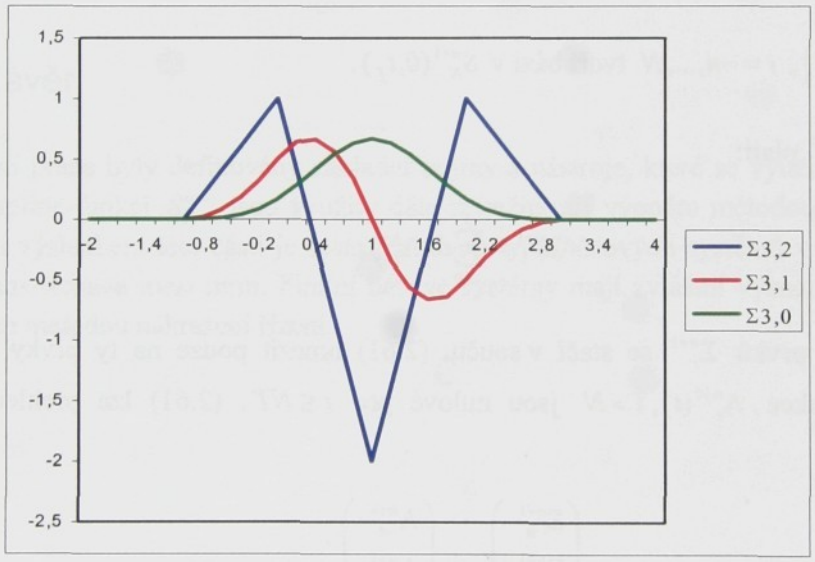
k / j	*0*	*1*	*2*	*3*	*4*	*5*	*6*
1	1						
2	1	-1					
3	1	-2	1				
4	1	-3	3	-1			
5	1	-4	6	-4	1		
6	1	-5	10	-10	5	-1	
7	1	-6	15	-20	15	-6	1

Tab. 2.3: Hodnoty koeficientů α_j pro různá k

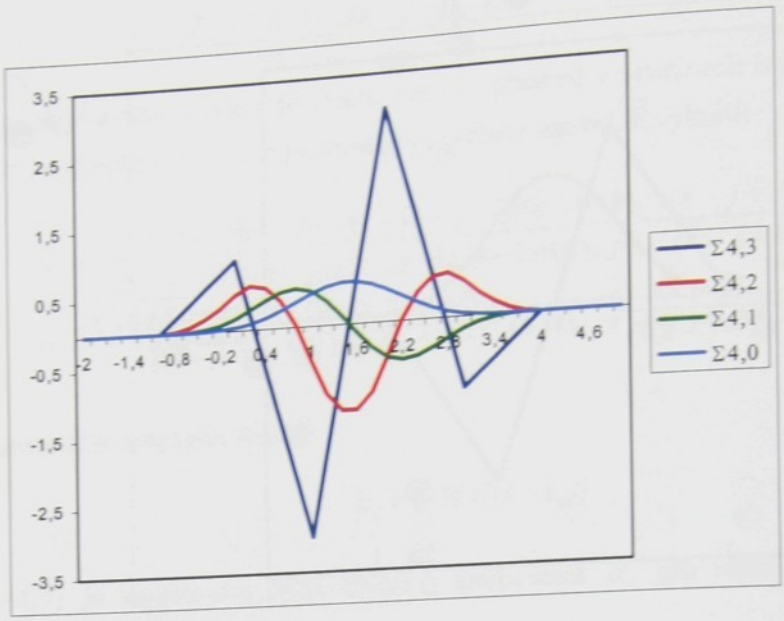
Následující obrázky znázorňují průběhy $(\Sigma^k)^{(i)}$ pro $k = 2,3,4$. i-tá derivace Σ^k je označena jako $\Sigma k,i$.



Obr. 2.5: Derivace funkce Σ^2



Obr. 2.6: Derivace funkce Σ^3



Obr. 2.7: Derivace funkce Σ^4

Obdobně jako v případě funkcí Λ^k označme $\Sigma_i^k \equiv \Sigma^k(t-iT)$.

Tvrzení: Systém $\left\{\Sigma_i^{n+1}\right\}$, $i=-n,...,N$ tvoří bázi v $S_N^{n+1}(0,t_f)$.

Důkaz: Pro prvky Σ_i^{n+1} platí:

$$\Sigma_i^{n+1} = \sum_{j=0}^{n} \alpha_j \Lambda_{i+j}^{n+1}. \tag{2.61}$$

Přitom u koncových prvků Σ_i^{n+1} se stačí v součtu (2.61) omezit pouze na ty prvky Λ_{i+j}^{n+1}, kde $i+j \le N$, neboť funkce $\Lambda_i^{n+1}(t), i>N$ jsou nulové pro $t \le NT$. (2.61) lze přehledně zapsat v maticovém tvaru

$$\begin{pmatrix} \Sigma_{-n}^{n+1} \\ \Sigma_{-n+1}^{n+1} \\ \cdots \\ \Sigma_N^{n+1} \end{pmatrix} = \mathbf{H} \begin{pmatrix} \Lambda_{-n}^{n+1} \\ \Lambda_{-n+1}^{n+1} \\ \cdots \\ \Lambda_N^{n+1} \end{pmatrix} \tag{2.62}$$

kde matice $\mathbf{H}$ má tvar

$$\mathbf{H} = \begin{pmatrix} \alpha_0, ..., \alpha_{k-1}, 0, ..., 0 \\ 0, \alpha_0, ..., \alpha_{k-1}, 0, ..., 0 \\ \cdots \\ 0, \quad \cdots \quad 0, \alpha_0 \end{pmatrix}. \tag{2.63}$$

Čtvercová matice **H** je zřejmě horní trojúhelníková a prvky na diagonále jsou jednotkové, je tedy regulární. Výše již bylo dokázáno, že v tomto případě systém $\left\{\Sigma_i^{n+1}\right\}_{-n}^{N}$ musí být báze v $S_N^{n+1}(0,t_f)$.

♦

Báze $\left\{\Sigma_i^{n+1}\right\}$ je tedy zcela ekvivalentní bázi $\left\{\Lambda_i^{n+1}\right\}$ a lze použít k vyjádření spline-funkce v intervalu $\left[0,t_f\right]$. Přitom má vlastnost finitní báze, tj. vliv jejích bázových prvků je zcela omezen na subintervaly pevné délky.

Jiným typem dostatečně hladkého systému, který je současně finitní, je např. již zmíněný systém s generující funkcí $L^n(t)=\left[(nT+t)(nT-t)\right]^n$ v $t\in\left[-nT,nT\right]$ (viz též příklad 3 z kap. 2.3). Vlastnosti tohoto bázového systému nebyly autorem dosud analyzovány, ani prakticky ověřovány. Na první pohled je však zřejmé, že nelze očekávat lepší vlastnosti, než nabízí systém $\left\{\Sigma_i^n\right\}$, neboť $\left\{L_i^n\right\}$ je pro stejný řád mocnin t dvakrát méně hladký (vyšší mocniny jsou v numerických metodách většinou nežádoucí). Poslední derivace je sice spojitou funkcí, avšak nikoliv po částech lineární. Pro praktické využití by bylo třeba dokázat kromě nezávislosti bázových funkcí i řešitelnost soustavy okrajových podmínek – pro systémy $\left\{\Lambda_i^n\right\}$ a $\left\{\Sigma_i^n\right\}$ bude tento problém analyzován později. Systém $\left\{L_i^n\right\}$ rovněž nenabízí řadu jiných užitečných vlastností, které jsou využívány u $\left\{\Sigma_i^n\right\}$, jako je např. efektivní transformace do systému $\left\{\Lambda_i^n\right\}$ a zpět.

2.6. Závěr

V této části práce byly definovány základní pojmy a nástroje, které se využívají v dalších částech. Prostory spline-funkcí S_N^{n+1} jsou využity dále zejména při výpočtu metodou nahrazení trajektorie. Důležitým výsledkem této části je sestrojení dvou typů bázových systémů v prostoru spline-funkcí S_N^{n+1} a transformace mezi nimi. Finitní bázové systémy mají zvláštní význam rovněž v souvislosti s výpočtem metodou nahrazení řízení.

Část 3.

Výpočet metodou nahrazení trajektorie

V této části práce je popsán aparát určený především pro efektivní výpočet optimálních trajektorií určité skupiny mechanických soustav. Popsané výsledky lze však využít i pro mnohem širší třídu dynamických systémů zavedením dodatečných omezení.

Nejprve je vymezena třída systémů, pro které jsou tyto postupy určeny a jsou uvedeny některé její vlastnosti. Dále je formulována úloha určení optimální trajektorie a je popsána metoda numerického výpočtu, která využívá charakteristických vlastností těchto systémů a speciálního tvaru okrajových podmínek. Zvlášť jsou popsána možná rozšíření pro úlohy s volným časem. Pro úplnost je zmíněn i možný způsob určení parametrů regulační smyčky pro zpětnovazební řízení podél vypočtené trajektorie, který využívá vlastností zmíněné třídy dynamických systémů. Na závěr jsou demonstrovány různé aspekty popsané metody na několika praktických příkladech.

Dále popsané výsledky vychází zejména z disertační práci autora, v mnoha směrech ji však překonávají. Novými myšlenkami, které jsou hlavním přínosem pro celkové urychlení výpočtu, je ortogonální rozklad soustavy okrajových podmínek a metoda urychleného výpočtu gradientu kritéria s využitím finitní spline-báze. Praktické úlohy ukazují, že popsané postupy jsou velmi účinné při použití kvalitních algoritmů optimalizace, kterým je věnována část 6 práce.

V případě, že v prostoru parametrů existuje více lokálních extrémů, které mohou vzniknout např. zavedením dodatečných omezení nebo nelineárních okrajových podmínek, je nutné využít algoritmů globální optimalizace, které jsou většinou časově mnohem náročnější a přitom teoreticky nezaručují nalezení globálního extrému v konečném čase. To je rovněž případ problému výpočtu optimálních trajektorií v prostoru s překážkami, který je diskutován v samostatné části práce.

3.1. Systémy regulární vzhledem k řízení

Postupy popsané v dalších kapitolách jsou určeny především pro mechanické systémy, které lze vyjádřit ve tvaru

$$\mathbf{H}(\mathbf{y})\,\mathbf{y}^{(n)} + \mathbf{T}(\mathbf{y}^{(n-1)},\ldots,\mathbf{y},t) = \mathbf{u}\,. \tag{3.1}$$

Vektorové funkce **y** a **u** jsou řádu m, **H** je regulární matice závislá na **y** a **T** je vektorová funkce spojitě diferencovatelná podle všech argumentů. Typickým příkladem jsou mechanické systémy s tuhými členy, jako průmyslové roboty, kdy na každou nezávislou osu působí samostatná akční síla.

Tvar (3.1) lze v tomto případě okamžitě získat z Lagrangeových rovnic:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{y}}} - \frac{\partial L}{\partial \mathbf{y}} = \mathbf{u} \tag{3.2}$$

kde $L = K - P$ je rozdíl kinetické a potenciální energie, **y** vektor zobecněných souřadnic a **u** vektor nezávislých působících sil, popř. momentů v jednotlivých osách. Přitom se předpokládá, že systém zobecněných souřadnic $(y_1,\ldots,y_m)$ je zvolen tak, aby byl [16]:

a) *úplný*, tj. libovolná pozice systému lze popsat pomocí souřadnic $(y_1,...,y_m)$ a současně každé dvě různé pozice systému jsou definovány odlišnými vektory souřadnic
b) *nezávislý*, tj. při zvolení libovolné podmnožiny souřadnic z $(y_1,...,y_m)$ a upevnění v těchto souřadnicích se systém při změně zbývajících souřadnic volně pohybuje

Souřadnice obvykle přímo odpovídají relativnímu pootočení, popř. posuvům jednotlivých členů vůči sobě. Lze dokázat [16], že v případě splnění podmínek a) a b) lze kinetickou energii soustavy vyjádřit ve tvaru

$$K = \frac{1}{2}\dot{\mathbf{y}}^T \mathbf{H}(\mathbf{y})\dot{\mathbf{y}} \tag{3.3}$$

kde matice setrvačnosti $\mathbf{H}(\mathbf{y})$ je symetrická a pozitivně definitní, tedy regulární.

Rovnice (3.2) pak přechází do podoby

$$\mathbf{H}(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{T}(\dot{\mathbf{y}},\mathbf{y}) = \mathbf{u}\,. \tag{3.4}$$

V případě, že některé hnací síly chybí, má soustava (3.4) tvar

$$\begin{aligned} \mathbf{H}_1(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{T}_1(\dot{\mathbf{y}},\mathbf{y}) &= \mathbf{u} \\ \mathbf{H}_2(\mathbf{y})\ddot{\mathbf{y}} + \mathbf{T}_2(\dot{\mathbf{y}},\mathbf{y}) &= \mathbf{0} \end{aligned} \tag{3.5}$$

který nelze převést do tvaru (3.1).

Pro přesnější vymezení třídy dynamických systémů, pro které lze aplikovat dále popsaný postup, uvažujme systém definovaný soustavou rovnic

$$\mathbf{g}(\mathbf{y}^{(n)},\mathbf{y}^{(n-1)},...,\mathbf{y},\mathbf{u},t) = \mathbf{0} \tag{3.6}$$

kde $\mathbf{y}$ a $\mathbf{u}$ jsou vektory (stejné) dimenze m.

Počáteční podmínky problému jsou

$$\mathbf{y}^{(n-1)}(0) = \mathbf{y}_0^{(n-1)},\ \ldots\ ,\mathbf{y}(0) = \mathbf{y}_0\,. \tag{3.7}$$

O funkcích $\mathbf{y}^{(n)}(.),\mathbf{u}(.)$ předpokládáme, že jsou spojité, tj. $\mathbf{y} \in C_n^m$, $\mathbf{u} \in C^m$ a funkce $\mathbf{g}(.)$ spojitě diferencovatelná podle všech argumentů.

Nazvěme systém (3.6) *regulárním vzhledem k řízení*, jestliže

a) Rovnice (3.6) má jediné řešení pro každé $\mathbf{u}(.) \in C^m$.
b) Pro každou n-krát spojitě diferencovatelnou funkci $\mathbf{y}(.)$, která vyhovuje počátečním podmínkám (3.7), při současném splnění rovnice (3.6) v každém bodě t, existuje právě jedna funkce $\mathbf{u}(.) \in C^m$.

Z podmínek a), b) vyplývá, že každému průběhu řízení $\mathbf{u}(.)$ odpovídá jediná trajektorie $\mathbf{y}(.)$ a naopak, každému průběhu $\mathbf{y}(.)$ odpovídá právě jediný průběh $\mathbf{u}(.)$. Mezi množinou řídicích funkcí a odpovídajících trajektorií tedy existuje bijektivní zobrazení.

Vyjádřeme diferenciální přírůstek funkce $\mathbf{g}$ v čase t pro $dt \to 0$, který musí být na základě (3.6) nulový:

$$\begin{aligned} d\mathbf{g} &= \frac{\partial \mathbf{g}}{\partial \mathbf{y}} d\mathbf{y} + \ldots + \frac{\partial \mathbf{g}}{\partial \mathbf{y}^{(n)}} d\mathbf{y}^{(n)} + \frac{\partial \mathbf{g}}{\partial t} dt + \frac{\partial \mathbf{g}}{\partial \mathbf{u}} d\mathbf{u} = \\ &= \left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}} \dot{\mathbf{y}} + \ldots + \frac{\partial \mathbf{g}}{\partial \mathbf{y}^{(n-1)}} \mathbf{y}^{(n)} + \frac{\partial \mathbf{g}}{\partial t} \right] dt + \frac{\partial \mathbf{g}}{\partial \mathbf{y}^{(n)}} d\mathbf{y}^{(n)} + \frac{\partial \mathbf{g}}{\partial \mathbf{u}} d\mathbf{u} = \mathbf{0}. \end{aligned} \tag{3.8}$$

Aby byla splněna podmínka b), rovnice (3.8) musí mít řešení (a to jediné) vzhledem k $d\mathbf{u}$ pro libovolné hodnoty členů na levé straně výrazu. To je splněno jedině pokud

$$\det\left(\frac{\partial \mathbf{g}}{\partial \mathbf{u}} \right) \neq 0. \tag{3.9}$$

Obdobně, aby rovnice (3.8) měla řešení vzhledem k přírůstku $d\mathbf{y}^{(n)}$ v bodě na trajektorii daném hodnotami $\mathbf{y},\ldots,\mathbf{y}^{(n)}$, pro dané $d\mathbf{u} \neq 0$ a $dt \neq 0$, musí platit

$$\det\left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}^{(n)}} \right) \neq 0. \tag{3.10}$$

Naopak, jestliže je splněna podmínka (3.9) v každém bodě, je důsledkem věty o implicitní funkci [10], že funkce řízení lze v určitém malém okolí bodu $(\mathbf{y}^{(n)}(t),\ldots,\mathbf{y}(t),\mathbf{u}(t),t)$ vyjádřit jako

$$\mathbf{u} = \mathbf{f}(\mathbf{y}^{(n)}, \mathbf{y}^{(n-1)}, \ldots, \mathbf{y}, t) \tag{3.11}$$

kde $\mathbf{f}$ je nějaká funkce spojitě diferencovatelná podle všech argumentů a tedy platí podmínka b). Platí-li (3.10), je možno stejným postupem separovat nejvyšší derivaci:

$$\mathbf{y}^{(n)} = \mathbf{h}(\mathbf{y}^{(n-1)}, \ldots, \mathbf{y}, \mathbf{u}, t). \tag{3.12}$$

Označme $\mathbf{x} = \left[\mathbf{y}^{(n-1)}, \ldots, \mathbf{y} \right]^T$. Pak lze soustavu (3.12) s počátečními podmínkami (3.7) snadno přepsat do standardního tvaru

$$\dot{\mathbf{x}} = \mathbf{G}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.13}$$

Rovnice (3.13) má na základě vět o řešitelnosti soustav obyčejných diferenciálních rovnic, např. [9], v určitém lokálním okolí bodu $\mathbf{x}_0$ jednoznačné řešení pro zadané počáteční podmínky (3.7), tj. platí a). Podmínky (3.9) a (3.10) lze tedy v určitém lokálním okolí trajektorie považovat za ekvivalentní podmínkám a) a b) výše.

Dále popsané postupy jsou navrženy pro ty systémy regulární vzhledem k řízení, u kterých lze navíc zajistit efektivní vyhodnocení funkce $\mathbf{f}$ (3.11). Proto je zpravidla třeba, aby rovnice (3.11) byla známa v explicitním tvaru. Podmínky a) a b) toto sice nevyžadují, avšak v obecném případě by bylo nutné určovat hodnoty funkce (3.11) numericky řešením (3.6). Rovněž se předpokládá, že platí podmínka a). Explicitní znalost funkce $\mathbf{h}$ (3.12) však v tomto případě zpravidla nutná není.

Z podmínek a) a b) vyplývá, že pro každou trajektorii, která je n-krát spojitě diferencovatelná a prochází počátečním bodem, existuje odpovídající průběh řízení. Okamžitým důsledkem je, že systém regulární vzhledem k řízení musí být řiditelný, neboť pro libovolný koncový bod $\mathbf{x}_f$ lze vždy nalézt dostatečně hladkou trajektorii takovou, že $\mathbf{x}(0) = \mathbf{x}_0$, $\mathbf{x}(t_f) = \mathbf{x}_f$.

3.2. Formulace problému

Nechť je dán systém regulární vzhledem k řízení přímo ve tvaru

$$\mathbf{u} = \mathbf{f}(\mathbf{y}^{(n)}, \mathbf{y}^{(n-1)}, ..., \mathbf{y}, t) \tag{3.14}$$

s okrajovými podmínkami

$$\mathbf{y}^{(n-1)}(0) = \mathbf{y}_0^{(n-1)}, \ \ldots \ , \mathbf{y}(0) = \mathbf{y}_0 \tag{3.15}$$

$$\mathbf{y}^{(n-1)}(t_f) = \mathbf{y}_f^{(n-1)}, \ \ldots \ , \mathbf{y}(t_f) = \mathbf{y}_f \,. \tag{3.16}$$

Přitom libovolný počet okrajových podmínek v (3.15), (3.16) může chybět.

Úlohou je nalézt trajektorii $\mathbf{y} \in C_n^m$, tj. která je n-krát spojitě diferencovatelná v každé složce, takovou, že funkcionál

$$J = \varphi(\mathbf{y}^{(n)}(t_f), ..., \mathbf{y}(t_f)) + \int_0^{t_f} f_0(\mathbf{y}^{(n)}, ..., \mathbf{y}, \mathbf{u}, t)\, dt \tag{3.17}$$

nabývá svého minima na přípustné množině trajektorií a příslušných funkcí řízení. Tato množina je zadána např. soustavou nerovností a rovností v každém bodě $t \in \left[0, t_f\right]$

$$\mathbf{g}(\mathbf{y}^{(n)}, ..., \mathbf{y}, \mathbf{u}, t) \geq \mathbf{0} \tag{3.18}$$

$$\mathbf{h}(\mathbf{y}^{(n)}, ..., \mathbf{y}, \mathbf{u}, t) = \mathbf{0} \tag{3.19}$$

kde znaménko „$\geq$“ v (3.18) znamená, že nerovnost platí pro každou složku. Přitom předpokládáme, že funkce f_0, φ, $\mathbf{g}$ a $\mathbf{h}$ jsou spojitě diferencovatelné vzhledem ke všem svým argumentům a $0 < t_f < \infty$ je známé.

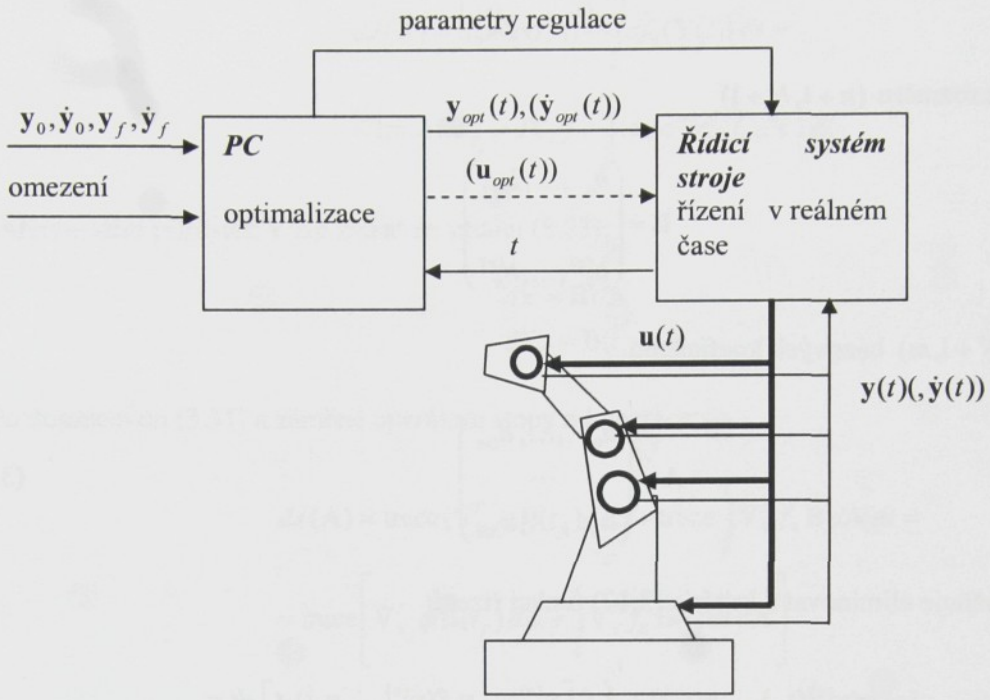
Nejběžnější omezení řídicí veličiny bývá tvaru $u_{i\min} \leq u_i(t) \leq u_{i\max}$. Omezení průběhu zobecněných souřadnic jsou zpravidla složitější. Například při návrhu trajektorie robota je často třeba uvažovat omezení rozsahu polohy jednotlivých členů, které je konstantní, ale i jejich vzájemnou polohu – části robota navzájem nesmí přijít do kolize.

Pomocí omezení typu rovností lze metodu použít i pro systémy ve tvaru (3.5), které nejsou regulární vzhledem k řízení. Je možné zavést vektor pomocných sil $\mathbf{v}$ a dodatečný omezující požadavek $\mathbf{v}(t) = \mathbf{0}$, $t \in \left[0, t_f\right]$:

$$\begin{aligned} \mathbf{H}_1\ddot{\mathbf{y}} + \mathbf{T}_1(\dot{\mathbf{y}}, \mathbf{y}) &= \mathbf{u} \\ \mathbf{H}_2\ddot{\mathbf{y}} + \mathbf{T}_2(\dot{\mathbf{y}}, \mathbf{y}) &= \mathbf{v} \\ \mathbf{v} &= \mathbf{0} \end{aligned} \tag{3.20}$$

Jelikož se hledá trajektorie $\mathbf{y} \in C_n^m$, je příslušná funkce řízení nutně spojitou funkcí. V tom je tento přístup méně obecný než klasické metody. Optimální funkce řízení je totiž pouze po částech spojitá. Ve skutečnosti se však v tomto případě hledá optimální trajektorie, a odpovídající průběh řízení je pouze podružným výsledkem, který pro realizaci systému řízení může nebo nemusí být využit. Jelikož optimální trajektorie v klasickém smyslu je spojitou funkcí, lze každou složku s libovolnou přesností aproximovat v normě $\|h(.)\| = \max_{t \in \left[0, t_f\right]} \{h(t)\}$ nějakou funkcí z množiny C_∞, a tedy i C_n

(každou spojitou funkci na $\left[0, t_f\right]$ lze např. dle Weierstrassovy věty [10] s libovolnou přesností aproximovat polynomem). Předpokládá se, že skutečné hodnoty řízení generuje řídicí systém na základě odchylky od optimální trajektorie $\mathbf{y}_{opt}$ (Obr. 3.1).



Obr. 3.1: Schéma systému řízení

3.3. Náhrada trajektorie pomocí bázových funkcí

Zvolme vhodný konečněrozměrný prostor n-krát spojitě diferencovatelných funkcí M. Nutnou podmínkou je, aby v tomto prostoru existovalo m funkcí $y_i \in M$ takových, že vektor $\mathbf{y}(t) = \left(y_1(t), ..., y_m(t)\right)^T$ vyhovuje okrajovým podmínkám (3.15) a (3.16).

Sestrojme v tomto prostoru bázi $\{b_i\}_0^N$ a vyjádřeme trajektorii jako vážený součet prvků báze, tj.

$$y_i(t) = \sum_{j=0}^{N} a_{ij} b_j(t) . \tag{3.21}$$

Pro derivace trajektorie řádu $1, ..., n$ platí:

$$y_i^{(k)}(t) = \sum_{j=0}^{N} a_{ij} b_j^{(k)}(t) . \tag{3.22}$$

Předchozí rovnosti lze psát v maticovém tvaru

$$\mathbf{Y}(t) = \mathbf{B}(t)\,\mathbf{A} \tag{3.23}$$

kde **Y** je matice $(n+1,m)$ derivací trajektorie

$$\mathbf{Y} = \begin{pmatrix} y_1 & ,\ldots, & y_m \\ & \cdots & \\ y_1^{(n)} & ,\ldots, & y_m^{(n)} \end{pmatrix} \tag{3.24}$$

B je matice báze rozměru $(n+1,N+1)$

$$\mathbf{B} = \begin{pmatrix} b_0 & ,\ldots, & b_N \\ & \cdots & \\ b_0^{(n)} & ,\ldots, & b_N^{(n)} \end{pmatrix} \tag{3.25}$$

a **A** je matice $(N+1,m)$ bázových koeficientů

$$\mathbf{A} = \begin{pmatrix} a_{01} & ,\ldots, & a_{0m} \\ & \cdots & \\ a_{N1} & ,\ldots, & a_{Nm} \end{pmatrix}. \tag{3.26}$$

Tvar (3.14) umožňuje eliminovat z kritéria (3.17) funkci řízení:

$$\begin{aligned} J &= \varphi(\mathbf{y}^{(n)}(t_f),...,\mathbf{y}(t_f)) + \int_0^{t_f} f_0\left[\mathbf{y}^{(n)},...,\mathbf{y},\mathbf{f}(\mathbf{y}^{(n)},...,\mathbf{y},t),t\right] dt \equiv \\ &\equiv \varphi(\mathbf{Y}(t_f)) + \int_0^{t_f} \tilde{f}_0(\mathbf{Y}(t))\, dt \to \min. \end{aligned} \tag{3.27}$$

Dosazením (3.23) dostáváme kritérium jako funkci koeficientů $\mathbf{A} \equiv (a_{ij})$:

$$J(\mathbf{A}) = \varphi(\mathbf{Y}(t_f)) + \int_0^{t_f} \tilde{f}_0(\mathbf{Y}(t))\, dt = \varphi(\mathbf{B}(t_f)\mathbf{A}) + \int_0^{t_f} \tilde{f}_0(\mathbf{B}(t)\mathbf{A})\, dt\,. \tag{3.28}$$

Pak je úloha hledání optimální trajektorie totožná s úlohou hledání složek matice **A** takových, že funkce $J(\mathbf{A})$ nabývá minima na množině dané okrajovými podmínkami (3.15), (3.16) a omezeními (3.18). Pro nalezení optima je nutno využít numerických metod.

Pro zjednodušení práce s funkcemi závislými na matici parametrů, které se v této části textu často vyskytují, definujme gradient skalární funkce J podle matice $\mathbf{A}(N+1,m)$ následovně:

$$\nabla_{\mathbf{A}} J = \begin{pmatrix} \dfrac{\partial J}{\partial a_{01}} & \cdots & \dfrac{\partial J}{\partial a_{0m}} \\ & \cdots & \\ \dfrac{\partial J}{\partial a_{N1}} & \cdots & \dfrac{\partial J}{\partial a_{Nm}} \end{pmatrix}. \tag{3.29}$$

Diferenciální přírůstek funkce J pak lze vyjádřit jako

$$dJ(\mathbf{A}) = \sum_{i=0}^{N}\sum_{j=1}^{m}\frac{\partial J}{\partial a_{ij}}da_{ij} = \sum_{j=1}^{m}\sum_{i=0}^{N}\frac{\partial J}{\partial a_{ij}}da_{ij} = \mathrm{trace}(\nabla_{\mathbf{A}}^{T} J\, d\mathbf{A}). \tag{3.30}$$

Dosadíme-li do (3.28):

$$\begin{aligned} dJ(\mathbf{A}) &= d\varphi(\mathbf{Y}(t_f)) + \int_0^{t_f} d\tilde{f}_0(\mathbf{Y}(t))\,dt = \\ &= \mathrm{trace}(\nabla_{\mathbf{Y}_f}^{T}\varphi\, d\mathbf{Y}_f) + \int_0^{t_f}\mathrm{trace}(\nabla_{\mathbf{Y}}^{T}\tilde{f}_0\, d\mathbf{Y})\,dt. \end{aligned} \tag{3.31}$$

Diferenciální přírůstek $\mathbf{Y}$ lze získat ze vztahu (3.23):

$$\begin{aligned} d\mathbf{Y} &= \mathbf{B}\,d\mathbf{A} \\ d\mathbf{Y}_f &= \mathbf{B}(t_f)\,d\mathbf{A}. \end{aligned} \tag{3.32}$$

Po dosazení do (3.31) a záměně operátoru stopy a integrace:

$$\begin{aligned} dJ(\mathbf{A}) &= \mathrm{trace}(\nabla_{\mathbf{Y}_f}^{T}\varphi\,\mathbf{B}(t_f)d\mathbf{A}) + \mathrm{trace}\int_0^{t_f}\nabla_{\mathbf{Y}}^{T}\tilde{f}_0\,\mathbf{B}\,d\mathbf{A}\,dt = \\ &= \mathrm{trace}\left[\nabla_{\mathbf{Y}_f}^{T}\varphi\,\mathbf{B}(t_f)\,d\mathbf{A} + \int_0^{t_f}\nabla_{\mathbf{Y}}^{T}\tilde{f}_0\,\mathbf{B}(t)\,dt\,d\mathbf{A}\right] \end{aligned} \tag{3.33}$$

tedy platí:

$$dJ(\mathbf{A}) = \mathrm{trace}(\mathbf{G}^{T}\,d\mathbf{A}) = \sum_{j=1}^{m}\sum_{i=0}^{N} g_{ij}\,da_{ij} \tag{3.34}$$

$$\mathbf{G} = \mathbf{B}^{T}(t_f)\nabla_{\mathbf{Y}_f}\varphi + \int_0^{t_f}\mathbf{B}^{T}(t)\nabla_{\mathbf{Y}}\tilde{f}_0\,dt. \tag{3.35}$$

Vzhledem k tomu, že diferenciální přírůstek je lineárně závislý na hodnotách da_{ij}, je funkce $J(\mathbf{A})$ spojitě diferencovatelná podle $\mathbf{A}$. Matice $\mathbf{G}$ je pak zřejmě gradientem funkce $J(\mathbf{A})$. Spojitá diferencovatelnost $J(\mathbf{A})$ umožňuje pro hledání optimálního průběhu použít efektivní algoritmy optimalizace.

Pozn.: Bylo by samozřejmě možné pracovat pouze s vektorovým zápisem, tj. složky matic $\mathbf{Y}$ a $\mathbf{A}$ organizovat ve vektorech. To je rovněž zápis, se kterým se obvykle pracuje v algoritmech optimalizace. Na druhé straně, v tomto případě by matice báze $\mathbf{B}$ byla značně rozsáhlá a řídká. Zápis využitý zde je kompaktní a přímo odpovídá tvaru datových struktur, které jsou použity při efektivní implementaci, proto jej autor považuje za výhodnější.

3.4. Okrajové podmínky a omezení

Okrajové podmínky (3.15), (3.16) představují pro problém (3.27) soustavu omezení typu rovností. Lineární povaha těchto omezení umožňuje jejich eliminaci a následně výrazně efektivnější výpočet než v případě nelineárních omezení.

Okrajové podmínky lze zapsat jako $2m$ soustav lineárních rovnic vzhledem k členům $\mathbf{A}$:

$$\begin{pmatrix} b_0(0), & \ldots, & b_N(0) \\ & \ldots & \\ b_0^{(n-1)}(0), & \ldots, & b_N^{(n-1)}(0) \end{pmatrix} \begin{pmatrix} a_{0i} \\ \ldots \\ a_{Ni} \end{pmatrix} = \begin{pmatrix} y_{0i} \\ \ldots \\ y_{0i}^{(n-1)} \end{pmatrix} \tag{3.36}$$

$$\begin{pmatrix} b_0(t_f), & \ldots, & b_N(t_f) \\ & \ldots & \\ b_0^{(n-1)}(t_f), & \ldots, & b_N^{(n-1)}(t_f) \end{pmatrix} \begin{pmatrix} a_{0i} \\ \ldots \\ a_{Ni} \end{pmatrix} = \begin{pmatrix} y_{fi} \\ \ldots \\ y_{fi}^{(n-1)} \end{pmatrix} \tag{3.37}$$

kde $i = 1,\ldots,m$. Přitom v soustavách mohou chybět řádky s některými indexy $j = 1,\ldots,n$, avšak současně pro všechny soustavy $i = 1,\ldots,m$. Označme n_0 počet počátečních podmínek a n_f počet koncových podmínek.

Soustavy (3.36) a (3.37) lze zapsat jako

$$\begin{aligned} \mathbf{C}_0\mathbf{A} &= \mathbf{R}_0 \\ \mathbf{C}_f\mathbf{A} &= \mathbf{R}_f \end{aligned} \tag{3.38}$$

nebo v kompaktním tvaru

$$\mathbf{C}\,\mathbf{A} = \mathbf{R} \tag{3.39}$$

kde matice $\mathbf{C}$ je sestavena z řádků matic $\mathbf{B}(0)$ a $\mathbf{B}(t_f)$ a $\mathbf{R}$ obsahuje požadované hodnoty derivací v krajních bodech.

Je hledána matice koeficientů $\mathbf{A}$ taková, že $J(\mathbf{A}) \to \min$ a současně jsou splněny podmínky (3.39). Základním postupem je rozdělení matice $\mathbf{C}$ na dvě části $\mathbf{C} = (\mathbf{C}_1, \mathbf{C}_2)$ tak, že matice $\mathbf{C}_1$ je čtvercová a regulární. Přitom se uvažuje i možné přeuspořádání sloupců v matici $\mathbf{C}$. Rozdělíme-li odpovídajícím způsobem i matici $\mathbf{A}$, přejde soustava (3.39) do tvaru

$$(\mathbf{C}_1\ \mathbf{C}_2)\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} = \mathbf{R}\,. \tag{3.40}$$

Potom

$$\mathbf{A}_1 = \mathbf{C}_1^{-1}(\mathbf{R} - \mathbf{C}_2\mathbf{A}_2)\,. \tag{3.41}$$

Hledají se pak koeficienty submatice $\mathbf{A}_2$ tak, že $J(\mathbf{A}) \to \min$. Složky $\mathbf{A}_1$ jsou přitom dopočítány z (3.41) jako součást výpočtu hodnoty kritéria. Inverzní matice $\mathbf{C}_1^{-1}$ však nemusí být určována při každém výpočtu J, neboť závisí pouze na hodnotách bázových funkcí v čase 0 a t_f, které se při výpočtu zpravidla nemění.

V případě, že některé triviální koncové podmínky jsou nahrazeny obecnější podmínkou ve tvaru

$$\boldsymbol{\psi}(\mathbf{Y}(t_f)) = \mathbf{0} \tag{3.42}$$

kde ψ je hladká vektorová funkce, je možné toto omezení zahrnout pokutovou metodou do terminální části funkcionálu (3.27):

$$\varphi(\mathbf{Y}(t_f)) = \sigma \left\| \psi(\mathbf{Y}(t_f)) \right\|_{\mathbf{K}}^2 . \tag{3.43}$$

kde $\mathbf{K}$ je diagonální matice s kladnými prvky a $\sigma > 0$ pokutová konstanta. Lze však využít i přímé metody nelineárního programování, jako např. SQP [11], které pracují které s omezeními ve tvaru (3.42) efektivnějším způsobem - viz též část 6 práce, kap. 6.2.

Nerovnosti (3.18) a rovnosti (3.19) představují nekonečně mnoho omezujících podmínek, neboť platí v každém bodě intervalu $\left[0, t_f\right]$. Při numerickém výpočtu je sice vyhodnocení funkcí $\mathbf{g}$ a $\mathbf{h}$ prováděno pouze v konečném počtu kroků, avšak tento počet zpravidla musí být značně vysoký – jinak je získána trajektorie, která výrazně překračuje omezení. Běžně je takto možné získat stovky až tisíce omezujících podmínek, což může být příliš mnoho pro efektivní praktickou realizaci.

Omezení (3.18) a (3.19) lze však snadno zahrnout pokutovou metodou. Rozšířené kritérium má tvar

$$\tilde{J} = \varphi(\mathbf{Y}_f) + \int_0^{t_f} f_0 \, dt + \sigma \int_0^{t_f} \left[\left\| \mathbf{g}_-(\mathbf{Y},\mathbf{u},t) \right\|_{\mathbf{K}_1}^2 + \left\| \mathbf{h}(\mathbf{Y},\mathbf{u},t) \right\|_{\mathbf{K}_2}^2 \right] dt \ . \tag{3.44}$$

kde $\mathbf{K}_1$ a $\mathbf{K}_2$ jsou diagonální matice s kladnými prvky, $\sigma > 0$ pokutová konstanta a $\mathbf{g}_- = (g_{i-})$ je vektor, pro jehož složky platí

$$g_{i-} = \min\{g_i, 0\} . \tag{3.45}$$

Podstatné je, že kritérium v tomto případě zůstane spojitě diferencovatelnou funkcí $\mathbf{A}$ a pro jeho minimalizaci je tedy možné použít běžné metody hledání minima bez omezení. Výpočet by měl být proveden sekvenčním způsobem se zvyšující se hodnotou σ (viz část 6 práce).

Omezení typu nerovností v každém čase lze zahrnout i bariérovou metodou, kde rozšířené kritérium nabývá nekonečně velkých hodnot mimo přípustnou oblast. Namísto členu $\sigma \left\| \mathbf{g}_-(\mathbf{Y},\mathbf{u},t) \right\|_{\mathbf{K}_1}^2$ v (3.44) je možné dosadit

$$-\frac{1}{\sigma} \sum_j k_{3j} \ln g_j(\mathbf{Y},\mathbf{u},t) . \tag{3.46}$$

Otázkou zůstává, zda je možné omezení v každém čase nahradit malým počtem omezujících podmínek. V tom případě by bylo možné aplikovat některou z přímých metod nelineárního programování a výpočet by byl pravděpodobné efektivnější než při použití pokutových, popř. bariérových členů. Uvažujme pouze omezení typu nerovností v každém bodě. Všechny výrazy

$$\gamma_{2i}^2(\mathbf{A}) = \int_0^{t_f} \left[g_{i-}(\mathbf{Y},\mathbf{u},t) \right]^2 dt = 0 \tag{3.47}$$

$$\gamma_{2i}(\mathbf{A}) = \sqrt{\int_0^{t_f} \left[g_{i-}(\mathbf{Y},\mathbf{u},t) \right]^2 dt} = 0 \tag{3.48}$$

$$\gamma_{1i}(\mathbf{A}) = \int_0^{t_f} \left| g_{i-}(\mathbf{Y},\mathbf{u},t) \right| dt = 0 \tag{3.49}$$

$$\gamma_{\infty i}(\mathbf{A}) = \min_{t \in \left[0, t_f\right]} \left\{ g_i(\mathbf{Y},\mathbf{u},t) \right\} \geq 0 \tag{3.50}$$

nahrazují omezující podmínky (3.18). Problém podmínek (3.47) je, že gradienty funkcí γ_{2i}^2 jsou v přípustné oblasti nulové. Na druhé straně, funkce γ_{2i}, γ_{1i} a $\gamma_{\infty i}$ jsou nehladké. Metody nelineárního programování však zpravidla předpokládají, že funkce omezení jsou spojitě diferencovatelné a že matice gradientů omezení má plnou hodnost. Je však možné, že lze vyvinout pro omezení v některém z tvarů (3.47) - (3.50) speciální účinné algoritmy. Tyto možnosti však dosud nebyly podrobně analyzovány.

3.5. Řešitelnost soustavy okrajových podmínek pro bázový systém $\{\Lambda_i^{n+1}\}$

Existence inverzní matice $\mathbf{C}_1^{-1}$ v soustavě (3.40) je základním požadavkem na volbu prostoru M. Zabývejme se splněním této podmínky nejprve pro prostor spline-funkcí $S_N^{n+1}(0, t_f)$, tj. prostor s bázovým systémem $\{\Lambda_i^{n+1}\}_{-n}^{N}$, který je v této práci preferován.

Je-li parametr n spline shodný s řádem systému (3.14), je n-tá derivace složek trajektorie po částech lineární funkcí. Řízení je pak spojitou funkcí s nespojitostí derivace v bodech $t = iT$, $T = t_f / N$. Je ale možné bez větších úprav zvolit n spline větší než n systému a získat tak hladší průběh. Praktické zkušenosti však naznačují, že v tomto případě je konvergence k řešení zpravidla výrazně pomalejší. V dalším textu pro jednoduchost předpokládejme, že řád spline je shodný s řádem systému. V případě použití hladšího bázového systému se pouze zvýší horní indexy v položkách matic **B**, **C** (např. ve vztahu (3.51)).

V případě, že stupeň spline je o 1 nižší, než řád systému, n-tá derivace je po částech konstantní a řízení je v uzlových bodech nespojité, což neodpovídá původním předpokladům. Přesto je možné s určitými omezeními provést rozšíření popsaného postupu i pro tento případ, neboť integrand kritéria zůstává integrovatelný. Pro dosažení kvalitních výsledků je však nutné zvolit mnohem vyšší N, což je nevýhodné z hlediska celkové náročnosti výpočtu.

Jelikož generující funkce $\Lambda^{n+1}(t)$ jsou nulové pro $t < -T$, má matice $\mathbf{C}_0$ v případě plně zadaných počátečních podmínek, tj. $n_0 = n$, tvar

$$\mathbf{C}_0 = \begin{pmatrix} \Lambda_{-n}^{n+1}(0), \cdots, \Lambda_0^{n+1}(0), 0 \cdots 0 \\ \cdots \\ \Lambda_{-n}^{2}(0), \cdots, \Lambda_0^{2}(0), 0 \cdots 0 \end{pmatrix} = \begin{pmatrix} \Lambda^{n+1}(nT), \cdots, \Lambda^{n+1}(0), 0 \cdots 0 \\ \cdots \\ \Lambda^{2}(nT), \cdots, \Lambda^{2}(0), 0 \cdots 0 \end{pmatrix}. \tag{3.51}$$

Rozdělme matici $\mathbf{C}_0$ vertikálně na tři části:

$$\mathbf{C}_0 = (\mathbf{C}_{01}, \mathbf{C}_{02}, \mathbf{0}) \tag{3.52}$$

V případě $n_0 = n$, $\mathbf{C}_{02}$ obsahuje pouze sloupcový vektor $(\Lambda^{n+1}(0), \ldots, \Lambda^{2}(0))^T$. Čtvercová matice

$$\mathbf{C}_{01} = \begin{pmatrix} \Lambda^{n+1}(nT), \cdots, \Lambda^{n+1}(T) \\ \cdots \\ \Lambda^{2}(nT), \cdots, \Lambda^{2}(T) \end{pmatrix}. \tag{3.53}$$

má tvar (2.41) a je tedy regulární. Pokud $n_0 < n$, v matici $\mathbf{C}_{01}$ oproti (3.53) chybí některé řádky a sloupce a regulární být nemusí (nebylo to dokázáno ani vyvráceno). Jelikož však matice (3.53) má

maximální hodnost n, po odstranění $n-n_0$ řádků má hodnost n_0 a je tedy možné v ní vybrat n_0 nezávislých sloupců. Po přeuspořádání matice $\mathbf{C}_0$ tak, že prvních n_0 sloupců je nezávislých, je matice $\mathbf{C}_{01}$ regulární.

Matice $\mathbf{C}_f$ má v případě $n_f = n$ tvar

$$\mathbf{C}_f = \begin{pmatrix} \Lambda_{-n}^{n+1}(NT), & \cdots, \Lambda_N^{n+1}(NT) \\ & \cdots \\ \Lambda_{-n}^{2}(NT), & \cdots, \Lambda_N^{2}(NT) \end{pmatrix} = \begin{pmatrix} \Lambda^{n+1}((N+n)T), & \cdots, \Lambda^{n+1}(0) \\ & \cdots \\ \Lambda^{2}((N+n)T), & \cdots, \Lambda^{2}(0) \end{pmatrix}. \tag{3.54}$$

Rozdělme $\mathbf{C}_f$ vertikálně na čtyři části:

$$\mathbf{C}_f = (\mathbf{C}_{f1}, \mathbf{C}_{f2}, \mathbf{C}_{f3}, \mathbf{C}_{f4}). \tag{3.55}$$

Části $\mathbf{C}_{f1}$ a $\mathbf{C}_{f2}$ odpovídají prvním blokům matice $\mathbf{C}_0$. Matice $\mathbf{C}_{f3}$ nechť je čtvercová (předpokládejme, že N je dostatečně vysoké). Pokud $n_f = n$, má $\mathbf{C}_{f3}$ tvar

$$\mathbf{C}_{f3} = \begin{pmatrix} \Lambda^{n+1}((N-1)T), & \cdots, \Lambda^{n+1}((N-n)T) \\ & \cdots \\ \Lambda^{2}((N-1)T), & \cdots, \Lambda^{2}((N-n)T) \end{pmatrix}. \tag{3.56}$$

Matice $\mathbf{C}_{f3}$ má rovněž tvar (2.41) je tedy regulární. Pokud $n_f < n$ je možné obdobně jako v předchozím případě provést přeuspořádání sloupců matice $\mathbf{C}_f$ v rozsahu indexů $n+1,\ldots,2n$ tak, že sloupce v $\mathbf{C}_{f3}$ jsou nezávislé a $\mathbf{C}_{f3}$ je regulární. Vzhledem k tvaru (3.54) je však možné nezávislé sloupce $\mathbf{C}_{f3}$ vybrat libovolně v rozsahu indexů $n+1,\ldots,N+n$, tedy např. na konci trajektorie (kromě posledního indexu). Celá soustava okrajových podmínek pak má tvar

$$\begin{pmatrix} \mathbf{C}_{01}, \mathbf{C}_{02}, & \mathbf{0}, & \mathbf{0} \\ \mathbf{C}_{f1}, \mathbf{C}_{f2}, \mathbf{C}_{f3}, \mathbf{C}_{f4} \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_f \end{pmatrix} \tag{3.57}$$

Zvolme libovolné hodnoty $\mathbf{A}_2, \mathbf{A}_4$. Z (3.57) pak lze jednoznačně určit hodnoty $\mathbf{A}_1, \mathbf{A}_3$:

$$\begin{aligned} \mathbf{A}_1 &= \mathbf{C}_{01}^{-1}(\mathbf{R}_0 - \mathbf{C}_{02}\mathbf{A}_2) \\ \mathbf{A}_3 &= \mathbf{C}_{f3}^{-1}(\mathbf{R}_f - \mathbf{C}_{f1}\mathbf{A}_1 - \mathbf{C}_{f2}\mathbf{A}_2 - \mathbf{C}_{f4}\mathbf{A}_4) \end{aligned} \tag{3.58}$$

To znamená, že existuje přeuspořádání (3.40) takové, že matice $\mathbf{C}_1^{-1}$ existuje, tedy matice celé soustavy $\mathbf{C}$ musí mít plnou hodnost $n_0 + n_f$.

Kromě toho, že dokazuje existenci $\mathbf{C}_1^{-1}$, vztah (3.58) přímo odpovídá možné praktické implementaci a nahrazuje formální zápis (3.41). Tento postup má tu výhodu, že umožňuje realizaci jednotného software pro různé typy (lineárních) okrajových podmínek a různé varianty indexů dopočítávaných a volitelných koeficientů. Praktické zkušenosti naznačují, že z hlediska celkové

efektivity je výhodnější volit dopočítávané koeficienty blíže k počátku trajektorie, než k jejímu konci, což odpovídá přímo zápisu (3.57) bez přeuspořádání sloupců.

Předpokladem této konstrukce je, že všechny submatice $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4$ v (3.57) obsahují nějaká data, což je ekvivalentní podmínce

$$N \geq n+1. \tag{3.59}$$

Hodnoty $\mathbf{A}_2$ a poslední sloupec $\mathbf{A}_4$ jsou přímo úměrné hodnotám $\mathbf{y}^{(n)}$ v koncových bodech. Je-li tedy např. dodatečným požadavkem $\mathbf{y}^{(n)}(0) = \mathbf{y}^{(n)}(t_f) = 0$, je možné $\mathbf{A}_2$ a poslední sloupec $\mathbf{A}_4$ nastavit nulový. V opačném případě jsou hodnoty $\mathbf{y}^{(n)}$ v koncových bodech ponechány jako volné, což odpovídá standardní formulaci úlohy.

Z předchozího odstavce přímo vyplývá, že jestliže uvažujeme rozšířenou soustavu okrajových podmínek, která navíc obsahuje požadované hodnoty n-té derivace v krajních bodech

$$\begin{pmatrix} \mathbf{C} \\ \mathbf{E} \end{pmatrix} \mathbf{A} \equiv \tilde{\mathbf{C}}\mathbf{A} = \begin{pmatrix} \mathbf{R} \\ \mathbf{S} \end{pmatrix} \tag{3.60}$$

kde $\mathbf{E}$ $(2,\ N+n+1)$ je matice hodnot n-tých derivací bázových prvků pro v bodě 0 a t_f a $\mathbf{S}$ $(2, m)$ je matice požadovaných hodnot v těchto bodech, má tato soustava řešení pro libovolnou stranu právě tehdy, když platí (3.59).

Jednodušší možností by bylo např. rozdělit matice $\mathbf{C}_0, \mathbf{C}_f$ na tři části, kde soustava má tvar

$$\begin{pmatrix} \mathbf{C}_{01}, \mathbf{C}_{02}, & \mathbf{0} \\ \mathbf{C}_{f1}, \mathbf{C}_{f2}, \mathbf{C}_{f3} \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_f \end{pmatrix}. \tag{3.61}$$

V tomto případě jsou $\mathbf{A}_1$, $\mathbf{A}_3$ čtvercové matice dopočítávaných parametrů a $\mathbf{A}_2$ obsahuje volitelné koeficienty. Matice $\mathbf{C}_{01}$ a $\mathbf{C}_{f3}$ pro úplné okrajové podmínky však mají jinou strukturu a nebylo dokázáno, že jsou v obecném případě regulární.

3.6. Řešitelnost soustavy okrajových podmínek pro některé další typy bázových systémů

Dokázat existenci $\mathbf{C}_1^{-1}$ pro jiné bázové systémy pro libovolné N většinou není snadné. Je-li však dokázána existence $\mathbf{C}_1^{-1}$ pro nějaký bázový systém $\{b_i\}_0^N$ v M, existuje $\mathbf{C}_1^{-1}$ i pro jakýkoliv jiný bázový systém $\{h_i\}_0^N$ v tomtéž prostoru. V části 2 bylo dokázáno, že v tomto případě existuje regulární matice $\mathbf{H}$ taková, že

$$\begin{pmatrix} h_0 \\ h_1 \\ \cdots \\ h_N \end{pmatrix} = \mathbf{H} \begin{pmatrix} b_0 \\ b_1 \\ \cdots \\ b_N \end{pmatrix}. \tag{3.62}$$

Jelikož matice $\mathbf{C}$ je sestavena z hodnot bázových funkcí v bodech 0 a t_f, hodnoty této matice v systémech $\{h_i\}$ a $\{b_i\}$ jsou vázány vztahem

$$\mathbf{C}^h = \mathbf{C}^b \mathbf{H}^T . \tag{3.63}$$

kde horní index označuje příslušnost k bázovému systému. Místo (3.41) dostáváme tedy

$$\mathbf{A}_1^h = (\mathbf{C}_1^b \mathbf{H}^T)^{-1} (\mathbf{R} - \mathbf{C}_2^b \mathbf{H}^T \mathbf{A}_2^h) \tag{3.64}$$

a inverzní matice $(\mathbf{C}_1^b \mathbf{H}^T)^{-1}$ existuje, neboť dle předpokladu existuje $(\mathbf{C}_1^b)^{-1}$. Důsledkem tohoto faktu je např. existence $\mathbf{C}_1^{-1}$ pro systém $\{\Sigma_i^{n+1}\}_{-n}^{N}$ v $S_N^{n+1}(0,t_f)$.

Zabývejme se dále existencí $\mathbf{C}_1^{-1}$ pro bázový systém $\{1,t,t^2,...,t^N\}$ v $P_N(0,1)$ (tj. v prostoru polynomů do stupně N na intervalu $[0,1]$), a tedy i pro libovolný jiný bázový systém v tomtéž prostoru. Matice $\mathbf{B}(t)$ má v tomto případě tvar

$$\mathbf{B}(t) = \begin{pmatrix} 1 & t & & \cdots & t^N \\ 0 & 1 & 2t & \cdots & Nt^{N-1} \\ 0 & 0 & 2 & \cdots & N(N-1)t^{N-2} \\ & & & \cdots & \\ 0 & \ldots & 0 & n! \ \ldots & [N\ldots(N-n+1)]t^{N-n} \end{pmatrix} \tag{3.65}$$

Předpokládejme úplné okrajové podmínky. Rozdělme matici $\mathbf{C}_0$ na tři části $\mathbf{C}_0 = (\mathbf{C}_{01}, \mathbf{C}_{02}, \mathbf{C}_{03})$, kde matice $\mathbf{C}_{01}$ a $\mathbf{C}_{02}$ jsou čtvercové, a stejně tak i matici $\mathbf{C}_f = (\mathbf{C}_{f1}, \mathbf{C}_{f2}, \mathbf{C}_{f3})$. Matice $\mathbf{C}_{01}, \mathbf{C}_{02}, \mathbf{C}_{f1}, \mathbf{C}_{f2}$ mají tvar:

$$\mathbf{C}_{01} = \operatorname{diag}(1,1,2,...,(n-1)!), \quad \mathbf{C}_{02} = \mathbf{C}_{03} = \mathbf{0} \tag{3.66}$$

$$\mathbf{C}_{f1} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & n-1 \\ & & \cdots & \\ 0 & & \cdots \ 0 & (n-1)! \end{pmatrix} \tag{3.67}$$

$$\mathbf{C}_{f2} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ n & n+1 & \cdots & 2n-1 \\ & & \cdots & \\ n! & \dfrac{(n+1)!}{2} & \cdots & \dfrac{(2n-1)!}{n!} \end{pmatrix}. \tag{3.68}$$

Celá soustava (3.38) pak má tvar

$$\begin{pmatrix} \mathbf{C}_{01} & \mathbf{0} & \mathbf{0} \\ \mathbf{C}_{f1} & \mathbf{C}_{f2}, & \mathbf{C}_{f3} \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_0 \\ \mathbf{R}_f \end{pmatrix}. \tag{3.69}$$

Zvolme libovolné hodnoty $\mathbf{A}_3$. Z (3.69) pak lze jednoznačně určit hodnoty $\mathbf{A}_1, \mathbf{A}_2$:

$$\begin{aligned} \mathbf{A}_1 &= \mathbf{C}_{01}^{-1}\mathbf{R}_0 \\ \mathbf{A}_2 &= \mathbf{C}_{f2}^{-1}(\mathbf{R}_f - \mathbf{C}_{f1}\mathbf{C}_{01}^{-1}\mathbf{R}_0 - \mathbf{C}_{f3}\mathbf{A}_3). \end{aligned} \tag{3.70}$$

Matice $\mathbf{C}_{01}^{-1}$ zřejmě existuje. Existenci $\mathbf{C}_{f2}^{-1}$ se bohužel nepodařilo obecně matematicky dokázat. Není však obtížné ověřit nenulovost determinantu $\mathbf{C}_{f2}$ s využitím počítače pro hodnoty $n = 2,...,20$, což zcela přesahuje rámec praktického využití.

V případě $t_f \neq 1$ se předpokládá transformace času do intervalu $[0,1]$, neboť v opačném případě mohou být hodnoty bázových prvků v t_f pro vyšší N buďto příliš velké nebo naopak příliš malé, což může způsobit numerické problémy. Transformaci času lze provést substitucí $\frac{t}{t_f}$ namísto t v (3.65). V tomto případě je však třeba násobit mocninami faktoru $\frac{1}{t_f}$ derivace bázových funkcí. Dokonalejší způsob je založen na transformaci celé úlohy (3.27) do intervalu $[0,1]$. Tento postup je podrobně popsán v kap. 3.11.

Na závěr uvažujme ortonormální systém

$$\left\{ \frac{1}{\sqrt{\pi}}, \sqrt{\frac{2}{\pi}}\cos t, \sqrt{\frac{2}{\pi}}\cos 2t, ..., \sqrt{\frac{2}{\pi}}\cos Nt \right\} \tag{3.71}$$

v intervalu $[0,\pi]$. V tomto případě má matice báze tvar

$$\mathbf{B}(t) = \sqrt{\frac{2}{\pi}} \begin{pmatrix} 1/\sqrt{2} & \cos t & \cos 2t & ... & \cos Nt \\ 0 & -\sin t & -2\sin 2t & ... & -N\sin Nt \\ 0 & -\cos t & -4\cos 2t & ... & -N^2\cos Nt \\ & & & ... & \end{pmatrix}. \tag{3.72}$$

Matice okrajových podmínek mají tvar

$$\mathbf{C}_0 = \sqrt{\frac{2}{\pi}} \begin{pmatrix} 1/\sqrt{2} & 1 & 1 & ... & 1 \\ 0 & 0 & 0 & ... & 0 \\ 0 & -1 & -4 & ... & -N^2 \\ & & & ... & \end{pmatrix} \tag{3.73}$$

$$\mathbf{C}_f = \sqrt{\frac{2}{\pi}} \begin{pmatrix} 1/\sqrt{2} & -1 & 1 & -1 & ... \\ 0 & 0 & 0 & 0 & ... \\ 0 & 1 & -4 & 9 & ... \\ & & & ... & \end{pmatrix}. \tag{3.74}$$

Soustava okrajových podmínek je řešitelná pouze tehdy, jestliže liché derivace v koncových bodech jsou nulové. Pak je možné tyto podmínky odstranit, neboť jsou automaticky splněny. V nejčastějším

případě $n = 2$ má pak matice **C** pouze dva řádky, které jsou zřejmě nezávislé. Bázový systém (3.71) je sice ortogonální v každé derivaci, ale ortonormální pouze v 0-té derivaci. Jelikož kritérium optimality je po eliminaci řízení zpravidla závislé na všech derivacích do řádu n, je z hlediska vyrovnaného vlivu bázových složek výhodnější zvolit např. systém

$$\left\{1, \cos t, \frac{1}{2^n}\cos 2t, ..., \frac{1}{N^n}\cos Nt\right\}. \tag{3.75}$$

Pro praktické využití se obdobně jako v předchozím případě předpokládá transformace času z intervalu $[0, t_f]$ do $[0, \pi]$.

Bázové systémy (3.71) a (3.75), třebaže mají teoretické přednosti, se nakonec při praktické realizaci neosvědčily. Obdobně nedobré zkušenosti byly překvapivě získány i v případě ortogonálních polynomiálních bázových systémů. Bázový systém $\{1, t, ..., t^N\}$ v $[0,1]$ se pro menší N (cca $N \approx 10$) poměrně osvědčil, třebaže zdaleka ne tak dobře jako $\{\Lambda_i^{n+1}\}$. Pro vyšší N je však konvergence k řešení velmi pomalá a výpočet zpravidla nevede ke kvalitnějšímu řešení. Viz též příklad 3, kap. 3.13.

3.7. Numerický výpočet kritéria

Pro výpočet integrální části funkcionálu $J(\mathbf{A})$ lze použít běžné metody numerické integrace. Funkce **f** je dle předpokladu spojitě diferencovatelná podle času. Je-li bázový systém $(n+1)$-krát spojitě diferencovatelný, je i integrand kritéria $\tilde{f}_0(\mathbf{Y}, t)$ funkcí spojitě diferencovatelnou podle času. To platí i v případě, kdy omezení jsou zahrnuta pomocí hladkých pokutových, popř. bariérových členů.

Výpočet funkcionálu je kritickým místem pro výpočet optimální trajektorie, protože je poměrně časově náročný a optimalizační algoritmus zpravidla potřebuje značně vysoký počet vyhodnocení kritéria. Výpočet je třeba provést numericky, avšak obvykle je možné spokojit se s nižší přesností. Třebaže byl prakticky využit pouze oblíbený Simpsonův kompozitní vzorec

$$\int_0^{t_f} f(x)\,dx \approx \frac{h}{3}\left[f_1 + 4f_2 + 2f_3 + 4f_4 + ... + 2f_{N-2} + 4f_{N-1} + f_N\right] + O(h^4) \tag{3.76}$$

pro liché N, který je založen na interpolaci parabolou v úsecích $[ih, (i+2)h]$, je možné patrně dosáhnout lepších výsledků pomocí sofistikovanějších metod. Např. Gaussovy metody integrace [14] pracují s nepravidelným rozmístěním vzorkovacích bodů a umožňují tímto způsobem snížit nutný počet vyhodnocení funkce pro dosažení požadované přesnosti na polovinu.

V případě omezení v každém čase, kdy je kritérium rozšířeno o pokutovou část, je však pravděpodobně vhodné zachovat vyšší počet kroků, protože úseky překročení omezení mohou být poměrně krátké. Také není jisté, jestli je korektní v tomto případě použít nepravidelné vzorkování (viz též kap. 4.5 v následující části práce, kdy se jedná o obdobný případ). Z tohoto důvodu nakonec nebyly využity metody vyššího řádu ani jiné sofistikovanější algoritmy, ale pouze vzorec (3.76).

V případě bázového systému $\{\Lambda_i^{n+1}\}_{-n}^{N}$ je třeba vzít v úvahu skutečnost, že bázové funkce jsou pouze n-krát spojitě diferencovatelné (pokud není řád spline vyšší než řád systému) a kritérium je tedy funkcí pouze po částech hladkou. Za těchto předpokladů je třeba nepočítat najednou integrál přes celý interval $[0, t_f]$, ale integrovat zvlášť jednotlivé podintervaly $[kT, (k+1)T]$. Je-li použit

vzorec (3.76), lze však integraci provést jednorázově, protože body nespojitosti derivace v tomto případě vždy oddělují dva úseky, kde je interpolována parabola.

Jiným aspektem výpočtu kritéria je efektivita vyhodnocení $\mathbf{Y}(t) = (\mathbf{y}(t), \dot{\mathbf{y}}(t), \ldots, \mathbf{y}^{(n)}(t))^T$ pro všechny časové body trajektorie. Obecně je třeba v každém kroku integrace určovat součin matic $\mathbf{B}(t)$ a $\mathbf{A}$ (3.23). Je-li N počet bázových funkcí, je třeba nmN součinů. V případě použití spline-báze je počet kroků integrace rovněž úměrný N, takže celkový počet operací pro výpočet kritéria je úměrný N^2. Pro vysoké hodnoty N se pak tento faktor může podílet značnou měrou na celkové výpočetní náročnosti. Výhodou bázového systému $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$ je, že umožňuje provést výpočet hodnot $\mathbf{Y}$ podél trajektorie výrazně efektivnějším způsobem, který je pouze lineárně závislý na N.

V prvé řadě, vzhledem k tomu, že $\Lambda_j^i(t) = 0$ pro $t \le jT$, je možné ze součtu vynechat prvky $\Lambda_j^i(t)$, kde $j \ge t/T$. Takto se v rámci celého výpočtu funkcionálu ušetří právě polovina součinů.

Dále předpokládejme, že je určena hodnota derivací $\mathbf{y}^{(i)}$ v čase $t = jT$. Z definice spline vyplývá, že v intervalu $[jT,(j+1)T]$ je $\mathbf{y}^{(n+1)}$ konstantní. Přitom hodnoty $\mathbf{y}^{(n)}$ v po sobě jdoucích uzlových bodech jsou právě

$$\begin{aligned} y_k^{(i)}(jT) &= a_{n+j,k}\,T \\ y_k^{(i)}((j+1)T) &= a_{n+j+1,k}\,T \end{aligned} \tag{3.77}$$

kde $a_{n+j,k}, a_{n+j+1,k}$ jsou koeficienty u příslušných bázových funkcí. $\mathbf{y}^{(n+1)}$ je pak v celém intervalu $[jT,(j+1)T]$ rovna

$$\mathbf{y}^{(n+1)} = \frac{\mathbf{y}^{(n)}((j+1)T) - \mathbf{y}^{(n)}(jT)}{T} = \mathbf{a}_{n+j+1} - \mathbf{a}_{n+j} = \mathbf{d} \tag{3.78}$$

kde $\mathbf{a}_k = (a_{k1}, \ldots, a_{km})^T$.

Rovnici (3.78) je možné postupně integrovat

$$\begin{aligned} &\mathbf{y}^{(n)}(\tau) = \mathbf{y}^{(n)}(t_j) + \tau\,\mathbf{d} \\ &\mathbf{y}^{(n-1)}(\tau) = \mathbf{y}^{(n-1)}(t_j) + \tau\,\mathbf{y}^{(n)}(t_j) + \frac{\tau^2}{2}\mathbf{d} \\ &\ldots \\ &\mathbf{y}(\tau) = \mathbf{y}(t_j) + \tau\,\dot{\mathbf{y}}(t_j) + \ldots + \frac{\tau^n}{n!}\mathbf{y}^{(n)}(t_j) + \frac{\tau^{n+1}}{(n+1)!}\mathbf{d} \end{aligned} \tag{3.79}$$

kde $\tau = t - t_j$ a $t_j = jT$.

Výpočet vztahů (3.79) je zpravidla mnohem rychlejší než součin matic (3.23) a nezávisí na N. Opakovaná aplikace (3.79) však kumuluje numerické chyby. Při praktické realizaci proto byly nejprve určeny hodnoty v uzlových bodech s využitím vztahu (3.23) a následně dopočítány hodnoty uvnitř intervalu $[jT,(j+1)T]$ pomocí (3.79). Hodnoty v uzlových bodech však mohou být určeny rovněž z rovnic (3.79):

$$\begin{aligned}
&\mathbf{y}^{(n)}(t_{j+1}) = \mathbf{y}^{(n)}(t_j) + T\,\mathbf{d} \\
&\ldots \\
&\mathbf{y}(t_{j+1}) = \mathbf{y}(t_j) + T\,\dot{\mathbf{y}}(t_j) + \ldots + \frac{T^n}{n!}\mathbf{y}^{(n)}(t_j) + \frac{T^{n+1}}{(n+1)!}\mathbf{d}
\end{aligned} \tag{3.80}$$

3.8. Výpočet s využitím prostoru finitních spline-funkcí

Finitní spline-funkce umožňují zajímavý způsob eliminace okrajových podmínek konstrukcí prostoru Z_N^{n+1}, jehož prvky mají v krajních bodech nulovou hodnotu včetně všech derivací do řádu n. Tento postup je však uveden pouze pro úplnost, neboť se ukázalo, že je málo flexibilní a konvergence k minimu v prostoru Z_N^{n+1} je obvykle pomalejší než v S_N^{n+1}. Hlavní význam bázového systému $\{\Sigma_i^{n+1}\}$ spočívá v urychleném způsobu výpočtu gradientu kritéria, který bude popsán později.

Definujme prostor $Z_N^{n+1}(0,t_f)$ funkcí φ takových, že platí:

$$\varphi \in S_N^{n+1}(0,t_f) \tag{3.81}$$

a současně

$$\begin{aligned}
&\varphi(0) = \dot{\varphi}(0) = \ldots = \varphi^{(n)}(0) = 0 \\
&\varphi(t_f) = \dot{\varphi}(t_f) = \ldots = \varphi^{(n)}(t_f) = 0.
\end{aligned} \tag{3.82}$$

Tvrzení: Systém $\{\Sigma_i^{n+1}\}$, $i = 1,\ldots,N-n-1$, tvoří bázi v $Z_N^{n+1}(0,t_f)$.

Důkaz: Napišme podmínky (3.82) jako soustavu lineárních rovnic

$$\begin{aligned}
&\sum_{i=-n}^{N} a_i \Lambda_i^{n+1}(0) = 0 \\
&\ldots \\
&\sum_{i=-n}^{N} a_i (\Lambda_i^{n+1})^{(n)}(0) = 0 \\
&\sum_{i=-n}^{N} a_i \Lambda_i^{n+1}(t_f) = 0 \\
&\ldots \\
&\sum_{i=-n}^{N} a_i (\Lambda_i^{n+1})^{(n)}(t_f) = 0
\end{aligned} \tag{3.83}$$

Matice soustavy (3.83) je až na pořadí řádků identická s rozšířenou maticí okrajových podmínek $\tilde{\mathbf{C}}$ (3.60), která má řešení pro libovolnou pravou stranu právě tehdy, když platí $N \geq n+1$. V tom případě má tedy $\tilde{\mathbf{C}}$ plnou hodnost $2(n+1)$ a musí obsahovat $2(n+1)$ nezávislých sloupců. Příslušné koeficienty a_i pak lze jednoznačně určit tak, že při libovolné volbě ostatních koeficientů je splněna pravá strana. Lze snadno ověřit, že $Z_N^{n+1}(0,t_f)$ je lineární prostor. Dimenze prostoru $Z_N^{n+1}(0,t_f)$ musí být shodná s počtem volitelných koeficientů, tj. $(N+n+1)-2(n+1) = N-n-1$.

Zřejmě $\Sigma_i^{n+1} \in Z_N^{n+1}(0,t_f)$ pro $i=1,...,N-n-1$ a bylo dokázáno, že tyto funkce jsou nezávislé. Pro dokončení lze použít tvrzení dokázané v části 2 práce, že systém k nezávislých funkcí tvoří v prostoru funkcí dimenze k bázi. ♦

Zvolme nyní pevně libovolnou spline funkci $y_p(t) \in S_N^{n+1}$ takovou, že pro ni platí okrajové podmínky (3.15), (3.16). Pak každá funkce $y(t) \in S_N^{n+1}$, která splňuje okrajové podmínky, může být jednoznačně vyjádřena jako součet

$$y = y_p + y_0 \tag{3.84}$$

kde $y_0(t) \in Z_N^{n+1}$. Sestrojme v prostoru $Z_N^{n+1}(0,t_f)$ bázi $\left\{\Sigma_i^n\right\}_1^{N-n-1}$. Problém je pak redukován na nalezení optimálních bázových koeficientů. Výhodou v tomto případě je, že okrajové podmínky jsou jednorázově eliminovány konstrukcí báze v prostoru $Z_N^{n+1}(0,t_f)$ a řešení je hledáno v lineárním prostoru bez dodatečných omezení.

Je-li tento způsob aproximace použit současně pro všechny složky trajektorie, vztah pro výpočet hodnot derivací má namísto (3.23) tvar

$$\mathbf{Y}(t) = \mathbf{Y}_p(t) + \boldsymbol{\beta}(t)\mathbf{A}_0 \tag{3.85}$$

kde $\boldsymbol{\beta}(t)$ je matice báze a $\mathbf{A}_0$ matice bázových koeficientů. Výpočet hodnoty kritéria přímo pomocí vztahu (3.85) je však velmi neefektivní. Výhodnější je provést transformaci koeficientů do systému $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$ a pokračovat postupem popsaným v předchozí kapitole. Transformace se provede s využitím rovnosti

$$\mathbf{A}^T \begin{pmatrix} \Lambda_{-n}^{n+1} \\ \cdots \\ \Lambda_N^{n+1} \end{pmatrix} = \mathbf{A}_p^T \begin{pmatrix} \Lambda_{-n}^{n+1} \\ \cdots \\ \Lambda_N^{n+1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_0 \\ \mathbf{0} \end{pmatrix}^T \begin{pmatrix} \Sigma_{-n}^{n+1} \\ \cdots \\ \Sigma_N^{n+1} \end{pmatrix}. \tag{3.86}$$

Odkud vyplývá

$$\mathbf{A} = \mathbf{A}_p + \mathbf{H}^T \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_0 \\ \mathbf{0} \end{pmatrix} \tag{3.87}$$

kde nulové bloky v matici $(\mathbf{0}, \mathbf{A}_0, \mathbf{0})^T$ mají $n+1$ řádků a matice $\mathbf{H}$ má již odvozený tvar (2.63).

Nevýhodou této metody je však nutnost zadání všech okrajových podmínek včetně dodatečného omezení

$$\mathbf{y}^{(n)}(0) = \mathbf{y}^{(n)}(t_f) = 0. \tag{3.88}$$

Pokud některé z okrajových podmínek chybí, je nutno uvažovat funkci $\mathbf{Y}_p$ v (3.85) jako závislou na dalších volitelných parametrech a celá konstrukce ztrácí smysl.

V případě návrhu optimálních trajektorií bohužel zpravidla chybí podmínky (3.88), neboť okrajové podmínky problému jsou zadány do řádu $(n-1)$. Dodatečné omezení (3.88) však může být smysluplné např. pro návrh trajektorií mechanických systémů, kde můžeme požadovat nulovou

hodnotu zrychlení v krajních bodech. Možností je rovněž snížit o 1 řád spline-báze. V tom případě je však n-tá derivace trajektorie nespojitou funkcí, což je v rozporu s původními předpoklady a vyžaduje mnohem vyšší dělení N.

3.9. Ortogonální rozklad soustavy okrajových podmínek

Rozklad matice bázových koeficientů popsaný v kapitole 3.5 byl využit v disertační práci. Pro dopočítání části koeficientů na základě okrajových podmínek a zbývajících koeficientů byl aplikován přímo vztah (3.58). Jako alternativní postup byla navržena konstrukce s využitím finitních spline-funkcí popsaná v předcházející kapitole.

Jak již bylo zmíněno, pro dostatečně vysoké N existuje více možností rozkladu matice $\mathbf{A}$ na část volitelnou ($\mathbf{A}_2$) a část dopočítanou ($\mathbf{A}_1$). Nabízí se tedy otázka, která z těchto variant je nejvýhodnější z hlediska efektivity výpočtu. V rámci disertační práce bylo prakticky porovnáno několik těchto rozkladů a byly učiněny určité závěry. Ve většině případů bylo výhodnější volit dopočítané koeficienty blíže počátku, popř. středu trajektorie, což lze vysvětlit snad tím, že v této části trajektorie měly optimální bázové koeficienty většinou nejnižší absolutní hodnoty.

V poslední verzi software byl použit rozklad soustavy okrajových podmínek, který vychází z metody řešení úloh nelineárního programování s lineárními omezujícími podmínkami doporučené v [11], [12]. Díky tomuto rozkladu bylo dosaženo mnohem lepších výsledků a to nejen z hlediska rychlosti výpočtu, ale i schopnosti konvergovat k řešení pro vysoké N. Jestliže elementární metoda optimalizace trajektorie s využitím spline-funkcí, navržená v [1], kterou autor prakticky realizoval v rámci diplomové práce, konvergovala uspokojivě k řešení pro $N<10$, zdokonalené postupy popsané v disertační práci byly použitelné přibližně pro $N<20$, tj. cca do 50 optimalizovaných parametrů. Postup popsaný dále pak dává uspokojivé výsledky v krátké době v řadě případů i pro $N\approx 100$, což odpovídá stovkám optimalizovaných parametrů. Nutno však uvést, že na tomto zdokonalení se nepodílí pouze jiný způsob rozkladu soustavy okrajových podmínek, ale rovněž využití kvalitnějších algoritmů optimalizace.

Pišme soustavu okrajových podmínek opět v tvaru

$$\mathbf{C}\,\mathbf{A} = \mathbf{R}\,. \tag{3.89}$$

Předpokládejme, že matice soustavy $\mathbf{C}$ má plnou hodnost. Matice $\mathbf{C}$ má plnou hodnost právě tehdy, když existuje rozklad (3.40) takový, že matice $\mathbf{C}_1$ je regulární. Výsledky kap. 3.5 jsou tedy potřebné i v tomto případě.

Řešení hledáme opět ve tvaru

$$\mathbf{A} = \mathbf{A}_p + \mathbf{A}_0 \tag{3.90}$$

kde $\mathbf{A}_p$ vyhovuje rovnici (3.89) a $\mathbf{A}_0$ patří do nulového prostoru matice $\mathbf{C}$, tj.

$$\begin{aligned} \mathbf{C}\mathbf{A}_p &= \mathbf{R} \\ \mathbf{C}\mathbf{A}_0 &= \mathbf{0} \end{aligned} \tag{3.91}$$

Zvolme

$$\begin{aligned} \mathbf{A}_p &= \mathbf{U}\mathbf{R} \\ \mathbf{A}_0 &= \mathbf{V}\mathbf{X} \end{aligned} \tag{3.92}$$

kde **U**, **V**, **X** jsou nějaké matice. Aby platilo (3.91) pro libovolné **R** a **X**, musí být

$$\begin{aligned} \mathbf{CU} &= \mathbf{1} \\ \mathbf{CV} &= \mathbf{0} \end{aligned} \tag{3.93}$$

Jsou-li sloupce matice **V** nezávislé, tvoří bázi v nulovém prostoru matice **C**, neboť každé $\mathbf{A}_0$ je možné vyjádřit jako lineární kombinaci **VX** těchto sloupců.

Jsou-li nalezeny matice **U**, **V** takové, že platí (3.93) a **V** má plnou hodnost, je problém, obdobně jako v předchozí kapitole, redukován na nalezení optimální matice **X**, která není ničím jiným, než maticí bázových koeficientů v nulovém prostoru matice **C**. Rozdíl spočívá v tom, že možných rozkladů (3.92) existuje více a není nutné pracovat s podmínkou pro n-tou derivaci (3.88).

Mezi možnými rozklady (3.92) je možné zvolit sloupce matice **V** jako ortogonální. Tím dojde k určitému „narovnání" deformace prostoru volitelných parametrů, která byla způsobena okrajovými podmínkami. V případě, že je navíc sloupcově ortogonální i složená matice $(\mathbf{U}, \mathbf{V})$, je bod $\mathbf{A}_p$ ortogonální s jakýmkoliv bodem $\mathbf{A}_0$ z nulového prostoru matice **C**, neboť

$$\mathbf{A}_p^T \mathbf{A}_0 = \mathbf{R}^T (\mathbf{U}^T \mathbf{V}) \mathbf{X} = \mathbf{0}. \tag{3.94}$$

Bod $\mathbf{A}_p$ je pak řešením soustavy (3.89), které je nejmenší ve smyslu kvadrátu Frobeniovy normy

$$\|\mathbf{A}\|^2 = \sum_{i=1}^{N+n+1} \sum_{j=1}^{m} a_{ij}^2 = \operatorname{trace}(\mathbf{A}^T \mathbf{A}) \tag{3.95}$$

neboť platí

$$\begin{aligned} \|\mathbf{A}\|^2 &= \|\mathbf{A}_p + \mathbf{A}_0\|^2 = \|\mathbf{A}_p\|^2 + \|\mathbf{A}_0\|^2 + 2\operatorname{trace}(\mathbf{A}_p^T \mathbf{A}_0) = \\ &= \|\mathbf{A}_p\|^2 + \|\mathbf{A}_0\|^2 \geq \|\mathbf{A}_p\|^2 \end{aligned} \tag{3.96}$$

a lze považovat za kvalitní počáteční odhad. V podstatě se jedná o bázové koeficienty optimální trajektorie, kde kritériem optimality je minimum normy (3.95). V případě spline-báze koeficienty odpovídají hodnotám n-té derivace v uzlových bodech a toto kritérium má tvar

$$J = \sum_{i=-n}^{N} \left\| \mathbf{y}^{(n)}(iT) \right\|^2. \tag{3.97}$$

Ortogonální rozklad (3.92) může být proveden více způsoby. Na základě doporučení z literatury [11], [15] byl použit postup založený na Hauseholderově QR rozkladu matice $\mathbf{C}^T$, který je považován za velmi robustní. Matice $\mathbf{C}^T$ je rozložena na součin matic

$$\mathbf{C}^T = (\mathbf{Q}_1, \mathbf{Q}_2) \begin{pmatrix} \mathbf{T} \\ \mathbf{0} \end{pmatrix} = \mathbf{Q}_1 \mathbf{T} \tag{3.98}$$

kde matice $\mathbf{Q} = (\mathbf{Q}_1, \mathbf{Q}_2)$ je sloupcově ortogonální a **T** je horní trojúhelníková. Zvolíme-li

$$\begin{aligned} \mathbf{U} &= \mathbf{Q}_1 (\mathbf{T}^{-1})^T \\ \mathbf{V} &= \mathbf{Q}_2 \end{aligned} \tag{3.99}$$

lze snadno ověřit, že platí podmínky (3.93) a (3.94). Předpokladem je existence matice $\mathbf{T}^{-1}$, což je ekvivalentní požadavku, aby matice $\mathbf{C}$ měla plnou hodnost.

Jelikož rozklad (3.99) je proveden jednorázově na začátku výpočtu, je limitujícím faktorem spíše součin $\mathbf{A}_0 = \mathbf{V}\mathbf{X}$, který je určován jako součást vyhodnocení kritéria. Náročnost tohoto výpočtu závisí kvadraticky na N; pro běžné hodnoty $N < 100$ je však tento faktor nevýznamný.

Na závěr ukažme, že původní metodu rozkladu soustavy okrajových podmínek (3.40) lze považovat za speciální případ rozkladu (3.92). Z (3.41) vyplývá

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1^{-1}\mathbf{R} - \mathbf{C}_1^{-1}\mathbf{C}_2\mathbf{A}_2 \\ \mathbf{A}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{C}_1^{-1} \\ \mathbf{0} \end{pmatrix}\mathbf{R} + \begin{pmatrix} -\mathbf{C}_1^{-1}\mathbf{C}_2 \\ \mathbf{1} \end{pmatrix}\mathbf{A}_2 . \tag{3.100}$$

Zvolíme-li

$$\mathbf{U} = \begin{pmatrix} \mathbf{C}_1^{-1} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} -\mathbf{C}_1^{-1}\mathbf{C}_2 \\ \mathbf{1} \end{pmatrix} \tag{3.101}$$

lze snadno dokázat, že platí podmínky (3.93).

3.10. Výpočet gradientu kritéria

Efektivní algoritmy lokální optimalizace využívají derivací účelové funkce. Vzhledem k složitosti účelové funkce je z praktického hlediska třeba předpokládat, že derivace budou určovány numericky. Metody druhého řádu v tomto případě obyčejně nelze použít pro vyšší N, neboť numerický výpočet Hessovy matice je velmi nepřesný a jeho náročnost roste s druhou mocninou počtu parametrů. Výhodnější jsou v tomto případě metody konjugovaných směrů nebo quasi-Newtonovy metody (viz část 6), pracující pouze s gradientem kritéria, který může být určován i čistě numericky.

V dalším textu opět předpokládejme, že volitelné koeficienty i derivace účelové funkce jsou organizovány v maticích. Pro vlastní optimalizaci se pak předpokládá, že sloupce matice koeficientů a gradientu jsou přeskupeny do vektorů.

Použijeme-li rozkladu

$$\mathbf{A} = \mathbf{A}_p + \mathbf{V}\mathbf{X} \tag{3.102}$$

platí pro přírůstek $d\mathbf{A}$

$$d\mathbf{A} = \mathbf{V}\, d\mathbf{X} . \tag{3.103}$$

Vyjádřeme diferenciální přírůstek kritéria:

$$\operatorname{trace}(\nabla_{\mathbf{X}}^T J\, d\mathbf{X}) = \operatorname{trace}(\nabla_{\mathbf{A}}^T J\, d\mathbf{A}) = \operatorname{trace}(\nabla_{\mathbf{A}}^T J\, \mathbf{V} d\mathbf{X}) . \tag{3.104}$$

Pro gradient kritéria v redukovaném prostoru parametrů pak dostaneme s využitím (3.35)

$$\nabla_{\mathbf{X}} J = \mathbf{V}^T \nabla_{\mathbf{A}} J = \mathbf{V}^T \left[\mathbf{B}^T(t_f) \nabla_{\mathbf{Y}_f} \varphi + \int_0^{t_f} \mathbf{B}^T(t) \nabla_{\mathbf{Y}} \tilde{f}_0 \, dt \right] . \tag{3.105}$$

Gradient kritéria daný vztahem (3.105) je možno použít v případě, že jsou známy parciální derivace funkcí φ a $\tilde{f}_0$ podle $\mathbf{Y}$, tj. $\nabla_{\mathbf{Y}_{ij}}\varphi$ a $\nabla_{\mathbf{Y}}\tilde{f}_0$. Pokud parciální derivace nejsou známy, což je obvyklý případ, není vhodné je určovat numericky, neboť integrace v (3.105) kumuluje chyby, které vznikají rozdílem blízkých hodnot. Určitým kompromisem mezi přesností a efektivitou je určovat gradient pomocí centrálních diferencí účelové funkce

$$\frac{\partial J}{\partial x_{ij}} \approx \frac{J(\mathbf{X}+h\mathbf{D}_{ij})-J(\mathbf{X}-h\mathbf{D}_{ij})}{2h} \tag{3.106}$$

kde $h>0$ je malé číslo a $\mathbf{D}_{ij}$ je matice, která má všechny prvky nulové kromě elementu na pozici (i,j), který je roven jedné. Nevýhodou vztahu (3.106) je značná výpočetní náročnost, neboť má-li matice $\mathbf{X}$ $N_0 \times m$ koeficientů, je třeba dvojnásobný počet vyhodnocení funkcionálu (3.28). Pro uvažovanou třídu problémů pak může být výpočet gradientu např. stokrát náročnější operací než výpočet hodnoty kritéria. Přesto se ukazuje, že numerické metody, které s gradientem pracují, jsou většinou mnohem účinnější než ostatní algoritmy. S využitím pouze dopředných diferencí lze ušetřit téměř polovinu výpočtů za cenu nižší přesnosti:

$$\frac{\partial J}{\partial x_{ij}} \approx \frac{J(\mathbf{X}+h\mathbf{D}_{ij})-J(\mathbf{X})}{h} \tag{3.107}$$

ale vztah (3.106) se prakticky osvědčil lépe a je doporučován i v literatuře [11], [15].

Výše popsaný postup je použitelný univerzálně pro jakýkoliv bázový systém. Existuje však možnost urychlení tohoto výpočtu pro bázové systémy typu spline. Tento postup je založen na konstrukci finitní spline-báze.

Zvolíme-li pro nahrazení trajektorie systém finitních spline funkcí $\left\{\Sigma_i^{n+1}\right\}_{-n}^{N}$, je vliv každého bázového prvku omezen na interval délky $(n+2)T$. $\nabla_{\mathbf{X}}J$ může být určen numerickým výpočtem $\nabla_{\mathbf{A}}J$ s využitím centrálních diferencí:

$$\frac{\partial J}{\partial x_{ij}} \approx \frac{1}{2h}\mathbf{V}^T\left[J(\mathbf{A}^{ij+})-J(\mathbf{A}^{ij-})\right] \tag{3.108}$$

kde $\mathbf{A}^{ij+}=\mathbf{A}+h\mathbf{D}_{ij}$ a $\mathbf{A}^{ij-}=\mathbf{A}-h\mathbf{D}_{ij}$. Pro rozdíl $\Delta_{ij}J=J(\mathbf{A}^{ij+})-J(\mathbf{A}^{ij-})$ pak platí

$$\begin{aligned}\Delta_{ij}J &= \left[\varphi(\mathbf{B}(t_f)\mathbf{A}^{+ij})-\varphi(\mathbf{B}(t_f)\mathbf{A}^{-ij})\right]+\int_0^{t_f}\left[\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{+ij})-\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{-ij})\right]dt = \\ &= \left[\varphi(\mathbf{B}(t_f)\mathbf{A}^{+ij})-\varphi(\mathbf{B}(t_f)\mathbf{A}^{-ij})\right]+\int_{t_1}^{t_2}\left[\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{+ij})-\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{-ij})\right]dt\end{aligned} \tag{3.109}$$

kde

$$\begin{aligned}t_1 &= \max\left\{(i-1)T,0\right\} \\ t_2 &= \min\left\{(i+n+1)T,t_f\right\}\end{aligned} \tag{3.110}$$

neboť změna trajektorie se při změně koeficientu a_{ij} projeví pouze v intervalu $\left[(i-1)T,(i+n+1)T\right]$ a mimo tento interval je rozdíl $\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{+ij})-\tilde{f}_0(\mathbf{B}(t)\mathbf{A}^{-ij})$ nulový. V případě, že $N \gg n$, je výpočet s využitím (3.108) a (3.109) zhruba N/n - krát rychlejší než základní vztah (3.106) a jeho celková

náročnost tedy nezávisí na N. V praxi se však ukazuje, že urychlení je patrné většinou až při $N > 10$ (pro $n = 2$), neboť matice $\mathbf{A}$ má více řádků než $\mathbf{X}$.

Nevýhodou základního postupu popsaného výše je bázový systém $\{\Sigma_i^{n+1}\}$. Praktické zkušenosti totiž ukazují, že v tomto bázovém systému je konvergence k optimu většinou podstatně horší než v bázi $\{\Lambda_i^{n+1}\}$. Proto byl tento algoritmus dále modifikován. Optimalizace probíhá v bázovém systému $\{\Lambda_i^{n+1}\}$. Systém $\{\Sigma_i^{n+1}\}$ je využit pouze pro určení gradientu. Součástí výpočtu gradientu je pak transformace koeficientů z báze $\{\Lambda_i^{n+1}\}$ do $\{\Sigma_i^{n+1}\}$ a zpětná transformace gradientu.

Transformace všech koeficientů z $\{\Lambda_i^{n+1}\}$ do $\{\Sigma_i^{n+1}\}$ není výhodná, neboť pro efektivní výpočet kritéria je stejně nutné koeficienty transformovat zpět do $\{\Lambda_i^{n+1}\}$ (viz kap. 3.7). Modifikaci koeficientů, tj. určení matic $\mathbf{A}^{ij+}$ a $\mathbf{A}^{ij-}$, lze však provést přímo v systému $\{\Lambda_i^{n+1}\}$, jak je popsáno dále.

Transformace koeficientů z $\{\Lambda_i^{n+1}\}$ do $\{\Sigma_i^{n+1}\}$ je definována s pomocí již sestrojené čtvercové regulární matice $\mathbf{H}$ (2.63):

$$\begin{pmatrix} \Sigma_{-n}^{n+1} \\ \Sigma_{-n+1}^{n+1} \\ \cdots \\ \Sigma_{N}^{n+1} \end{pmatrix} = \mathbf{H} \begin{pmatrix} \Lambda_{-n}^{n+1} \\ \Lambda_{-n+1}^{n+1} \\ \cdots \\ \Lambda_{N}^{n+1} \end{pmatrix} \tag{3.111}$$

Vynásobením zleva transponovanou maticí bázových koeficientů v systému $\{\Sigma_i^{n+1}\}$ dostáváme (spodní index označuje příslušnost k bázovému systému)

$$\mathbf{A}_\Lambda = \mathbf{H}^T \mathbf{A}_\Sigma . \tag{3.112}$$

Označíme-li $(\mathbf{A}^{ij+})_\Sigma = \mathbf{A}_\Sigma + h\mathbf{D}_{ij}$, $(\mathbf{A}^{ij-})_\Sigma = \mathbf{A}_\Sigma - h\mathbf{D}_{ij}$ a $(\mathbf{A}^{ij+})_\Lambda$, $(\mathbf{A}^{ij-})_\Lambda$ odpovídající změny v systému $\{\Lambda_i^{n+1}\}$, je

$$\begin{aligned} (\mathbf{A}^{ij+})_\Lambda &= \mathbf{H}^T(\mathbf{A}_\Sigma + h\mathbf{D}_{ij}) = \mathbf{A}_\Lambda + h\,\mathbf{H}^T\mathbf{D}_{ij} \\ (\mathbf{A}^{ij-})_\Lambda &= \mathbf{A}_\Lambda - h\,\mathbf{H}^T\mathbf{D}_{ij} . \end{aligned} \tag{3.113}$$

K bázovým koeficientům v systému $\{\Lambda_i^{n+1}\}$ tady stačí přičíst a následně odečíst h-násobek matice $\mathbf{H}^T\mathbf{D}_{ij}$, což odpovídá přičtení, resp. odečtení h-násobku i-tého řádku matice $\mathbf{H}$ k j-tému sloupci matice $\mathbf{A}_\Lambda$.

Pomocí vztahů (3.108), (3.109) je pak určen gradient v systému $\{\Sigma_i^{n+1}\}$. Pro získání gradientu v $\{\Lambda_i^{n+1}\}$, lze použít vztah (3.112) aplikovaný na diferenciální přírůstek J:

$$\operatorname{trace}(\nabla_\Lambda^T d\mathbf{A}_\Lambda) = \operatorname{trace}(\nabla_\Lambda^T \mathbf{H}^T d\mathbf{A}_\Sigma) . \tag{3.114}$$

Odtud vyplývá, že

$$\mathbf{H}\nabla_{\Lambda} = \nabla_{\Sigma} \tag{3.115}$$

neboli

$$\nabla_{\Lambda} = \mathbf{H}^{-1}\nabla_{\Sigma} \,. \tag{3.116}$$

Bylo již dokázáno, že matice $\mathbf{H}^{-1}$ existuje. Pro výpočet ∇_{Λ} však není vhodné použít přímo vztah (3.116), neboť matice $\mathbf{H}$ může být značně rozsáhlá. Místo toho se využije speciálního tvaru této matice, která je horní trojúhelníková a navíc nenulové prvky tvoří s diagonálou pás šířky $n+1$. Řešení (3.115) lze získat velmi efektivně zpětnou substitucí, tj. zpětným krokem Gaussovy eliminační metody [15], pro každý sloupec ∇_{Λ} zvlášť.

Na závěr shrňme celý postup:

1. Pro dané $\mathbf{X}$ je třeba určit $\mathbf{A}_{\Lambda}$ ze vztahu (3.102). Tím je určena trajektorie.
2. Pro každé (i,j) určíme $(\mathbf{A}^{ij+})_{\Lambda}$, $(\mathbf{A}^{ij-})_{\Lambda}$ ze vztahu (3.113)
3. Gradient $\nabla_{\Sigma} J$ se určí z parciálních derivací, pro každé (i,j):

$$\frac{\partial J}{\partial (a_{ij})_{\Sigma}} \approx \frac{1}{2h}\left[J(\mathbf{A}^{ij+})_{\Lambda} - J(\mathbf{A}^{ij-})_{\Lambda} \right] . \tag{3.117}$$

4. Gradient $\nabla_{\Lambda} J$ se vypočítá řešením soustavy (3.115).
5. Gradient $\nabla_{\mathrm{X}} J$ se určí jako $\nabla_{\mathrm{X}} J = \mathbf{V}^{T}\nabla_{\Lambda} J$.

Popsaný postup byl úspěšně prakticky implementován. Zajímavé je, že tento algoritmus výpočtu gradientu je pro dostatečně velké N rychlejší než základní vztah (3.105), který je založen na znalosti parciálních derivací funkce $\tilde{f}_0$. Lze konstatovat, že díky tomuto postupu je možné pracovat i s hodnotami $N \approx 100$.

3.11. Úlohy s volným časem

Uvažujme nyní, že čas t_f v kritériu

$$J(\mathbf{A};t_f) = \varphi(\mathbf{Y}(t_f)) + \int_0^{t_f} \tilde{f}_0(\mathbf{Y}(t))\,dt \to \min \tag{3.118}$$

není předem znám, ale předpokládejme, že je konečný.

Základní možností je druhá iterační smyčka, kdy řešení je získáno jednorozměrným hledáním optimální hodnoty t_f a minimalizovanou funkcí je hodnota řešení problému s pevným časem:

$$J(\mathbf{A}^{*};t_f^{*}) = \min_{t_f}\left\{ \min_{\mathbf{A}}\left\{ J(\mathbf{A};t_f) \right\} \right\} . \tag{3.119}$$

Tento postup je ale většinou prakticky nerealizovatelný, neboť vnitřní optimalizační úloha je značně časově náročná.

Je rovněž možné přidat parametr t_f mezi optimalizované parametry. Ani tento způsob však není efektivní. Je nutné přepočítávat matici báze a realizovat rozklad soustavy okrajových podmínek jako součást výpočtu každé hodnoty kritéria. I při akceptování této režie však bylo prakticky

ověřeno, že po přidání t_f mezi optimalizované parametry se konvergence k řešení zpravidla výrazně zhorší.

Jinou alternativou, která byla realizována v rámci disertační práce, je hledání minima (3.17) střídáním optimalizačních kroků v proměnných $\mathbf{A}$ a t_f

$$J(\mathbf{A}_2;t_{f1}) = \min_{\mathbf{A}}\{J(\mathbf{A};t_{f1})\} \tag{3.120}$$

$$J(\mathbf{A}_2;t_{f2}) = \min_{t_f}\{J(\mathbf{A}_2;t_f)\}. \tag{3.121}$$

Předpokládejme, že všechny prvky matice $\mathbf{A}$ a parametr t_f jsou uloženy je jediném vektoru $\mathbf{z}$. Funkce $J(\mathbf{A};t_f) \equiv J(\mathbf{z})$ je spojitě diferencovatelná. Derivace podle parametru t_f je dána vztahem

$$\frac{\partial J}{\partial t_f} = \frac{\partial \varphi(\mathbf{Y}(t_f))}{\partial t_f} + \tilde{f}_0(\mathbf{Y}(t_f)). \tag{3.122}$$

Hodnota funkce $J(\mathbf{z})$ lze v okolí bodu $\mathbf{z}_0$ vyjádřit Taylorovým rozvojem

$$J(\mathbf{z}) = J(\mathbf{z}_0) + \frac{\partial J}{\partial \mathbf{z}}\mathbf{h} + \frac{1}{2}\mathbf{h}^T\frac{\partial^2 J}{\partial \mathbf{z}^2}\mathbf{h} + o\left(\|\mathbf{h}\|^2\right) \tag{3.123}$$

kde $\mathbf{h} = \mathbf{z} - \mathbf{z}_0$.

Tvrzení. Má-li problém (3.118) řešení, získaná posloupnost hodnot

$$J(\mathbf{A}_1;t_1), J(\mathbf{A}_2;t_2), \ldots \tag{3.124}$$

konverguje. Limita této posloupnosti J_∞ je lokálním minimem $J(\mathbf{A}^*;t_f^*)$ problému (3.118), jestliže matice druhých derivací $\frac{\partial^2 J}{\partial \mathbf{z}^2}$ je v každém bodě $(\mathbf{A};t_f)$ takovém, že $J(\mathbf{A};t_f) = J_\infty$, pozitivně definitní.

Důkaz: Z (3.120), (3.121) vyplývá, že $J(\mathbf{A}_i;t_i) \le J(\mathbf{A}_{i+1};t_i) \le J(\mathbf{A}_{i+1};t_{i+1})$, tj. posloupnost (3.124) je klesající a je zdola omezená hodnotou řešení (3.118), proto má limitu J_∞. Označme $\mathbf{z}_0 \equiv (\mathbf{A};t_f)$. Jestliže $J(\mathbf{z}_0) = J_\infty$ není lokálním minimem (3.118), pak v libovolně malém okolí bodu $\mathbf{z}_0$ existuje bod $\mathbf{z}$ takový, že $J(\mathbf{z}) < J(\mathbf{z}_0)$. Z (3.123) pak vyplývá, že musí existovat alespoň jeden index i takový, že součin $\frac{\partial J}{\partial z_i}h_i$ je záporný, neboť poslední dva členy v (3.123) jsou kladné. Potom je však možné v tomto směru nalézt nižší hodnotu úlohy (3.120) nebo (3.121), takže J_∞ nemůže být limitou (3.124). ♦

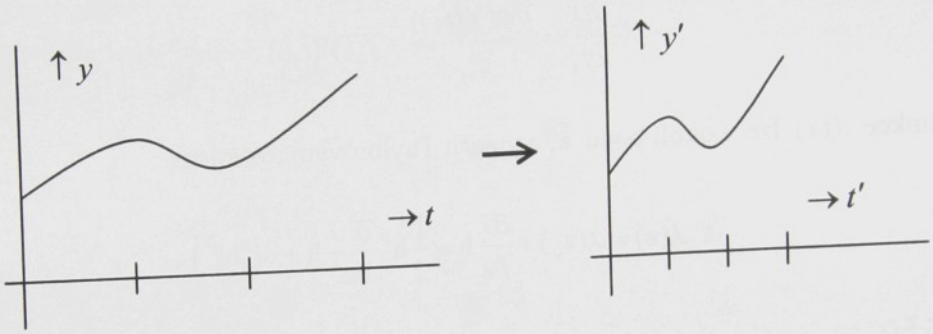
Pozitivní definitnost $\frac{\partial^2 J}{\partial \mathbf{z}^2}$ však obecně zaručena není. Při praktické realizaci se hledání minima v každém kroku (3.120), (3.121) sice provádí s omezenou přesností, ale to většinou nepředstavuje podstatný problém. Pro efektivní výpočet je třeba zaručit, aby náročnost optimalizačních podproblémů (3.120), (3.121) postupně klesala. Toho lze dosáhnout využitím předchozího řešení pro sestrojení nového počátečního odhadu.

V případě iterací (3.121) je výchozím odhadem okamžitě posledně získaná hodnota t_f, avšak u kroků (3.120) je situace složitější, neboť při změně t_f v předchozím kroku (3.121) došlo ke změně bázových funkcí a tím i k deformaci trajektorie. Konstrukce kvalitního počátečního odhadu pro podproblémy (3.120) tedy vyžaduje vhodnou transformaci bázových koeficientů získaných z minulého řešení.

V ideálním případě jsou koeficienty přepočítány tak, že trajektorie si zachová stejný tvar a jsou splněny všechny okrajové podmínky (Obr. 3.2). V tom případě platí

$$y(t) = y'(t') \tag{3.125}$$

$$\frac{t}{t_f} = \frac{t'}{t'_f}. \tag{3.126}$$



Obr. 3.2: Transformace trajektorie při změně t_f

Pro diferenciální přírůstky trajektorie $y'(t')$ platí

$$dy = dy'$$

$$\frac{dt}{t_f} = \frac{dt'}{t'_f}. \tag{3.127}$$

Odtud dostáváme

$$\frac{d^k y'}{dt'^k} = \left(\frac{t_f}{t'_f}\right)^k \frac{d^k y}{dt^k}. \tag{3.128}$$

Ze vztahu (3.128) ovšem vyplývá, že při zachování tvaru trajektorie dojde ke změně hodnot derivací v koncových bodech. V případě, že původní trajektorie vyhovovala okrajovým podmínkám, po transformaci času to již neplatí. Přípustná trajektorie je tímto způsobem získána pouze pokud jsou okrajové podmínky ve speciálním tvaru

$$\dot{y}(0) = \ldots = y^{(n)}(0) = 0$$

$$\dot{y}(t_f) = \ldots = y^{(n)}(t_f) = 0 \tag{3.129}$$

a hodnoty $y(0)$, $y(t_f)$ mohou být libovolné. Jedná se však o nejběžnější případ návrhu optimálních trajektorií. Pokud podmínky (3.129) neplatí, je možné pro konkrétní typ bázového systému se omezit pouze na zachování podmínky nulté (nebo některé vyšší) derivace nebo provést přepočítání

bázových koeficientů např. pomocí metody nejmenších čtverců. Tento postup byl popsán v disertační práci pro případ bázových systémů $\{\Lambda_i^{n+1}\}$ a $\{\Sigma_i^{n+1}\}$.

Uvažujme dále pouze transformaci koeficientů v případě, že platí (3.129) pro všechny složky trajektorie. V případě polynomiální báze bylo řečeno, že je zpravidla nutná transformace času do intervalu $[0,1]$. To znamená, že místo matice báze $\mathbf{B}(t), t \in [0,t_f]$ pracujeme s maticí báze

$$\mathbf{B}\left(\frac{t}{t_f}\right), \frac{t}{t_f} \in [0,1]. \tag{3.130}$$

Potom bázový systém je pro nultou derivaci stejný pro všechna t_f. To však znamená, že při změně t_f pro zachování tvaru trajektorie není nutné transformovat bázové koeficienty. Hodnoty i-tých derivací bázových funkcí jsou pak na základě vztahu (3.128) násobeny faktorem $\frac{1}{(t_f)^i}$ (ve vztahu (3.128) je $t_f = 1$ a t_f' je místo t_f). V soustavě okrajových podmínek jsou rovnice obsahující derivace trajektorie oproti původnímu tvaru násobeny rovněž tímto faktorem. Jelikož na pravé straně těchto rovnic zůstává nula, konstantní faktor lze vykrátit a proto získané matice $\mathbf{U},\mathbf{V}$, popř. $\mathbf{C}_1^{-1},\mathbf{C}_2$ zůstávají stejné. To však platí pro jakýkoliv bázový systém, kde hodnoty času jsou transformovány do intervalu pevné délky, např. $[0,\pi]$ pro systém $\{1, \cos t,...,\cos Nt\}$.

Bázový systém $\{\Lambda_i^{n+1}\}$ byl zkonstruován jako závislý na parametru $T = \frac{t_f}{N}$, avšak z výše uvedených důvodů je výhodnější pro úlohy s volným časem i v tomto případě využít transformaci času (3.130). Jinou možností, která byla detailně rozvedena v rámci disertační práce, je provést přepočet bázových koeficientů při změně t_f. Jelikož platí (3.78), je v tomto případě třeba transformovat koeficienty podle vztahu

$$\mathbf{A}' = \left(\frac{t_f}{t_f'}\right)^{n+1} \mathbf{A}_k. \tag{3.131}$$

Změna t_f, která je prováděna v každé iteraci podproblému (3.121), však rovněž vyžaduje odpovídající korekci matic $\mathbf{U},\mathbf{V}$, popř. $\mathbf{C}_1^{-1},\mathbf{C}_2$.

Výše uvedený postup byl s poměrně dobrými zkušenostmi prakticky realizován. V případě okrajových podmínek ve tvaru (3.129) se tvar trajektorie po prvním optimalizačním kroku mění minimálně, a proto stačí zpravidla jen několik kroků (3.120), (3.121). Vnější iterační smyčka je z praktického hlediska přesto značnou nevýhodou. Existuje však transformace na problém s pevným časem, která je obdobou postupu popsaného v části 4, kap. 4.4, pro úlohy optimálního řízení v standardním tvaru.

Uvažujme tvar kritéria s eliminovanou funkcí řízení

$$J = \varphi(\mathbf{y}^{(n)}(t_f),...,\mathbf{y}(t_f)) + \int_0^{t_f} \tilde{f}_0(\mathbf{y}^{(n)},...,\mathbf{y},t)\,dt. \tag{3.132}$$

Považujme t za funkci parametru $\tau \in [0,1]$ definovanou diferenciální rovnicí s počáteční podmínkou

$$\frac{d^n t}{d\tau^n} = t^{(n)} = v \tag{3.133}$$

$$t(0) = 0 \tag{3.134}$$

kde $v(t)$ je nějaká spojitá nezávislá řídicí proměnná. Okrajové podmínky derivací t jsou ponechány volné. Použijeme-li jako substituci v integrálu (3.132) řešení rovnice (3.133), platí

$$\mathbf{y}^{(i)} = \frac{d^n \mathbf{y}}{dt^n} = \frac{d^n \mathbf{y}}{d\tau^n} \frac{1}{t^{(n)}} . \tag{3.135}$$

Potom (3.132) přejde do tvaru s pevným časem $\tau_f = 1$:

$$J = \varphi\left(\frac{d^n \mathbf{y}}{d\tau^n}(t(1)) \frac{1}{t^{(n)}(1)}, ..., \mathbf{y}(t(1)) \right) + \int_0^1 \tilde{f}_0 \left(\frac{d^n \mathbf{y}}{d\tau^n} \frac{1}{t^{(n)}}, ..., \mathbf{y}(t(\tau)), t \right) \frac{dt(\tau)}{d\tau} d\tau . \tag{3.136}$$

Rozšíříme-li vektor $\mathbf{y}$ o složku t, má kritérium (3.136) tvar (3.132) a lze tedy použít popsané postupy pro řešení úloh s pevným časem pod podmínkou, že funkce $t(\tau)$ je rostoucí. Monotónnost lze zaručit dodatečným omezujícím požadavkem

$$\frac{d}{d\tau} t(\tau) \geq \varepsilon > 0 \tag{3.137}$$

kde $\varepsilon \geq 0$ je malé číslo, což je speciální případ omezení typu (3.18).

Výše popsaný způsob transformace problému dosud nebyl prakticky realizován. Lze předpokládat, že výpočet bude trvat podstatně déle, než u stejných úloh s pevným časem z důvodu složitější funkce $\tilde{f}_0$, ale odpadá vnější iterační smyčka a řada praktických problémů, které souvisí s střídáním optimalizačních podproblémů (3.120), (3.121). Z hlediska teoretického pak není dále nutnost se zvlášť zabývat problémy s volným časem.

3.12. Nastavení parametrů regulační smyčky

Není cílem této práce popisovat způsoby nastavení parametrů zpětnovazební smyčky, která má udržovat systém v pohybu podél vypočítané optimální trajektorie. Přesto systémy regulární vzhledem k řízení mají i pro zpětnovazební řízení mimořádné výhody, které by měly být alespoň stručně zmíněny.

Uvažujme, že systém regulární vzhledem k řízení je ve tvaru s vektorem $\mathbf{u}$ na pravé straně

$$\mathbf{f}(\mathbf{y}^{(n)}, ..., \mathbf{y}, t) = \mathbf{u} . \tag{3.138}$$

V blízkosti optimální trajektorie lze systém linearizovat:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}^{(n)}} \delta \mathbf{y}^{(n)} + ... + \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \delta \mathbf{y} = \delta \mathbf{u} \tag{3.139}$$

kde matice parciálních derivací jsou závislé na čase. Veličina $\delta \mathbf{y}$ vyjadřuje odchylku trajektorie a $\delta \mathbf{u}$ odchylku řízení. Jelikož systém (3.138) je dle předpokladu regulární vzhledem k řízení, je

$$\det\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}^{(n)}}\right) \neq 0 \tag{3.140}$$

a soustava (3.139) je pak rovněž regulární vzhledem k řízení, tedy řiditelná.

V případě odchylky od optimální trajektorie je tedy možné navrhnout libovolný průběh kompenzace této chyby a rovnice (3.139) jednoznačně definuje odpovídající průběh řízení. Zvolme např.

$$\delta y_i(t) = C_i e^{-t/\tau_i}, \; i = 1,...,m \tag{3.141}$$

kde $\tau_i > 0$ je časová konstanta zvolená zvlášť pro každý člen. Zvolený průběh je nekmitavý, což je obzvláště důležité u mechanických systémů, a asymptoticky stabilní.

(3.141) je pro libovolné $i > 0$ řešením soustavy diferenciálních rovnic

$$\delta \mathbf{y}^{(i)}(t) = \begin{pmatrix} (-1/\tau)^i, 0, \cdots, 0 \\ \cdots \\ 0, \cdots, 0, (-1/\tau_m)^i \end{pmatrix} \delta \mathbf{y}(t) = (-\mathbf{T})^i \delta \mathbf{y}(t) \tag{3.142}$$

kde

$$\mathbf{T} = \begin{pmatrix} 1/\tau_1, 0, \cdots, 0 \\ \cdots \\ 0, \cdots, 0, 1/\tau_m \end{pmatrix}. \tag{3.143}$$

Dosaďme do rovnice (3.139):

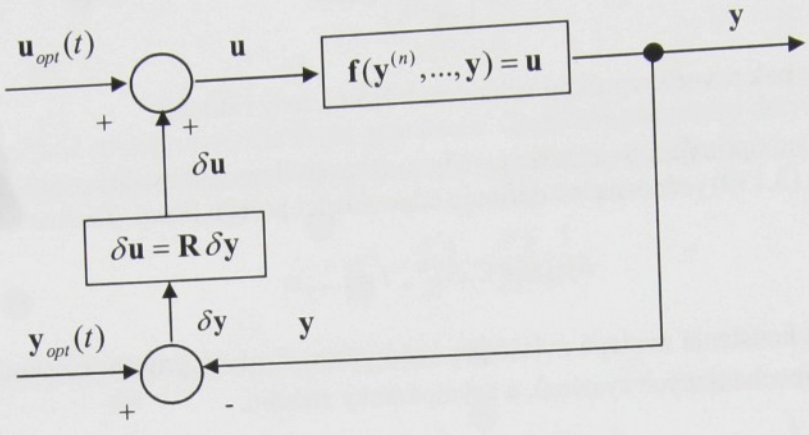
$$\sum_{i=0}^{n} \frac{\partial \mathbf{f}}{\partial \mathbf{y}^{(i)}} \delta \mathbf{y}^{(i)} = \left[\sum_{i=0}^{n} \frac{\partial \mathbf{f}}{\partial \mathbf{y}^{(i)}} (-\mathbf{T})^i \right] \delta \mathbf{y} = \delta \mathbf{u} \,. \tag{3.144}$$

Odtud dostáváme ihned odpovídající matici regulátoru, která však závisí na čase

$$\mathbf{R}(t) = \sum_{i=0}^{m} \frac{\partial \mathbf{f}}{\partial \mathbf{y}^{(i)}} (-\mathbf{T})^i \,. \tag{3.145}$$

Schéma regulační smyčky je znázorněno na Obr. 3.3.

Jelikož průběh parciálních derivací $\dfrac{\partial \mathbf{f}}{\partial \mathbf{y}^{(n)}}$ podél vypočítané trajektorie je předem znám a může být uložen v paměti řídicího systému, není problém v každém bodě efektivně určit odpovídající hodnotu vektoru řízení.



Obr. 3.3: Schéma regulační smyčky

3.13. Řešené úlohy

Následující řešené příklady demonstrují různé aspekty metody nahrazení trajektorie. Jako testovací materiál jsou vesměs využity matematické modely mechanických systémů popsané v části 7.

Úlohy 1-3 jsou bez omezujících podmínek. Součástí úloh 4 a 5 je jednoduché omezení a obecnější tvar koncových podmínek. Pro výpočet byly ve všech případech využity pouze lokální algoritmy optimalizace.

1. Úloha

Model č.1 robota cylindrického typu, kap. 7.1. Okrajové podmínky problému jsou

$$\begin{aligned} \mathbf{y}_0 &= (0,0,0)^T \\ \dot{\mathbf{y}}_0 &= (0,0,0)^T \\ \mathbf{y}_f &= (1,2,1)^T \\ \dot{\mathbf{y}}_f &= (0,0,0)^T \end{aligned} \tag{3.146}$$

kde $\mathbf{y} = (z,\varphi,x)^T$. Jako kritérium optimality je zvolen kvadratický funkcionál

$$J = \int_0^1 \mathbf{u}^T\mathbf{u}\,dt \to \min . \tag{3.147}$$

Je použit bázový systém typu spline $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$, kde $n = 2$, $N = 20$. Celkový počet optimalizovaných parametrů je $n_z = 57$.

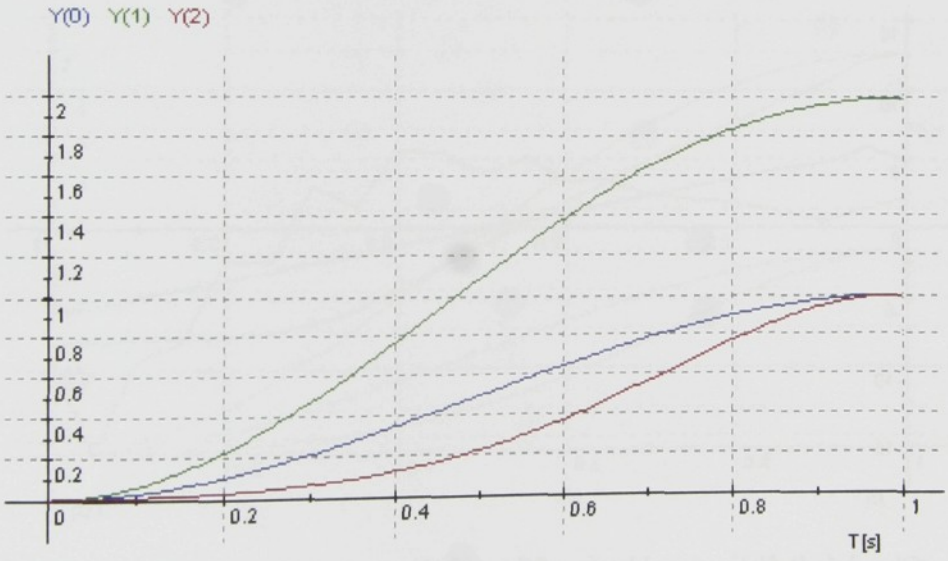
Nejsou uvažována žádná dodatečná omezení. Výpočet byl pro srovnání proveden algoritmy Největšího spádu, Polak-Ribière, PARTAN a BFGS. Za jeden iterační krok je u všech těchto metod považováno určení směru hledání a přibližné určení bodu minima v tomto směru. Výchozí odhad byl zvolen náhodně, ale u všech algoritmů stejně. Tab. 3-1 obsahuje dosažené hodnoty kritéria v určitém počtu kroků pro jednotlivé algoritmy.

Iterace	Největšího spádu	Polak-Ribière	PARTAN	BFGS
0	62.048	62.048	62.048	62.048
2	53.9686	54.6575	53.7459	53.7463
4	53.3792	53.6261	53.3425	53.412
10	53.0961	53.1878	53.0657	53.1931
20	53.0308	53.0447	53.0126	53.0304
50	53.0113	53.0121	53.0107	53.0107

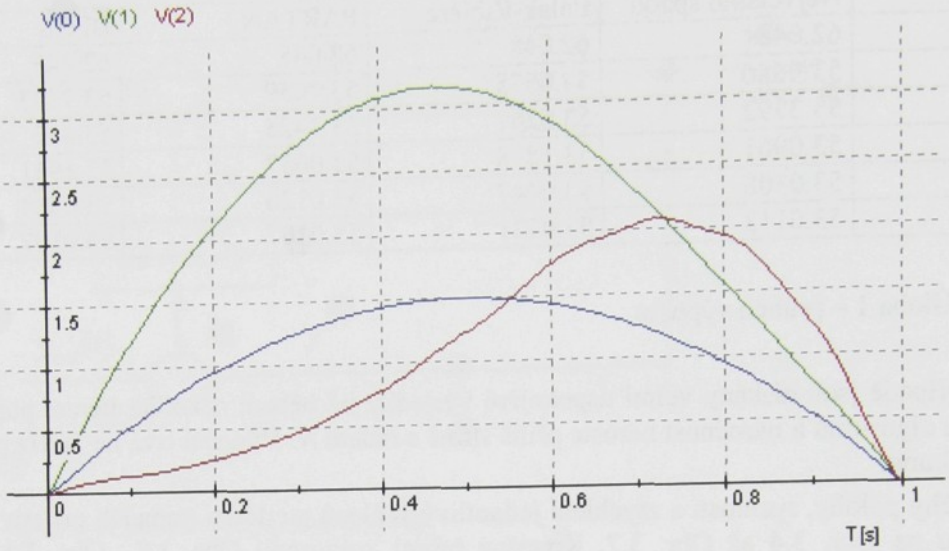
Tab. 3-1: Úloha 1 – průběh výpočtu

V tomto případě jsou získány velmi uspokojivé výsledky již během několika iterací pomocí všech uvedených algoritmů a náročnost neroste příliš strmě s řádem *N*. Výpočet trvá na 1GHz počítači PC okolo 5 sekund.

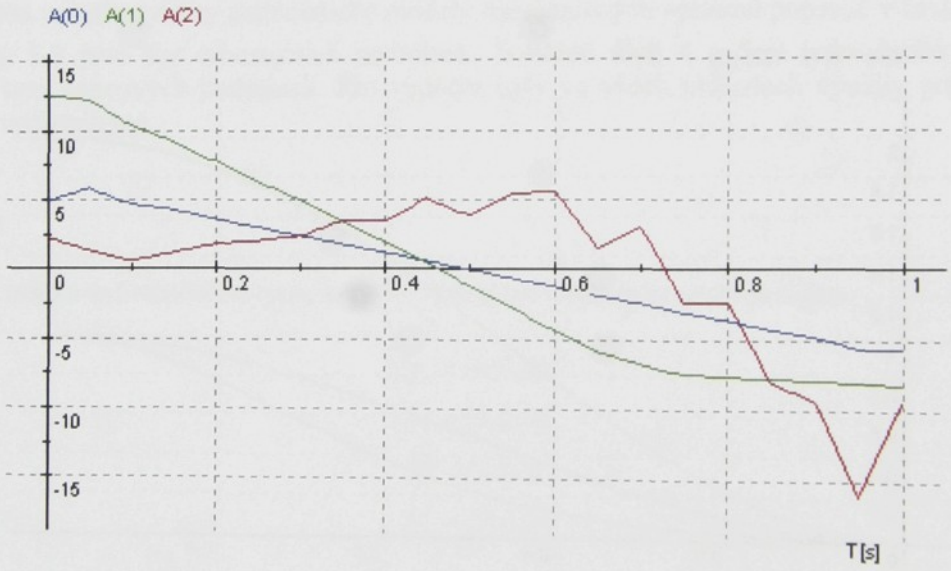
Průběhy polohy, rychlosti a zrychlení jednotlivých členů po deseti iteracích metody největšího spádu jsou na Obr. 3.4 až Obr. 3.7. Konečné řešení znázorňují Obr. 3.8 - Obr. 3.11. Získaný průběh zrychlení je velmi blízký hladké funkci, ačkoliv je ve skutečnosti pouze spojitý. Proto se lze domnívat, že výsledek je skutečným řešením, alespoň v lokálním smyslu.



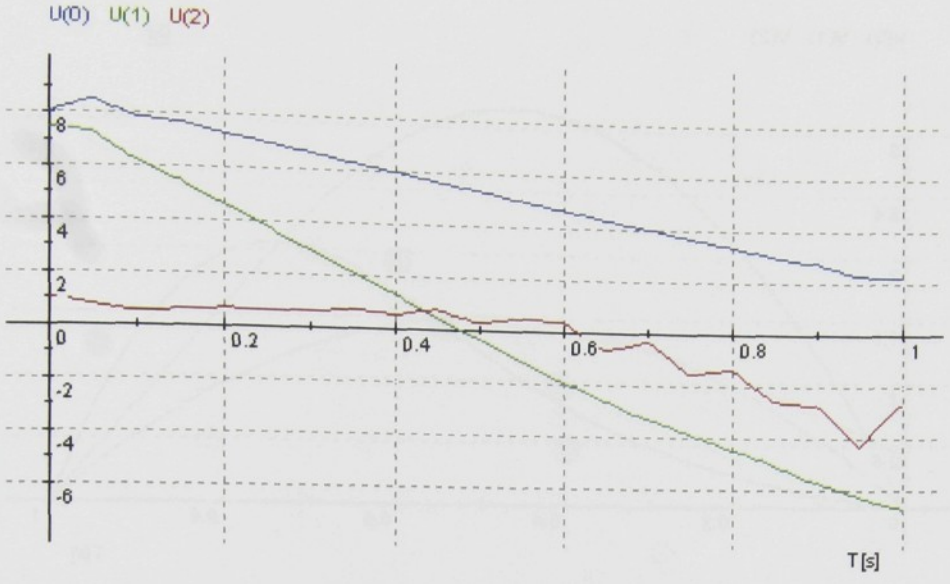
Obr. 3.4: Průběhy polohy po 10 iteracích



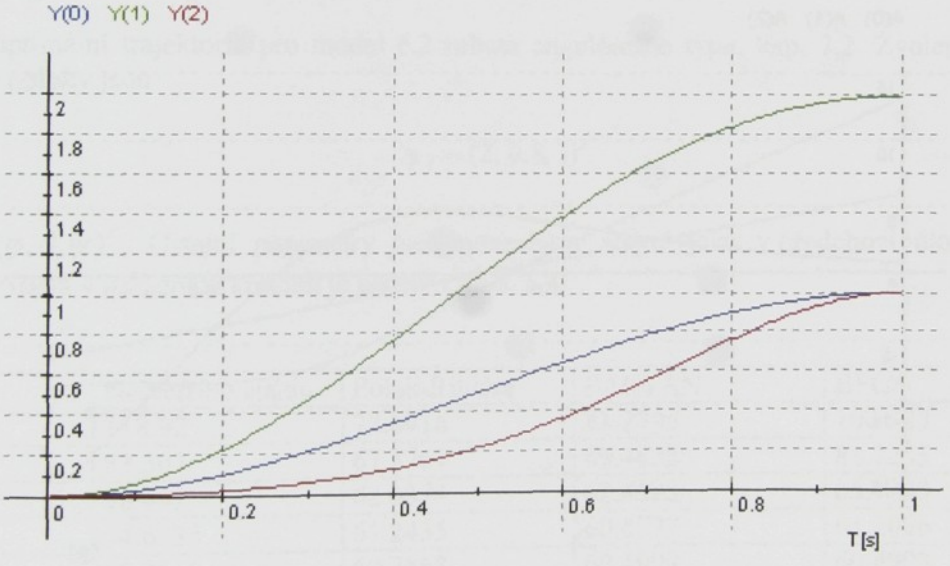
Obr. 3.5: Průběhy rychlosti po 10 iteracích



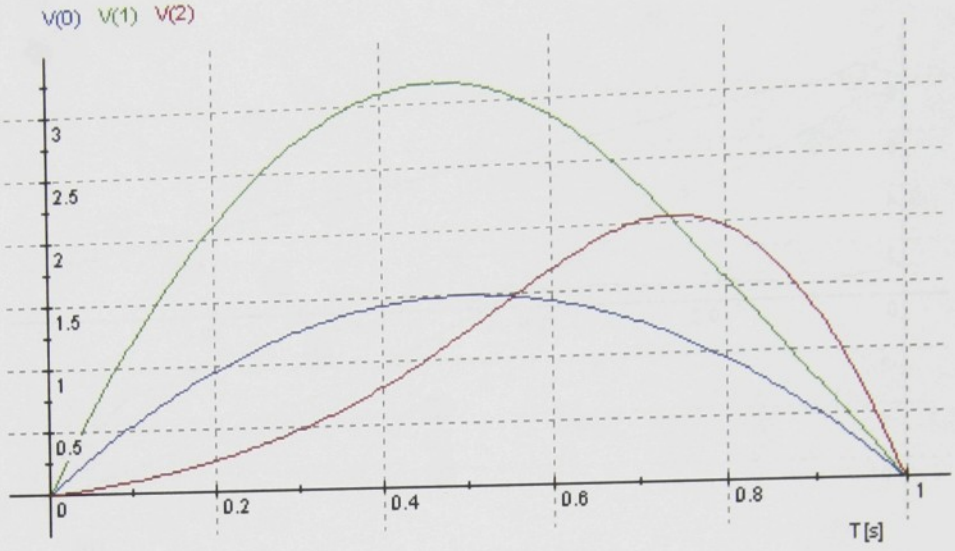
Obr. 3.6: Průběhy zrychlení po 10 iteracích



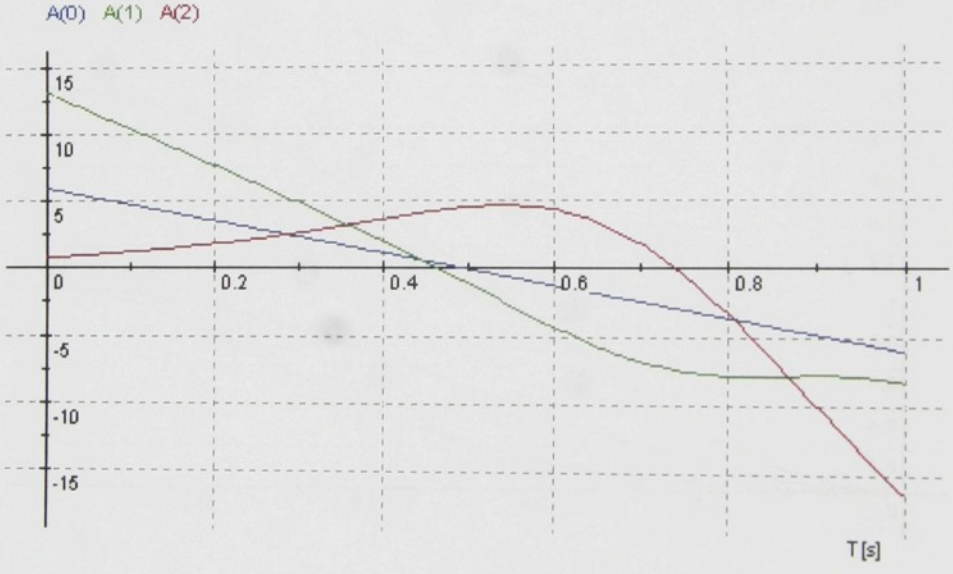
Obr. 3.7: Průběhy řízení po 10 iteracích



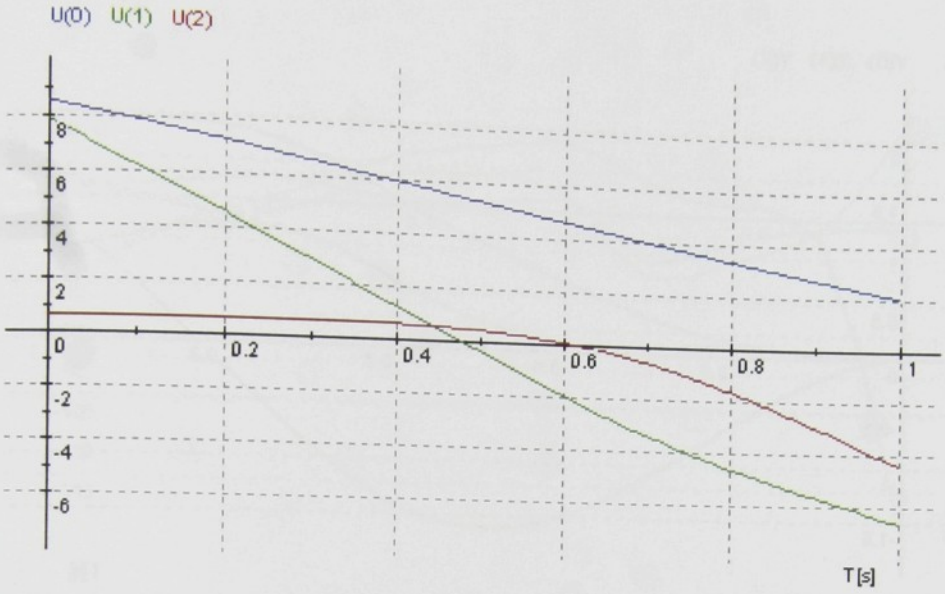
Obr. 3.8: Úloha 1 - konečné průběhy polohy



Obr. 3.9: Úloha 1 - konečné průběhy rychlosti



Obr. 3.10: Úloha 1 - konečné průběhy zrychlení



Obr. 3.11: Úloha 1 - konečné průběhy řízení

2. Úloha

Výpočet optimální trajektorie pro model č.2 robota angulárního typu, kap. 7.2. Zvolené koncové podmínky polohy jsou

$$\mathbf{y}_f = (2, 0.8, 1)^T \tag{3.148}$$

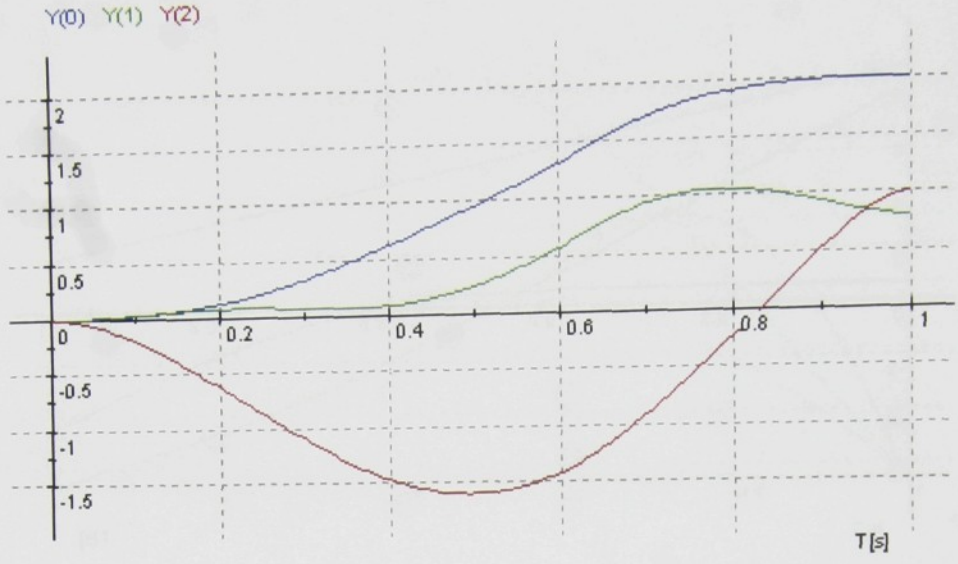
kde $\mathbf{y} = (\varphi, \vartheta, \psi)^T$. Ostatní parametry jsou ponechány stejné jako v předchozí úloze. Průběh hodnoty kritéria v iteračních krocích je patrný z Tab. 3-2.

Iterace	Největšího spádu	Polak-Ribière	PARTAN	BFGS
10	118.92	72.6916	81.2995	100.523
20	83.307	67.3767	69.4838	81.2255
50	68.492	62.6374	62.4523	65.7203
100	64.6025	61.2455	60.8777	61.2096
200	62.3618	60.7868	60.4909	60.4903
500	61.0375	60.4913	60.4903	-
1000	60.687	60.4903	-	-

Tab. 3-2: Úloha 2 – průběh výpočtu

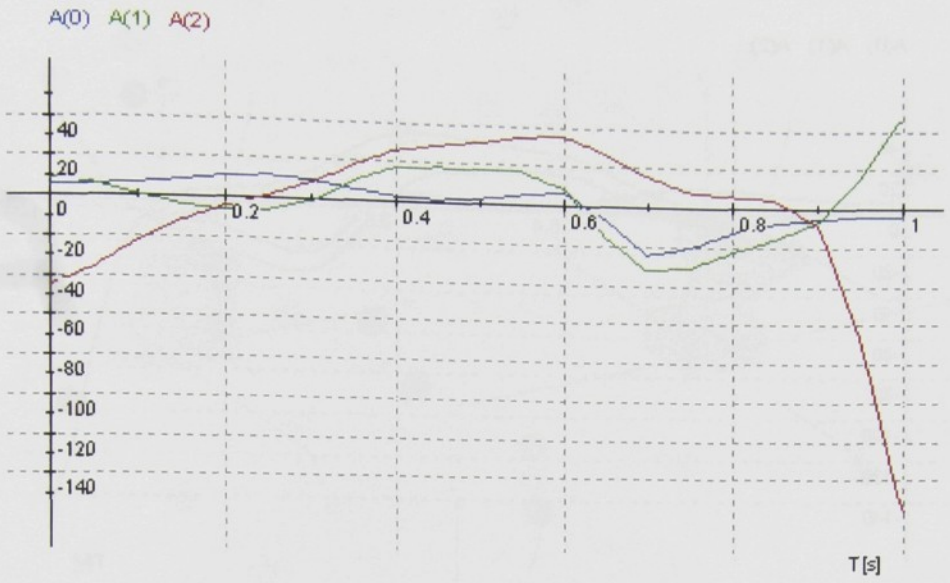
V tomto případě již výpočet trvá podstatně déle a je patrný rozdíl mezi jednotlivými algoritmy. V blízkosti extrému nejrychleji konverguje metoda BFGS, třebaže v počátečních krocích jsou účinnější metody konjugovaných gradientů. Účinnost metody PARTAN je překvapivá. Metoda největšího spádu konverguje mnohem pomaleji než ostatní metody.

Průběhy polohy, rychlosti a zrychlení jednotlivých členů jsou znázorněny na Obr. 3.12 až Obr. 3.15. Průběh řízení, který je spojitý, je v tomto případě poměrně vzdálený od hladké funkce. Zajímavé je porovnání s výsledky pro vyšší hodnotu $N = 40$. Průběhy zrychlení a řízení pro $N = 40$ jsou na Obr. 3.16 a Obr. 3.17.

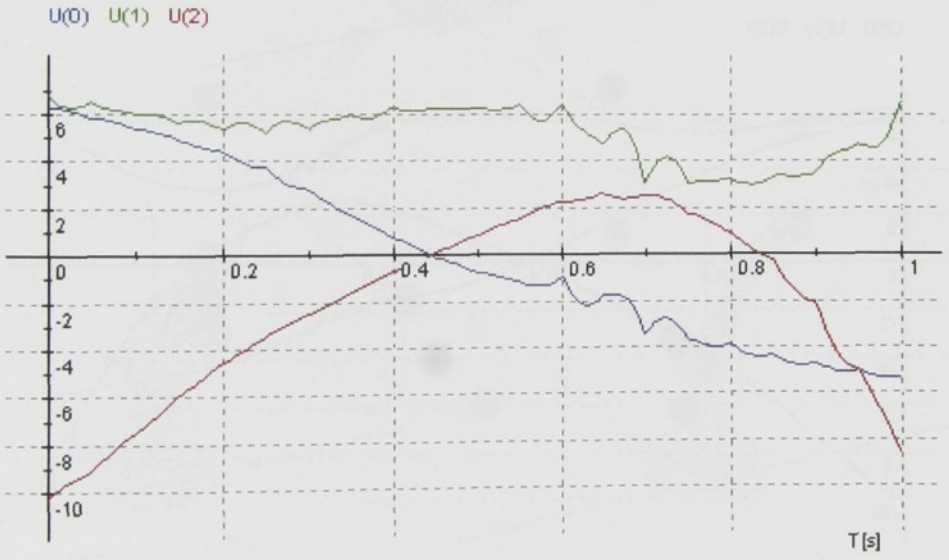


Obr. 3.12: Úloha 2 - průběhy polohy

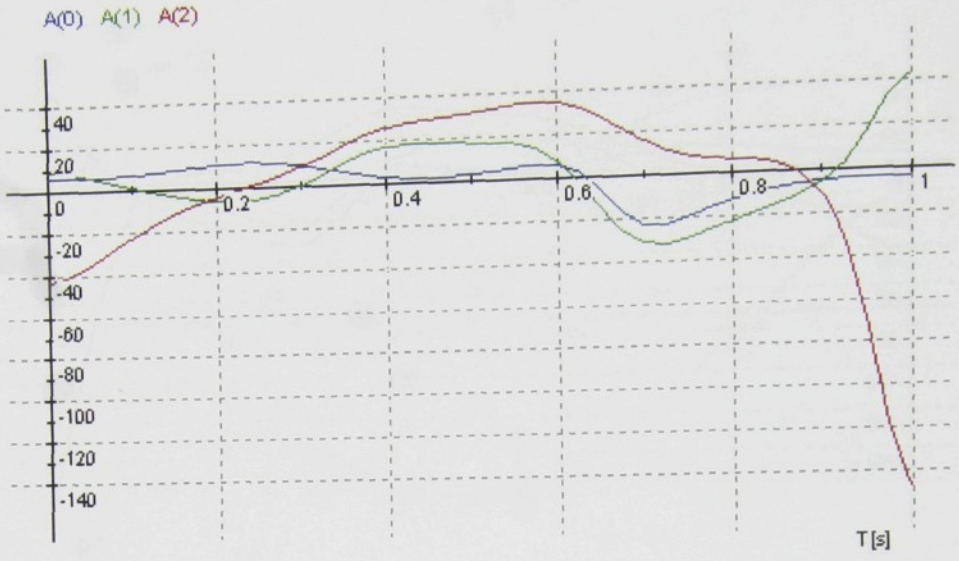
Obr. 3.13: Úloha 2 - průběhy rychlosti



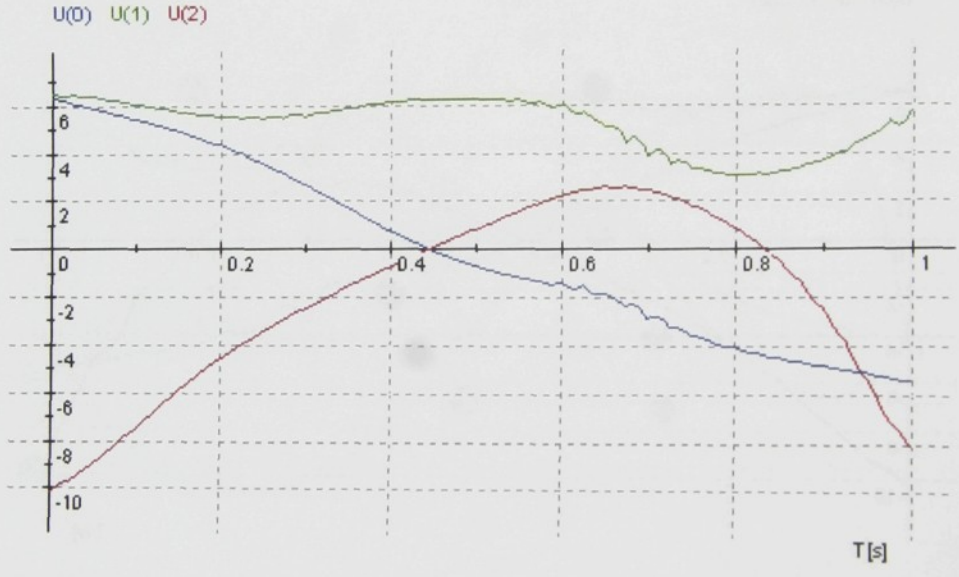
Obr. 3.14: Úloha 2 - průběhy zrychlení



Obr. 3.15: Úloha 2 - průběhy řízení

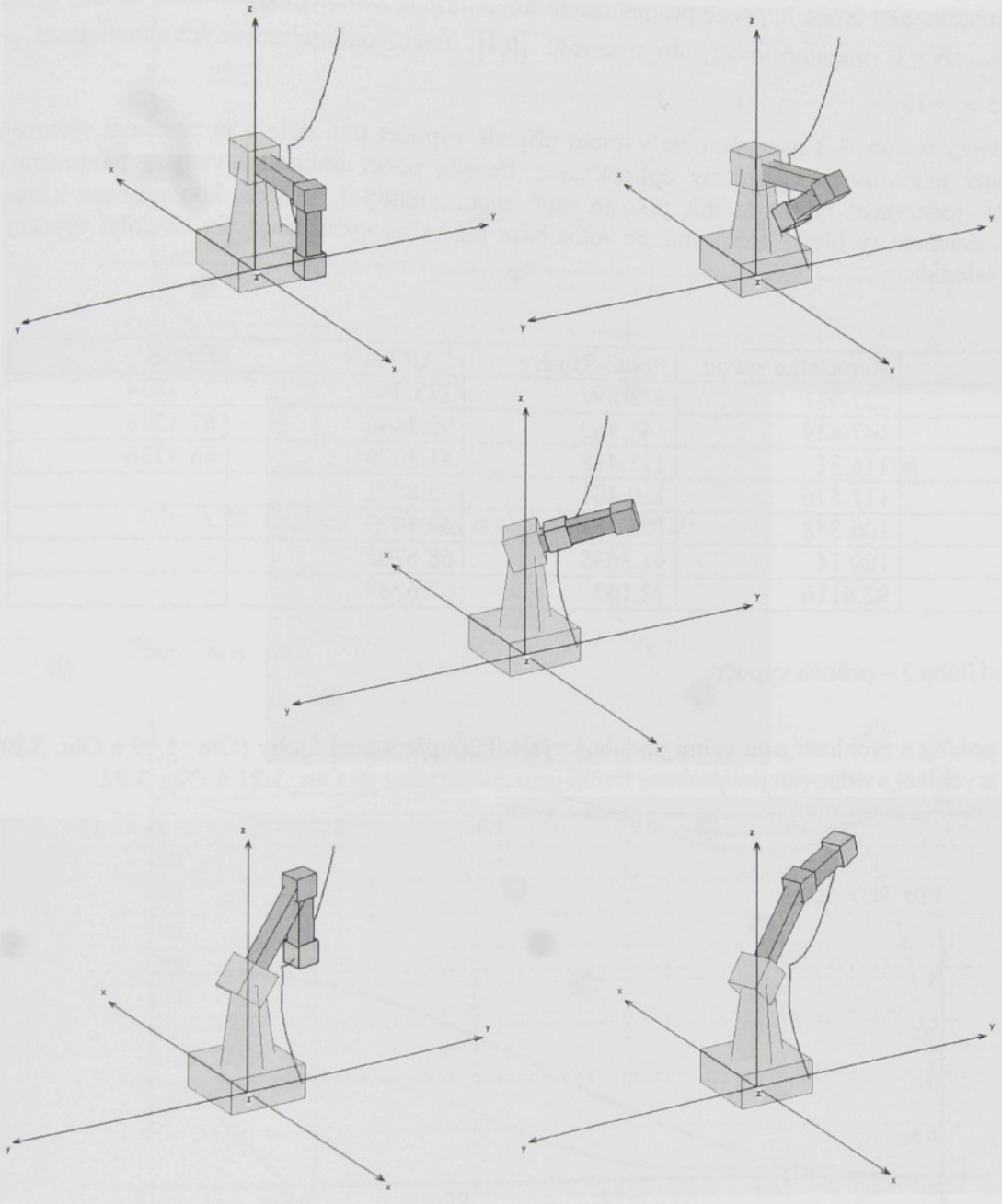


Obr. 3.16: Úloha 2 - průběhy zrychlení pro $N = 40$



Obr. 3.17: Úloha 2 - průběhy řízení pro $N = 40$

Pro návrh optimální trajektorie však nebyl uvažován omezený rozsah polohy členů. Výsledný pohyb robota v prostoru je proto nereálný, jak znázorňuje Obr. 3.18.



Obr. 3.18: Úloha 2 – pohyb robota v prostoru

3. Úloha

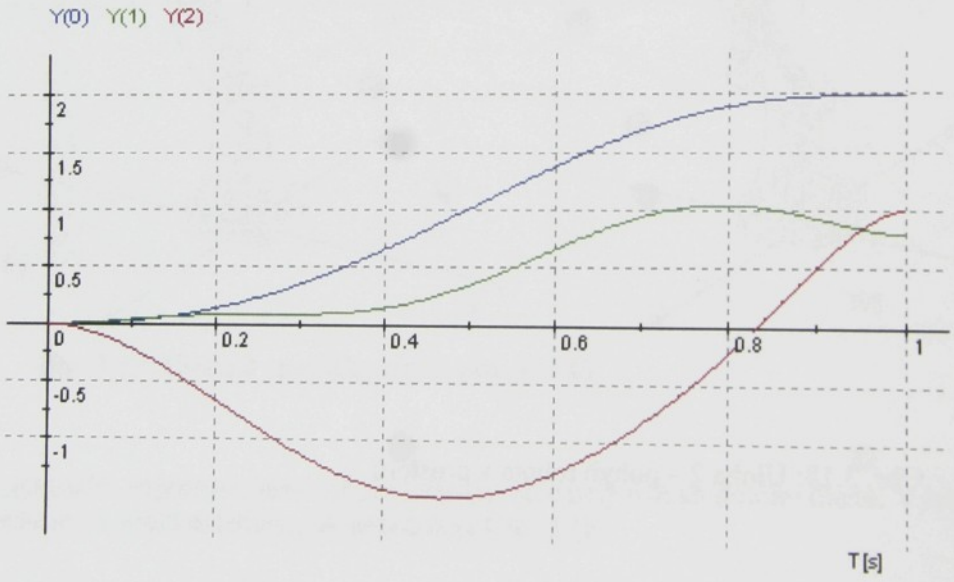
Stejný problém jako úloha 2, pouze pro nahrazení trajektorie je zvolen polynomiální bázový systém $\{t^i\}, i = 0,...,N-1$, transformovaný do intervalu $[0,1]$. Počet optimalizovaných koeficientů pro $N = 10$ je $n_z = 18$.

Hodnoty v Tab. 3-3 naznačují, že v tomto případě výpočet trvá déle a je mnohem výraznější rozdíl mezi jednotlivými algoritmy optimalizace, třebaže počet optimalizovaných parametrů je menší. Pro jednodušší typy systémů, jako je např. robot z úlohy 1, je však konvergence k řešení zpravidla mnohem rychlejší. Je zřejmé, že volba báze má velmi významný vliv na dobu výpočtu a kvalitu výsledků.

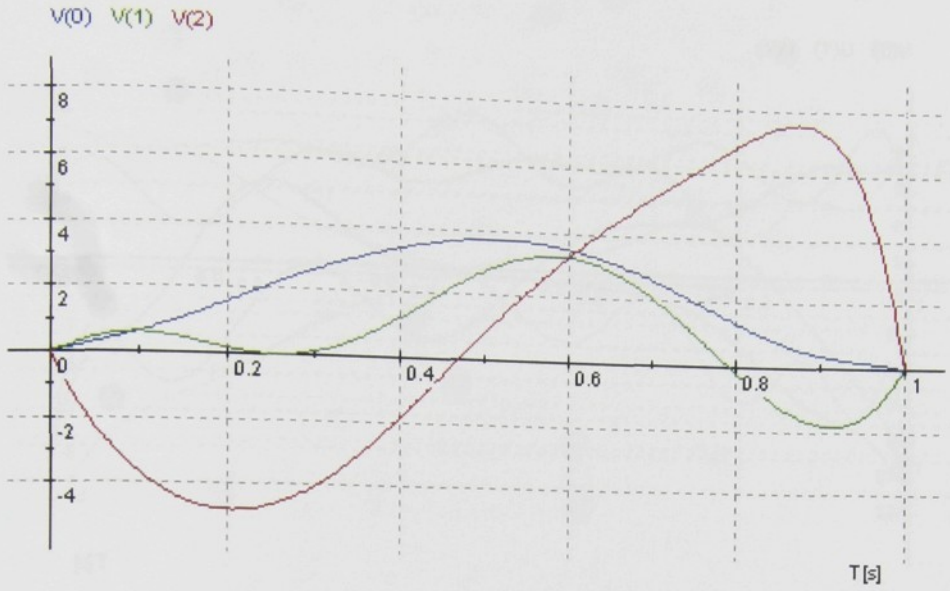
Iterace	Největšího spádu	Polak-Ribière	PARTAN	BFGS
20	327.081	323.897	123.51	72.7834
50	147.439	144.862	90.3446	67.1218
100	116.31	115.418	81.3079	65.7736
200	111.576	110.401	72.8571	-
500	106.358	102.142	69.1423	-
1000	100.14	91.3635	68.6702	-
2000	92.6116	72.162	67.6249	-

Tab. 3-3: Úloha 3 – průběh výpočtu

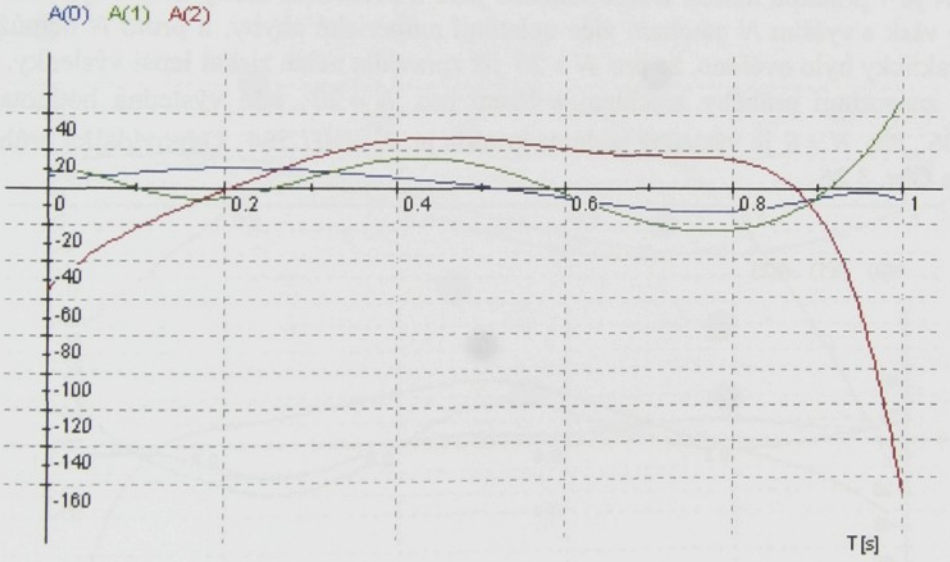
Průběhy polohy a rychlosti jsou velmi podobné výsledkům předchozí úlohy (Obr. 3.19 a Obr. 3.20). Průběhy zrychlení a odpovídající průběhy řízení jsou znázorněny na Obr. 3.21 a Obr. 3.22.



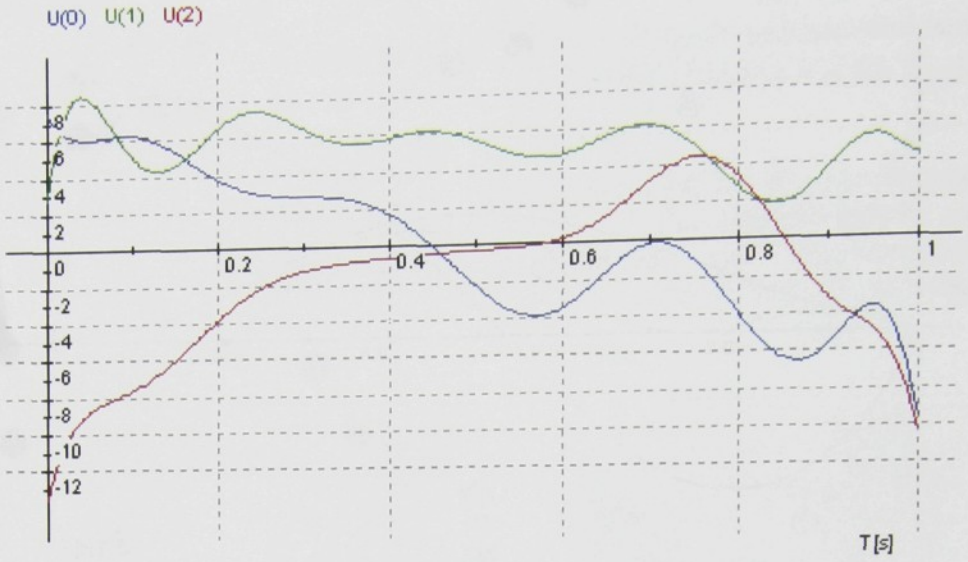
Obr. 3.19: Úloha 3 - průběhy polohy pro $N = 10$



Obr. 3.20: Úloha 3 - průběhy rychlosti pro $N = 10$

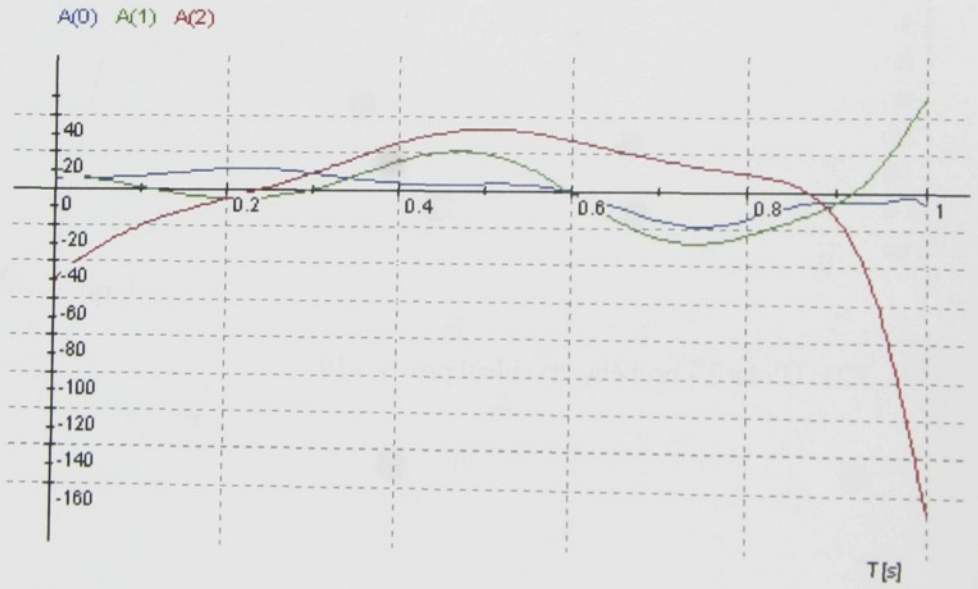


Obr. 3.21: Úloha 3 - průběhy zrychlení pro $N = 10$

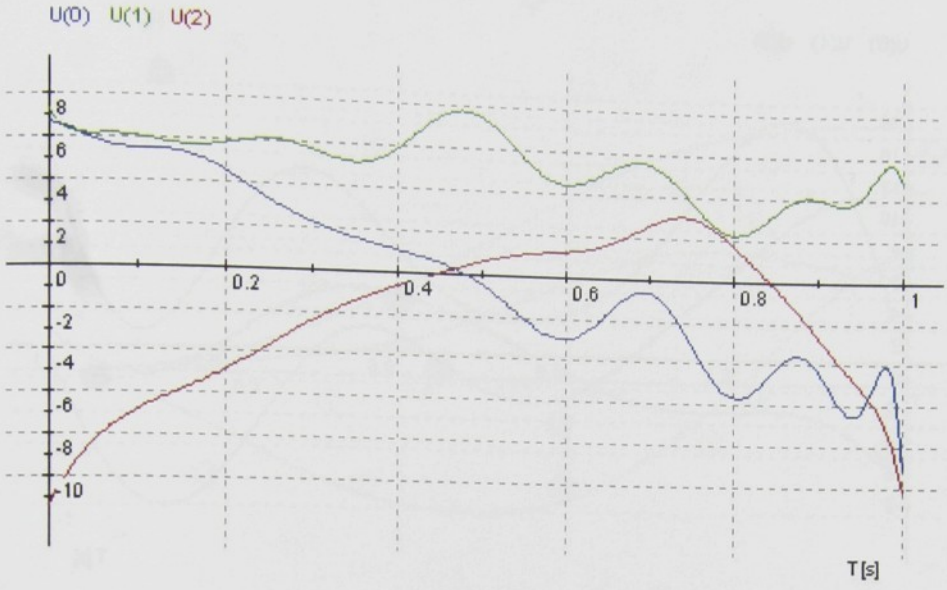


Obr. 3.22: Úloha 3 - průběhy řízení pro $N = 10$

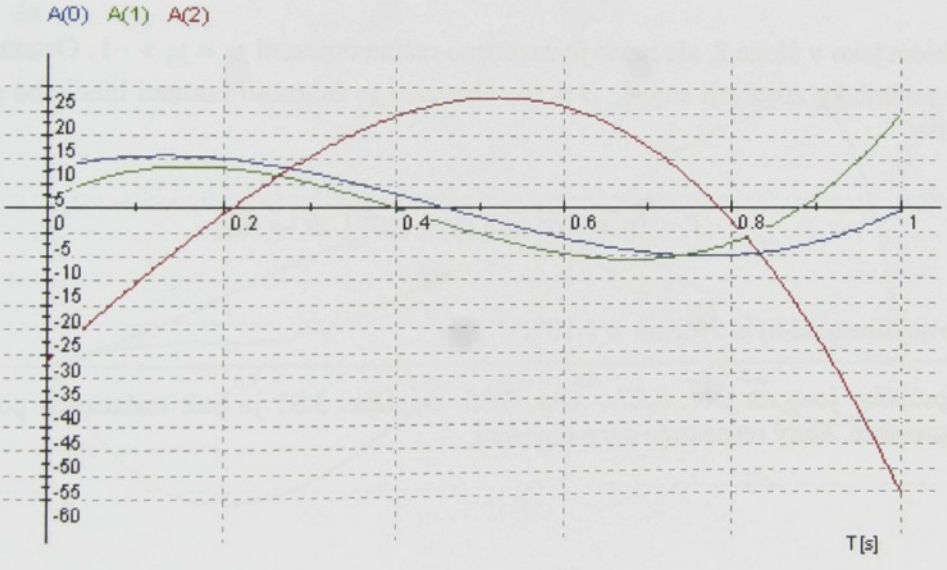
Pro vyšší N je v principu možné získat obdobně jako u předchozí úlohy kvalitnější řešení. V tomto případě se však s vyšším N mnohem více uplatňují numerické chyby, a proto N nemůže být příliš vysoké. Prakticky bylo ověřeno, že pro $N > 20$ již zpravidla nelze získat lepší výsledky. Obr. 3.23 a Obr. 3.24 znázorňují průběhy zrychlení a řízení pro $N = 20$, kde výsledná hodnota kritéria je $J^* = 62.006$. Pro $N = 6$ je výsledná hodnota kritéria je $J^* = 107.564$. Odpovídající průběhy jsou na Obr. 3.25 a Obr. 3.26.



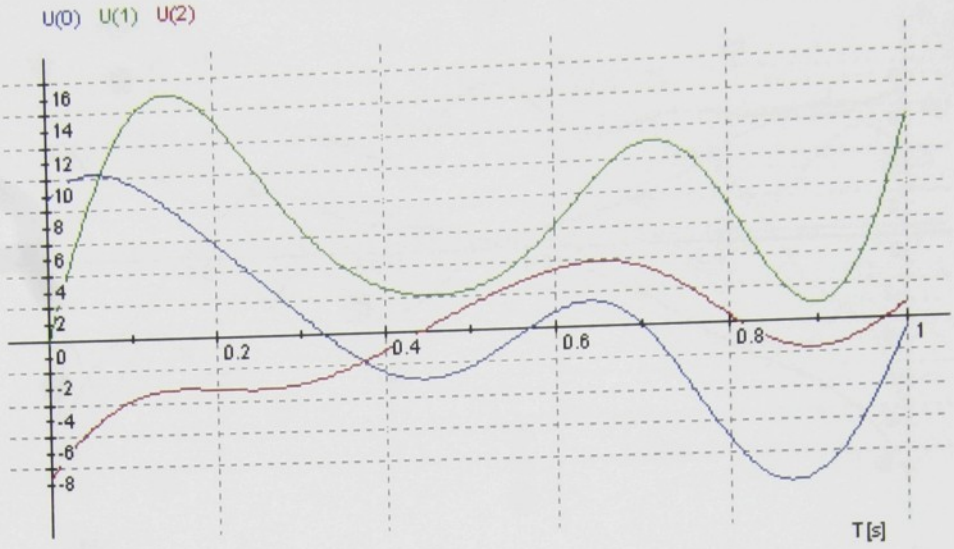
Obr. 3.23: Úloha 3 - průběhy řízení pro $N = 20$



Obr. 3.24: Úloha 3 - průběhy řízení pro $N = 20$



Obr. 3.25: Úloha 3 - průběhy řízení pro $N = 6$



Obr. 3.26: Úloha 3 - průběhy řízení pro $N = 6$

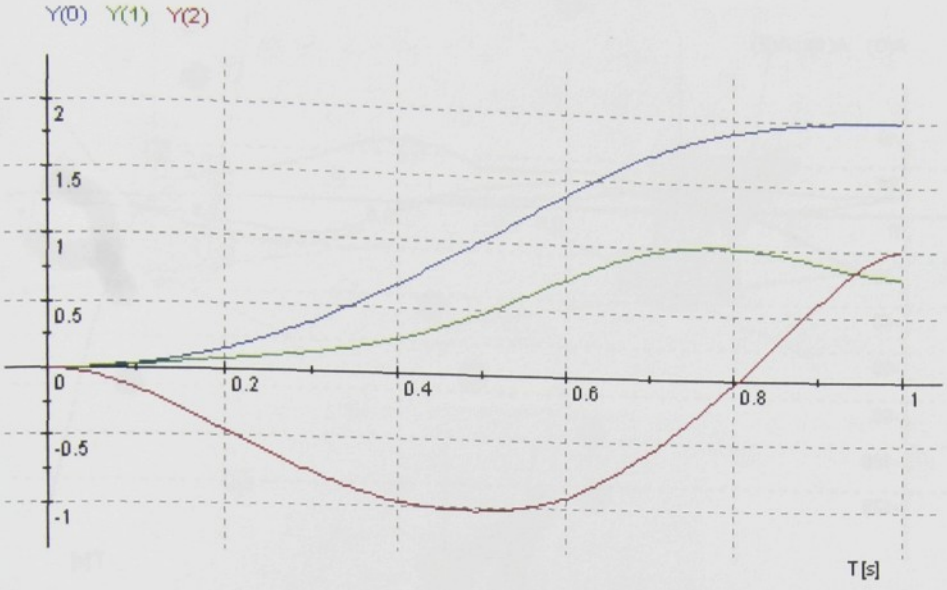
4. Úloha

Stejný problém jako v úloze 2, ale navíc je zavedeno reálné omezení $\psi \equiv y_3 > -1$. Ostatní parametry a okrajové podmínky zůstávají stejné, $N = 20$. Omezení je zahrnuto pomocí hladkého pokutového členu kritéria:

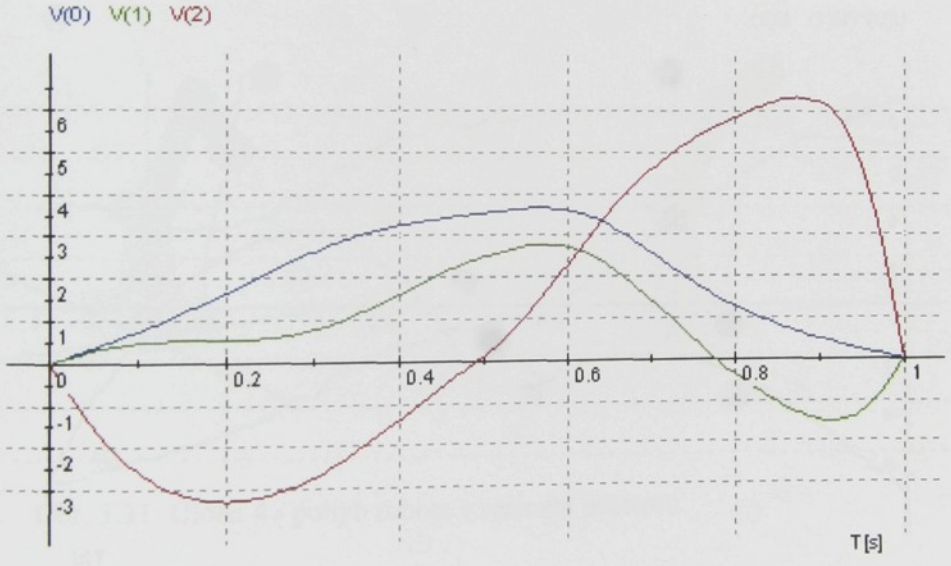
$$J' = \int_0^1 \mathbf{u}^T \mathbf{u} + \sigma \left[\min \left\{ y_3 + 1, 0 \right\} \right]^2 dt \to \min \tag{3.149}$$

kde pokutová konstanta byla zvolena $\sigma = 10^4$.

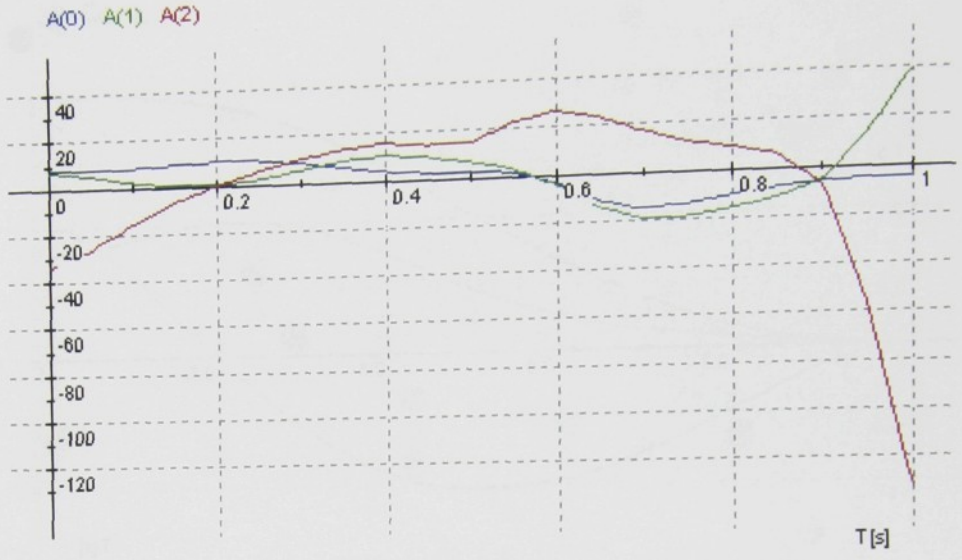
Výsledné průběhy jsou na Obr. 3.27 - Obr. 3.30. Na Obr. 3.31 je pak znázorněn pohyb robota v reálném prostoru, který respektuje dané omezení.



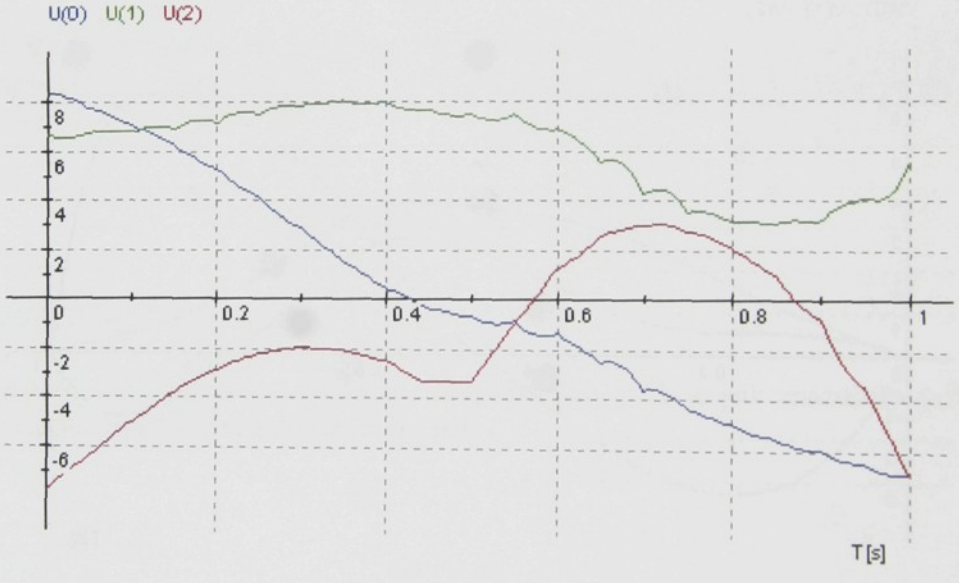
Obr. 3.27: Úloha 4 - průběhy polohy



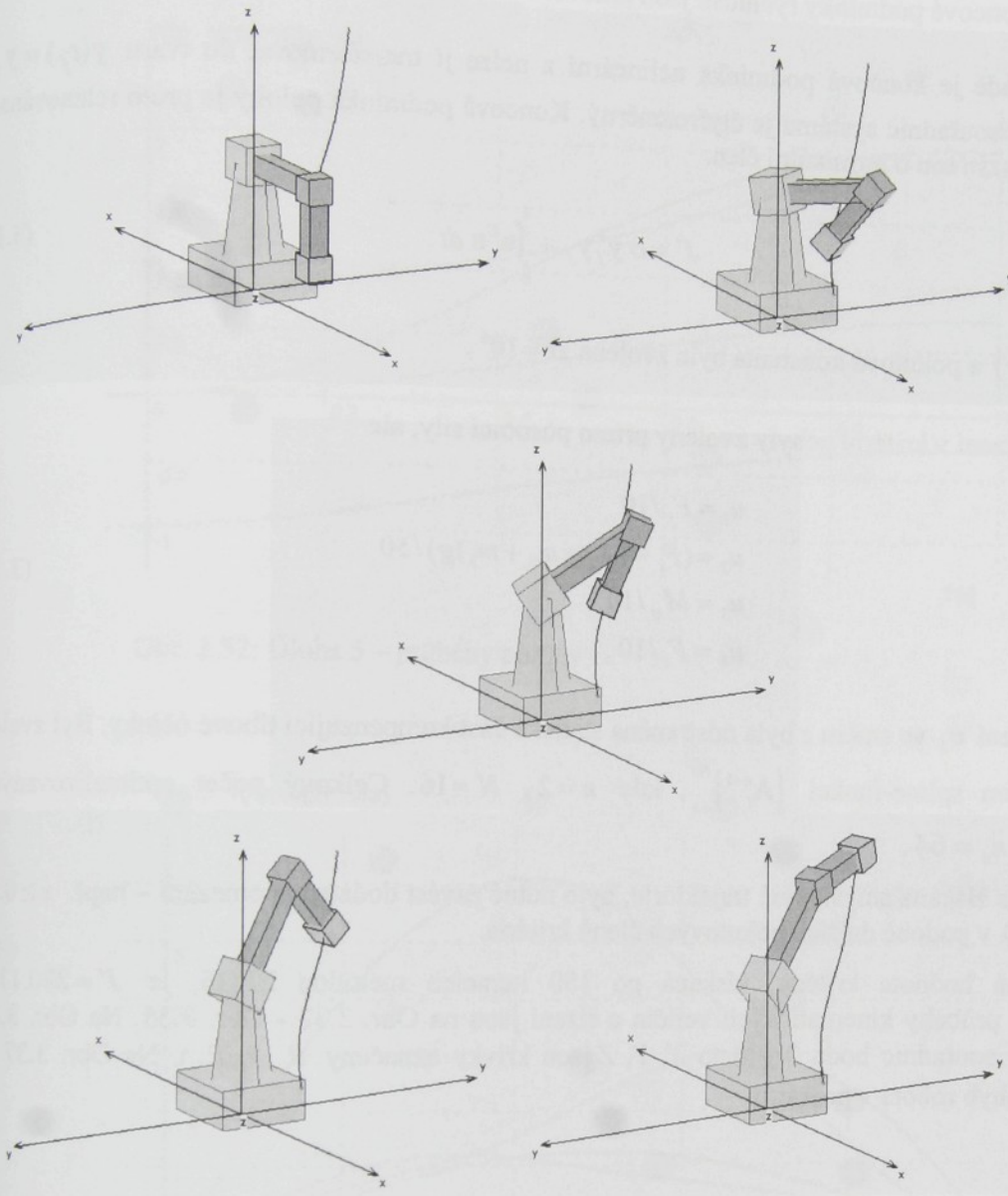
Obr. 3.28: Úloha 4 - průběhy rychlosti



Obr. 3.29: Úloha 4 - průběhy zrychlení



Obr. 3.30: Úloha 4 - průběhy řízení



Obr. 3.31: Úloha 4 - pohyb robota v reálném prostoru

5. Úloha

Model č.3, kap. 7.3, robota cylindrického typu na pojezdu. Je zadána koncová podmínka bodu **3** v kartézských souřadnicích:

$$\begin{aligned} X_3 &= -1 \\ Y_3 &= 1.5 \\ Z_3 &= 1.5\,. \end{aligned} \tag{3.150}$$

Počáteční podmínka polohy je v souřadnicích systému

$$\mathbf{y}_0 = (y, z, \varphi, x)^T(0) = (-1, 0, 0, 0.8)^T\,. \tag{3.151}$$

Počáteční a koncové podmínky rychlosti jsou nulové.

V tomto případě je koncová podmínka nelineární a nelze ji transformovat do tvaru $\mathbf{y}(t_f) = \mathbf{y}_f$, neboť prostor souřadnic systému je čtyřrozměrný. Koncová podmínka polohy je proto relaxována a kritérium je rozšířeno o terminální člen:

$$J' = \sigma \mathbf{y}_f^T \mathbf{y}_f + \int_0^1 \mathbf{u}^T \mathbf{u}\, dt\,. \tag{3.152}$$

kde $\mathbf{y}_f = \mathbf{y}(t_f)$ a pokutová konstanta byla zvolena $\sigma = 10^4$.

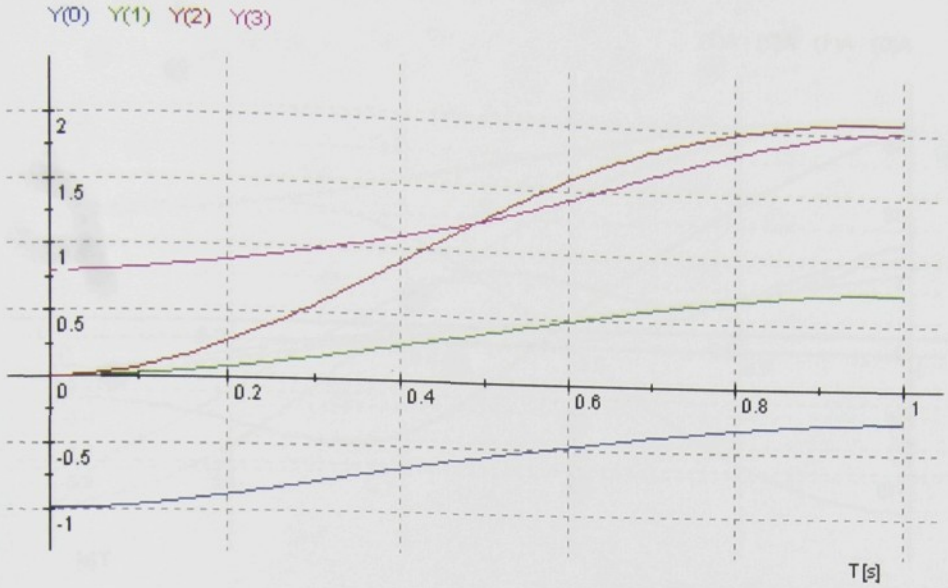
Jako složky řízení v kritériu nebyly zvoleny přímo působící síly, ale

$$\begin{aligned} u_1 &= F_y / 10 \\ u_2 &= (F_z - (m_2 + m_3 + m_3)g)/50 \\ u_3 &= M_\varphi / 10 \\ u_4 &= F_x / 10\,. \end{aligned} \tag{3.153}$$

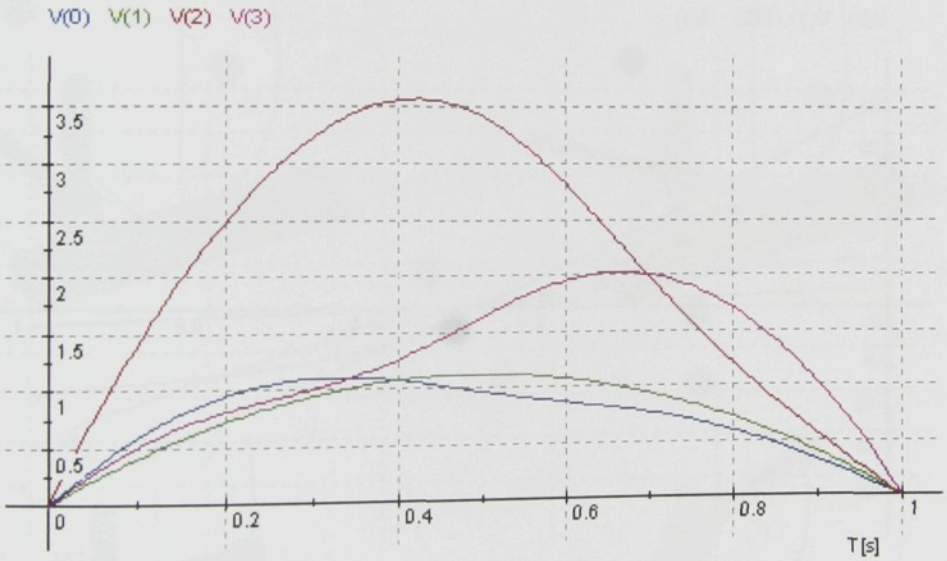
Ze složky řízení u_2 ve směru z byla odstraněna statická část kompenzující tíhové účinky. Byl zvolen bázový systém spline-funkcí $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$, kde $n = 2$, $N = 16$. Celkový počet optimalizovaných parametrů je $n_z = 64$.

Aby byla získána smysluplná trajektorie, bylo nutné zavést dodatečná omezení – např. $x \geq 0.3$, $x \leq 2$ a $\varphi \geq 0$ v podobě dalších pokutových členů kritéria.

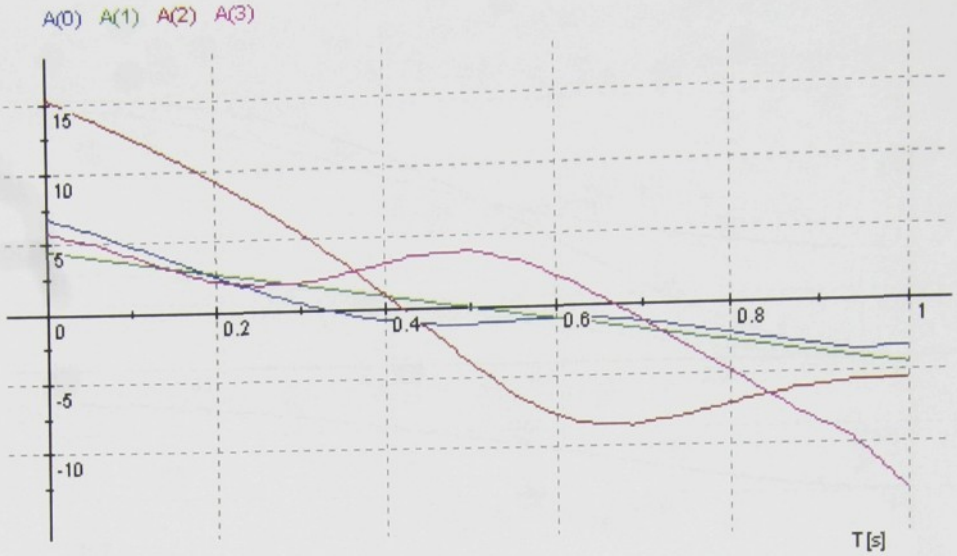
Výsledná hodnota kritéria, získaná po 150 iteracích metodou BFGS, je $J' = 280.113$. Odpovídající průběhy kinematických veličin a řízení jsou na Obr. 3.32 - Obr. 3.35. Na Obr. 3.36 jsou průběhy souřadnic bodu **3** (místo X, Y, Z jsou křivky označeny R_0, R_1, R_2). Na Obr. 3.37 je znázorněn pohyb robota v prostoru.



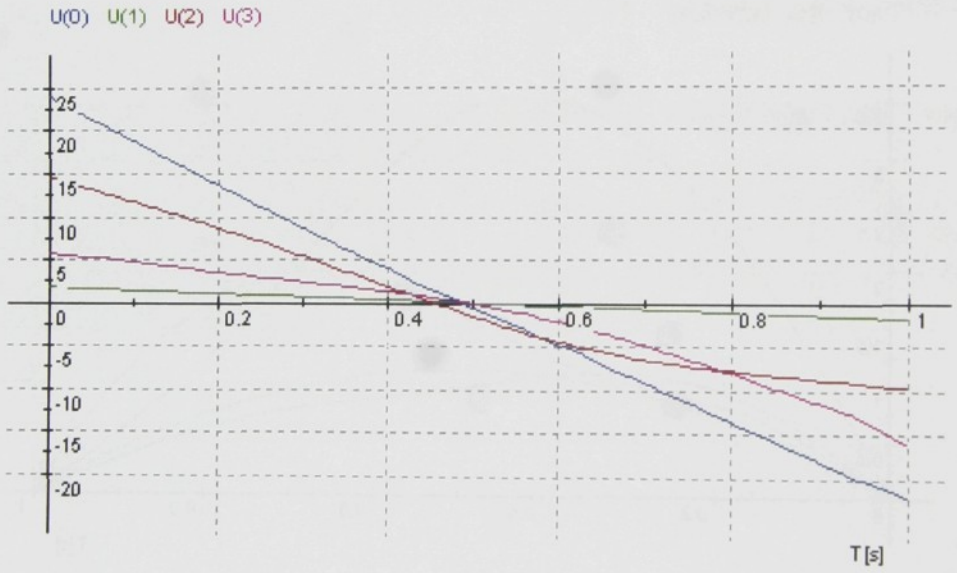
Obr. 3.32: Úloha 5 – průběhy polohy



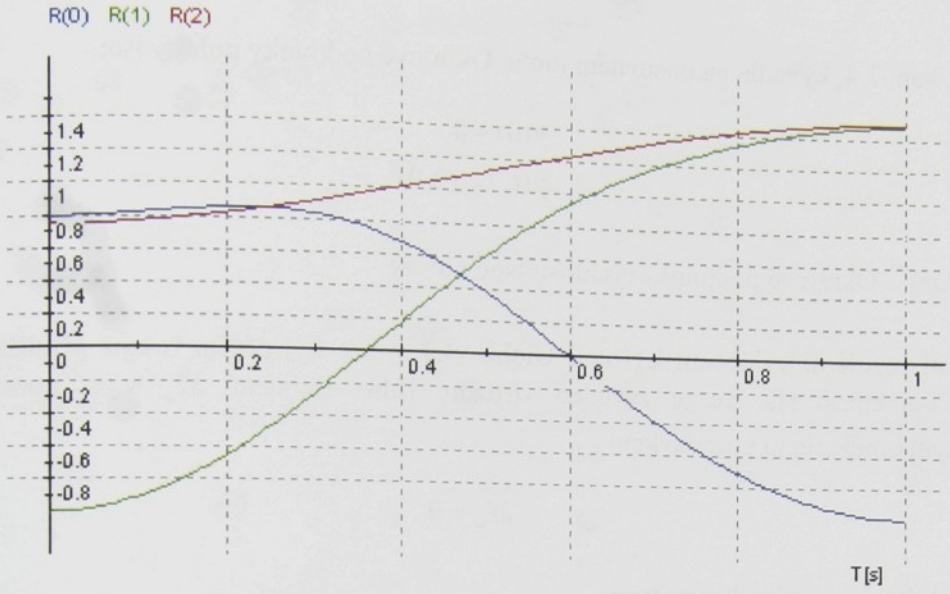
Obr. 3.33: Úloha 5 – průběhy rychlosti



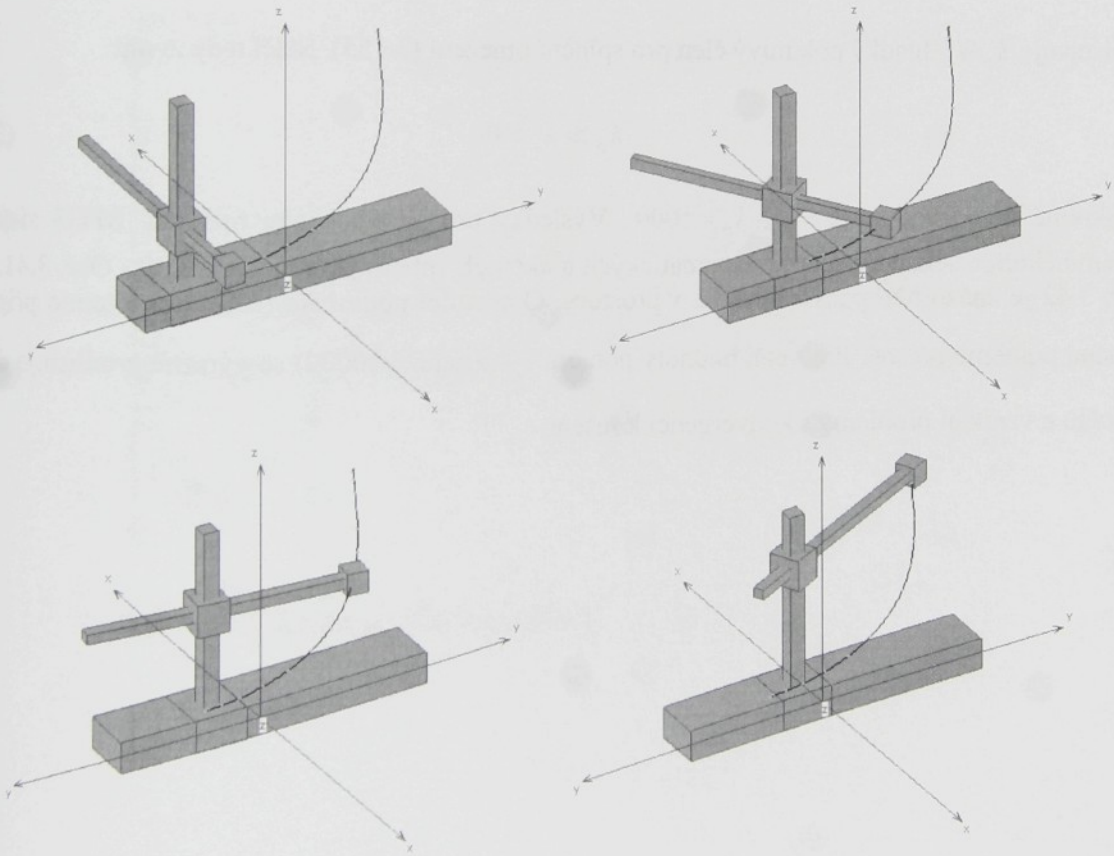
Obr. 3.34: Úloha 5 – průběhy zrychlení



Obr. 3.35: Úloha 5 – průběhy zrychlení



Obr. 3.36: Úloha 5 – průběh polohy koncového bodu



Obr. 3.37: Úloha 5 - pohyb robota v prostoru

6. Úloha

Model č.4, kap. 7.4, kyvadla na posuvném rámu. Okrajové podmínky polohy jsou:

$$\begin{aligned} \mathbf{y}(0) &= \mathbf{0} \\ \mathbf{y}(t_f) &= (2,0)^T \end{aligned} \tag{3.154}$$

kde $\mathbf{y} = (x,\varphi)^T$. Okrajové podmínky rychlosti jsou nulové.

Systém není regulární vzhledem k řízení. Úloha však může být přesto řešena přibližně metodou nahrazení trajektorie tak, že je zaveden virtuální řídicí moment M_φ v ose rotace členu **2** s dodatečným omezujícím požadavkem

$$M_\varphi = 0\,. \tag{3.155}$$

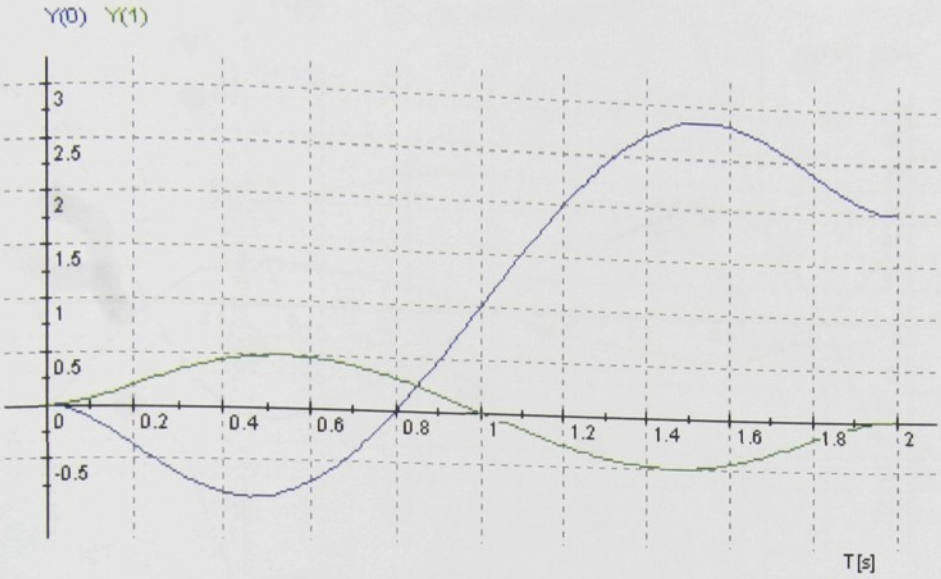
Zvolíme-li kritérium optimality ve tvaru

$$J = \int_0^2 k_x F_x^2 + k_\varphi M_\varphi^2 \, dt \to \min \tag{3.156}$$

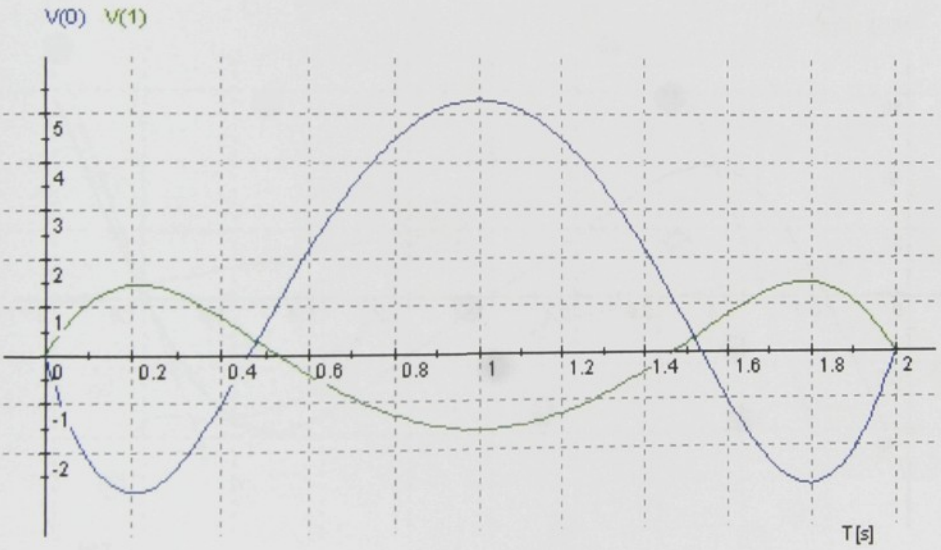
představuje $k_\varphi M_\varphi^2$ hladký pokutový člen pro splnění omezení (3.155). Stačí tedy zvolit

$$k_\varphi \gg k_x > 0\,. \tag{3.157}$$

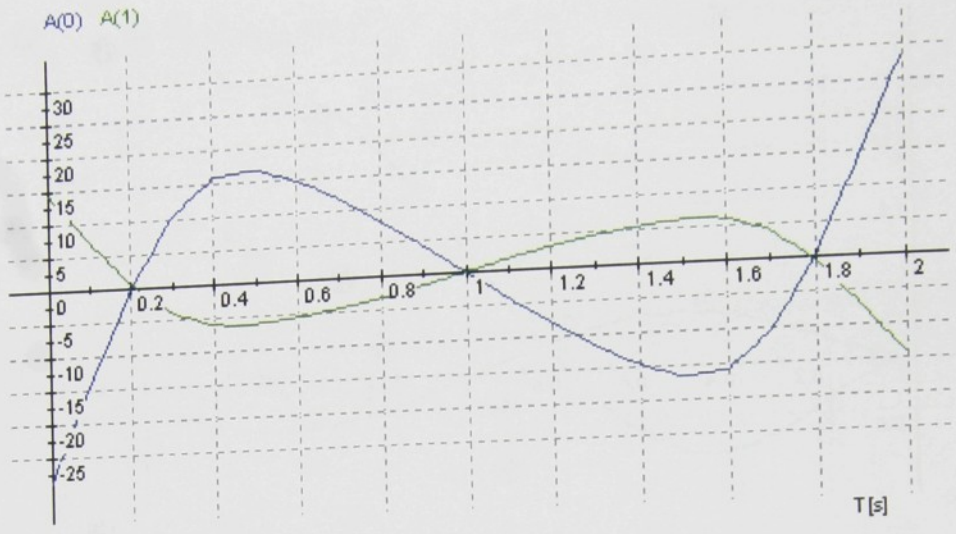
Konkrétně bylo zvoleno $k_x = 1$, $k_\varphi = 1000$. Výsledky pro $N = 20$ jsou metodou BFGS získány během několika sekund. Průběhy kinematických a akčních veličin jsou na Obr. 3.38 - Obr. 3.41. Na Obr. 3.42 je znázorněn pohyb kyvadla v prostoru. Omezující podmínka (3.155) je v tomto případě splněna poměrně přesně. Pro větší hodnoty poměru $\frac{k_\varphi}{k_x}$ (např. >10000) se výrazně prodlužuje doba výpočtu a vznikají problémy s konvergencí k řešení.



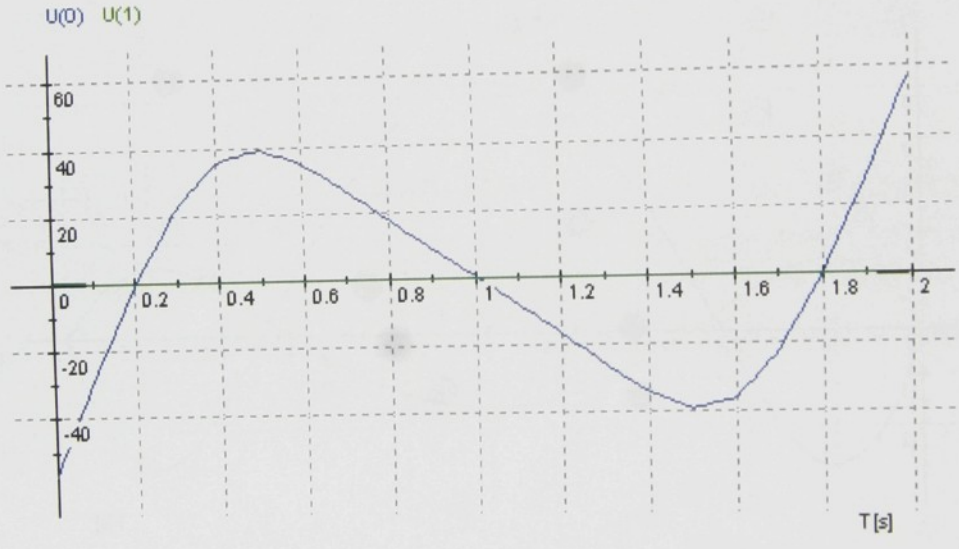
Obr. 3.38: Úloha 6 – průběhy polohy



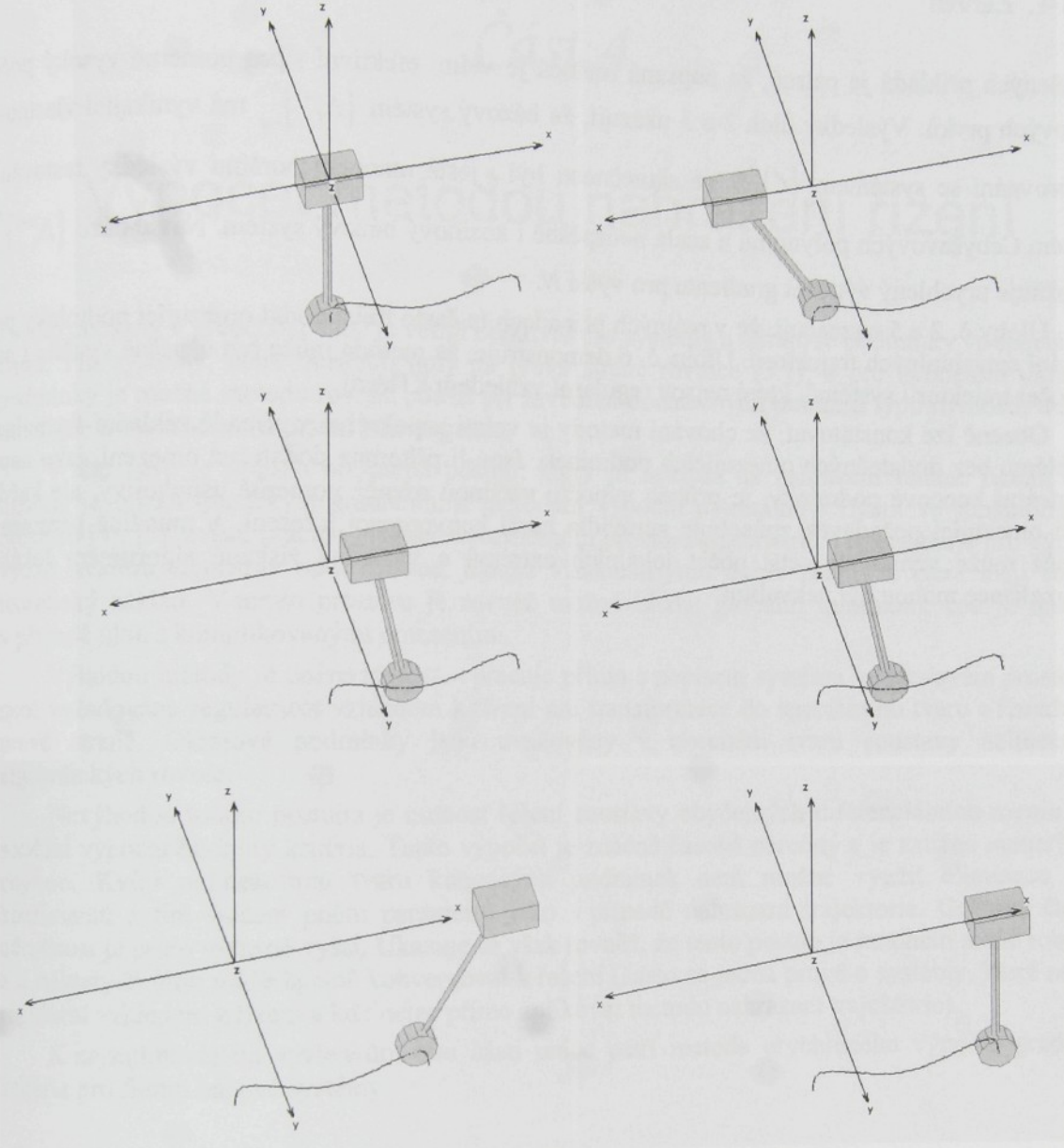
Obr. 3.39: Úloha 6 – průběhy rychlosti



Obr. 3.40: Úloha 6 – průběhy zrychlení



Obr. 3.41: Úloha 6 – průběhy řízení



Obr. 3.42: Úloha 6 – pohyb kyvadla v prostoru

3.14. Závěr

Z řešených příkladů je patrné, že popsaná metoda je velmi efektivní i pro poměrně vysoký počet bázových prvků. Výsledky úloh 2 a 3 ukazují, že bázový systém $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$ má vynikající vlastnosti v porovnání se systémem $\left\{t^i\right\}_0^N$. Ve skutečnosti byl s ještě mnohem horšími výsledky testován i systém Čebyševových polynomů a zcela neúspěšně i kosinový bázový systém. Navíc báze $\left\{\Lambda_i^{n+1}\right\}_{-n}^{N}$ umožňuje urychlený výpočet gradientu pro vyšší N.

Úlohy č. 2 a 5 naznačují, že v reálných případech je často třeba dodat omezující podmínky pro získání smysluplných trajektorií. Úloha č. 6 demonstruje, že metoda může být úspěšně využita i pro výpočet trajektorií systémů, které nejsou regulární vzhledem k řízení.

Obecně lze konstatovat, že chování metody je velmi uspokojivé v případě základní formulace problému bez dodatečných omezujících podmínek. Jsou-li přítomna dodatečná omezení, jako např. nelineární koncové podmínky, je průběh výpočtu většinou rovněž poměrně uspokojivý, ale každý další omezující požadavek způsobuje zpravidla horší konvergenci k řešení. V množině parametrů rovněž může vzniknout větší počet lokálních extrémů a výsledky získané algoritmem lokální optimalizace mohou být nekvalitní.

Část 4.

Výpočet metodou nahrazení řízení

Postup popsaný v předchozí části je velmi efektivní pro systémy a okrajové podmínky ve speciálním tvaru. Pro systémy, které obsahují nuly na pravé straně rovnice (3.1), a pro nelineární okrajové podmínky je možné metodu rovněž použít při zavedení dodatečných omezení typu rovností, třebaže tím dojde k výraznému zhoršení konvergence.

V této části je popsán obecnější postup, který je založen na nahrazení funkce řízení. Tato metoda je blízká klasickým gradientním metodám výpočtu optimálního řízení ve funkcionálním prostoru [4], [5], avšak pracuje v konečněrozměrném prostoru parametrů. To umožňuje pro výpočet využít kvalitní algoritmy optimalizace, jejichž vlastnosti jsou dobře známy a které mají solidní teoretický základ. V tomto prostoru je rovněž možné hledat globální minimum, což je důležité v případě úloh s komplikovanými omezeními.

Výhodou metody je univerzálnost – pracuje přímo s popisem systému ve stavovém prostoru a není vyžadována regulárnost vzhledem k řízení ani transformace do speciálního tvaru s řízením na pravé straně. Okrajové podmínky jsou uvažovány v obecném tvaru soustavy nelineárních algebraických rovnic.

Nevýhodou tohoto postupu je nutnost řešení soustavy obyčejných diferenciálních rovnic jako součást výpočtu hodnoty kritéria. Tento výpočet je značně časově náročný a je zatížen numerickou chybou. Kvůli nelineárnímu tvaru koncových podmínek není možné využít eliminace části koeficientů a tím snížení počtu parametrů jako v případě nahrazení trajektorie. Celková časová náročnost je proto výrazně vyšší. Ukazuje se však rovněž, že tento postup je mnohem méně robustní a u některých úloh může špatně konvergovat k řešení (často se jedná právě o systémy, které nejsou regulární vzhledem k řízení a kde nelze přímo aplikovat metodu nahrazení trajektorie).

K nejzajímavějším výsledkům této části práce patří metoda urychleného výpočtu gradientu kritéria pro finitní bázové systémy.

4.1. Formulace problému

Uvažujme systém popsaný soustavou obyčejných diferenciálních rovnic

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},t)\,. \tag{4.1}$$

Řád vektoru stavu $\mathbf{x}$ označme n, řád vektoru řízení $\mathbf{u}$ m, $m \le n$.

Okrajové podmínky nechť jsou ve tvaru

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{x}_0 \\ \boldsymbol{\psi}(\mathbf{x}(t_f)) &= \mathbf{0}\,. \end{aligned} \tag{4.2}$$

Uvažujme rovněž omezení typu rovností a nerovností, která platí v každém čase $t \in \left[0, t_f\right]$:

$$\mathbf{g}(\mathbf{x},\mathbf{u},t) \geq \mathbf{0} \tag{4.3}$$
$$\mathbf{h}(\mathbf{x},\mathbf{u},t) = \mathbf{0}. \tag{4.4}$$

Úkolem je nalézt integrovatelnou funkci řízení takovou, že funkcionál

$$J = \int_0^{t_f} f_0(\mathbf{x},\mathbf{u},t)\, dt \rightarrow \min . \tag{4.5}$$

Předpokládá se, že funkce $\mathbf{f}, f_0, \mathbf{g}, \mathbf{h}$ jsou spojitě diferencovatelné podle svých parametrů $\mathbf{x}$, $\mathbf{u}$ (ψ podle $\mathbf{x}(t_f)$) a po částech spojité podle času. Čas t_f uvažujme nejprve jako předem známý (tj. pevný). Řešení úloh s volným časem je diskutováno v kap. 4.4.

4.2. Princip metody

Nahraďme funkci řízení pro každou složku váženým součtem bázových funkcí $\{b_0,...,b_N\}$, tj.

$$u_i(t) = \sum_{j=0}^{N} a_{ij}\, b_j(t) \tag{4.6}$$

kde $\mathbf{A} = (a_{ij})$ je matice koeficientů.

V dalším textu je využíván vektorový zápis:

$$\mathbf{u}(t) = \mathbf{A}\,\mathbf{b}(t). \tag{4.7}$$

Dosaďme (4.7) do (4.5):

$$J = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{A}\mathbf{b}(t), t)\, dt . \tag{4.8}$$

Tento vztah lze rovněž zapsat jako

$$\frac{dJ}{dt} = f_0(\mathbf{x}, \mathbf{A}\mathbf{b}(t), t)$$
$$J(0) = 0. \tag{4.9}$$

Rovnice (4.9) spolu s pohybovými rovnicemi systému tvoří rozšířenou soustavu

$$\frac{d\hat{\mathbf{x}}}{dt} = \hat{\mathbf{f}}(\hat{\mathbf{x}}, \mathbf{u}, t) \tag{4.10}$$

kde

$$\hat{\mathbf{x}} = \begin{pmatrix} J \\ \mathbf{x} \end{pmatrix}, \quad \hat{\mathbf{f}} = \begin{pmatrix} f_0 \\ \mathbf{f} \end{pmatrix}. \tag{4.11}$$

Rovnice (4.10) jsou využity pro výpočet hodnoty kritéria na základě dané matice **A**. Kritérium je tedy funkcí **A** a jeho minimum je hledáno v prostoru složek této matice

$$J^* = \min\{J(\mathbf{A})\}. \tag{4.12}$$

Nejsou-li uvažována omezení (4.3) a (4.4), je problém (4.12) s koncovými podmínkami (4.2) úlohou nelineárního programování, neboť $\mathbf{x}(t_f)$ závisí na **A** prostřednictvím řešení soustavy (4.10). Na rozdíl od metody nahrazení trajektorie je nelineární omezující podmínka typu rovnosti vždy přítomná. Tato omezení je možné zahrnout pokutovou metodou do terminální části kritéria

$$J = \sigma \left\| \boldsymbol{\psi}(\mathbf{x}(t_f)) \right\|_{\mathbf{K}}^2 + \int_0^{t_f} f_0(\mathbf{x},\mathbf{u},t)\, dt \tag{4.13}$$

kde **K** je diagonální matice s kladnými prvky a $\sigma > 0$ pokutová konstanta. V tomto případě však může být rychlejší konvergence k řešení dosaženo pomocí přímých metod nelineárního programování, jako např. SQP (viz část 6, kap. 6.2). Omezení typu rovností a nerovností v každém bodě (4.3) a (4.4) lze snadno zahrnout pokutovou metodou do kritéria, obdobně jako u metody nahrazení trajektorie, viz kap. 3.4.

Z výše uvedených závěrů vyplývá, že při použití pokutových členů je možné se v dalších úvahách omezit na minimalizaci kritéria ve tvaru

$$J = \varphi(\mathbf{x}_f) + \int_0^{t_f} f_0(\mathbf{x},\mathbf{u},t)\, dt. \tag{4.14}$$

4.3. Teoretické opodstatnění metody

Je možné považovat za teoretický nedostatek, že optimální funkce řízení, která je na základě výsledků variačního počtu a teorie optimálního řízení pouze po částech spojitá [3] (tj. má na intervalu $[0,t_f]$ konečný počet bodů nespojitosti), nemůže být aproximována s libovolnou přesností pomocí váženého součtu posloupnosti spojitých funkcí ve smyslu normy

$$\|\mathbf{h}(.)\| = \max_{t\in[0,t_f]} \|\mathbf{h}(t)\|. \tag{4.15}$$

Cílem však je ve skutečnosti nalézt trajektorii (tj. průběh **x**), která je dostatečně blízká extremále, a odpovídající průběh řízení je pouze vedlejším výsledkem.

Dále je ukázáno, že i když optimální průběh $\mathbf{u}^*(t)$ je pouze po částech spojitý, pro dostatečně vysoké N lze nalézt spojitou funkci $\mathbf{u}(t)$, jejíž složky lze vyjádřit součtem (4.6), že trajektorie $\mathbf{x}(t)$ a $\mathbf{x}^*(t)$ jsou libovolně blízké ve smyslu metriky $\max_{t\in[0,t_f]} \|\mathbf{x}(t) - \mathbf{x}^*(t)\|$. Tím je tento postup teoreticky opodstatněn.

Uvažujme nejprve, že daná optimální funkce řízení $\mathbf{u}^*(t)$ je spojitá. Předpokládejme, že bázový systém má tu vlastnost, že pro každé $\delta > 0$ lze nalézt N a bázové koeficienty a_{ij} v součtu (4.6) takové, že platí

$$\max_{t\in[0,t_f]} \|\mathbf{u}(t) - \mathbf{u}^*(t)\| < \delta. \tag{4.16}$$

Takový bázový systém jistě existuje, neboť prostor spojitých funkcí na intervalu $\left[0,t_f\right]$ je separabilní [2] (speciálně, každá spojitá funkce na intervalu $\left[0,t_f\right]$ lze podle Weierstrassovy věty [10] s libovolnou přesností aproximovat polynomem). Na základě vět o spojité závislosti řešení soustav diferenciálních rovnic na počátečních podmínkách a parametru, např. [9], lze ke každému $\varepsilon > 0$ nalézt $\delta > 0$ a $\nu > 0$, tedy i N tak, že

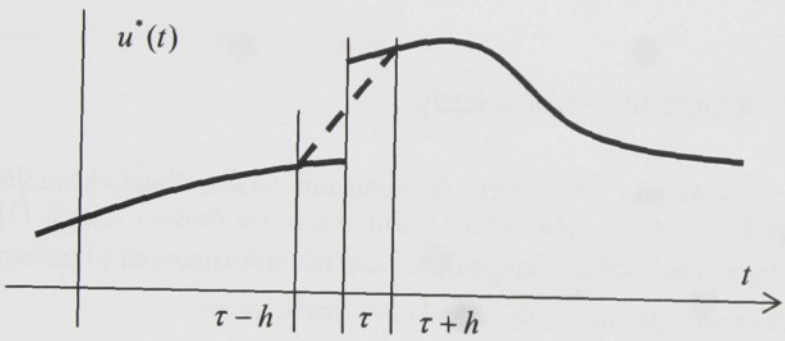
$$\max_{t\in\left[0,t_f\right]} \left\|\mathbf{x}(t) - \mathbf{x}^*(t)\right\| < \varepsilon \tag{4.17}$$

pokud platí (4.16) a $\left\|\mathbf{x}(0) - \mathbf{x}^*(0)\right\| < \nu$.

V případě, že $\mathbf{u}^*(t)$ je pouze po částech spojitá, (4.17) nemusí platit, neboť odchylka $\delta\mathbf{u}$ je v bodech nespojitosti velká. Pro jednoduchost uvažujme nejprve pouze jediný bod nespojitosti τ. Předpokládejme, že bázový systém má tu vlastnost, že pro libovolné $\delta > 0$ a $h > 0$ lze nalézt N tak, že platí

$$\left\|\mathbf{u}(t) - \mathbf{u}^*(t)\right\| < \delta \text{ pro } t \in \left[0, \tau - h\right] \cup \left[\tau + h, t_f\right]. \tag{4.18}$$

To je jistě možné, neboť nahradíme-li průběh funkce $\mathbf{u}^*(t)$ v intervalu $\left[\tau - h, \tau + h\right]$ úsečkou, která spojuje body $\mathbf{u}^*(\tau - h)$ a $\mathbf{u}^*(\tau + h)$, je výsledná funkce spojitá (Obr. 4.1).



Obr. 4.1: Nahrazení po částech spojitého průběhu

Označme $t_1 = \tau - h$, $t_2 = \tau + h$. Pro hodnoty trajektorie v bodě t_2 platí

$$\mathbf{x}(t_2) = \mathbf{x}(t_1) + \int_{t_1}^{t_2} \mathbf{f}(\mathbf{x},\mathbf{u},t)\,dt \tag{4.19}$$

$$\mathbf{x}^*(t_2) = \mathbf{x}^*(t_1) + \int_{t_1}^{t_2} \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\,dt\,. \tag{4.20}$$

Z (4.19), (4.20) je možné vyjádřit maximální odchylku trajektorie v bodě t_2:

$$\delta\mathbf{x}(t_2) = \mathbf{x}(t_2) - \mathbf{x}^*(t_2) = \delta\mathbf{x}(t_1) + \int_{t_1}^{t_2} \left[\mathbf{f}(\mathbf{x},\mathbf{u},t) - \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\right] dt \tag{4.21}$$

$$\begin{aligned}\|\delta\mathbf{x}(t_2)\| &\le \|\delta\mathbf{x}(t_1)\| + \int_{t_1}^{t_2} \|\mathbf{f}(\mathbf{x},\mathbf{u},t) - \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\|\, dt \le \\ &\le \|\delta\mathbf{x}(t_1)\| + \int_{t_1}^{t_2} \max_{t\in[t_1,t_2]} \|\mathbf{f}(\mathbf{x},\mathbf{u},t) - \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\|\, dt = \\ &= \|\delta\mathbf{x}(t_1)\| + h \max_{t\in[t_1,t_2]} \|\mathbf{f}(\mathbf{x},\mathbf{u},t) - \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\|.\end{aligned} \tag{4.22}$$

Přitom pro libovolné $\varepsilon > 0$ lze nalézt N takové, že $\|\delta\mathbf{x}_1\| < \varepsilon$, neboť funkce $\mathbf{u}^*$ je v intervalu $[0,t_1]$ spojitá. Potom pro každé $\varepsilon_2 > 0$ lze nalézt $h > 0$ a $\varepsilon > 0$, tj. také N tak, že platí $\|\delta\mathbf{x}_2\| < \varepsilon_2$ a tedy také

$$\|\delta\mathbf{x}\| < \varepsilon_2 \text{ pro } \forall t \in [0,t_2]. \tag{4.23}$$

Stačí tyto hodnoty zvolit tak, aby platilo

$$\varepsilon_2 > \varepsilon + h \max_{t\in[t_1,t_2]} \|\mathbf{f}(\mathbf{x},\mathbf{u},t) - \mathbf{f}(\mathbf{x}^*,\mathbf{u}^*,t)\|. \tag{4.24}$$

Jelikož funkce $\mathbf{u}^*$ je na intervalu $[t_2,t_f]$ spojitá a $\|\delta\mathbf{x}_2\| < \varepsilon_2$, lze pro libovolné $\varepsilon_3 > 0$ nalézt N takové, že $\|\delta\mathbf{x}\| < \varepsilon_3$ v celém intervalu $[0,t_f]$.

V případě, že optimální funkce řízení má více (ale konečný počet) bodů nespojitosti, lze platnost předchozích výsledků indukčním způsobem rozšířit i na tento případ. Tím bylo dokázáno, že optimální trajektorie může být dobře aproximována, je-li funkce řízení vyjádřena v bázovém systému spojitých funkcí.

4.4. Problémy s volným časem

Metoda je založena na apriorní znalosti času t_f. V mnoha praktických případech se však jedná o tzv. problém s volným časem, kdy doba pohybu t_f má být rovněž určena. Nejjednodušší možností je tuto úlohu řešit jako posloupnost úloh s pevným časem. Označíme-li $J^*(t_f)$ řešení úlohy s pevným časem t_f, pak řešení problému s volným časem je

$$J^* = \min\left\{J^*(t_f)\right\}. \tag{4.25}$$

Tento postup však mnohonásobně prodlužuje dobu výpočtu, která může být značná i v případě pevného času. V případě, že optimální čas t_f^* je konečný, existuje však transformace na úlohu s pevným časem [5]:

$$J = \int_0^1 f_0(\mathbf{x}(s),\mathbf{u}(s),t(s))\, v^2 ds \tag{4.26}$$

kde parametr s je novou nezávislou proměnnou, t novou stavovou proměnnou, v novou vstupní proměnnou a platí

$$\frac{dt}{ds} = v^2, \quad t(0) = 0 \tag{4.27}$$

$$\frac{d\mathbf{x}}{ds} = v^2 \mathbf{f}(\mathbf{x}(s), \mathbf{u}(s), t(s)). \tag{4.28}$$

Díky této transformaci není nutné se případem s volným časem zvlášť zabývat. Tento postup však dosud nebyl v rámci této práce prakticky realizován.

4.5. Numerický výpočet kritéria

Pro integraci soustavy (4.11) je nutné použít některou numerickou metodu řešení soustav obyčejných diferenciálních rovnic. Pro známý průběh řízení je

$$\hat{\mathbf{f}}(\hat{\mathbf{x}}, \mathbf{u}, t) = \tilde{\mathbf{f}}(\hat{\mathbf{x}}, t). \tag{4.29}$$

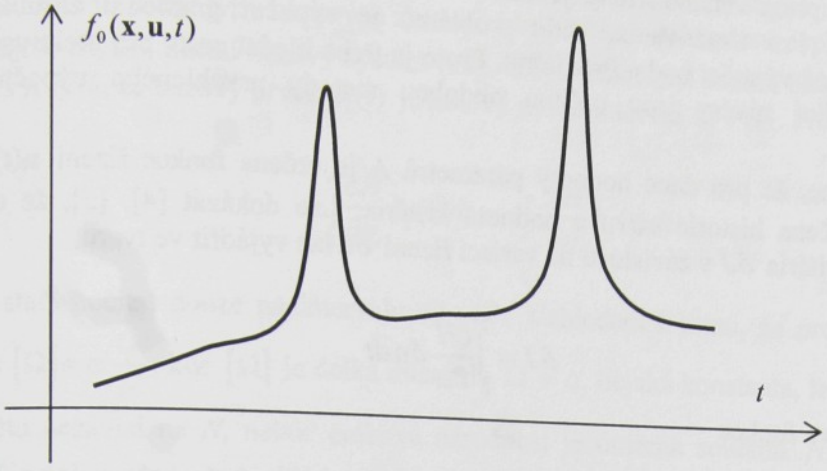
Integrace soustavy (4.11) je poměrně náročným výpočtem. Délka kroku proto musí být zvolena co největší, ale tak, aby současně nedošlo k výraznému zkreslení řešení. Vhodná délka kroku však obecně nelze odhadnout předem. Moderní metody pracují většinou s proměnným krokem, který se určuje automaticky v průběhu výpočtu na základě určitých chybových kritérií.

K nejefektivnějším metodám, které současně umožňují dynamické řízení délky kroku, jsou metody typu prediktor-korektor [14], které jsou založeny na interpolaci polynomu několika posledními body. Tyto metody jsou však poměrně komplikované z hlediska implementace. Musí být např. startovány nějakou jinou metodou, neboť pro určení nového bodu potřebují znalost několika předcházejících bodů. V praxi se nejčastěji využívají metody prediktor-korektor čtvrtého a pátého řádu.

Volba metody je rovněž ovlivněna typem bázového systému. Např. v případě bázového systému $\{\Lambda_i^1\}_0^N$ je průběh řízení funkcí po částech hladkou a metody prediktor-korektor, které předpokládají hladkost funkce $\mathbf{f}$ v celém intervalu integrace, nelze použít. Vhodnou volbou v tomto případě jsou např. metody Runge-Kutta [15], kterým pro určení nového bodu $\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k + h)$ stačí znalost $\mathbf{x}(t_k)$ a hladkost $\mathbf{f}$ v intervalu $[t_k, t_k + h]$. Tyto metody rovněž umožňují dynamické řízení délky kroku, většinou jsou však méně efektivní.

U úloh s omezeními je délka kroku integrace shodná s krokem zjišťování, zda nedošlo k překročení omezení a odpovídá určitému faktoru rozlišení. V tomto případě je pravděpodobně vhodné ponechat krok integrace konstantní a musí být poměrně krátký. V opačném případě může být získáno nepřípustné řešení. Jsou-li omezení zahrnuta pomocí hladkých pokutových členů, nejsou sice porušeny podmínky hladkosti, avšak ani v tomto případě krok integrace nemůže být příliš prodloužen, neboť integrand kritéria má v bodech překročení omezení výrazné úzké výběžky (Obr. 4.2).

Uvažujeme-li všechny tyto aspekty, zůstávají metody Runge-Kutta s konstantní délkou kroku vhodnou volbou. Efektivnější postupy jako prediktor-korektor s proměnnou délkou kroku lze doporučit pouze u úloh bez omezení v každém čase a v případě hladkých bázových systémů s menším počtem prvků.



Obr. 4.2: Průběh integrandu kritéria při využití pokutových funkcí

Prakticky byla s dobrými zkušenostmi implementována známá metoda Runge-Kutta čtvrtého řádu definovaná vzorcem

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)\\ t_{k+1} &= t_k + h\end{aligned} \tag{4.30}$$

kde

$$\begin{aligned}\mathbf{k}_1 &= \tilde{\mathbf{f}}(\mathbf{x}_k, t_k)\\ \mathbf{k}_2 &= \tilde{\mathbf{f}}(\mathbf{x}_k + \frac{\mathbf{k}_1}{2}, t_k + \frac{h}{2})\\ \mathbf{k}_3 &= \tilde{\mathbf{f}}(\mathbf{x}_k + \frac{\mathbf{k}_2}{2}, t_k + \frac{h}{2})\\ \mathbf{k}_4 &= \tilde{\mathbf{f}}(\mathbf{x}_k + \mathbf{k}_3, t_k + h).\end{aligned} \tag{4.31}$$

4.6. Výpočet gradientu

Pro nalezení lokálního minima rozšířeného kritéria (4.14) lze použít libovolný z algoritmů hledání minima bez omezení. Vzhledem ke skutečnosti, že počet parametrů je poměrně vysoký (běžně 20-30 a více), je vhodné zvolit metodu, která využívá derivací účelové funkce. Tyto algoritmy jsou zpravidla efektivnější než ty, které derivací nevyužívají, i když derivace nejsou explicitně známé a jsou určovány numericky.

Pro numerický výpočet gradientu kritéria je většinou doporučeno [11], [15] použít centrální diference:

$$\frac{\partial J}{\partial a_{ij}} \approx \frac{J(\mathbf{A} + h\mathbf{D}_{ij}) - J(\mathbf{A} - h\mathbf{D}_{ij})}{2h} \tag{4.32}$$

kde $h > 0$ je malé číslo a $\mathbf{D}_{ij}$ je matice mající všechny prvky nulové kromě prvku na pozici (i, j), který je roven jedné.

Vztah (4.32) v praxi většinou funguje bez potíží, avšak pro větší počet parametrů je značn výpočetně náročný. Pro uvažovanou třídu problémů je výpočet gradientu zhruba $50-100$ náročnější operací než výpočet hodnoty kritéria. Proto je třeba hledat cesty pro efektivnější výpoče gradientu. Následující závěry jsou určitou obdobou metody urychleného výpočtu gradient z předcházející části.

Předpokládejme, že pro dané hodnoty parametrů $\mathbf{A}$ je určena funkce řízení $\mathbf{u}(t)$ a integrac rovnic (4.10) je určena historie stavu a hodnota kritéria. Lze dokázat [4], [5], že diferenciáln přírůstek hodnoty kritéria δJ v závislosti na variaci řízení $\delta\mathbf{u}$ lze vyjádřit ve tvaru

$$\delta J = \int_0^{t_f} \frac{\partial H}{\partial \mathbf{u}} \delta\mathbf{u}\, dt \tag{4.}$$

kde H je Hamiltonova funkce

$$H = f_0 + \boldsymbol{\lambda}^T \mathbf{f}\,. \tag{4.}$$

Pro funkci $\boldsymbol{\lambda}(t)$ přitom platí

$$\frac{d\boldsymbol{\lambda}^T}{dt} = -\frac{\partial H}{\partial \mathbf{x}} \tag{4.}$$

$$\boldsymbol{\lambda}^T(t_f) = \left[\frac{\partial \varphi}{\partial \mathbf{x}_f}\right]_{x_f = x(t_f)}. \tag{4.}$$

Pozn.: lineární zobrazení definované vztahem (4.33), které každé variaci $\delta\mathbf{u}$ přiřazuje nějaké dJ, představuje derivaci funkcionálu podle funkce řízení v bodě $\mathbf{u}(t)$ [5] .

Průběh $\boldsymbol{\lambda}(t)$ je tedy určen integrací rovnice (4.35) ve zpětném směru s koncovou podmínkou (4.36). Ze vztahu (4.7) lze vyjádřit variaci řízení

$$\delta\mathbf{u}(t) = d\mathbf{A}\,\mathbf{b}(t)\,. \tag{4.}$$

Označme $\mathbf{a}_i^T$ řádky matice $\mathbf{A}$. Po dosazení do (4.33) dostáváme

$$\delta J = \int_0^{t_f} \frac{\partial H}{\partial \mathbf{u}} d\mathbf{A}\,\mathbf{b}(t)\,dt = \sum_{i=1}^{m} \int_0^{t_f} \frac{\partial H}{\partial u_i} d\mathbf{a}_i^T \mathbf{b}(t)\,dt =$$
$$= \sum_{i=1}^{m}\sum_{j=1}^{N} \int_0^{t_f} \frac{\partial H}{\partial u_i} b_j(t)\,dt\, da_{ij}. \tag{4.}$$

To však znamená, že

$$\frac{\partial J}{\partial a_{ij}} = \int_0^{t_f} \frac{\partial H}{\partial u_i} b_j(t)\,dt\,. \tag{4.}$$

Pro výpočet gradientu je v první řadě třeba provést integraci rovnic (4.10) a následně integraci (4.35), (4.36) ve zpětném směru. Tato část výpočtu nezávisí na řádu bázového systému N a je shodná s určením gradientu ve funkcionálním prostoru. Následně je třeba vypočítat $m(N+1)$

integrálů (4.39). Náročnost je tedy obecně opět lineárně závislá na N, stejně jako v případě použití diferencí (4.32). Pro určité typy bázových systémů však může být dosaženo lepších výsledků.

Uvažujme nějaký finitní bázový systém (kap. 2.3) s generující funkcí nulovou mimo nosič Ω. Z definice vyplývá, že bázový prvek $b_i(t)$ je nulový mimo interval $\Omega + iT$. Proto

$$\int_0^{t_f} \frac{\partial H}{\partial u_i} b_j(t)\,dt = \int_{\Omega+iT} \frac{\partial H}{\partial u_i} b_j(t)\,dt \tag{4.40}$$

a integrál stačí počítat pouze na intervalu $\Omega + iT$. Vzhledem k tomu, že pro daný typ bázového systému je $[\Omega] \approx \alpha \frac{t_f}{N}$, kde $[\Omega]$ je délka intervalu Ω a α nějaká konstanta, lze konstatovat, že tato část výpočtu nezávisí na N, neboť celková náročnost je úměrná součinu $N[\Omega] \approx \alpha t_f$. Pro větší hodnoty N je pak možné předpokládat, že dojde k značnému urychlení. Tyto závěry jsou prakticky potvrzeny, třebaže k výraznějšímu rozdílu dochází až zhruba pro $N > 10$, neboť samotný výpočet integrálů je méně náročný než integrace rovnic (4.10) a (4.35).

Koncové podmínky $\boldsymbol{\psi}(\mathbf{x}_f) = \mathbf{0}$ představují z hlediska metody optimalizace omezení typu rovností. Algoritmy nelineárního programování, jako např. SQP (viz část 6, kap. 6.2), pracují s gradienty funkcí ψ_j podle parametrů $\mathbf{A}$. Uvážíme-li však, že $\boldsymbol{\psi}(\mathbf{x}_f)$ lze vyjádřit jako

$$\begin{aligned} \boldsymbol{\psi}(\mathbf{x}_f) &= \boldsymbol{\psi}(\mathbf{x}_0) + \int_0^{t_f} d\,\boldsymbol{\psi}(\mathbf{x}) = \boldsymbol{\psi}(\mathbf{x}_0) + \int_0^{t_f} \frac{\partial \boldsymbol{\psi}(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}}\,dt = \\ &= \boldsymbol{\psi}(\mathbf{x}_0) + \int_0^{t_f} \frac{\partial \boldsymbol{\psi}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}(t))\,dt \end{aligned} \tag{4.41}$$

lze pro každou složku $\boldsymbol{\psi}$ použít stejný postup jako v předchozím případě, kde však opět chybí terminální člen $\varphi(\mathbf{x}_f)$, neboť $\boldsymbol{\psi}(\mathbf{x}_0)$ je konstanta, a místo f_0 jsou složky vektoru $\frac{\partial \boldsymbol{\psi}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}$. V případě, že koncové podmínky jsou triviální, tj. $\boldsymbol{\psi}(\mathbf{x}(t_f)) \equiv \mathbf{x}(t_f) - \mathbf{x}_f$, je $\frac{\partial \boldsymbol{\psi}(\mathbf{x})}{\partial \mathbf{x}}$ jednotkovou maticí a stačí uvažovat gradient jednotlivých složek integrálu $\int_0^{t_f} \mathbf{f}(\mathbf{x}(t))\,dt$.

4.7. Numerické aspekty metody výpočtu gradientu

Popsaný postup má dvě nevýhody. První z nich je nutnost uložení průběhů $\boldsymbol{\lambda}(t)$ a $\mathbf{x}(t)$ v paměti počítače. To však u současných počítačů většinou není na závadu. Závažnějším problémem je potřeba parciálních derivací $\frac{\partial f_0}{\partial \mathbf{u}}, \frac{\partial f_0}{\partial \mathbf{x}}, \frac{\partial \mathbf{f}}{\partial \mathbf{u}}, \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$. V případě komplikovaných funkcí $\mathbf{f}$, f_0, popř. $\mathbf{g}$, může být nesnadné dodat explicitní vztahy, i když je možné využít softwarové nástroje k symbolickému výpočtu derivací. Pro potřeby realizace univerzálního software je proto třeba zabývat se možností numerického výpočtu parciálních derivací.

Situace je komplikovaná tím, že chyby vznikající přibližným výpočtem derivací v každém kroku jsou integrovány v rovnicích (4.35) a (4.39). Zde se nepříznivě projevuje efekt rozdílu dvou blízkých hodnot [14], který se při integraci kumuluje. Proto se lze domnívat, že výsledek může být zatížen příliš velkou chybou. Mnohem lepších výsledků však lze dosáhnout reorganizací vztahů.

Parciální derivace podle **u** se vyskytují pouze v integrálu (4.39). Ten je možné určit rozdílem dvou integrálů

$$\frac{\partial J}{\partial a_{ij}} \approx \frac{1}{2h}\left[\int_0^{t_f} H(\mathbf{u}+h\mathbf{e}_i,t)b_j\,dt - \int_0^{t_f} H(\mathbf{u}-h\mathbf{e}_i,t)b_j\,dt\right] \tag{4.42}$$

kde $\mathbf{e}_i$ je i-tý jednotkový vektor a $H(\mathbf{u},t)$ hodnota Hamiltonovy funkce podél aktuální trajektorie.

Parciální derivace podle **x** jsou využity v rovnici (4.35) pro výpočet funkce $\boldsymbol{\lambda}$. Soustava (4.35) je lineární:

$$\frac{d\boldsymbol{\lambda}^T}{dt} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = -\frac{\partial f_0}{\partial \mathbf{x}}. \tag{4.43}$$

Nahraďme pravou stranu pomocí centrálních diferencí

$$\left(\frac{\partial f_0}{\partial \mathbf{x}}\right)^T \approx \frac{1}{2h}\left[\mathbf{v}^+ - \mathbf{v}^-\right] \tag{4.44}$$

kde pro složky vektorů $\mathbf{v}^+, \mathbf{v}^-$ platí:

$$\begin{aligned} v_j^+ &= f_0(\mathbf{x}+h\mathbf{e}_j,t) \\ v_j^- &= f_0(\mathbf{x}-h\mathbf{e}_j,t) \end{aligned} \tag{4.45}$$

Řešení lze pak vyjádřit jako

$$\boldsymbol{\lambda} = -\frac{1}{2h}\left[\boldsymbol{\lambda}^+ - \boldsymbol{\lambda}^-\right] \tag{4.46}$$

kde $\boldsymbol{\lambda}^+$, $\boldsymbol{\lambda}^-$ jsou řešení rovnic

$$\frac{d\boldsymbol{\lambda}^+}{dt} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda}^+ = \mathbf{v}^+ \tag{4.47}$$

$$\frac{d\boldsymbol{\lambda}^-}{dt} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda}^- = \mathbf{v}^- \tag{4.48}$$

s koncovými podmínkami

$$\lambda_i^+(t_f) = \varphi(\mathbf{x}_f + h\mathbf{e}_i) \tag{4.49}$$

$$\lambda_i^-(t_f) = \varphi(\mathbf{x}_f - h\mathbf{e}_i). \tag{4.50}$$

Obdobným způsobem však bohužel není možné odstranit parciální derivace $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, které by měly být určeny symbolicky, je-li to možné. Přesto je však dosaženo významného zjednodušení, neboť derivace $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ neobsahují pokutové složky omezení.

Bylo prakticky ověřeno, že popsaný postup je funkční.

4.8. Řešené příklady

Úloha 1

Model robota cylindrického typu popsaný v kap. 7.1. Pro vyjádření soustavy (7.4) ve standardním stavovém tvaru $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ označme

$$H = \begin{vmatrix} Ax^2 + 2Bx + C & -eA \\ eA & A \end{vmatrix} = A^2x^2 + 2ABx + A(C + Ae^2) \tag{4.51}$$

$$H_\varphi = \begin{vmatrix} M_\varphi - 2\dot{\varphi}\dot{x}(Ax + B) & -eA \\ F_x + \dot{\varphi}^2(Ax + B) & A \end{vmatrix} = AM_\varphi - 2A\dot{\varphi}\dot{x}(Ax + B) + eA\left[F_x + \dot{\varphi}^2(Ax + B)\right] \tag{4.52}$$

$$\begin{aligned} H_x &= \begin{vmatrix} Ax^2 + 2Bx + C & M_\varphi - 2\dot{\varphi}\dot{x}(Ax + B) \\ eA & F_x + \dot{\varphi}^2(Ax + B) \end{vmatrix} = \\ &= (Ax^2 + 2Bx + C)\left[F_x + \dot{\varphi}^2(Ax + B)\right] - eA\left[M_\varphi - 2\dot{\varphi}\dot{x}(Ax + B)\right] \end{aligned} \tag{4.53}$$

Jelikož musí platit $H \neq 0$, lze vyjádřit druhé derivace:

$$\begin{aligned} \ddot{z} &= (F_z - gD)/D \\ \ddot{\varphi} &= H_\varphi / H \\ \ddot{x} &= H_x / H \end{aligned} \tag{4.54}$$

Definujeme-li stavový vektor

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \dot{\mathbf{y}} \end{pmatrix} = (z, \varphi, x, \dot{z}, \dot{\varphi}, \dot{x})^T \tag{4.55}$$

je možné z (4.54) soustavu (7.4) snadno převést do tvaru $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$:

$$\begin{aligned} \dot{x}_1 &= x_4 \\ \dot{x}_2 &= x_5 \\ \dot{x}_3 &= x_6 \\ \dot{x}_4 &= (F_z - gD)/D \\ \dot{x}_5 &= H_\varphi / H \\ \dot{x}_6 &= H_x / H \end{aligned} \tag{4.56}$$

Pro nahrazení řízení byl zvolen bázový systém $\{\Lambda_i^1\}_0^N$, $N = 20$. Počet optimalizovaných parametrů je $n_z = 63$. Integrace rozšířené soustavy (4.10) byla provedena čtyřbodovou metodou Runge-Kutta s krokem $h = \frac{1}{100} t_f$, tj. pro výpočet funkcionálu je třeba 400 vyhodnocení $\mathbf{f}(\mathbf{x},\mathbf{u})$ podél trajektorie.

Okrajové podmínky jsou obdobné příkladu č. 1, kap. 3.13:

$$\mathbf{x}(0) = (0,0,0,0,0,0)^T$$
$$\mathbf{x}(t_f) = \mathbf{x}_f = (1,2,1,0,0,0)^T . \tag{4.57}$$

Kritériem optimality je kvadratický funkcionál s terminálním členem

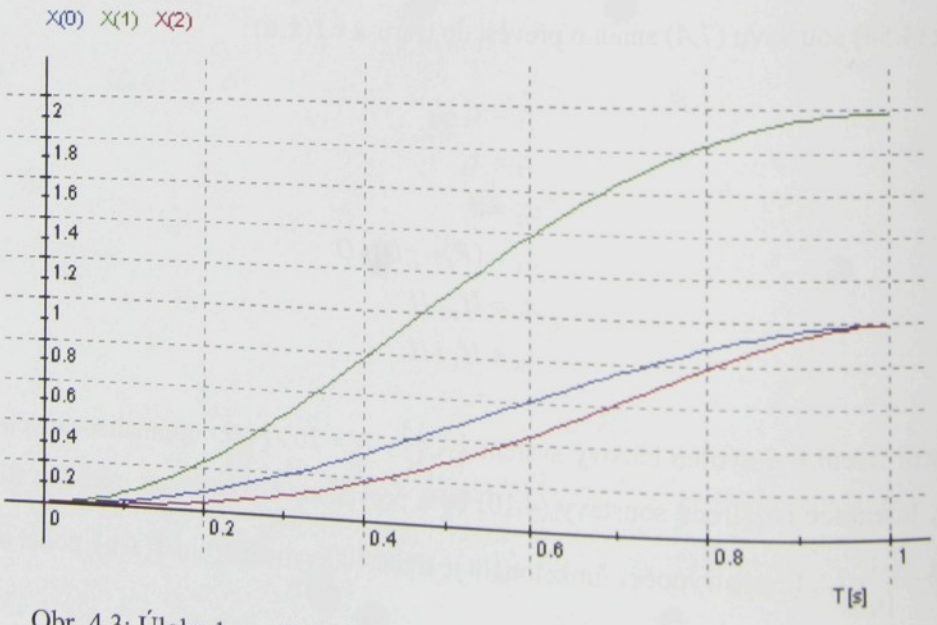
$$J = \sigma \left\| \mathbf{x}(t_f) - \mathbf{x}_f \right\|_2^2 + \int_0^1 \mathbf{u}^T \mathbf{u} \, dt \to \min \tag{4.58}$$

kde pokutová konstanta pro splnění okrajových podmínek byla zvolena $\sigma = 1000$.

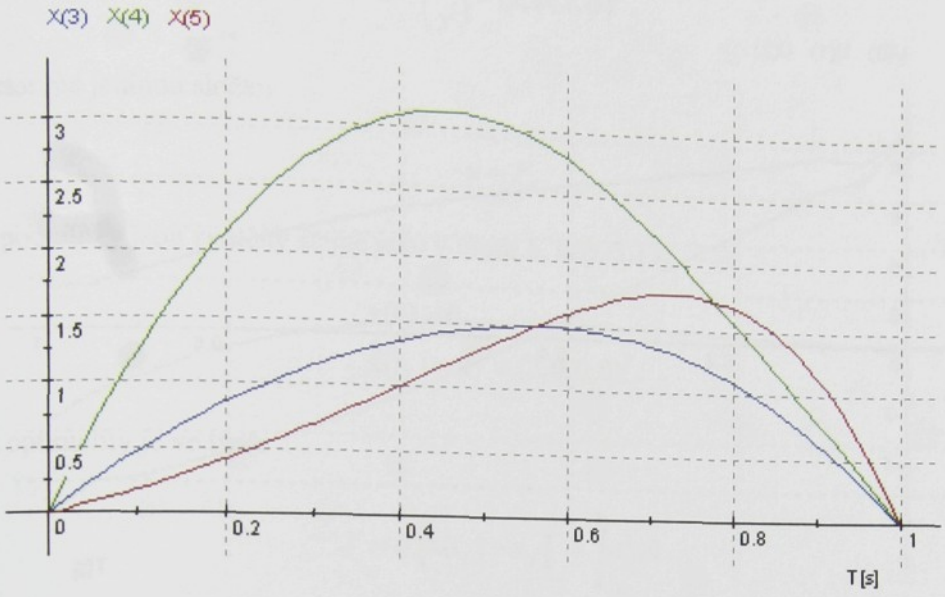
Tab. 4-1 ukazuje průběh výpočtu několika algoritmy optimalizace. Metodou BFGS jsou výsledky získány přibližně během 30 s, což je více než 5x déle než v případě nahrazení trajektorie. Výsledné průběhy na Obr. 4.3 - Obr. 4.5 jsou prakticky shodné s průběhy z příkladu č. 1, kap. 3.13. Výpočet je rychlejší, je-li zvolen hrubší krok integrace, avšak např. pro $h = \frac{1}{40} t_f$ je již patrné značné zkreslení výsledné řídicí funkce v krajních bodech (Obr. 4.6).

Iterace	Největšího spádu	PARTAN	BFGS
10	1872.77	1742.77	67.105
20	1322.01	590.709	64.4149
50	559.084	64.9393	58.8992
100	190.332	54.1661	53.1487
200	68.2172	53.3894	-
500	54.0258	53.1488	-
1000	53.9607	53.1487	-

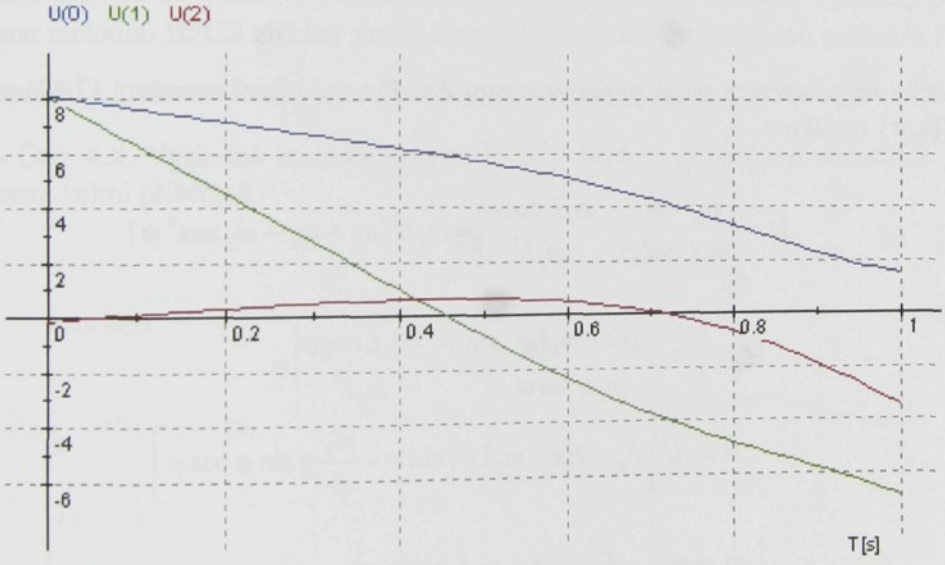
Tab. 4-1: Úloha 1 – průběh výpočtu



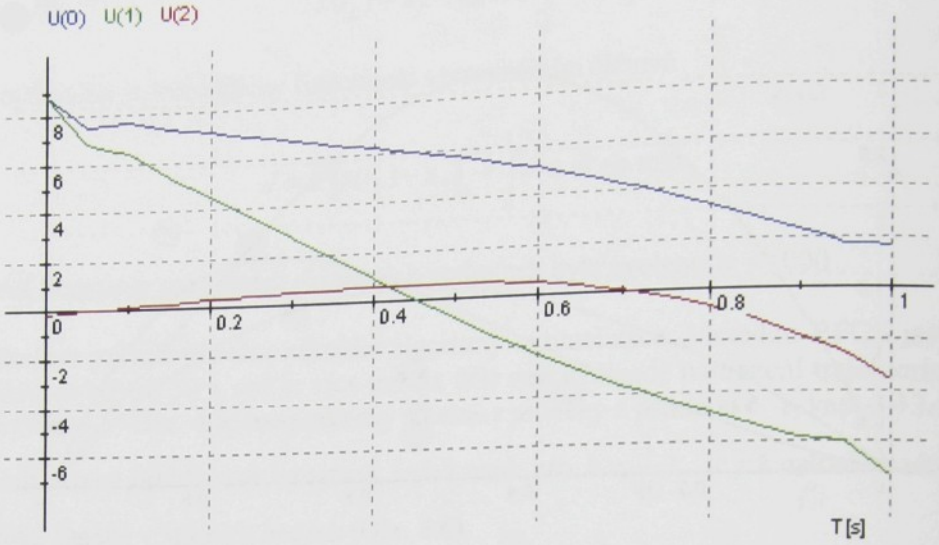
Obr. 4.3: Úloha 1 – průběhy polohy



Obr. 4.4: Úloha 1 – průběhy rychlosti



Obr. 4.5: Úloha 1 – průběhy řízení



Obr. 4.6: Úloha 1 – průběhy řízení pro $h = \frac{1}{40} t_f$

Úloha 2

Model kyvadla na posuvném rámu popsaný v kap. 7.4. Pro vyjádření soustavy (7.43) ve stavovém tvaru $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ označme

$$H = \begin{vmatrix} m_1 + m_2 & m_2 L \cos\varphi \\ m_2 L \cos\varphi & m_2 L^2 \end{vmatrix} = m_2 L^2 [m_1 + m_2 - m_2 \cos^2 \varphi] \tag{4.59}$$

$$\begin{aligned} H_x &= \begin{vmatrix} F_x - \beta\dot{x} + m_2 L \dot{\varphi}^2 \sin\varphi & m_2 L \cos\varphi \\ m_2 g \sin\varphi & m_2 L^2 \end{vmatrix} = \\ &= m_2 L^2 \left[F_x - \beta\dot{x} + m_2 L \dot{\varphi}^2 \sin\varphi - \frac{m_2}{L} g \sin\varphi \cos\varphi \right] \end{aligned} \tag{4.60}$$

$$\begin{aligned} H_\varphi &= \begin{vmatrix} m_1 + m_2 & F_x - \beta\dot{x} + m_2 L \dot{\varphi}^2 \sin\varphi \\ m_2 L \cos\varphi & m_2 g \sin\varphi \end{vmatrix} = \\ &= m_2 L \left[\frac{m_2}{L} (m_1 + m_2) g \sin\varphi - (F_x - \beta\dot{x} + m_2 L \dot{\varphi}^2 \sin\varphi) \cos\varphi \right] \end{aligned} \tag{4.61}$$

Po vyjádření druhých derivací:

$$\begin{aligned} \ddot{x} &= \frac{H_x}{H} \\ \ddot{\varphi} &= \frac{H_\varphi}{H} \end{aligned} \tag{4.62}$$

Stavový vektor je

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \dot{\mathbf{y}} \end{pmatrix} = (x, \varphi, \dot{x}, \dot{\varphi})^T . \tag{4.63}$$

Řídicí vektor má jedinou složku

$$u = F_x . \tag{4.64}$$

Okrajové podmínky jsou zvoleny stejně jako v úloze 6, kap. 3.13:

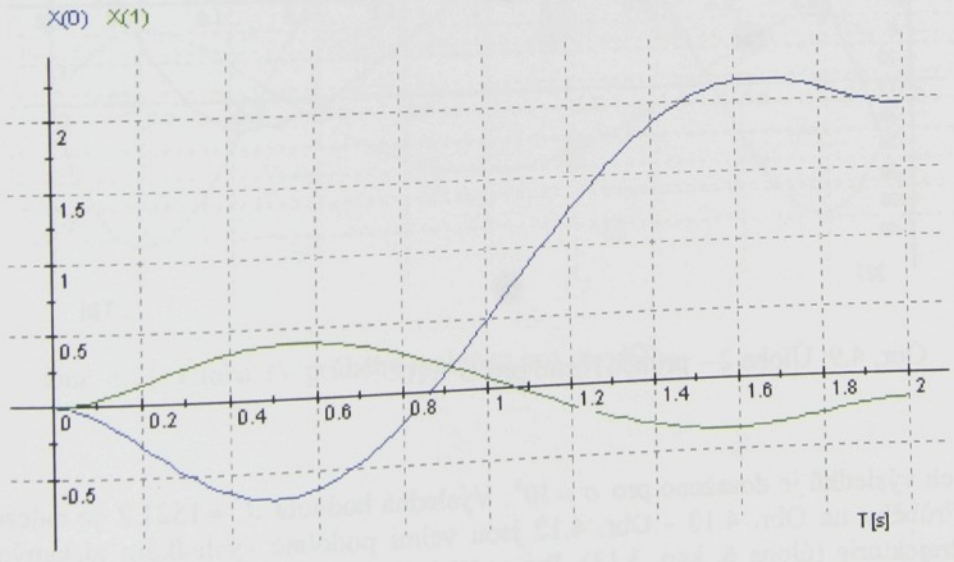
$$\begin{aligned} &\mathbf{x}(0) = \mathbf{0} \\ &\mathbf{x}(t_f) = \mathbf{x}_f = (2,0,0,0)^T . \end{aligned} \tag{4.65}$$

Kritérium optimality je ve tvaru

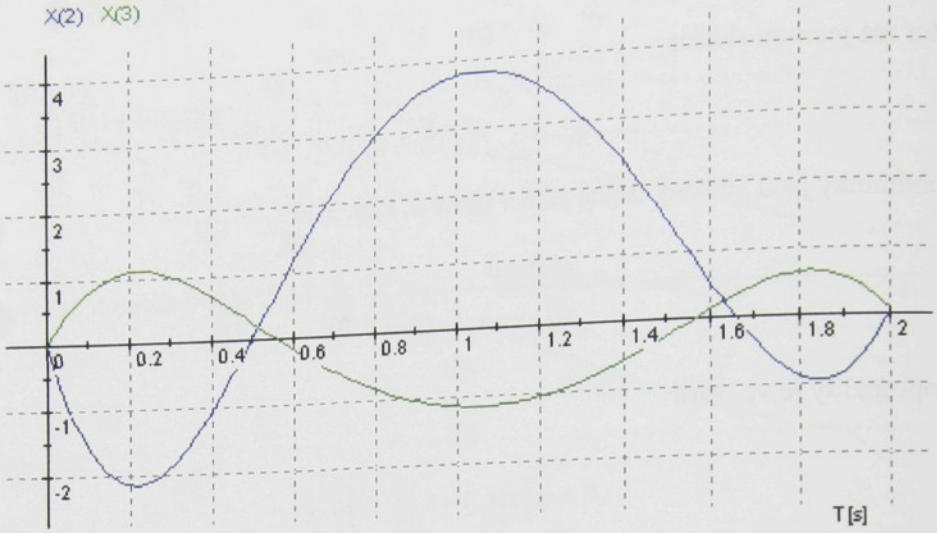
$$J' = \sigma \left\| \mathbf{x}(t_f) - \mathbf{x}_f \right\|^2 + \int_0^2 u^2 \, dt . \tag{4.66}$$

Pro nahrazení řízení byl opět zvolen bázový systém $\left\{ \Lambda_i^1 \right\}_0^N$, $N = 20$. Počet optimalizovaných parametrů je $n_z = 21$.

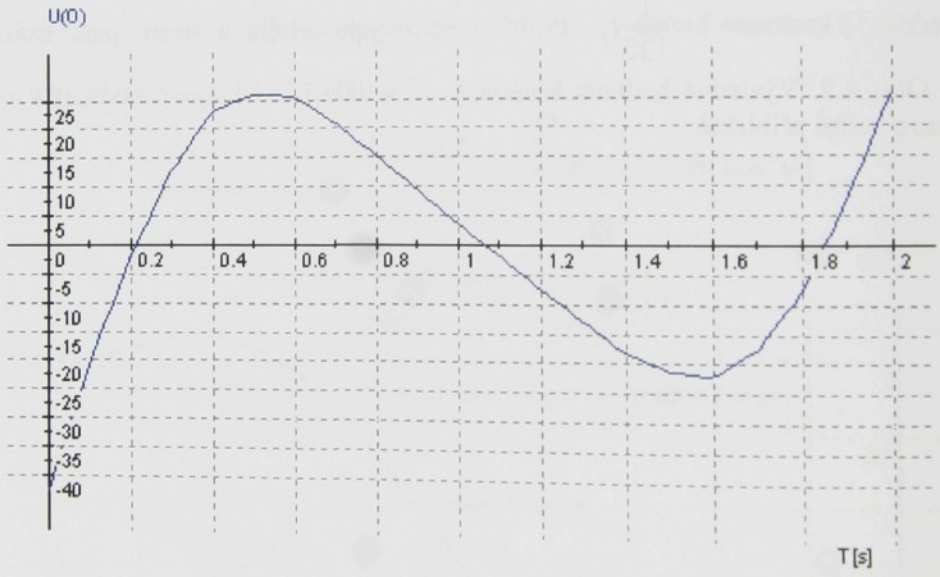
V tomto případě se ukazuje, že hodnota σ musí být značně vysoká. Výsledky pro $N = 20$, $\sigma = 10^4$ jsou metodou BFGS získány po 70 iteracích. Integrace čtyřbodovou metodou Runge-Kutta byla provedena s krokem $h = \frac{1}{120} t_f$. Průběhy stavových veličin a řízení jsou znázorněny na Obr. 4.7 - Obr. 4.9. Výsledná hodnota kritéria je $J^* = 1006.15$. Okrajové podmínky jsou zřejmě splněny pouze velmi přibližně.



Obr. 4.7: Úloha 2 – průběhy polohy pro $\sigma = 10^4$

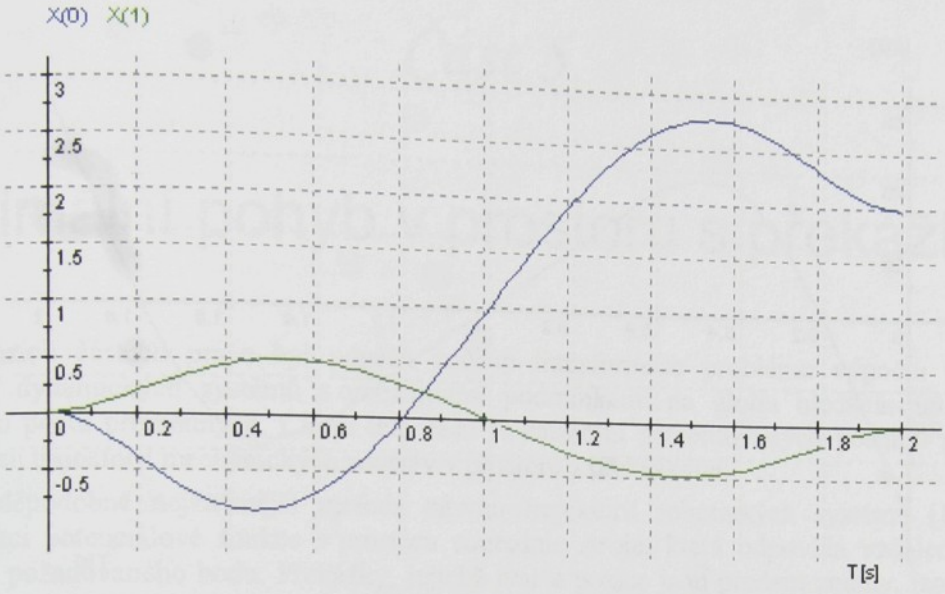


Obr. 4.8: Úloha 2 – průběhy rychlosti pro $\sigma = 10^4$

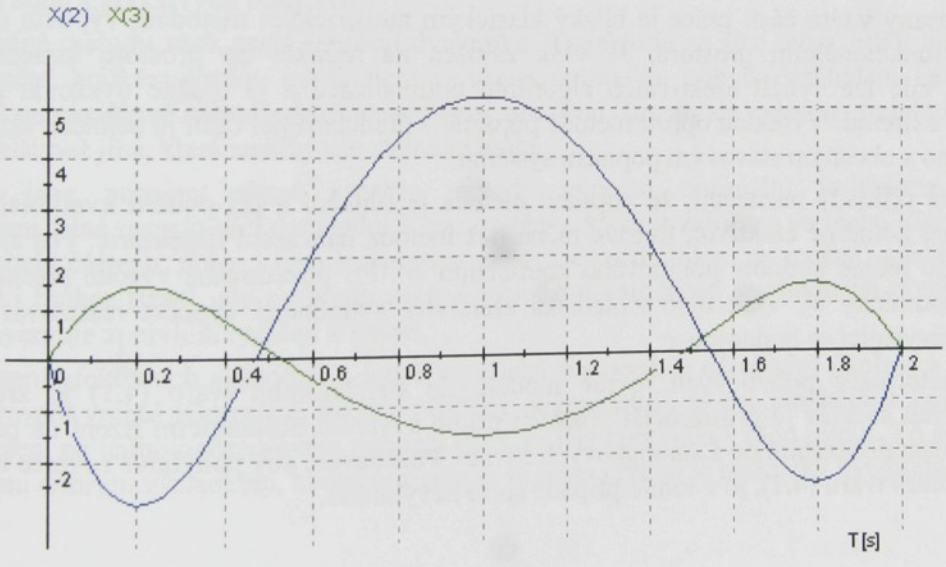


Obr. 4.9: Úloha 2 – průběh řízení pro $\sigma = 10^4$

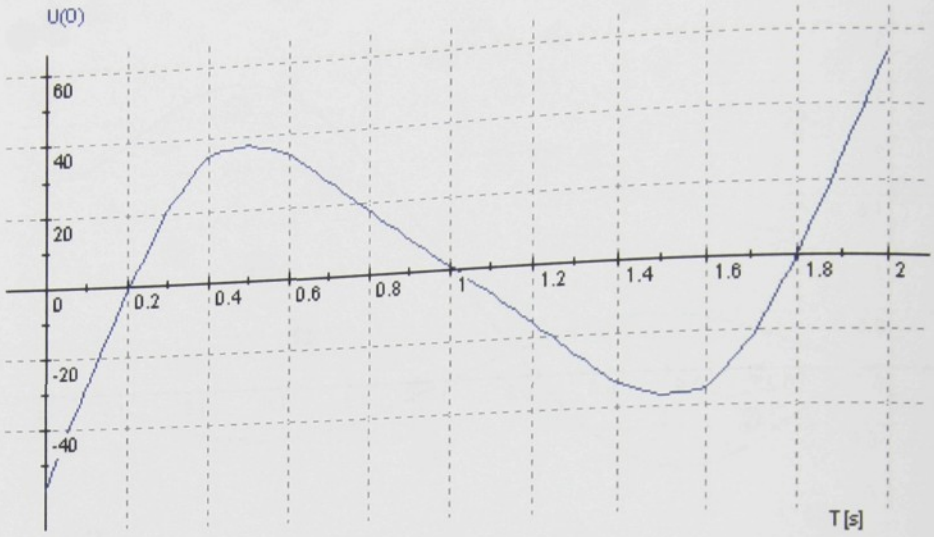
Kvalitnějších výsledků je dosaženo pro $\sigma = 10^5$. Výsledná hodnota $J^* = 1521.2$ je nalezena po 120 iteracích. Průběhy na Obr. 4.10 - Obr. 4.12 jsou velmi podobné výsledkům získaným metodou nahrazení trajektorie (úloha 6, kap. 3.13). Pro vyšší hodnoty σ již algoritmy optimalizace špatně konvergují k řešení.



Obr. 4.10: Úloha 2 – průběhy polohy pro $\sigma = 10^5$



Obr. 4.11: Úloha 2 – průběhy rychlosti pro $\sigma = 10^5$



Obr. 4.12: Úloha 2 – průběhy zrychlení pro $\sigma = 10^5$

4.9. Závěr

Postup popsaný v této části práce je blízký klasickým numerickým metodám výpočtu optimálního řízení ve funkcionálním prostoru. Je však založen na redukci do prostoru konečného počtu parametrů, kde lze využít efektivních algoritmů optimalizace a je možné uvažovat i o hledání globálního extrému. Výhodou oproti metodě popsané v předcházející části je zejména skutečnost, že je pracováno s obecným stavovým popisem systému.

Řešené příklady naznačují, že popsaná metoda je funkční a při použití kvalitních algoritmů optimalizace poměrně efektivní, třebaže méně než metoda nahrazení trajektorie. Pro zjednodušení bylo využito jediné hodnoty pokutového koeficientu σ. Při požadované vysoké přesnosti splnění koncové podmínky by však bylo z hlediska efektivity výhodnější výpočet realizovat v několika krocích se zvyšující se hodnotou σ.

Z transformace pohybových rovnic modelů do standardního tvaru (4.1) je zřejmé, že u mechanických soustav je přirozenější pracovat přímo s tvarem obsahujícím řízení na pravé straně, který je získán dosazením do Lagrangeových rovnic. Požadavek, aby pohybové rovnice byly zadány ve standardním tvaru (4.1), je v tomto případě spíše nevýhodou.

Část 5.

Optimální pohyb v prostoru s překážkami

V předchozích částech práce byl popsán způsob transformace problému výpočtu optimálních trajektorií dynamických systémů s omezujícími podmínkami na úlohu hledání minima funkce konečného počtu proměnných. Cílem této části je rozšíření prezentovaných postupů pro výpočet optimálních trajektorií mechanických soustav v prostoru s překážkami.

Pravděpodobně nejznámější metoda návrhu trajektorií robotických systémů [17] spočívá v konstrukci potenciálové funkce v prostoru souřadnic stroje, která odpovídá vzdálenosti pozice robota od požadovaného bodu. Překážky, jejichž tvar a pozice jsou předem známy, jsou do tohoto potenciálu zahrnuty v principu pokutovou nebo bariérovou metodou. Směr trajektorie v každém bodě je pak dán záporným gradientem potenciálu. V potenciálové funkci mohou vznikat lokální extrémy (místa s nulovým gradientem), které se většinou překonávají pomocí náhodných faktorů. Výhodou této metody je jednoduchost a efektivita, třebaže u složitých scén není zaručeno, že v každé pozici, která není koncovou, bude nalezen bod s menším potenciálem. Jiným možným přístupem je např. konstrukce sítě cest, které spojují určitou dostatečně hustou diskrétní množinu výchozích bodů s koncovými body [18].

Zmíněné metody však zcela ignorují dynamiku. Trajektorie se sice může blížit hladké funkci při dostatečně krátkém kroku, avšak není zaručena optimalita pohybu vzhledem ke zvolenému kritériu. U potenciálové metody dokonce ani není zaručeno, že nebude nalezena trajektorie, která je výrazně delší než jiná, která směřuje do stejného bodu.

Dále jsou popsány různé aspekty metody výpočtu optimální trajektorie v prostoru s překážkami, plně respektující dynamiku a tvar systému, která je založena na technikách popsaných v předchozích částech práce. Jsou-li splněny příslušné podmínky, je možné využít nahrazení trajektorie i funkce řízení, ale z důvodu značné výpočetní náročnosti je vhodnější první z přístupů, neboť konverguje zpravidla rychleji k řešení.

Nalezení efektivních algoritmů a jejich praktická realizace je náročným úkolem a představuje v jistém smyslu vrchol této práce. Řešené příklady dokumentují, že pro složitější scény je získáno kvalitní řešení na jednoprocesorovém počítači PC 1.2 GHz zpravidla v řádu minut. To je dle názoru autora velmi dobrým výsledkem, neboť naznačuje, že tento přístup je prakticky využitelný.

5.1. Specifické vlastnosti problému

Uvažujeme-li optimální pohyb mechanických systémů daného tvaru v prostoru s překážkami, lze sice použít výše popsané metody, avšak objevuje se řada komplikací, které vyžadují samostatný rozbor.

K nejdůležitějším problémům patří:

1. Testování kolize mezi geometrickými modely systému a překážek v daném čase je zpravidla časově náročnou operací, která může být mnohem náročnější než výpočet hodnoty kritéria.

2. Trajektorie je přípustná, jestliže nenastává kolize v žádném časovém bodě $t \in [0, t_f]$. Algoritmus vyšetření kolize však může být vykonán pouze v konečném počtu bodů.

3. Jedná se o typický problém globální optimalizace s omezeními v obecném tvaru $\mathbf{z} \in \Omega$, kde $\mathbf{z}$ je vektor parametrů trajektorie a kompaktní množina Ω je zadána logickou funkcí příslušnosti ω (viz též kapitola Heuristické algoritmy v části 6).

K prvnímu bodu je třeba uvést, že systém i překážky jsou množinami nekonečného počtu bodů. V paměti počítače jsou však aproximovány modely, které umožňují rozhodnout o případné kolizi v konečném čase. Typ modelu musí být zvolen s ohledem na co nejvyšší rychlost vyšetření kolize.

Druhá část se týká určení časových bodů testování kolize t_i. V ideálním případě jsou tyto body určeny tak, že:

- Celkový počet bodů testování kolize t_i je co nejmenší.
- Je garantováno, že nedojde ke kolizi v žádném čase $t \in (t_{i-1}, t_i)$.

Pokud jde o třetí bod, je-li tvar systému v reálném prostoru v čase t označen jako $Q(t)$ a tvar překážek jako $R(t)$, je požadováno, aby

$$Q(t_i) \cap R(t_i) = \varnothing \tag{5.1}$$

pro všechny diskrétní body testování kolize $t_i \in [0, t_f]$. Přitom se předpokládá, že je zvolen takový typ modelů, že platnost (5.1) lze zjistit pomocí konečného počtu operací.

Jelikož poloha modelu $Q(t)$ závisí na souřadnicích $\mathbf{y}$ a ty dle předpokladu závisí na složkách vektoru parametrů $\mathbf{z}$, je požadavek (5.1) speciálním případem obecného zápisu $\mathbf{z} \in \Omega$.

Jsou možné dva přístupy:

- Pracovat přímo s obecným tvarem omezení $\mathbf{z} \in \Omega$ pomocí algoritmů optimalizace, které to umožňují
- Převést omezení na tvar

$$\mathbf{g}(\mathbf{z}) \geq \mathbf{0} \tag{5.2}$$

popř.

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{5.3}$$

kde funkce $\mathbf{g}$, $\mathbf{h}$ jsou spojité, popř. hladké v prostoru parametrů $\mathbf{z}$.

Praktické zkušenosti naznačují, že obecný tvar omezení $\mathbf{z} \in \Omega$ většinou není výhodný pro optimalizaci, třebaže mnohé heuristické algoritmy jsou schopné s ním přímo pracovat. Na druhé straně, jsou známé typy modelů (viz dále), které umožňují zjistit kolizi i pro složité tvary velmi efektivně. Při nalezení jediného bodu t_i, kde dochází ke kolizi, je pak možné vyhodnocení $\mathbf{z} \in \Omega$ ukončit.

V druhém případě je třeba v každém bodě určitým způsobem změřit rozsah kolize, což může být mnohem náročnější než pouhá detekce. Navíc výpočet není možné předčasně ukončit při nalezení t_i, pro které $Q(t_i) \cap R(t_i) \neq \varnothing$, jako v předchozím případě.

Není tedy jisté, který z přístupů je výhodnější. V rámci této práce byl vyvíjen a prakticky implementován spíše druhý přístup, který se zdá být výhodnější z hlediska potřebného počtu

iteračních kroků, neboť obsahuje informaci o míře kolize a umožňuje přiblížení k minimu i ze strany omezení. Existuje zde rovněž možnost urychlení výpočtu v blízkosti minima pomocí lokálních algoritmů optimalizace (viz též kapitola Hybridní metody v části 6 práce). Autor se domnívá, že tento přístup bude účinnější zejména pro vyšší n_z (počet optimalizovaných parametrů).

Pro převedení na tvar (5.3) definujme funkci $\mu(Q,R)$, která dvojici uzavřených podmnožin $\mathbb{R}^3$ přiřazuje reálné číslo např. tak, že platí tyto axiomy:

$$\mu(Q,R) \geq 0 \tag{5.4}$$

$$\mu(Q,R) = \mu(R,Q) \tag{5.5}$$

$$\mu(Q,R) = 0 \Leftrightarrow Q \cap R = \varnothing \tag{5.6}$$

$$\mu(Q_1 \cup Q_2, R) \geq \mu(Q_1, R)$$
$$\mu(Q_1 \cup Q_2, R) \geq \mu(Q_2, R) \tag{5.7}$$

Funkci $\mu(Q,R)$ nazvěme *mírou kolize*.

Požadavek (5.1) pak může být nahrazen omezující podmínkou ve tvaru (5.3):

$$\mu(Q,R) = 0 \ , \ \forall t_i \in \left[0, t_f\right]. \tag{5.8}$$

Omezení může být zahrnuto do kritéria pokutovou metodou:

$$J' = J + \sigma \int_0^{t_f} \mu(Q(t), R(t))\, dt \ . \tag{5.9}$$

kde $\sigma > 0$. Pro praktické využití je třeba hledat cesty maximálního urychlení výpočtu (5.9).

5.2. Modely pro efektivní zjištění kolize

Pro reprezentaci tvaru systému a překážek je třeba zvolit určitý typ modelů, se zřetelem na maximální efektivitu detekce kolize, popř. zjištění míry kolize. Pro první přiblížení je možné např. nahradit systém určitým počtem bodů na jeho povrchu a testovat, zda některý z těchto bodů se v nějakém čase nachází uvnitř překážky. Tento jednoduchý postup autor v podstatě ověřil již ve své diplomové práci, je však možné na něj narazit i v odborné literatuře v souvislosti s návrhem trajektorií robotických systémů. Pro praktické využití je však třeba většinou definovat velký počet těchto bodů a určovat jejich pozici v každém čase, což mimořádně prodlouží dobu výpočtu. Proto další postup směřoval k plnému nahrazení tvaru. V tomto případě je v každém časovém bodě testována kolize mezi tělesy reprezentujícími systém a překážky.

V disertační práci bylo využito nahrazení tvaru soustavou mnohostěnů. Pro detekci a zjištění míry kolize byly implementovány jednoduché algoritmy, které jsou efektivní pouze pro mnohostěny s malým počtem vrcholů a jsou pouze přibližné. Autor v té době nebyl informován o sofistikovanějších metodách vyšetření kolize mnohostěnů, např. [20], které jsou však na druhé straně mnohem komplikovanější z hlediska implementace. Realizované algoritmy lze přesto považovat za určitý kompromis z hlediska:

- rychlosti detekce kolize v konkrétním čase
- efektivity při aproximaci složitého tvaru
- náročnosti implementace datových struktur a algoritmů kolize
- efektivity detekce kolize podél trajektorie
- možnosti konstrukce pokutové funkce $\mu(Q,R)$ a její integrace

- náročnosti vytváření modelu z hlediska uživatele

Pro nahrazení tvaru systému a překážek jsou v podstatě možné dva základní přístupy:

a) použití povrchových modelů
b) nahrazení tvaru soustavou těles

Povrchové modely jsou běžně využívané v počítačové grafice pro kvalitní nahrazení tvaru reálných těles soustavou segmentů rovinných, popř. křivých povrchových ploch, které na sebe spojitě nebo hladce navazují. Pro určení kolize mezi dvěma modely v každém čase je třeba vyšetřit průsečíky těchto segmentů. Tento výpočet je třeba provést numericky už v případě ploch druhého řádu. Samotné zjištění kolize podél celé trajektorie je pak časově značně náročnou operací. Jinou nevýhodou z praktického hlediska je náročnost vytváření modelu, které vyžaduje výraznou softwarovou podporu.

Pod přístupem b) se rozumí aproximace tvaru systému a překážek sjednocením konečného počtu jednoduchých, většinou konvexních těles:

$$Q=\bigcup_{i=1}^{n_q} q_i \ , \ R=\bigcup_{j=1}^{n_r} r_j \ . \tag{5.10}$$

Vyšetření kolize je pak redukováno na zjištění kolize mezi všemi dvojicemi q_i, r_j.

Jednou z možností je nahrazení složitějších tvarů pomocí soustavy elementárních mnohostěnů, jako jsou hranoly, jehlany a komolé jehlany. Tento typ modelů je zřejmě výhodný pro nahrazení hranatých tvarů, které se často vyskytují u technických systémů. Hlavním nedostatkem je velmi hrubá aproximace oblých tvarů. Zjištění kolize je poměrně efektivní operací v případě, že počet vrcholů jednotlivých mnohostěnů není velký. Rozšíříme-li třídu elementárních těles o oblé prvky, jako např. části kuželů a koulí, není možné bohužel použít jednotné algoritmy popsané v dalších kapitolách. Zjištění kolize mezi některými dvojicemi elementárních těles potom může být náročnou operací z hlediska doby výpočtu i implementace.

Jinou alternativou je nahradit tvar přibližně sjednocením dostatečného počtu množin jednoduchého tvaru, mezi kterými je zjištění kolize triviální. Jsou-li zvoleny např. krychle se stranami rovnoběžnými se souřadnými osami, vzniká problém při změně orientace těles v prostoru. Výhodnější je nahrazení tvaru sjednocením koulí, kde při změně orientace dojde pouze k transformaci středů. Počet množin, který odpovídá rozlišovací úrovni, na které chceme s tělesy pracovat, může být pro aproximaci složitého tvaru poměrně vysoký (pro nahrazení tvaru průmyslového robota to může být např. řádově $n_q = 10^3$ koulí). Celková náročnost detekce kolize, která odpovídá součinu n_q x n_r pak může být značná.

Podstatného zdokonalení výše uvedeného postupu lze dosáhnout hierarchickým způsobem, např. [19]. Tvar systému a překážky je na počátku pokryt soustavou malého počtu koulí (ve skutečnosti stačí pokrýt pouze okraj tvaru). Jsou označeny dvojice koulí, mezi kterými dochází ke kolizi. V dalším kroku je použito jemnější pokrytí pomocí koulí s menším poloměrem a je testována kolize pouze mezi koulemi, které pokrývají oblast kolize z předchozího kroku.

U výše zmíněné metody roste náročnost testu kolize se zvyšující se složitostí modelů jen mírně. Na druhé straně však tato metoda neumožňuje efektivní určení spojité, popř. hladké míry kolize. Obtížným krokem je i samotné vytváření modelů, kdy je třeba vhodným způsobem definovat soustavy koulí pokrývajících těleso v jednotlivých krocích.

Pro urychlení algoritmů vyšetření kolize je výpočet často rozdělen na dvě fáze:

- *široká fáze*, kdy se pomocí jednoduchých kritérií určí menší podoblasti, popř. elementární tělesa či povrchové plochy, kde může dojít ke kolizi

- *úzká fáze*, kdy se provede vlastní vyšetření kolize mezi částmi, které byly určeny v široké fázi

U zmíněné hierarchické kulové metody však tyto fáze nejsou přesně odděleny a dochází mezi nimi k plynulému přechodu. V případě nahrazení tvaru pomocí soustavy mnohostěnů je úzkou fází vyšetření kolize mezi dvěma mnohostěny a do široké fáze patří všechny kroky, které zužují množinu dvojic mnohostěnů, mezi kterými se bude kolize vyšetřovat.

5.3. Výpočet polohy bodů

Při pohybu modelu je třeba v každém časovém bodě vypočítat pozice některých bodů systému, popř. překážek (pokud jsou pohyblivé) v závislosti na souřadnicích stroje **y**. Např. v případě hierarchických kulových modelů to jsou souřadnice středů koulí, zatímco u mnohostěnových modelů to jsou souřadnice rohových bodů.

Pozice těchto bodů je konstantní v souřadném systému daného členu, neboť členy jsou dle předpokladu tuhé. Ten vůči souřadnému systému předcházejícího členu vykonává v obecném případě současně rotační a translační pohyb. Označme $\mathbf{r}_i$ polohu daného bodu v souřadném systému i-tého členu. Je-li $\mathbf{s}_i$ pozice souřadného systému i-tého členu vůči členu $(i-1)$ a $\mathbf{T}_i$ ortogonální matice rotace i-tého členu vůči členu $(i-1)$, platí pro $\mathbf{r}_{i-1}$:

$$\mathbf{r}_{i-1} = \mathbf{s}_i + \mathbf{T}_i \mathbf{r}_i . \tag{5.11}$$

Matice $\mathbf{T}_i = \mathbf{T}(\alpha_i, \beta_i, \gamma_i)$ je definována úhly rotace podle souřadných os x, y a z:

$$\mathbf{T}(\alpha_i, \beta_i, \gamma_i) = \mathbf{T}_z(\gamma_i)\mathbf{T}_y(\beta_i)\mathbf{T}_x(\alpha_i) \tag{5.12}$$

kde

$$\mathbf{T}_x(\alpha_i) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{pmatrix}, \tag{5.13}$$

$$\mathbf{T}_y(\beta_i) = \begin{pmatrix} \cos\beta_i & 0 & \sin\beta_i \\ 0 & 1 & 0 \\ -\sin\beta_i & 0 & \cos\beta_i \end{pmatrix}, \tag{5.14}$$

$$\mathbf{T}_z(\gamma_i) = \begin{pmatrix} \cos\gamma_i & -\sin\gamma_i & 0 \\ \sin\gamma_i & \cos\gamma_i & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.15}$$

Polohu bodu v základním, tj. nultém souřadném systému, lze pak určit rekurzivní aplikací vztahu (5.11). Pro formální zjednodušení se často vztah (5.11) zapisuje jako

$$\begin{pmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{T}_i & \mathbf{s}_i \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}. \tag{5.16}$$

Zavedeme-li rozšířené vektory polohy $\tilde{\mathbf{r}}_i$, má vztah (5.16) tvar

$$\tilde{\mathbf{r}}_{i-1} = \tilde{\mathbf{T}}_i \, \tilde{\mathbf{r}}_i . \tag{5.17}$$

Pro rozšířený vektor polohy v základním souřadném systému platí:

$$\tilde{\mathbf{r}}_0 = (\tilde{\mathbf{T}}_1 ... \tilde{\mathbf{T}}_i) \, \tilde{\mathbf{r}}_i = \tilde{\mathbf{T}}_{i,0} \, \tilde{\mathbf{r}}_i . \tag{5.18}$$

Pro vyšetření kolize je většinou třeba pro každý člen určit pozice většího počtu bodů najednou. Určeme $\mathbf{r}_{i-2}$ ze vztahu (5.11):

$$\begin{aligned} \mathbf{r}_{i-2} &= \mathbf{s}_{i-1} + \mathbf{T}_{i-1} \mathbf{r}_{i-1} = \mathbf{s}_{i-1} + \mathbf{T}_{i-1} (\mathbf{s}_i + \mathbf{T}_i \mathbf{r}_i) = \\ &= \mathbf{s}_{i-1} + \mathbf{T}_{i-1} \mathbf{s}_i + \mathbf{T}_{i-1} \mathbf{T}_i \mathbf{r}_i = \mathbf{s}_{i,i-2} + \mathbf{T}_{i-1} \mathbf{T}_i \mathbf{r}_i \end{aligned} \tag{5.19}$$

kde $\mathbf{s}_{i,i-2}$ je pozice i-tého souřadného systému v souřadném systému členu $(i-2)$. Opakovanou aplikací (5.19) dostáváme:

$$\mathbf{r}_0 = \mathbf{s}_{i,0} + \mathbf{T}_{i,0} \mathbf{r}_i \tag{5.20}$$

$$\begin{aligned} \mathbf{s}_{i,0} &= \mathbf{s}_{i-1,0} + \mathbf{T}_{i-1} \mathbf{s}_i \\ \mathbf{T}_{i,0} &= \mathbf{T}_{i-1,0} \mathbf{T}_i . \end{aligned} \tag{5.21}$$

Pomocí rekurzivních vztahů (5.21) se nejprve postupem zdola nahoru pro $i = 2,3,...$ určí vektory $\mathbf{s}_{i,0}$ a matice absolutních rotací $\mathbf{T}_{i,0}$ pro všechny členy. Hodnoty $\mathbf{s}_{1,0}$ a $\mathbf{T}_{1,0}$ odpovídají umístění a orientaci celé soustavy těles v prostoru. Absolutní pozice jednotlivých bodů uvnitř členů se pak dopočítají pomocí vztahu (5.20).

5.4. Nahrazení soustavou elementárních těles

Výsledky popsané v této kapitole se týkají případu, kdy je tvar nahrazen soustavou elementárních těles. Specifické vlastnosti mnohostěnových modelů, které jsou speciálním případem, jsou diskutovány v dalších kapitolách.

V základní verzi je problém redukován na zjištění kolize mezi všemi dvojicemi elementárních těles q_i, r_j. Algoritmy zjištění kolize, popř. jejího rozsahu, mohou být poměrně časově náročné, dokonce i pokud se jedná o mnohostěny. Jelikož počet těchto operací roste zhruba kvadraticky se zvyšující se složitostí modelu, je tento základní přístup velmi neefektivní pro komplikovanější tvary.

Velmi významného urychlení je však možné dosáhnout, pokud se ve většině případů podaří rozhodnout pomocí jednodušších podmínek. Jednou z možností je před vlastním zjišťováním kolize mezi dvěma elementárními tělesy testovat kolizi jim opsaných koulí. Takto lze snadno vyloučit případy, kdy jsou tělesa dostatečně vzdálená. Definujme $C_P \in P$ střed tělesa P a číslo r_P, které je poloměrem minimální koule se středem C_P, která obsahuje P. Číslo r_P může být určeno jednorázově, neboť je invariantní vzhledem k pohybu P. Dvě tělesa nemají společný průnik, jestliže

$$\left\| C_P - C_Q \right\|_2^2 > \left(r_P + r_Q \right)^2 . \tag{5.22}$$

U podlouhlých tvarů může být test (5.22) málo účinný. Je však možné navíc snadno sestrojit opsané válce nekonečné délky zadané osou a poloměrem. V tomto případě lze velmi efektivně rozhodnout o prázdném nebo neprázdném průniku koule a válce. Je-li střed válce tělesa Q identický se středem jeho opsané koule, r_q' je jeho poloměr a $\mathbf{v}$ vektor určující jeho osu, mají P a Q prázdný průnik, jestliže

$$\|C_P - C_Q\|^2 - \frac{1}{\|\mathbf{v}\|^2}\langle C_P - C_Q | \mathbf{v}\rangle^2 > (r_P + r_Q)^2 \tag{5.23}$$

kde $\langle a | b \rangle$ označuje skalární součin. Vyšetření průniku dvou nekonečných válců je již náročnější operací, neboť vyžaduje vyřešení soustavy lineárních rovnic 3 x 3. Proto před vlastním testováním kolize byly při praktické implementaci provedeny pouze tyto testy:

- opsaná koule 1. tělesa - opsaná koule 2. tělesa
- opsaný válec 1. tělesa - opsaná koule 2. tělesa
- opsaná koule 1. tělesa - opsaný válec 2. tělesa

Algoritmy vyšetření kolize popsané v dalších kapitolách pracují s rohovými body a normálami stěn mnohostěnů. Pro výpočet rohových bodů se použije vztah (5.20) a normály se určí na základě rohových bodů. Pro větší počet bodů je tento výpočet poměrně náročný. Velmi důležité z hlediska efektivity je, že jej není nutné provádět, jestliže je některý z výše popsaných testů negativní. Stačí tedy nejprve určit středy opsaných koulí a jeden další bod, který určí osu opsaného válce. Výhodné je zvolit střed souřadného systému členu identický se středem opsané koule a válce – pak lze ušetřit výpočet pozice jednoho bodu.

Uvažujeme-li, že každý pevný člen, popř. každá překážka může být nahrazena několika elementárními tělesy, je přirozené modely organizovat ve více úrovních. Model systému a překážek je v této obecnější podobě definován jako stromová struktura, kde každý vrchol obsahuje konečný počet následníků, z nichž některé představují elementární tělesa a ostatní jsou stromové struktury stejného typu. Souřadné systémy, které jsou společné pro všechna elementární tělesa a podstromy daného členu, jsou reprezentovány vnitřními vrcholy stromu. Změna pozice a orientace vnitřního vrcholu se projeví změnou absolutní pozice a orientace všech jeho následníků.

Výpočet polohy bodů v stromě lze provést opět pomocí vztahů (5.20), (5.21). Rozdíl spočívá v tom, že místo cyklu je pro průchod stromem nutno použít jednoduchý rekurzivní algoritmus:

Pozice(i) (předpokládá známé $\mathbf{s}_{i,0}$ a $\mathbf{T}_{i,0}$ pro $i = 1$):

- Je-li i list stromu, tj. elementární těleso, urči všechny jeho potřebné body pomocí (5.20).
- Je-li i vnitřní vrchol:

a) urči $\mathbf{s}_{k,0}$ a $\mathbf{T}_{k,0}$ pro všechny jeho přímé následníky $\{k\}$ pomocí vztahů

$$\begin{aligned} \mathbf{s}_{k,0} &= \mathbf{s}_{i,0} + \mathbf{T}_i \mathbf{s}_k \\ \mathbf{T}_{k,0} &= \mathbf{T}_{i,0} \mathbf{T}_k \,. \end{aligned} \tag{5.24}$$

b) vyvolej **Pozice**(k).

Vyšetření kolize se provede rovněž rekurzivním způsobem. V tomto případě se kolize zjišťuje mezi vrcholy stromu (tj. podstromy):

Kolize(i, j):

- Lze-li snadno zjistit, že ke kolizi mezi vrcholy i, j nemůže dojít, vrať negativní výsledek.
- Je-li i vnitřní vrchol:

- je-li j vnitřní vrchol, vyvolej **Kolize**(k,l) pro všechny dvojice (k,l), kde $\{k\}$ jsou přímí následníci vrcholu i a $\{l\}$ jsou přímí následníci vrcholu j.
- je-li j list stromu, vyvolej **Kolize**(k,j) pro všechny přímé následníky $\{k\}$ vrcholu i.

- Je-li i list stromu:
  - je-li j vnitřní vrchol, vyvolej **Kolize**(i,l) pro všechny přímé následníky $\{l\}$ vrcholu j.
  - je-li j list stromu, proveď zjištění kolize pro mnohostěny q_i, r_j.
- Je-li některý z výsledků v předchozích krocích pozitivní (tj. došlo ke kolizi), vrať pozitivní výsledek, jinak vrať negativní výsledek

První řádek výše uvedeného algoritmu odpovídá vyloučení případů, kdy modely jsou dostatečně vzdálené. Urychlení pomocí opsaných koulí a válců lze beze změny aplikovat, jedná-li se o listy, tj. elementární tělesa. Navíc je však možné pro urychlení definovat opsané koule a válce u vnitřních vrcholů, které odpovídají skupinám těles.

Nejefektivnější variantou v tomto případě je sestrojit opsané koule a válce u skupin těles s konstantní relativní vzdáleností před začátkem výpočtu. V tomto případě je možné využít techniky opožděného vyhodnocování, kdy souřadnice potřebných bodů se určí až v algoritmu vyšetření kolize. Tím dojde k ušetření výpočtu pozic bodů v těch částech modelu, kde lze rozhodnout o kolizi pouze na základě opsaných koulí, popř. válců. Tento přístup se však zdá být implementačně značně náročný, ale dosud nebyl podrobně analyzován. Komplikací, která však lze vyřešit softwarově, je definování opsaných koulí a válců pro skupiny těles při vytváření modelu.

Jednodušší, i když zřejmě méně efektivní variantou je automaticky určovat koule (popř. i válce, třebaže to je již obtížnější) opsané skupinám těles reprezentovaným vnitřními vrcholy stromu zpětným postupem v každém bodě trajektorie. Pro výpočet středu a poloměru koule odpovídající konkrétnímu vrcholu se využije znalost koulí opsaných jeho následníkům. Tento výpočet není náročný a lze jej provést v rámci určení koulí opsaných elementárním tělesům:

Koule(i):
- Je-li i list stromu, urči opsanou kouli a válec.
- Je-li i vnitřní vrchol:
  - vyvolej **Koule**(k) pro všechny jeho přímé následníky $\{k\}$
  - na základě znalosti středů a poloměrů koulí opsaných následníkům urči opsanou kouli pro vrchol i.

Dosud bylo uvažováno, že kolize se určuje pouze mezi modely systému a překážek. Ve skutečnosti toto rozdělení v praxi často nestačí, neboť může dojít k nežádoucí kolizi některých částí systému vůči sobě. Výše popsané algoritmy lze snadno upravit i pro tento případ. Na druhé straně však mechanicky spojené členy systému jsou vždy v kolizi, což je v pořádku. Je třeba odlišit tyto dva případy. Jednoduchým technickým řešením je opatřit jednotlivá elementární tělesa q_i celočíselnými indexy $L(q_i)$, které odpovídají příslušnosti k členům. Předpokládá se, že indexy sousedních členů se liší o 1. Současně je možné vyhradit nulový index neaktivním částem, které se neuplatní při výpočtu. Kolize je pak testována pouze mezi tělesy q_i, q_j, u kterých platí

$$L(q_i) > 0, L(q_j) > 0 \text{ a } \left|L(q_i) - L(q_j)\right| > 1. \tag{5.25}$$

5.5. Mnohostěnové modely

V této kapitole je popsána základní verze prakticky realizované metody detekce kolize mezi dvěma mnohostěny.

Mnohostěn P v $\mathbb{R}^3$ je definován konvexní obal m konvexně nezávislých „rohových" bodů $X_1,\ldots,X_m$, které neleží v jedné rovině

$$P=\left\{X \mid X=\sum_{i=1}^{m} a_i X_i,\ 0\le a_i \le 1,\ \sum_{i=1}^{m} a_i = 1\right\} \tag{5.26}$$

kde $X_i=(x_i,y_i,z_i)$. Zkráceně označme

$$P=\text{conv}\,\{X_i \mid i=1,\ldots,m\}. \tag{5.27}$$

Negativní poloprostor H_i^- je definován čtveřicí reálných čísel $[a_i,\ b_i,\ c_i,\ d_i]$:

$$H_i^- =\{X \mid a_i x+b_i y+c_i z+d_i \le 0\}. \tag{5.28}$$

Pozitivní poloprostor H_i^+ definujme jako doplněk k H_i^-, tj.

$$H_i^+ =\{X \mid a_i x+b_i y+c_i z+d_i > 0\}. \tag{5.29}$$

Řekneme, že rovina H *odděluje* množiny A a B, jestliže

$$A\subset H^+ \text{ a } B\subset H^-. \tag{5.30}$$

Uvažujme, že systém Q a překážky R jsou nahrazeny soustavou mnohostěnů ve smyslu předchozí kapitoly. Problém je tak redukován na vyšetření kolize mezi dvěma mnohostěny.

Navíc je možné např. požadovat, aby se systém pohyboval v omezeném pracovním prostoru, který je definován průnikem poloprostorů:

$$W=\bigcap_{i=1}^{n_w} W_i^-. \tag{5.31}$$

Pro vyšetření kolize mnohostěnů bylo využito dvou jednoduchých tvrzení konvexní analýzy, jejichž důkaz je vynechán:

Každý mnohostěn je možné vyjádřit jako průnik nějakých poloprostorů $P=\bigcap_{i=1}^{p} H_i^-$.

Jestliže body $X_1,\ldots,X_m$ leží v poloprostoru H^+, resp. H^-, pak také jejich konvexní obal leží v H^+, resp. H^-.

Je-li dán mnohostěn P a množina poloprostorů $\{H_i^-,\ i=1,\ldots,p\}$ takových, že $P=\bigcap_{i=1}^{p} H_i^-$, označme tuto množinu jako *minimální*, jestliže po vyjmutí jakéhokoliv z prvků z této množiny je $\bigcap_{i=1}^{p-1} H_i^- \neq P$.

Použitý princip detekce kolize je založen na hledání oddělující roviny. Vyjádřeme mnohostěny $P = \text{conv}\{P_i \mid i = 1,...n\}$ a $S = \text{conv}\{S_i \mid i = 1,...,m\}$ jako průniky poloprostorů

$$P = \bigcap_{i=1}^{p} H_i^- \ , \ S = \bigcap_{i=1}^{s} K_i^- \ . \tag{5.32}$$

Jestliže existuje index i takový, že všechny rohové body S (a tedy na základě tvrzení výše všechny body S) leží v H_i^+, P a Q musí mít prázdný průnik, neboť všechny body tělesa P leží v H_i^-. Obdobně, jestliže existuje index j takový, že všechny rohové body P leží v K_j^+, potom rovněž $P \cap S = \varnothing$.

Zjištění, zda mnohostěn $P = \text{conv}\{P_i \mid i = 1,...n\}$ leží v pracovním prostoru (5.31) je jednodušší – stačí ověřit, zda ve W leží všechny jeho rohové body, tj.

$$P_i \in W_j^- \ , \ \forall i, j \ . \tag{5.33}$$

Zjištění, zda daný bod $X \in H^+$ je lacinou operací - stačí pouze dosadit souřadnice bodu do nerovnosti (5.29). Pro praktickou implementaci je výhodnější poloprostory definovat bodem $V \in H$ a normálovým vektorem $\mathbf{n}$. Poloprostory jsou pak snadno určeny na základě polohy rohových bodů. V tomto případě platí

$$X \in H^+ \Leftrightarrow \langle \mathbf{n} \mid X - V \rangle > 0 \, . \tag{5.34}$$

Zajímavé je srovnání uvedeného algoritmu využívajícího konvexnosti mnohostěnů s postupem založeným na vyšetření průsečnic povrchových ploch pomocí nástrojů analytické geometrie. Uvažujme příklad vyšetření kolize dvou kvádrů.

1. Hledání oddělující roviny

Každý kvádr má 6 poloprostorů a 8 rohových bodů. Proto je třeba provést maximálně 2.(6.8.3) = 288 operací součinu a přibližně stejně tolik operací součtu (výpočet je možné ukončit dříve, jestliže byl nalezen oddělující poloprostor).

2. Analytická geometrie

Předpokládejme, že operace násobení a dělení jsou stejně náročné a že porovnání dvou reálných hodnot je stejně náročné jako sčítání. Pro určení průniku dvou polygonů v prostoru je třeba nejprve nalézt průsečnici rovin. Předpokládáme, že jsou známy bázové vektory rovin v_1^i, v_2^i, bod polygonu X^i a normály n^i rovin, $i = 1,2$:

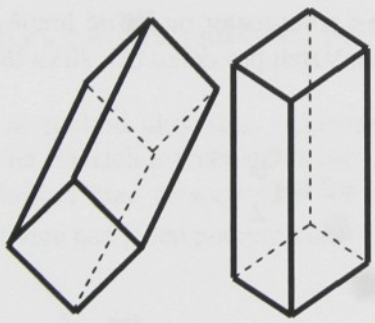
- vektorový součin normál $\rho = n^1 \times n^2$ ... 6 *, 3 +
- nalezení bodu na průsečnici znamená vyřešit soustavu rovnic 3 x 3. Výpočet Gaussovou eliminační metodou vyžaduje cca 17 *, 11 +
- dohromady 23 *, 14 +

Dále je třeba určit průsečíky průsečnice s hranami polygonů pro oba polygony. Průsečíky je nejvýhodnější hledat v bázi vektorů v_1^i, v_2^i, tj. v rovině:

- Vyjádříme průsečnici v bázi v_1^i, v_2^i. Je třeba řešit soustavu rovnic 2 x 2 - 8 *, 4 +.
- Průsečík s hranou vyžaduje opět řešení soustavy 2 x 2 rovnic, tj. 8 *, 4 +. Přitom každý z polygonů má 4 hrany
- Hrana protíná průsečnici, pokud průsečík leží uvnitř známého uzavřeného intervalu. Toto zjištění vyžaduje 2 operace porovnání, tj. 2 +
- dohromady 2 [(8 *, 4 +) + 4 (8 *, 4 +) + (2 +)] = 80 *, 44 +

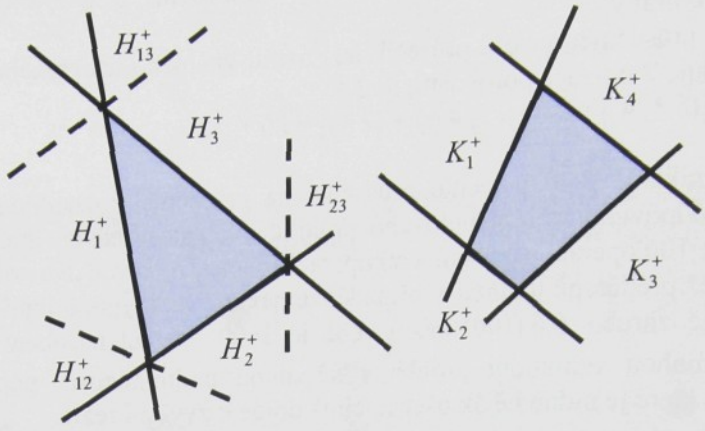
Na závěr je třeba na základě polohy těchto průsečíků na průsečnici rozhodnout, zda ke kolizi dochází nebo ne, což odpovídá ověření nulového průniku dvou uzavřených intervalů, tedy 4 +. Dohromady tedy zhruba 100 operací násobení a 60 operací součtu pro každé dva polygony.
Výpočet lze však rovněž předčasně ukončit v případě, že některé z polygonů mají průnik. Celková náročnost je maximálně zhruba 6.6 (100*, 60+), což je 3600 operací násobení a 2160 operací součtu. Kromě toho mohou vzniknout problémy s špatnou podmíněností, popř. neexistujícím řešením soustav rovnic, které je nutno nějak ošetřit, čímž dojde k zvýšení režie.

První z metod je tedy přibližně 10-krát rychlejší než druhá a přitom se jedná o velmi jednoduchý algoritmus. Bohužel však lze nalézt případ, kdy je tímto způsobem detekována kolize, i když nenastává (Obr. 5.1). Z konvexní analýzy je sice známo, že oddělující rovina musí existovat, nemusí však být některou z rovin H_i, K_j, které definují mnohostěny. Ukazuje se však, že tento případ nastává poměrně zřídka. Při využití pro návrh optimálních trajektorií se tím poněkud zmenší množina přípustných řešení. Z tohoto hlediska je možné metodu považovat za přibližnou. Podstatné je, že zůstává zaručeno, že ke kolizi nedochází v případech, kdy metoda kolizi nesignalizuje.



Obr. 5.1: Případ, kdy minimální množiny poloprostorů neobsahují oddělující poloprostor

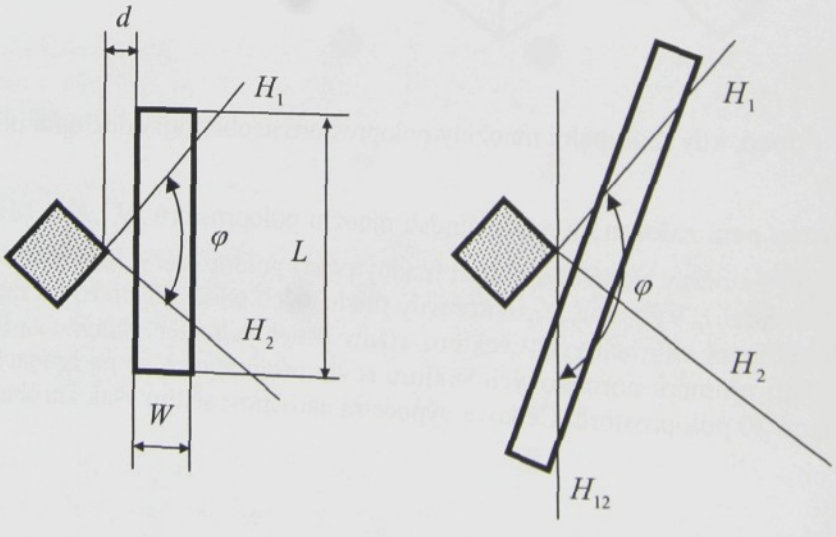
Výše uvedený postup není založen na minimálnosti množin poloprostorů H_i^-, K_j^-. Metoda se stane přesnější, pokud tyto množiny rozšíříme o další (nadbytečné) poloprostory definované na hranách a rozích (viz Obr. 5.2 vlevo). Výhodou je, že normály příslušných oddělujících rovin lze snadno určit jako konvexní kombinaci normálových vektorů rovin přilehlých stěn. Např. v případě kvádru pomocí aritmetických průměrů normálových vektorů rovin protínajících se na hranách a vrcholech takto vznikne dalších 20 poloprostorů. Celková výpočetní náročnost se tím však zhruba zčtyřnásobí.



Obr. 5.2: Nadbytečné poloprostory definované v rozích

Nejhorší případ nastává, jestliže dva protáhlé mnohostěny jsou vůči sobě obráceny hranou, jako na Obr. 5.1, a jejich osy jsou na sebe kolmé. Na základě úhlu oddělujících rovin lze odhadnout minimální dovolenou vzdálenost v nejhorším případě, která je úměrná délce mnohostěnu. Obr. 5.3 znázorňuje tuto situaci v případě minimální sady poloprostorů, kdy úhel oddělujících rovin φ je 90° (vlevo) a rovněž v případě rozšířené sady o jeden poloprostor na každé hraně určený aritmetickým průměrem, kdy příslušný úhel φ je 135° (vpravo). Vztah pro délku (L), šířku tělesa (W) a dovolenou vzdálenost v nejhorším případě d je zřejmě

$$\frac{L}{d+W} = 2 \tan\frac{\varphi}{2}. \tag{5.35}$$



Obr. 5.3: Případ chybné detekce kolize

Z (5.35) je možné vyjádřit poměr d/L:

$$\frac{d}{L} = \frac{1}{2}\operatorname{cotg}\frac{\varphi}{2} - \frac{W}{L}. \tag{5.36}$$

Parametr W/L může být chápan jako určitý omezující faktor pro vytváření modelu – příliš dlouhá a současně úzká tělesa je třeba nahradit pomocí většího počtu kratších. Je-li tento faktor zadán, lze pomocí poměru d/L odhadnout přesnost metody.

Tab. 5-1 ukazuje hodnoty L/d pro $W/L = 0.1$ v případě kvádru. První sloupec odpovídá celkovému počtu poloprostorů. Třetí řádek odpovídá případu, kdy na každé hraně jsou definovány dva nadbytečné poloprostory a v rozích tři. V tomto případě je však celková náročnost již srovnatelná s postupem založeným na hledání průsečnic povrchových ploch, proto pro praktické využití jsou zajímavější první dva řádky tabulky, které odpovídají situacím na Obr. 5.2 a Obr. 5.3. Druhý řádek představuje určitý kompromis mezi přesností a efektivitou. Při náhodném generování mnohostěnů v tomto případě je kolize detekována téměř vždy správně. Proto byla tato varianta zvolena pro praktickou realizaci.

p	φ	L/d
6	90°	2.5
26	135°	9.33
54	150°	29.43

Tab. 5-1: Poměr délky a dovolené vzdálenosti v nejhorším případě pro různé úhly rovin φ

U obecných tvarů se mohou úhly stěn vzájemně výrazně lišit. Pak je vhodné přidat dodatečné poloprostory pouze na hranách a vrcholech s ostrými, popř. málo tupými úhly – např. tak, aby úhel φ byl u všech vrcholů a hran alespoň 120°. U hran, které svírají úhel menší než 60° je v tomto případě nutno přidat více než jeden poloprostor.

5.6. Zjištění míry kolize

Jednou z důležitých předností mnohostěnových modelů pro návrh optimálních trajektorií je možnost konstrukce rychle vypočitatelné míry kolize $\mu(Q,R)$, která je navíc spojitá, nebo dokonce hladká v prostoru parametrů trajektorie **z**. Integrál této funkce může být využit jako pokutový člen kritéria (5.9).

Uvažujme, že platí:

- Pozice bodů množiny Q je v každém čase hladce závislá na vektoru polohy členů **y**
- Vektor **y** je v každém čase spojitě diferencovatelnou funkcí bázových koeficientů trajektorie, popř. řízení
- Bázové koeficienty jsou lineárně závislé na složkách vektoru **z**

Z těchto podmínek pak okamžitě vyplývá, že pozice libovolného bodu množiny Q je spojitě diferencovatelnou funkcí **z**.

První podmínka v případě mechanických systémů s tuhými členy přímo vyplývá z algoritmu výpočtu polohy bodů, který byl popsán v kap. 5.3. Druhou a třetí podmínku lze snadno ověřit v případě nahrazení trajektorie součtem bázových funkcí. V tomto případě je závislost **y** na bázových koeficientech lineární, tedy hladká.

Pokud je nahrazen průběh řízení, je **u** spojitou funkcí. Jelikož **f** je spojitě diferencovatelná, lze vyjádřit variaci trajektorie v závislosti na variaci řízení linearizací soustavy (4.1):

$$\delta\dot{\mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \delta\mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \delta\mathbf{u} \,. \tag{5.37}$$

Rovnice (5.37) je lineární a její řešení lze tedy explicitně vyjádřit ve tvaru

$$\delta\mathbf{x}(t) = \boldsymbol{\Phi}(t,0)\ \delta\mathbf{x}(0) + \int_0^t \boldsymbol{\Psi}(t,\tau)\,\delta\mathbf{u}(\tau)\,d\tau \tag{5.38}$$

kde $\boldsymbol{\Phi}(t,t_0)$ je fundamentální matice řešení homogenní soustavy

$$\delta\dot{\mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \delta\mathbf{x} \tag{5.39}$$

a funkce $\boldsymbol{\Psi}(t,\tau)$ má tvar

$$\boldsymbol{\Psi}(t,\tau) = \boldsymbol{\Phi}(t,\tau)\,\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \,. \tag{5.40}$$

Jelikož v obou případech je stejná počáteční podmínka $\mathbf{x}(0) = \mathbf{x}_0$, je $\delta\mathbf{x}(0) = \mathbf{0}$.

Dosadíme-li za variaci řízení

$$\delta\mathbf{u}(t) = d\mathbf{A}\,\mathbf{b}(t) \tag{5.41}$$

lze vyjádřit složky δx_i v každém čase jako

$$\delta x_i(t) = \int_0^t \sum_{j=1}^{m} \left(\Psi_{ij}(t,\tau) \sum_{k=0}^{N} da_{jk}\, b_k(\tau) \right) d\tau = \sum_{j=1}^{m} \sum_{k=0}^{N} \int_0^t \Psi_{ij}(t,\tau)\, b_k(\tau)\, d\tau\ da_{jk} \,. \tag{5.42}$$

Jelikož diferenciální přírůstek složek δx_i v daném čase t lze vyjádřit jako lineární kombinaci přírůstků koeficientů da_{ij}, musí být členy $\int_0^t \Psi_{ij}(t,\tau) b_k(\tau)\, d\tau$ spojitými parciálními derivacemi podle a_{ij}.

Třetí podmínka je v případě nahrazení řízení triviální, neboť složky vektoru **z** přímo odpovídají bázovým koeficientům.

Označme $h(H^-, X)$ vzdálenost bodu X od poloprostoru H^-. Je-li X_0 nějaký bod na oddělující rovině H a $\mathbf{n}(H^-)$ jednotkový normálový vektor roviny H orientovaný ven z H^-, platí:

$$h(H^-, X) = \left\langle \mathbf{n}(H^-) \,|\, X - X_0 \right\rangle . \tag{5.43}$$

Definujme dále

$$h_-(H^-, X) = \min\left\{ h(H^-, X), 0 \right\} . \tag{5.44}$$

Vyjádřeme mnohostěny P, S jako průnik poloprostorů:

$$P = \text{conv}\{P_i \mid i = 1,...,n\} = \bigcap_{i=1}^{p} H_i^-$$
$$S = \text{conv}\{S_i \mid i = 1,...,m\} = \bigcap_{i=1}^{s} K_i^- \tag{5.45}$$

Pak je možné definovat následující funkce:

$$\mu_1(P,S) \equiv \min\left\{ -\max_{i=1,..p}\min_{j=1,..m}\left\{h\_(H_i^-,S_j)\right\}, -\max_{i=1,..q}\min_{j=1,..n}\left\{h\_(K_i^-,P_j)\right\}\right\} \tag{5.46}$$

$$\mu_2(P,S) \equiv \min\left\{ -\max_{i=1,..p}\left\{\frac{1}{m}\sum_{j=1}^{m} h\_(H_i^-,S_j)\right\}, -\max_{i=1,..q}\left\{\frac{1}{n}\sum_{j=1}^{n} h\_(K_i^-,P_j)\right\}\right\} \tag{5.47}$$

$$\mu_3(P,S) \equiv \frac{1}{p+q}\ln\left(1+\frac{1}{m^p}\prod_{i=1}^{p}\sum_{j=1}^{m}\left[h\_(H_i^-,S_j)\right]^2 \times \frac{1}{n^q}\prod_{i=1}^{q}\sum_{i=1}^{n}\left[h\_(K_i^-,P_j)\right]^2\right) \tag{5.48}$$

Lze se snadno přesvědčit, že funkce μ_1 a μ_2 spojitě závisí na pozicích bodů (předpokládá se, že oddělující roviny poloprostorů jsou určovány z jejich souřadnic). Funkce μ_3 je pak vzhledem k polohám bodů navíc spojitě diferencovatelná. Pro ověření hladkosti μ_3 uvažujme derivaci zjednodušené funkce

$$\frac{d}{dx}\ln\left(1+\prod_{i=1}^{k} f_i(x)\right) = \frac{1}{1+\prod_{i=1}^{k} f_i(x)}\left[f_1' f_2 ... f_k + f_1 f_2' ... f_k + ... + f_1 f_2 ... f_k'\right]. \tag{5.49}$$

Předpokládejme, že $\prod_{i=1}^{k} f_i(x) \geq 0$, což lze snadno ověřit v (5.48). Derivace (5.49) jsou tedy spojité, pokud jsou spojité funkce f_i'. To však v uvažovaném případě platí, neboť funkce $\left[h\_(H_i^-,S_j)\right]^2, \left[h\_(K_i^-,P_j)\right]^2$ jsou spojitě diferencovatelné a funkce f_i odpovídají součtům těchto členů.

Z úvah na počátku této kapitoly pak vyplývá, že funkce μ_1, μ_2 jsou spojité podle parametrů trajektorie **z** a μ_3 je navíc podle **z** spojitě diferencovatelná.

Funkce μ_1 - μ_3 pro mnohostěny zřejmě vyhovují axiomům (5.4) - (5.7) a jsou tedy vhodným vyjádřením míry kolize. Vyhodnocení μ_3 je časově náročnější, ale pokutový člen (5.9) s touto funkcí umožňuje výpočet gradientu, který může být využit pro efektivnější optimalizaci – viz část 6. I v tomto případě je však možné výpočet předčasně ukončit, pokud je některý ze součinitelů nulový.

Pro celé modely je míra kolize určena jako součet přes všechny dvojice mnohostěnů:

$$\mu(Q,R) = \sum_{i=1}^{n}\sum_{j=1}^{m}\mu(q_i, r_j) \tag{5.50}$$

kde μ je jedna z funkcí μ_1 - μ_3. Jinou možností je vzít maximum

$$\mu(Q,R) = \max_{i=1,\dots,n} \max_{j=1,\dots,m} \left\{ \mu(q_i, r_j) \right\}. \tag{5.51}$$

Lze snadno ověřit, že axiomy (5.4) - (5.7) jsou v obou případech splněny. Součet (5.50) je výhodnější, neboť zachovává případnou hladkost a lépe charakterizuje rozsah kolize.

Pro výpočet (5.50), popř. (5.51) u stromových modelů se použijí postupy obdobné zjištění kolize. S drobnými úpravami se využijí i výše popsané metody urychlení – např. pokud koule opsané q_i a r_j nejsou v kolizi, jistě platí $\mu(q_i, r_j) = 0$ a je možné tento výpočet vynechat. Celková časová náročnost je však větší, neboť výpočet není možné ukončit při nalezení první dvojice, kde $\mu(q_i, r_j) \neq 0$, jako v případě pouhé detekce kolize.

5.7. Kolize mnohostěnů s mnoha vrcholy

Hlavním nedostatkem metody popsané v předchozí kapitole je hrubá aproximace oblých tvarů. Je sice možné nahradit např. část koule mnohostěnem s dostatečným počtem vrcholů, avšak náročnost vyšetření kolize mezi dvěma mnohostěny je v nejhorším případě přibližně úměrná druhé mocnině počtu vrcholů (pokud předpokládáme, že počet vrcholů a stěn je zhruba stejný a oba mnohostěny mají podobný počet vrcholů). Mají-li mnohostěny již několik desítek vrcholů, může být náročnost této operace příliš vysoká. Pro návrh optimálních trajektorií byla původně předpokládána pouze hrubá aproximace tvaru technických systémů, a z tohoto hlediska byly mnohostěnové modely relativně vyhovující.

Základní možností, jak výpočet v tomto případě urychlit, je nahradit mnohostěny s mnoha vrcholy soustavou menších mnohostěnů. Rozdělení může být realizováno i hierarchickým způsobem. V tomto případě se pouze stromová struktura celého modelu, popsaná v kap. 5.4, dále větví. Nevýhodou je komplikovanější vytváření modelu, i když rozdělení mnohostěnů na menší části lze realizovat softwarově.

Jiná možnost výrazného urychlení pro mnohostěny s větším počtem vrcholů existuje v případě vyhodnocení nehladké funkce μ_1 (5.46). Tento postup, popsaný dále, však doposud nebyl prakticky realizován.

Uvažujme problém zjištění míry kolize dvou mnohostěnů $P = \text{conv}\{P_i \mid i = 1,\dots n\} = \bigcap_{i=1}^{p} H_i^-$ a $S = \text{conv}\{S_j \mid j = 1,\dots,m\} = \bigcap_{j=1}^{q} K_j^-$. Lze snadno ověřit, že pro funkci $\mu_1(P,S)$ platí:

$$\mu_1(P,S) = \max\left\{ \min\left\{ -\max_{i=1,..p} \min_{j=1,..m} \left\{ h(H_i^-, S_j) \right\}, -\max_{i=1,..q} \min_{j=1,..n} \left\{ h(K_i^-, P_j) \right\} \right\}, 0 \right\}. \tag{5.52}$$

Problém lze tedy redukovat na efektivní vyhodnocení funkce

$$\max_{i=1,..p} \left\{ \min_{j=1,..m} \left\{ h(H_i^-, S_j) \right\} \right\}. \tag{5.53}$$

Uvažujme nejprve, že $p \ll m$. V tom případě záleží zejména na efektivním vyhodnocení funkce $\min_{j=1,..m} \left\{ h(H_i^-, S_j) \right\}$. Vnější maximalizace se provede běžným cyklem přes všechna i. Pro pevně dané H_i^- je

$$h(H_i^-, S_j) = h(S_j) = \left\langle \mathbf{n}_i \mid S_j - X_{0i} \right\rangle = \left\langle \mathbf{n}_i \mid S_j \right\rangle + d_i \tag{5.54}$$

Lineární funkce $h(X)$ má na konvexním obalu bodů S_j minimum, které je současně jediné v lokálním smyslu, právě v některém z rohových bodů. Odtud okamžitě vyplývá algoritmus pro jeho nalezení, který je analogií simplexové metody lineárního programování:

- z daného rohového bodu S_j určit hodnotu funkce pro všechny sousedící rohové body S_k
- jestliže $h(S_j) \le h(S_k)$ pro všechna k, je $h(S_j)$ minimem. V opačném případě dosadit místo S_j nějaký bod z $\{S_k\}$ s nižší hodnotou (nejvhodnější volba je patrně $S_j \leftarrow \arg\min\{h(S_k)\}$, tj. hledá se ve směru největšího spádu)

Výhodou tohoto postupu je, že potřebný počet kroků může být mnohem menší než počet vrcholů. Algoritmus však zřejmě vyžaduje uchovávat pro každý vrchol odkazy na okolní vrcholy.

Předchozí postup lze rozšířit i pro urychlení vnějšího cyklu:

- z daného poloprostoru H_i^- určit hodnotu funkce $\nu_k = \min_{j=1,\dots,m}\{h(H_k^-, S_j)\}$ pro všechny roviny sousedících stěn H_k.
- jestliže $\nu_i \ge \nu_k$ pro všechna k, ukonči výpočet. V opačném případě zvolit místo H_i stěnu z $\{H_k\}$ s nejvyšší hodnotou ν_k

Výpočet každé hodnoty vyžaduje sice minimalizaci $h(H_i^-, S_j)$ podle j, avšak řešení $h(H_i^-, S_j) \to \min_j$ poskytuje velmi dobrý počáteční odhad pro okolní body $\{S_k\}$. Výpočetní náročnost proto není o mnoho větší než v předchozím případě. Ukončující podmínka algoritmu je však v tomto případě pouze nutnou, ale nikoliv postačující podmínkou extrému. Konečný bod tedy v některých případech nemusí být skutečným řešením, třebaže lze předpokládat, že od něj nebude vzdálen. Detailní rozbor dosud nebyl dokončen.

5.8. Vyšetření kolize podél trajektorie

V předchozích kapitolách byly popsány typy modelů a algoritmy pro efektivní detekci kolize, popř. určení její míry v konkrétním čase. Pro výpočet optimálních trajektorií je však třeba kolizi vyšetřit v dostatečně vysokém počtu časových bodů podél trajektorie. Je zřejmé, že pro reálné situace tento výpočet může být i při použití nejefektivnějších postupů velmi časově náročný. Jelikož vyšetření kolize podél trajektorie je z hlediska algoritmu optimalizace pouze elementární operací, která může být vykonána např. 10^5 - krát během výpočtu, je tato část úzkým místem, které se na celkové časové náročnosti podílí převážnou měrou.

Uvažujme, že systém $Q(t)$ se pohybuje v intervalu $t \in \left[0, t_f\right]$. Překážky nejprve uvažujme jako statické. Základní přístup je založen na zvolení dostatečně krátké konstantní periody vzorkování dt. Krok vzorkování dt v tomto případě je možné volit úměrný převrácené hodnotě maximální rychlosti systému v intervalu $t \in \left[0, t_f\right]$. V úsecích, kde se systém pohybuje pomaleji, je však algoritmus vyšetření kolize vyvolán zbytečně často a dojde k plýtvání výpočetním výkonem. Přitom není nijak zaručeno, že kolize nenastane. Jelikož kolize může být testována pouze v diskrétních časových krocích t_i, algoritmus nemusí detekovat kolizi, třebaže ve skutečnosti existuje časový bod t takový, že $Q(t) \cap R \neq \varnothing$.

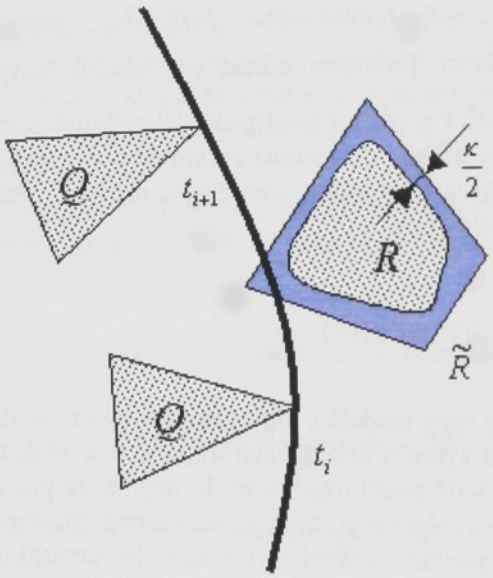
Kvalitnějších výsledků je možné dosáhnout, jestliže časové body t_i nejsou ekvidistantní. Cílem dále uvedeného postupu je navrhnout body t_i tak, aby:

- bylo zaručeno, že ke kolizi mezi překážkou a systémem nedojde v žádném bodě $t \in \left[0, t_f\right]$ (tj. ani mezi testovacími body t_i)
- počet bodů t_i byl co nejmenší

Předpokládejme, že v daném bodě t_i je $Q(t_i) \cap R = \varnothing$. Pokusme se určit maximální $t_{i+1} > t_i$ tak, aby v případě, že $Q(t_{i+1}) \cap R = \varnothing$, platilo rovněž $Q(t) \cap R = \varnothing$ pro všechna $t \in (t_i, t_{i+1})$. Takto formulovaný problém ovšem v obecném případě není řešitelný, neboť v čase t_i může být Q libovolně blízko překážce R a tato vzdálenost není předem známá.

Uvažujme však, že překážka R je aproximována modelem $\tilde{R}$ s určitým kladným přesahem, který je znám (Obr. 5.4). Kolize je testována mezi Q a $\tilde{R}$. Pak lze nalézt t_{i+1} tak, aby bylo zaručeno $Q(t) \cap R = \varnothing$ pro všechna $t \in (t_i, t_{i+1})$ (zde se však nejedná o kolizi s modelem $\tilde{R}$, ale se skutečnou překážkou R). Přesná formulace problému je následující:

Uvažujme, že pro daný časový bod t_i platí $Q(t_i) \cap \tilde{R} = \varnothing$. Úlohou je určit $t_{i+1} > t_i$ co největší tak, že jestliže $Q(t_{i+1}) \cap \tilde{R} = \varnothing$, pak platí $Q(\tau) \cap R = \varnothing$ pro všechna $\tau \in (t_i, t_{i+1})$. Model překážky $\tilde{R}$ může být v libovolné pozici a platí $R \subset \tilde{R}$.



Obr. 5.4: Určení bodu t_{i+1}

Nejprve uvažujme pohyb izolovaného bodu X. Označme $S_X(t_1, t_2)$ délku dráhy bodu X od t_1 do t_2, $t_2 \geq t_1$. Platí:

$$S_X(t_1, t_2) = \int_{t_1}^{t_2} dS_X \tag{5.55}$$

kde dS_X je diferenciální přírůstek dráhy bodu $X = (x, y, z)$

$$dS_X = \|dX\| = \sqrt{dx^2 + dy^2 + dz^2}\,. \tag{5.56}$$

Dle předpokladů, pro každý bod $X \in Q$ platí $X(t_1) \notin \tilde{R}$, $X(t_2) \notin \tilde{R}$ a $X(\tau) \notin R$ pro všechna $\tau \in (t_1, t_2)$. Definujme

$$\kappa = \inf \left\{ S_X(t_1,t_2) \mid X(t_1) \notin \tilde{R},\ X(t_2) \notin \tilde{R} \text{ a } \exists \tau \in (t_1,t_2) : X(\tau) \in R \right\} \tag{5.57}$$

kde $X(t_1), X(t_2)$ jsou libovolné body v prostoru a $X(\tau)$ je libovolný bod na jakékoliv trajektorii mezi $X(t_1)$ a $X(t_2)$.

Z (5.57) vyplývá, že jestliže $X(t_i) \notin \tilde{R}$, $X(t_{i+1}) \notin \tilde{R}$ a $S_X(t_i,t_{i+1}) < \kappa$, pak musí platit $X(\tau) \notin R$ pro všechna $\tau \in (t_i,t_{i+1})$. Pokud toto platí pro všechny body $X \in Q$, pak $Q(\tau) \cap R = \varnothing$ pro všechna $\tau \in (t_i,t_{i+1})$. Odtud vyplývá, že t_{i+1} je možné určit tak, aby byl co nejvyšší a současně $S_X(t_i,t_{i+1}) < \kappa$ pro všechna $X \in Q$:

$$t_{i+1} = \sup \left\{ t \mid S_X(t_i,t) < \kappa, \forall X \in Q \right\}. \tag{5.58}$$

Veličina κ má jasný geometrický význam. Jestliže je překážka R nahrazena modelem $\tilde{R}$ a trajektorie všech bodů $X \in Q$ je spojitá v čase, je κ dvojnásobkem minimální vzdálenosti od okraje R k okraji $\tilde{R}$ (Obr. 5.4).

Přímý výpočet t_{i+1} ze vztahu (5.58) není prakticky proveditelný, neboť Q obsahuje nekonečné množství bodů. Je však možné použít odhad s využitím nerovností. Z (5.58) vyplývá, že

$$\max_{X \in Q} \left\{ S_X(t_i,t_{i+1}) \right\} < \kappa. \tag{5.59}$$

Dosaďme (5.55):

$$\max_{X \in Q} \left\{ S_X(t_i,t_{i+1}) \right\} = \max_{X \in Q} \left\{ \int_{t_1}^{t_2} dS_X \right\} \le \int_{t_1}^{t_2} dS_Q \tag{5.60}$$

kde

$$dS_Q = \max_{X \in Q} \left\{ dS_X \right\}. \tag{5.61}$$

Označme dále

$$S_Q(t_1,t_2) = \int_{t_1}^{t_2} dS_Q. \tag{5.62}$$

Čas t_{i+1} pak lze určit tak, že platí

$$t_{i+1} = \sup \left\{ t \mid S_Q(t_i,t) < \kappa \right\}. \tag{5.63}$$

Při praktickém výpočtu je veličina dS_Q integrována od t_i, dokud neplatí $S_Q \ge \kappa$. Pak t_{i+1} je poslední časový krok, kde platilo $S_Q < \kappa$.

Veličinu dS_Q je možné určit v závislosti na dt a současné pozici pomocí geometrických parametrů systému: platí $dS_Q = \left\| dX_Q \right\|$, kde X_Q je bod s nejvyšší rychlostí ze všech bodů $X \in Q$.

Jestliže Q je mnohostěn, následující tvrzení umožňuje určit dS_Q pouze pomocí přírůstků trajektorie rohových bodů, což je užitečné pro praktickou implementaci:

Tvrzení. Jsou-li X_i rohové body Q, platí:

$$dS_Q = \max_i \left\{ dS_{X_i} \right\}. \tag{5.64}$$

Důkaz: Pro bod $X \in Q$, který lze vyjádřit jako konvexní kombinaci

$$X = \sum_{i=1}^{m} a_i X_i \,,\ \sum_{i=1}^{m} a_i = 1 \,,\ a_i \geq 0 \tag{5.65}$$

platí:

$$\begin{aligned} (dS_X)^2 &= \left(\sum_{i=1}^{m} a_i \, dX_i^T \right) \left(\sum_{i=1}^{m} a_i \, dX_i \right) = \sum_{j=1}^{m} \sum_{i=1}^{m} a_i a_j \, dX_i^T dX_j \leq \\ &\leq \sum_{j=1}^{m} \sum_{i=1}^{m} a_i a_j \left\| dX_i^T \right\| \left\| dX_j \right\| \leq \sum_{j=1}^{m} \sum_{i=1}^{m} a_i a_j \max_k \left\{ \left\| dX_k \right\|^2 \right\} = \max_k \left\{ \left\| dX_k \right\|^2 \right\}. \ \blacklozenge \end{aligned} \tag{5.66}$$

Uvažujme nyní mnohostěnové modely $Q = \bigcup_{j=1}^{q} q_j$. Body testování kolize mohou být určeny pro celý model jako celek pomocí rozšíření veličin dS_Q a κ:

$$dS_Q = \max_i \left\{ dS_{q_i} \right\}, \ \kappa = \min_j \left\{ \kappa_j \right\}. \tag{5.67}$$

Mnohem efektivnější řešení však spočívá v určení testovacích bodů zvlášť pro každý mnohostěn q_i. V tom případě je kolize testována pouze pro takové dvojice mnohostěnů q_i, r_j kde platí $S_{q_i} \geq \kappa_j$ (κ_j je geometrickou charakteristikou každé části překážky zvlášť, i když většinou stačí volit tento parametr jako globální konstantu). Implementace tohoto algoritmu je ale mnohem náročnější, neboť již není oddělen výpočet testovacích bodů a vyšetření kolize v daném čase. Veličina S_{q_i} je integrována zvlášť pro každý mnohostěn.

Popsaná metoda může být rozšířena i pro situace, kdy překážky jsou pohyblivé. Toto rozšíření je rovněž nezbytné v případě, kdy některé části systému se mohou dostat do vzájemné kolize, což je typická situace např. u průmyslových robotů.

Uvažujme pohyblivý bod $X \in Q$, který protíná model překážky $\tilde{R}$ v intervalu $[t_1, t_2]$. Označme $\Xi(t_1, t_2)$ množinu bodů $Y \in \tilde{R}$ takových, že $X(\tau) = Y(\tau)$ pro některá τ. Ξ je z fyzikálního hlediska stopou bodu X na překážce $\tilde{R}$. Označme $S_\Xi(t_1, t_2)$ délku stopy $\Xi(t_1, t_2)$. Platí:

$$S_\Xi(t_1, t_2) = \int_{t_1}^{t_2} \left\| dX - dY \right\| \leq \int_{t_1}^{t_2} \left(\left\| dX \right\| + \left\| dY \right\| \right) \leq S_Q(t_1, t_2) + S_{\tilde{R}}(t_1, t_2) \ . \tag{5.68}$$

Analogicky s předchozím postupem pak může být bod t_{i+1} určen tak, že $S_\Xi(t_i, t_{i+1}) < \kappa$:

$$t_{i+1} = \sup \left\{ t \mid S_Q(t_i, t) + S_{\tilde{R}}(t_i, t) < \kappa \right\}. \tag{5.69}$$

Na závěr uvažujme případ, kdy je požadováno, aby se systém pohyboval v omezeném konvexním pracovním prostoru ve tvaru $W = \bigcap_{i=1}^{w} W_i^-$. Nechť $\tilde{Q}$ je model systému definovaný stejným způsobem jako $\tilde{R}$ v případě překážky na Obr. 5.4. Definujme

$$\eta = \inf_{X \in \tilde{Q}} \left\{ S_X(t_1,t_2) \mid X(t_1) \in W,\ X(t_2) \in W \text{ a } \exists \tau \in (t_1,t_2) : Q(\tau) \cap W \neq Q(\tau) \right\}. \tag{5.70}$$

Další testovací bod může být určen analogicky s výše popsaným postupem:

$$t_{i+1} = \sup \left\{ t \mid S_X(t_i,t) < \eta, \forall X \in \tilde{Q} \right\}. \tag{5.71}$$

Geometrický význam η je velmi podobný κ. Jediný rozdíl je, že η je definováno pro systém a nikoliv pro překážku jako κ.

Předchozí výsledky lze využít i pro efektivní vyhodnocení pokutového členu kritéria $\int_0^{t_f} \mu(Q,R)\,dt$. V tomto případě je možné vynechat integraci v úseku, kde ke kolizi nedochází. Přesněji řečeno, výpočet integrálu probíhá ve dvou režimech:

- Po nalezení bodu t_i, kde $\mu(Q,R) > 0$ probíhá integrace běžným způsobem, dokud není nalezen bod t_{i+1} takový, že $\mu(Q,R) = 0$.
- Je-li $\mu(Q,R) = 0$ pro $t = t_i$, je další bod t_{i+1} stanoven stejným způsobem jako v případě pouhé detekce kolize, neboť v tomto useku je zaručeno, že $\mu(Q,R) = 0$.

Popsané výsledky, které ukazují souvislost mezi geometrickým tvarem těles, jejich pohybem a krokem vzorkování, jsou do značné míry obecné, i když při praktické implementaci se s výhodou využijí specifické vlastnosti mnohostěnových modelů. Uvažujeme-li vyšetření kolize mnohostěnů podél trajektorie, existuje však další velmi podstatné urychlení, které je z praktického hlediska ještě účinnější.

V případě, že v bodě t_i byla mezi mnohostěny q_i, r_j nalezena oddělující rovina, bude se s velkou pravděpodobností jednat o oddělující rovinu i pro $t = t_{i+1}$. Jelikož algoritmus zjištění kolize mezi mnohostěny, stejně jako výpočtu $\mu(q_i,r_j)$, končí při nalezení oddělující roviny, je vhodné jako první vyhodnotit pozici bodů druhého mnohostěnu vůči oddělujícímu poloprostoru H^+, popř. K^+ z předchozího kroku. Urychlení zrychlení získané touto jednoduchou heuristikou je mnohonásobné.

Oba zmíněné postupy urychlení detekce kolize podél trajektorie, popř. výpočtu její míry, se nevylučují a je možné je implementovat v rámci jednoho algoritmu. Výsledné rutiny jsou však již značně komplikované. Je třeba vykonat následující kroky zhruba v tomto pořadí:

- zjistit případnou kolizi opsaných koulí, popř. válců
- vypočítat pozice rohových bodů a určit poloprostory
- jestliže $t = t_i$, určit t_{i+1}
- vyšetřit kolizi. Jako první testovat oddělující poloprostory z předchozího kroku.

5.9. Praktické příklady

Následující příklady demonstrují výpočet optimálních trajektorií v prostoru s překážkami. Ve všech případech bylo využito nahrazení trajektorie v bázovém systému $\{\Lambda_i^{n+1}\}_{-n}^{N}$, kde $n=2$, $N=10$. Překážky byly zahrnuty do kritéria pokutovou metodou s hladkou mírou kolize μ_3. Pro optimalizaci byla použita druhá z variant diferenciální evoluce popsaných v části 6, kap. 6.5. Na počítači PC 1.2 GHz byly u všech úloh výsledky získány během několika minut.

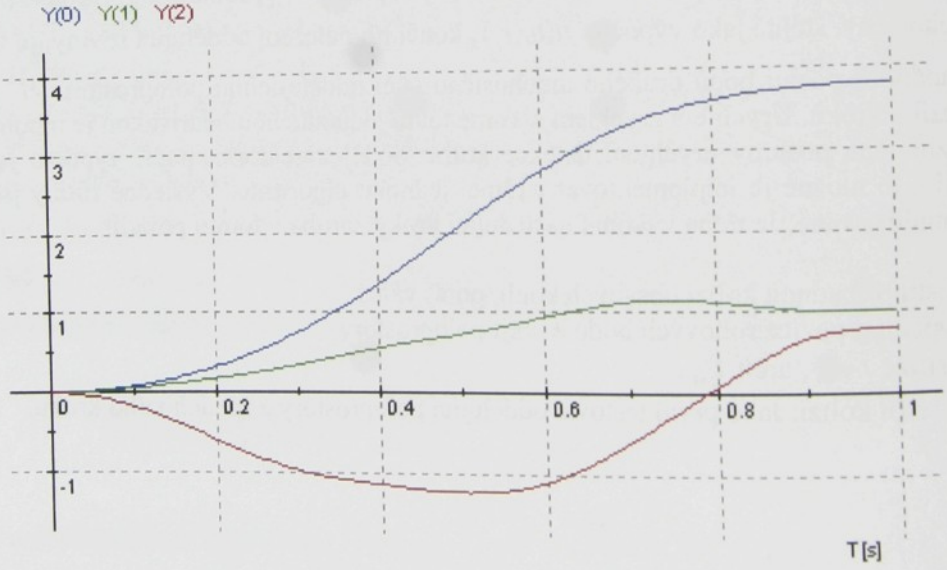
1. Úloha

Model č. 2, kap. 7.2, robota angulárního typu. V pracovním prostoru jsou umístěny překážky. Okrajové podmínky polohy jsou

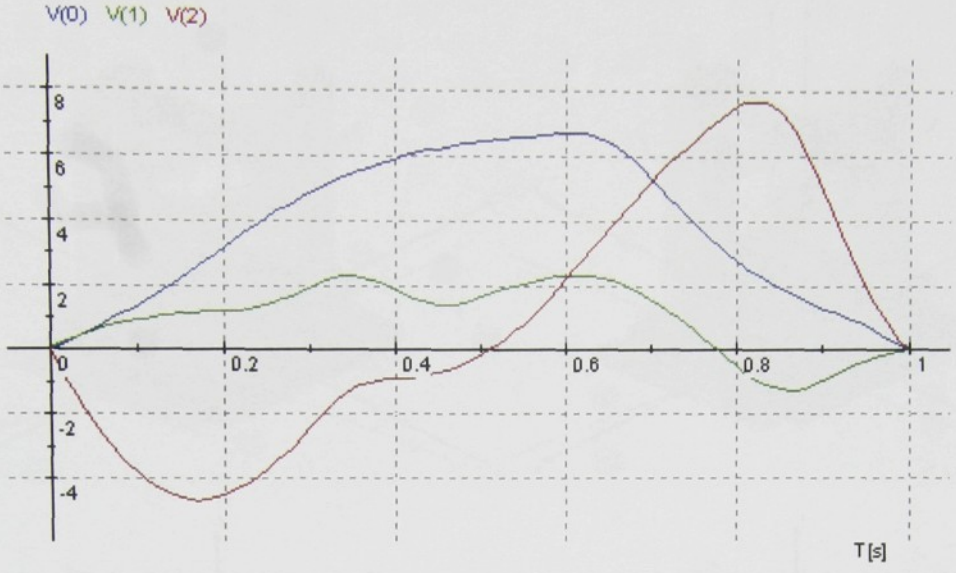
$$\mathbf{y}_0 = (0,0,0)^T \qquad (5.72)$$
$$\mathbf{y}_f = (4,1,1)^T$$

kde $\mathbf{y} = (\varphi, \vartheta, \psi)^T$. Okrajové podmínky rychlosti jsou nulové.

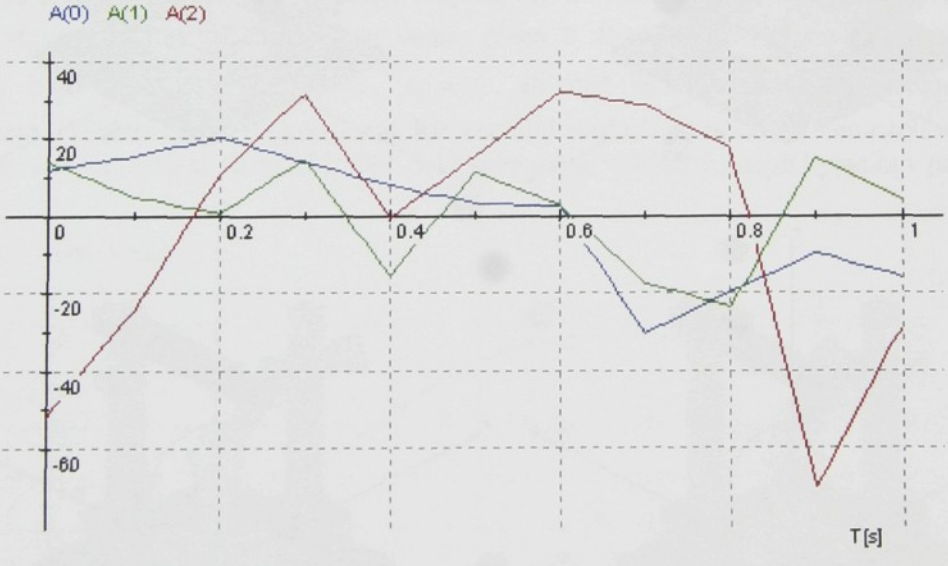
Průběhy kinematických veličin členů jsou na Obr. 5.5 - Obr. 5.7. Na Obr. 5.8 je znázorněn pohyb robota v prostoru. Pro získání smysluplných výsledků bylo třeba uvažovat vzájemnou kolizi členů robota vůči sobě, jak demonstruje Obr. 5.9.



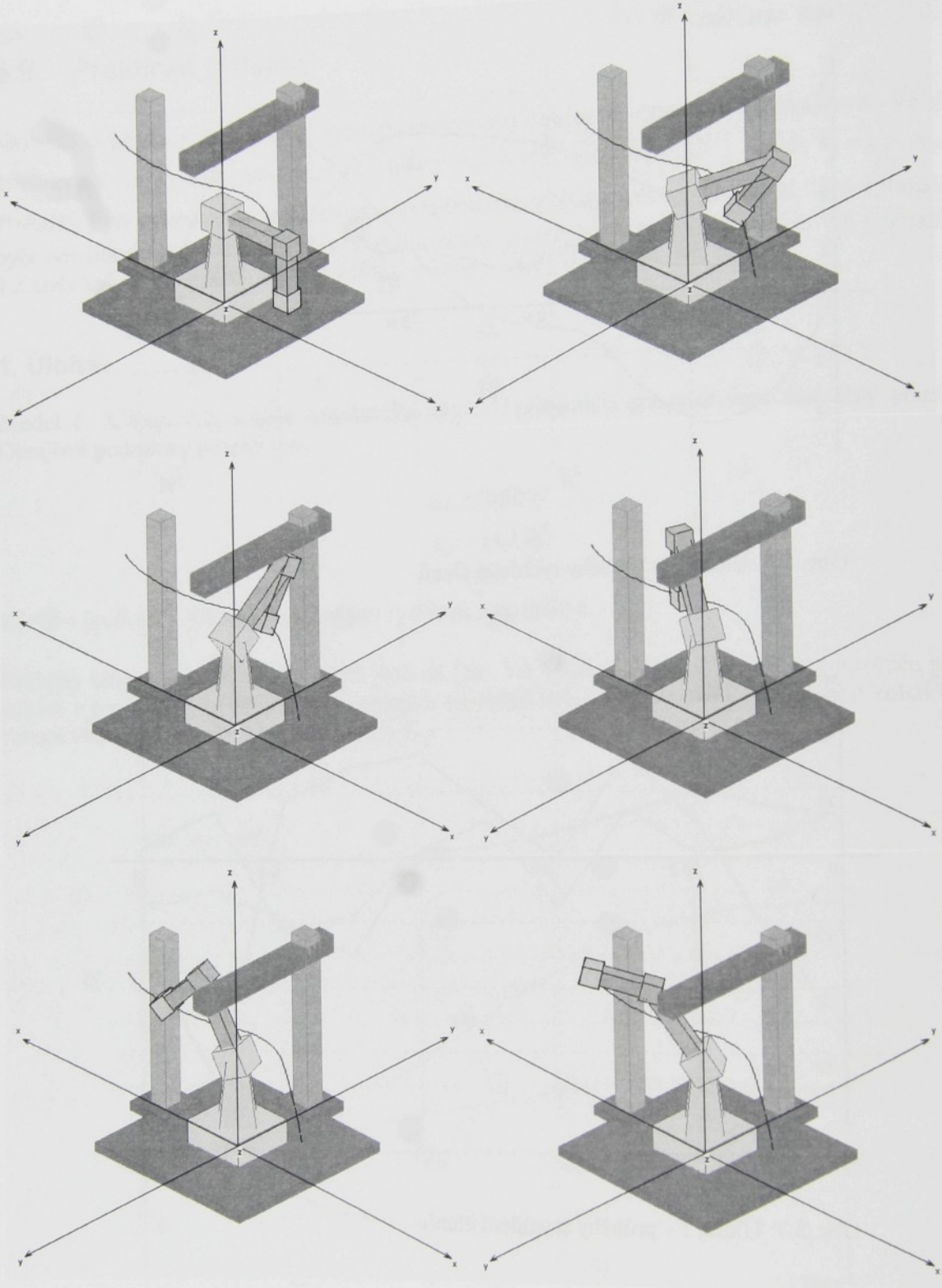
Obr. 5.5: Úloha 1 – průběhy polohy členů



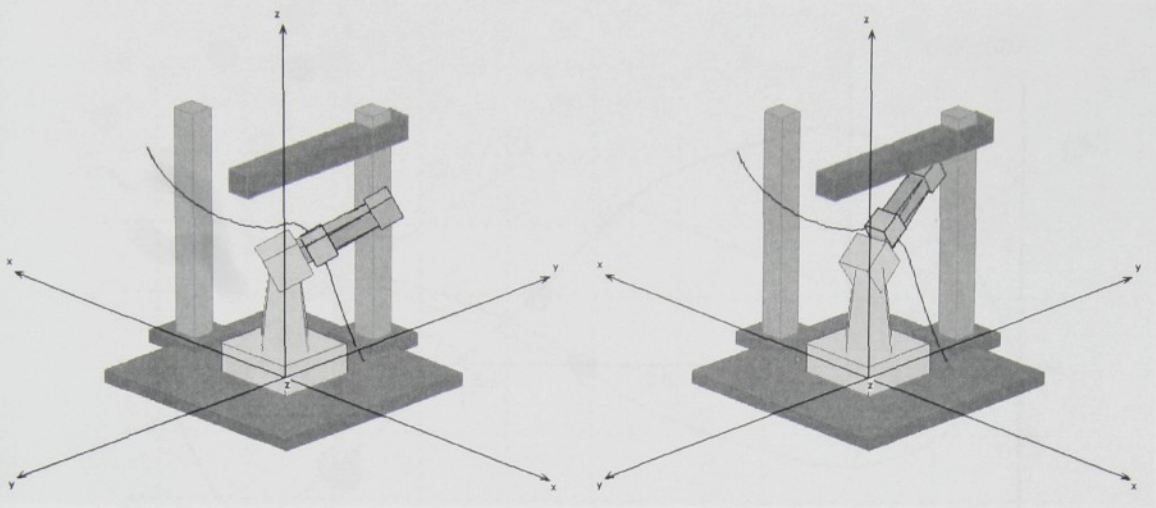
Obr. 5.6: Úloha 1 – průběhy rychlosti členů



Obr. 5.7: Úloha 1 – průběhy zrychlení členů



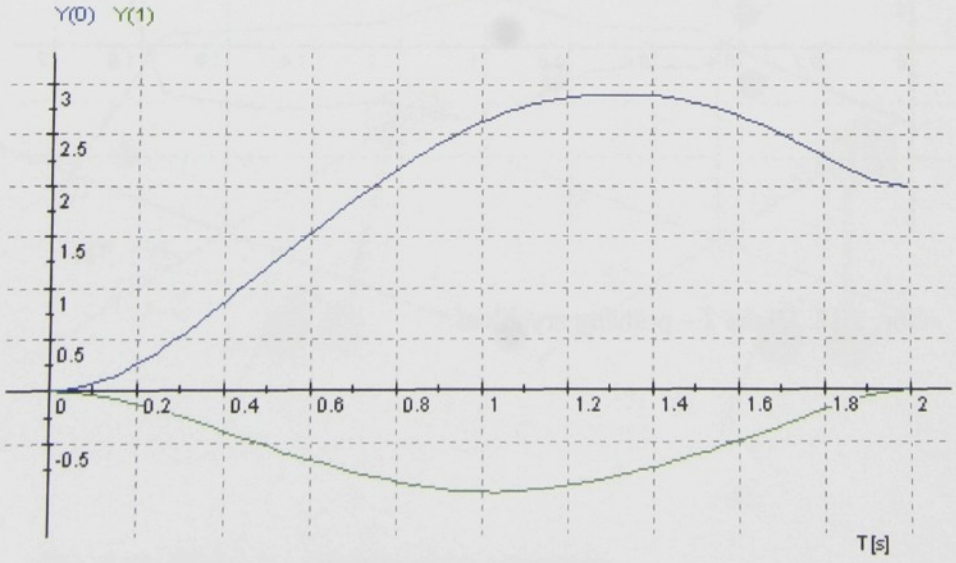
Obr. 5.8: Úloha 1 – výsledný pohyb v prostoru



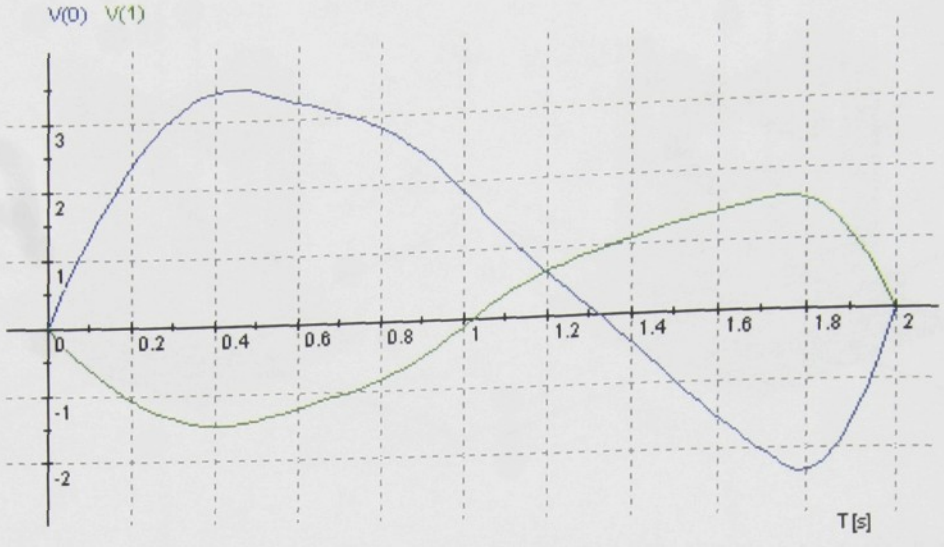
Obr. 5.9: Úloha 1 – výsledný pohyb v prostoru bez uvažování kolize částí robota vůči sobě

2. Úloha

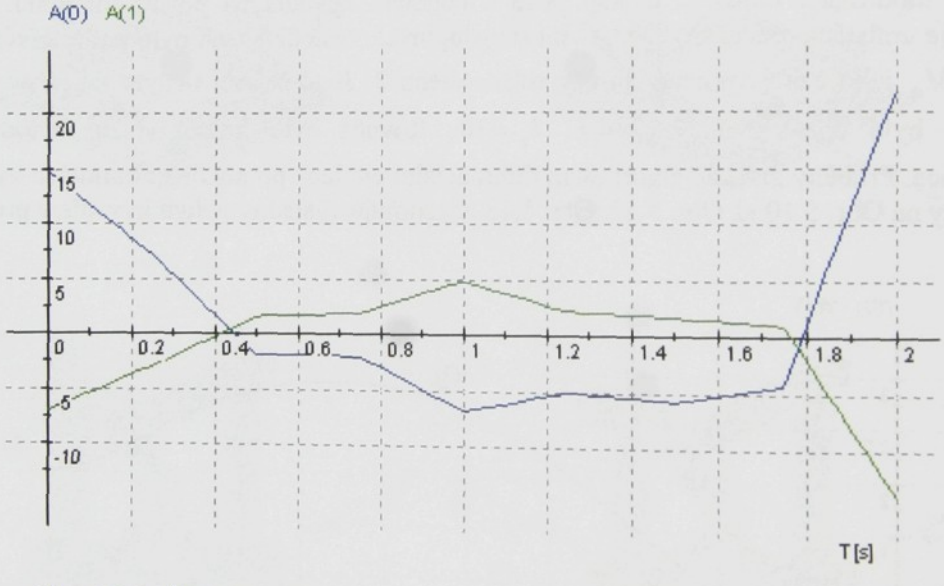
Zadání je modifikací úlohy č. 6, kap. 3.13 s modelem kyvadla na posuvném rámu. Na pozici $(1,0,-1)$ je umístěna překážka. Pro získání smysluplných výsledků však bylo nutné zavést skutečný moment M_φ jako řídicí veličinu do osy rotace členu **2**. Jinak řešení nebylo nalezeno. V kritériu optimality bylo proto zvoleno $k_x = 1$, $k_\varphi = 10$. Rovněž bylo nutné použít metodu globální optimalizace. Průběhy získané algoritmem diferenciální evoluce po několika minutách výpočtu jsou znázorněny na Obr. 5.10 až Obr. 5.12. Obr. 5.13 znázorňuje výsledný pohyb kyvadla v prostoru.



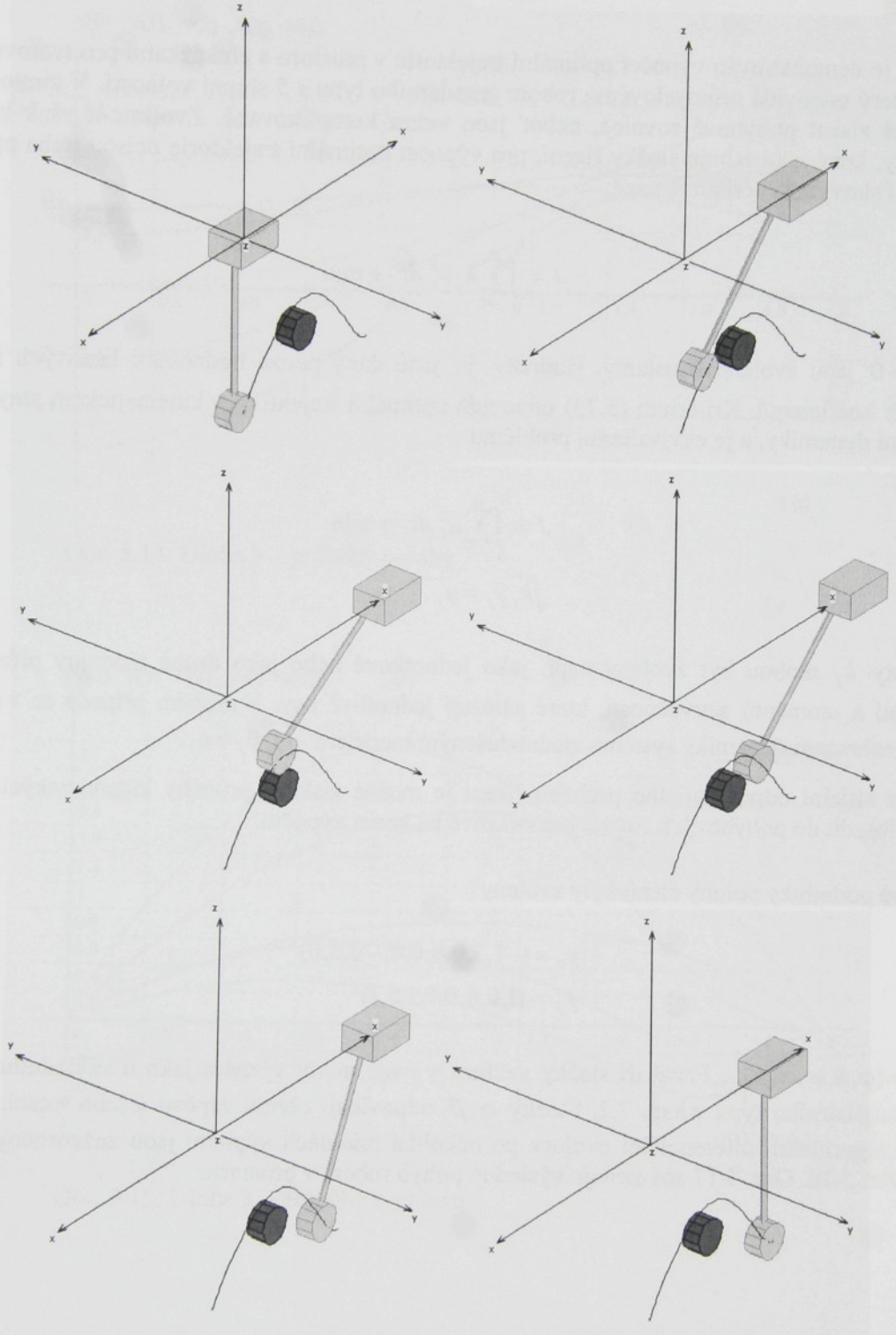
Obr. 5.10: Úloha 2 – průběhy polohy



Obr. 5.11: Úloha 2 – průběhy rychlosti



Obr. 5.12: Úloha 2 – průběhy zrychlení



Obr. 5.13: Úloha 2 – pohyb kyvadla v prostoru

3. Úloha

Na závěr je demonstrován výpočet optimální trajektorie v prostoru s překážkami pro tvarově složitý model, který odpovídá průmyslovému robotu angulárního typu s 5 stupni volnosti. V tomto případě je obtížné získat pohybové rovnice, neboť jsou velmi komplikované. Zvolíme-li však kritérium optimality, které neobsahuje složky řízení, pro výpočet optimální trajektorie nejsou třeba pohybové rovnice. Takovým kritériem je např.

$$J = \int_0^{t_f} \sum_{j=1}^{m} k_j \ddot{y}_j^2 \, dt \to \min \tag{5.73}$$

kde $k_j > 0$ jsou zvolené konstanty. Hodnoty $\ddot{y}_j$ jsou dány pouze hodnotami bázových funkcí a bázových koeficientů. Kritérium (5.73) odpovídá optimální trajektorii v kinematickém smyslu, bez uvažování dynamiky, a je ekvivalentní problému

$$\begin{aligned} J &= \int_0^{t_f} \sum_{j=1}^{m} u_j^2 \, dt \to \min \\ \sqrt{k_j}\,\ddot{y}_j &= u_j \,. \end{aligned} \tag{5.74}$$

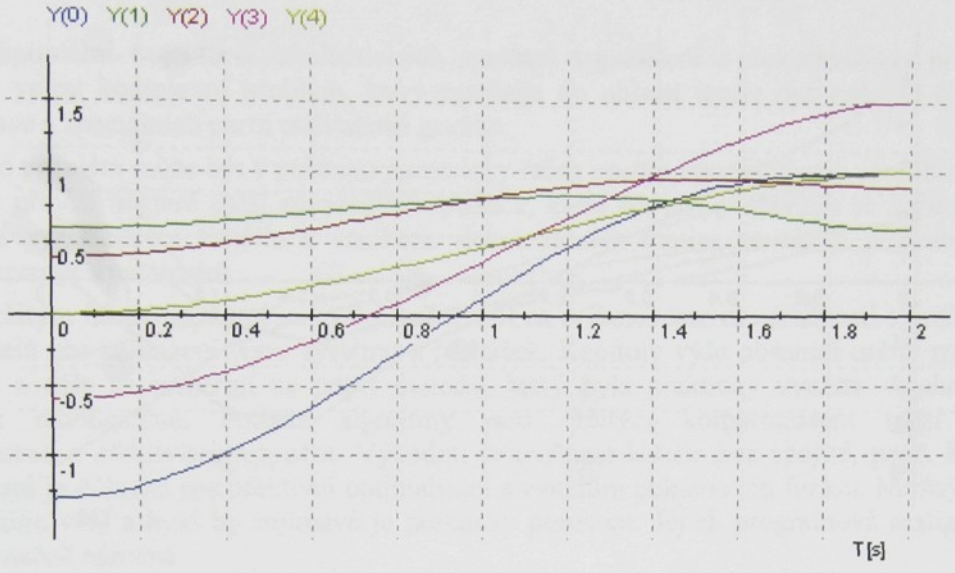
Konstanty k_j mohou být zvoleny např. jako jednotkové nebo jako druhé mocniny průměrných hmotností a momentů setrvačnosti, které zatěžují jednotlivé osy. V druhém případě se v podstatě jedná o nahrazení dynamiky systému zjednodušeným modelem $\sqrt{k_j}\,\ddot{y}_j = u_j$.

Pro získání odpovídajícího průběhu řízení je možné získané průběhy kinematických veličin $\ddot{\mathbf{y}}, \dot{\mathbf{y}}, \mathbf{y}$ dosadit do pohybových rovnic jednorázově na konci výpočtu.

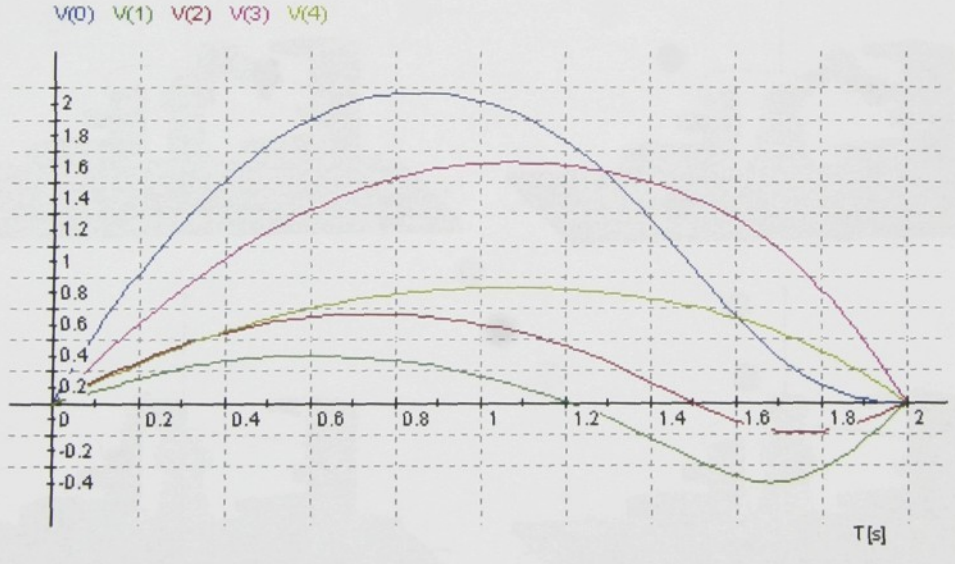
Okrajové podmínky polohy členů byly zvoleny

$$\begin{aligned} \mathbf{y}_0 &= (-1.3, 0.6, 0.4, -0.6, 0)^T \\ \mathbf{y}_f &= (1, 0.6, 0.9, 1.5, 1)^T \end{aligned} \tag{5.75}$$

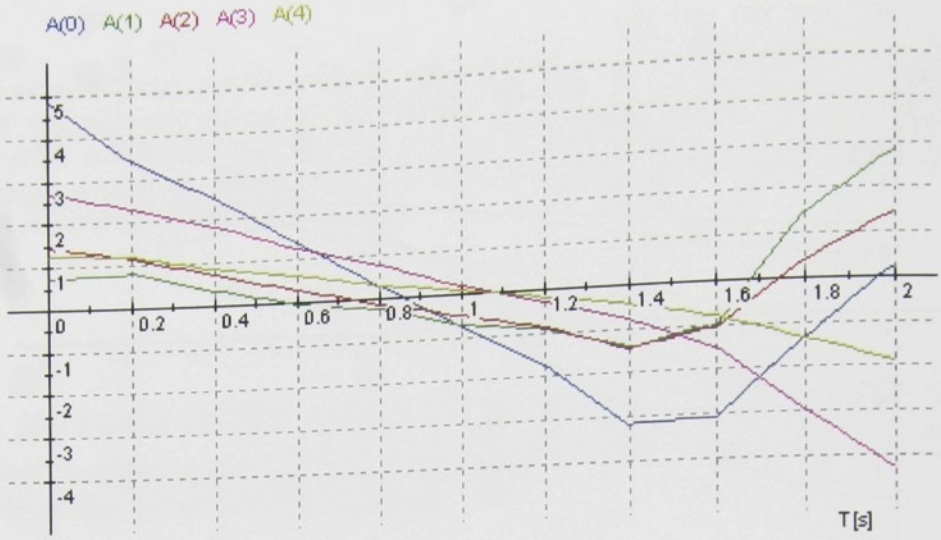
kde $\mathbf{y} = (\varphi, \vartheta, \psi, \alpha, \beta)^T$. První tři složky vektoru $\mathbf{y}$ mají stejný význam jako u základního modelu robota angulárního typu v kap. 7.2. Složky α, β odpovídají ohnutí zápěstí a jeho rotaci. Průběhy získané algoritmem diferenciální evoluce po několika minutách výpočtu jsou znázorněny na Obr. 5.14 - Obr. 5.16. Obr. 5.17 znázorňuje výsledný pohyb robota v prostoru.



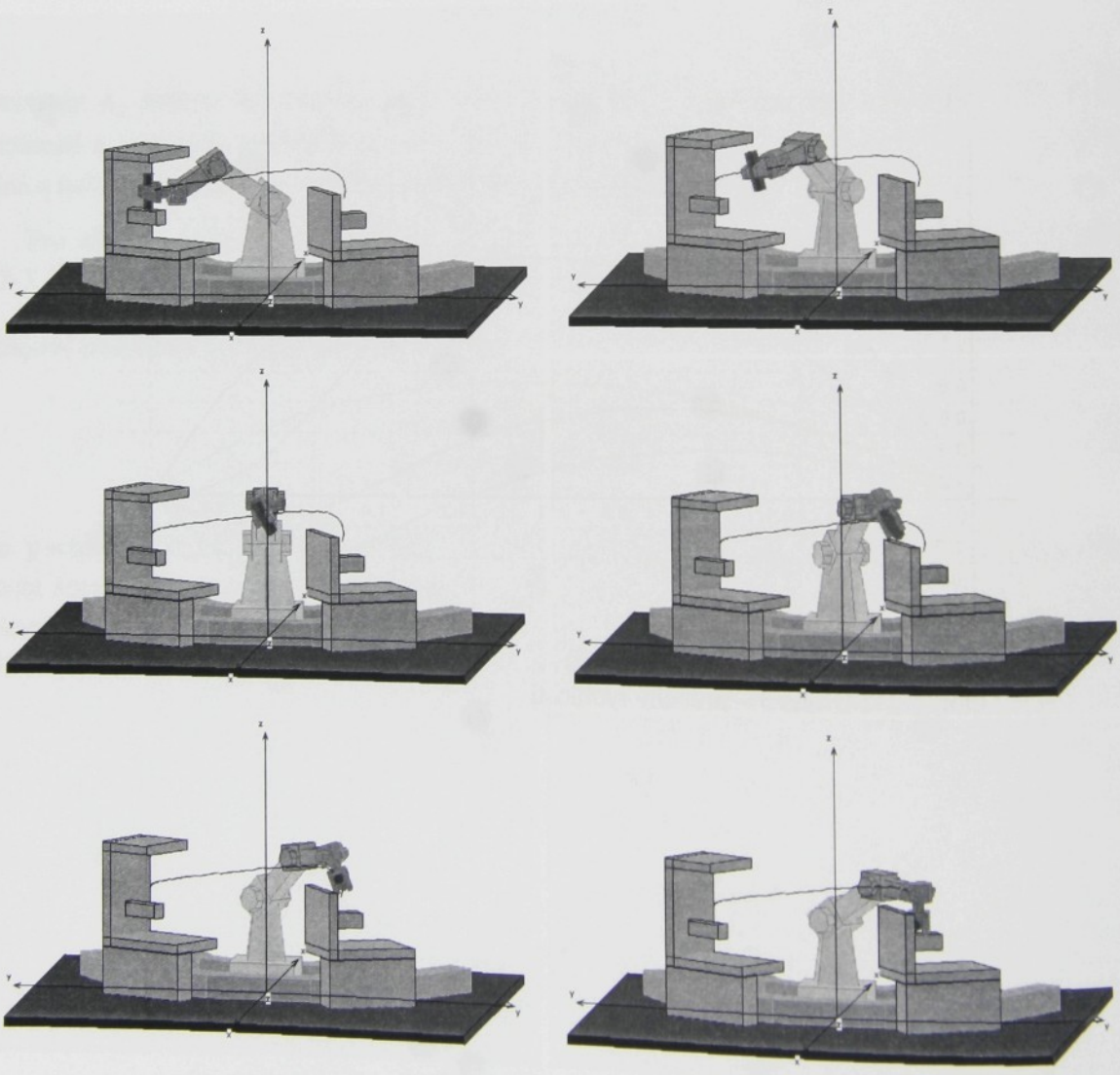
Obr. 5.14: Úloha 3 – průběhy polohy



Obr. 5.15: Úloha 3 – průběhy rychlosti



Obr. 5.16: Úloha 3 – průběhy zrychlení



Obr. 5.17: Úloha 3 – pohyb robota v prostoru

5.10. Závěr

Výpočet optimální trajektorie mechanických systémů v prostředí s překážkami je při uvažování dynamiky velmi komplexní problém, který zasahuje do oblasti teorie optimálního řízení, metod optimalizace a speciálních partií počítačové grafiky.

Tento problém může být v principu numericky řešen postupy popsanými v předchozích částech práce, ale přináší některé další závažné komplikace, které souvisí především se značnou časovou náročností vyhodnocení kritéria a vznikem více lokálních minim na okraji přípustné množiny optimalizovaných parametrů.

Důležitým faktorem, který se významně podílí na celkové časové náročnosti výpočtu, je volba typu modelů pro nahrazení tvaru systému a překážek. Kapitoly výše obsahují určitý rozbor tohoto problému a dále se zaměřují na popis metody, který byla prakticky využita - nahrazení tvaru soustavou mnohostěnů. Použité algoritmy jsou určitým kompromisem mezi náročností implementace a efektivitou výpočtu. Výhodou je možnost konstrukce spojité, popř. hladké míry kolize, která je důležitá pro efektivní optimalizaci s využitím pokutových funkcí. Možných přístupů však existuje více a bylo by zajímavé je prakticky porovnat. Jejich programová realizace je však bohužel značně náročná.

Řešené úlohy naznačují, že popsaná metoda je funkční. Doba výpočtu a schopnost nalezení kvalitního řešení je však závislá na použitém algoritmu globální optimalizace. Kvalitní metody řeší složitější úlohy zpravidla v řádu minut. Následující část práce obsahuje shrnutí zkušeností získaných s těmito algoritmy.

Část 6.

Algoritmy optimalizace

V předchozích částech práce byly popsány možné způsoby transformace problému návrhu optimální trajektorie na úlohu hledání minima reálné funkce konečného počtu parametrů.

Jádrem výpočtu optimální trajektorie je však numerický algoritmus optimalizace. Kvalita algoritmu optimalizace je faktorem, který se významně podílí na celkové době výpočtu. Za měřítko kvality algoritmu je považován potřebný počet vyhodnocení kritéria pro dosažení potřebné přesnosti. Jelikož výpočet hodnoty kritéria je časově náročnou operací, je možné zpravidla zanedbat přídavnou režii, která souvisí s manipulací s body a vypočítanými hodnotami kritéria.

Výhodou přístupu diskutovaného v této práci je možnost využití univerzálních algoritmů, které mají solidní teoretický základ a jejichž vlastnosti jsou dobře známy. Jinou předností je rozdělení celého problému na dvě části, které mohou být řešeny a implementovány do značné míry nezávisle.

Není záměrem autora v této části vytvořit přehled obsahující detailní popis a rozbor jednotlivých metod, který lze najít v literatuře, ale spíše provést určité shrnutí zkušeností s algoritmy, které byly testovány pro řešení popsaného problému. Cílem bylo dosáhnout spolehlivé a současně co nejrychlejší konvergence k řešení. Byly testovány postupy doporučené v literatuře a v případě metod globální optimalizace i vlastní modifikace některých algoritmů.

Použité algoritmy optimalizace lze rozdělit podle několika kritérií. Z hlediska typu řešeného problému se jedná o dvě hlavní skupiny:

1. algoritmy efektivní pro řešení úloh bez překážek
2. algoritmy použitelné pro výpočet trajektorií v prostoru s překážkami

Algoritmy z první skupiny jsou popsány v následujících dvou kapitolách. Dosažené výsledky v této oblasti lze považovat do značné míry za uspokojivé. Zejména v případě nahrazení trajektorie je kvalitní řešení zpravidla získáno během několika sekund i pro poměrně vysoký počet optimalizovaných parametrů. V případě bázových systémů typu spline lze kvalitu získaného řešení snadno posoudit podle průběhu n-té derivace. Jestliže se tato po částech lineární funkce blíží hladké funkci, jedná se velmi pravděpodobně o optimální řešení (alespoň v lokálním smyslu).

Postupy patřící do druhé skupiny jsou diskutovány zejména v následujících kapitolách Heuristické algoritmy a Hybridní metody. Výsledky v této oblasti však dosud zdaleka nejsou uzavřené. V případě výpočtu optimálních trajektorií v prostoru s překážkami pro modely připomínající reálné situace se délka výpočtu při použití nejefektivnějších metod pohybuje v řádu několika minut. Významným úspěchem však je, že tyto metody vůbec byly objeveny, neboť mnoho algoritmů globální optimalizace popisovaných v literatuře pro tuto třídu problémů zcela selhává. Na výrazném prodloužení doby výpočtu se bohužel nepodílí pouze složitost minimalizované funkce, nýbrž také mnohonásobné zvýšení režie pro určení hodnoty kritéria, které v sobě zahrnuje informaci o kolizi modelů systému a překážek podél trajektorie.

Pro srovnání účinnosti různých algoritmů bylo třeba porovnat průběh konvergence k řešení pro

- různé typy problémů
- různé počáteční odhady
- různé parametry algoritmu

Pro získání těchto dat byla nutná poměrně rozsáhlá experimentální práce. Detailní prezentace výsledků by byla námětem pro samostatnou studii značného rozsahu. Dále jsou popsány pouze nejdůležitější empirické poznatky.

6.1. Algoritmy pro řešení úloh bez překážek

V předchozích částech bylo ukázáno, že nejsou-li přítomna omezení typu rovností a nerovností v každém bodě, lze uvažovanou třídu problémů transformovat do tvaru

$$J(\mathbf{z}) \to \min \tag{6.1}$$

$$\mathbf{g}(\mathbf{z}) = \mathbf{0} \tag{6.2}$$

kde funkce J a $\mathbf{g}$ jsou spojitě diferencovatelné podle složek vektoru parametrů $\mathbf{z}$ dimenze n_z. Z hlediska algoritmu optimalizace není podstatné, jestli optimalizované koeficienty jsou ve skutečnosti organizovány v nějaké jiné datové struktuře – jako např. v matici. Předpokládáme, že algoritmus je vybaven rozhraním, které zajistí transformaci těchto struktur do vektoru a zpět.

U metody nahrazení trajektorie, kdy funkce $\mathbf{f}$ je ve speciálním tvaru, část (6.2) odpadá a lze přímo použít algoritmy pro řešení hladkých úloh bez omezení.

Tyto metody lze dále rozdělit do dvou skupin:

1. algoritmy využívající derivací účelové funkce
2. algoritmy bez výpočtu derivací

K hlavním specifikům problému patří:

- Výpočet kritéria je časově náročnou operací.
- Počet parametrů je zhruba v rozsahu $10 \le n_z \le 100$, tedy poměrně velký, avšak ne tolik, aby bylo nutné uvažovat režii operace násobení matice velikosti (n_z, n_z) a vektoru (pro podstatně vyšší n_z však tato režie může být podstatná, neboť je úměrná $(n_z)^2$). Stejně tak není problém s uložením matic této velikosti v paměti.
- Parciální derivace kritéria nejsou explicitně známé a je nutné je určovat numericky.
- Existují postupy pro urychlení numerického výpočtu gradientu, přesto je však výpočet gradientu podstatně náročnější operací než výpočet hodnoty kritéria.

Z prvních dvou bodů vyplývá nutnost zvolit sofistikovanější algoritmy, které jsou efektivní i pro poměrně vysoké hodnoty n_z. Z druhého a třetího bodu pak vyplývá, že druhé parciální derivace nejsou k dispozici a jejich numerický výpočet je nepřesný a značně časově náročný. Je tedy nutné omezit se na metody využívající nanejvýše první parciální derivace.

Prakticky bylo potvrzeno, že kvalitní algoritmy využívající derivací účelové funkce jsou pro řešení daného problému zpravidla efektivnější než metody z druhé skupiny i v případě, že gradient je určován numericky a nejsou použity kroky pro urychlení jeho výpočtu. V tom případě může být jeden výpočet gradientu např. 50-krát časově náročnější než výpočet hodnoty kritéria. Získané závěry jsou však potvrzeny i v literatuře [11].

Uvedeným požadavkům vyhovují především dvě skupiny algoritmů [11]:

- metody konjugovaných gradientů
- quasi-Newtonovy metody

Obě skupiny algoritmů jsou založeny na kvadratickém modelu funkce v okolí minima a lze je považovat za zdokonalení elementární metody největšího spádu:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha^* \mathbf{s}_k \tag{6.3}$$

$$\mathbf{s}_k = -\nabla J \tag{6.4}$$

kde ∇J je gradient kritéria a $\alpha^* > 0$ je parametr nalezený jednorozměrným hledáním minima ve směru záporného gradientu

$$\alpha^* = \arg\min_{\alpha} \{J(\mathbf{z}_k + \alpha \mathbf{s}_k)\} . \tag{6.5}$$

V prvním případě je využito skutečnosti, že jestliže směry hledání jsou konjugované s maticí $\mathbf{Q}$ druhých derivací kvadratické funkce $J(\mathbf{z}) = J_0 + \mathbf{b}^T\mathbf{z} + \mathbf{z}^T\mathbf{Q}\mathbf{z}$, je minima této funkce dosaženo nejvýše v n_z krocích hledání. Konjugovanými směry se rozumí takové, že

$$\begin{aligned} \mathbf{s}_i^T \mathbf{Q} \mathbf{s}_j &= 0 \quad \text{pro} \quad i \neq j \\ \mathbf{s}_i^T \mathbf{Q} \mathbf{s}_j &\neq 0 \quad \text{pro} \quad i = j . \end{aligned} \tag{6.6}$$

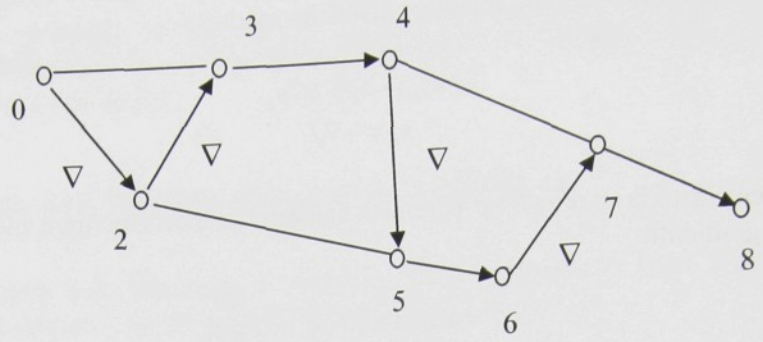
Pro nekvadratické funkce je pak dosaženo přiblížení k minimu, které je tím přesnější, čím blíže je výchozí odhad bodu minima. Konjugované směry mohou být určeny mnoha způsoby, bez explicitní znalosti matice $\mathbf{Q}$. Např. u metody Fletchera a Reevese je směr hledání určen pomocí gradientu a směru v předchozím kroku:

$$\begin{aligned} \mathbf{s}_k &= -\nabla_k + \frac{\nabla_k^T \nabla_k}{\nabla_{k-1}^T \nabla_{k-1}} \mathbf{s}_{k-1} \\ \mathbf{s}_0 &= -\nabla_0 . \end{aligned} \tag{6.7}$$

Po n_z krocích je třeba metodu restartovat. V literatuře je spíše doporučována varianta Polak-Ribière

$$\mathbf{s}_k = -\nabla_k + \frac{(\nabla_k^T - \nabla_{k-1}^T)\nabla_k}{\nabla_{k-1}^T \nabla_{k-1}} \mathbf{s}_{k-1} . \tag{6.8}$$

Bylo testovány obě varianty s nepříliš dobrými výsledky, snad z důvodu nepřesného výpočtu gradientu. Mnohem lepších výsledků bylo dosaženo metodou PARTAN, která je motivována kompenzací střídání kolmých směrů v metodě nejmenšího spádu. Algoritmus je znázorněn na Obr. 6.1, kde šipky odpovídají směrům hledání a znakem ∇ jsou označeny gradientní směry. U této metody je však teoreticky nižší účinnost, neboť konjugované jsou pouze směry $z_2 - z_0$, $z_4 - z_2$, $z_6 - z_4$, atd. [13].



Obr. 6.1: Kroky metody PARTAN

Tento algoritmus se velmi osvědčil, třebaže v současné literatuře je většinou opomíjen nebo je zmiňován pouze okrajově. Později byl překonán metodou BFGS, jejíž implementace je pro $n_z \approx 20-30$ ve většině případů několikrát rychlejší.

Jako kritérium ukončení výpočtu je většinou doporučována podmínka

$$\|\nabla_k\| < \varepsilon \tag{6.9}$$

kde ε je zvolené malé kladné číslo odpovídající požadované přesnosti výpočtu.

Podstatnou součástí zmíněných metod je algoritmus hledání minima v daném směru. Při praktické implementaci je hledání provedeno pouze s omezenou přesností. Při větší vzdálenosti od minima je většinou dostačující nižší přesnost hledání. Vhodná volba ukončujících podmínek zaručuje u některých algoritmů teoreticky konvergenci k minimu i pro poměrně nepřesné určení lokálního minima α v každém kroku. Jednou z možností jsou Goldsteinovy podmínky:

$$J(\alpha) \le J(0) + \alpha\rho J'(0) \tag{6.10}$$

$$J(\alpha) \ge J(0) + \alpha(1-\rho)J'(0) \tag{6.11}$$

kde $\rho \in (0,0.5)$ je pevný parametr. Vzhledem k tomu, že hledání je prováděno v pevném směru $\alpha \mathbf{s}_k$, je kritérium závislé pouze na parametru α. J' je derivace ve směru $\mathbf{s}_k$

$$J'(0) = \nabla_k^T \mathbf{s}_k . \tag{6.12}$$

Místo podmínky (6.11) je v knize [11] doporučován Wolfeho test

$$J'(\alpha) \ge \sigma J'(0), \quad \sigma \in (\rho,1) . \tag{6.13}$$

Nevýhodou vztahu (6.13) však je, že vyžaduje výpočet gradientu v každém bodě, a proto pro uvažovanou třídu problémů tato ukončovací podmínka není vhodná.

Pro nalezení minima v daném směru bylo prakticky využito algoritmu zlatého řezu [5], který byl rozšířen i pro dopředné hledání. Po upřesnění výsledku získaného po omezeném počtu kroků byla provedena interpolace kubického polynomu body v nejbližším okolí extrému.

Quasi-Newtonovy metody jsou založeny na postupné aproximaci matice druhých derivací, popř. přímo její inverze:

$$\mathbf{s}_k = -\mathbf{H}_k \nabla_k . \tag{6.14}$$

Výhodou těchto metod je, že nejsou založeny na přesném určení minima ve směru $\mathbf{s}_k$ a nevyžadují restartování. Matice $\mathbf{H}_0$ je většinou zvolena jako jednotková. Pro aktualizaci matice $\mathbf{H}_k$ existuje řada vztahů. K nejúspěšnějším patří:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\boldsymbol{\delta}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}} - \frac{\mathbf{H}_k\boldsymbol{\gamma}\boldsymbol{\gamma}^T\mathbf{H}_k}{\boldsymbol{\gamma}^T\mathbf{H}_k\boldsymbol{\gamma}} \tag{6.15}$$

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left(1 + \frac{\boldsymbol{\gamma}^T\mathbf{H}_k\boldsymbol{\gamma}}{\boldsymbol{\delta}^T\boldsymbol{\gamma}}\right)\frac{\boldsymbol{\delta}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}} - \frac{\boldsymbol{\delta}\boldsymbol{\gamma}^T\mathbf{H}_k + \mathbf{H}_k\boldsymbol{\gamma}\boldsymbol{\delta}^T}{\boldsymbol{\delta}^T\boldsymbol{\gamma}} \tag{6.16}$$

kde $\boldsymbol{\delta} = \mathbf{z}_{k+1} - \mathbf{z}_k$, $\boldsymbol{\gamma} = \nabla_{k+1} - \nabla_k$.

Metoda (6.15) je označována jako DFP (Davidon, Fletcher, Powel) a (6.16) jako BFGS (Broyden, Fletcher, Goldfarb, Shanno). V současnosti je na základě praktických zkušeností i řady teoretických výsledků doporučována zejména metoda BFGS, která je považována za jeden z nejlepších algoritmů pro nižší počet parametrů. Tomu odpovídají i autorovy praktické zkušenosti. Metoda BFGS se ukázala z testovaných algoritmů jako nejefektivnější (viz též příklady 1-3, kap. 3.13). Na druhé straně je však implementace tohoto algoritmu komplikovanější než např. u metod konjugovaných gradientů a paměťové nároky s vnitřní režií rostou s n_z^2.

6.2. Omezující podmínky

Omezení typu rovností (6.2) je přítomné v případě nelineárních koncových podmínek. Základním přístupem je rozšíření kritéria o hladký pokutový člen. Většinou je vyhovující kvadratická pokutová funkce:

$$J'(\mathbf{z}) = J(\mathbf{z}) + \sigma \|\mathbf{g}(\mathbf{z})\|_{\mathbf{K}}^2 \tag{6.17}$$

kde $\|\mathbf{g}\|_{\mathbf{K}}^2 = \mathbf{g}^T\mathbf{K}\mathbf{g}$, $\mathbf{K}$ je diagonální matice s kladnými členy a $\sigma > 0$ pokutová konstanta.

Je-li dána rostoucí posloupnost $\{\sigma_i\}_1^\infty$ pokutových konstant v (6.17) a $\{\mathbf{z}_i^*\}_1^\infty$ je odpovídající řešení hladkého problému bez omezení (6.17), lze dokázat [5], že posloupnost $\{\mathbf{z}_i^*\}_1^\infty$ konverguje k řešení původního problému s omezeními.

Třebaže v praxi je často proveden jediný optimalizační krok s dostatečně vysokou konstantou σ, sekvenční výpočet je často účinnější a je nezbytný v případě, že je požadováno splnění omezujících podmínek s vysokou přesností. Řešení z předchozího kroku je využito jako počáteční odhad pro nový výpočet. Důvodem tohoto postupu je deformace prostoru parametrů způsobená pokutovým členem. Pro vysoké hodnoty σ, které odpovídají vyšší požadované přesnosti splnění omezujících podmínek, tvoří rozšířená účelová funkce velmi úzké zakřivené údolí, které neodpovídá kvadratickému modelu. Zmíněné algoritmy optimalizace pak konvergují k minimu velmi pomalu. V tomto případě se rovněž velmi negativně uplatňují numerické chyby způsobené výpočtem gradientu.

Pokutová metoda lze využít i v případě omezení typu rovností a nerovností v každém čase, jak bylo popsáno v předchozích částech.

Praktické zkušenosti jsou poměrně dobré v případě, že σ není příliš vysoké. Přesto se jedná o elementární přístup, který z hlediska efektivity zdaleka není vyhovující. V daném případě je pravděpodobně dostačující provést výpočet ve dvou až třech krocích. Počáteční kroky představují určení počátečního odhadu pro vlastní výpočet a lze je provést s nižší přesností. V následujících krocích se σ násobí např. koeficientem 10 a faktor přesnosti ε se stejným koeficientem dělí.

Byla rovněž testována možnost použít v počátečních krocích nižší počet bázových prvků. V tom případě je však třeba provést přepočet bázových koeficientů – např. u bázových systémů typu spline je možné provést snadno transformaci koeficientů $N \to 2N$ pouze při nahrazení funkce řízení. Tento postup se však nakonec ukázal jako nepřínosný, protože i při stejném N je výpočet pro výrazně nižší σ zpravidla mnohem rychlejší.

V případě omezení typu nerovností lze rovněž použít bariérové členy, které na okraji přípustné množiny nabývají nekonečně velkých hodnot. Uvažujme omezení typu

$$\mathbf{g}(\mathbf{z}) \geq \mathbf{0}. \tag{6.18}$$

U daného problému se sice omezení typu (6.18) přímo nevyskytují, ale následující úvahy jsou využitelné analogicky u případu omezení typu nerovností v každém čase. Omezení (6.18) může být zahrnuto rozšířením kritéria o bariérový člen

$$J'(\mathbf{z}) = J(\mathbf{z}) - \frac{1}{\sigma}\sum_i k_i \log g_i(\mathbf{z}). \tag{6.19}$$

Výhodou bariérové metody je, že posloupnost bodů $\mathbf{z}_k$ získaná jednotlivými iteracemi leží v přípustné množině. Rozšířené kritérium (6.19), na rozdíl od pokutové metody, obsahuje určitou informaci o vzdálenosti od omezení i uvnitř přípustné množiny, a proto algoritmy optimalizace většinou rychleji konvergují k řešení. Na druhé straně, výpočet vyžaduje na počátku nalezení přípustného odhadu a algoritmus optimalizace musí být upraven tak, aby generoval vždy pouze body uvnitř přípustné množiny.

Efektivnější, třebaže většinou méně robustní a mnohem náročnější z hlediska implementace, jsou přímé metody. Tyto postupy však dosud nebyly prakticky realizovány, neboť vyžadují zásadní zásah do struktury existujícího software. K nejznámějším metodám řešení úloh s omezeními typu rovností (6.1), (6.2) patří projekční metody a sekvenční kvadratické programování (SQP) [11].

V případě projekčních metod je směr hledání v bodě $\mathbf{z}_k$ takovém, že $\mathbf{g}(\mathbf{z}_k) = \mathbf{0}$, odvozen od kroku gradientních, popř. Newtonových metod projekcí na plochu $\mathbf{g}(\mathbf{z}) = \mathbf{0}$. Projekce však vyžaduje vnitřní iterační smyčku.

Algoritmus SQP je možné odvodit z podmínek optimality. V bodě lokálního minima platí

$$\nabla \mathcal{L}(\mathbf{z}^*, \boldsymbol{\lambda}) = \mathbf{0} \tag{6.20}$$

kde $\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda})$ je Lagrangeova funkce

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}) = J(\mathbf{z}) - \sum_i \lambda_i g_i(\mathbf{z}) = J(\mathbf{z}) - \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{z}). \tag{6.21}$$

Na základě postačujících podmínek druhého řádu je $\nabla^2 \mathcal{L}$ (matice druhých derivací podle $\mathbf{z}$) pozitivně definitní pro všechny body $\mathbf{z}$ takové, že $\nabla \mathbf{g}(\mathbf{z}^*)^T \mathbf{h} = 0$, kde $\mathbf{h} = \mathbf{z} - \mathbf{z}^*$. Z toho vyplývá, že (6.20) je v okolí minima řešením problému

$$\mathcal{L}(\mathbf{z}^* + \mathbf{h}, \boldsymbol{\lambda}) \to \min_{\mathbf{h}} \tag{6.22}$$

$$\nabla \mathbf{g}^T \mathbf{h} = \mathbf{0}. \tag{6.23}$$

Nahradíme-li (6.22) kvadratickou funkcí v bodě $\mathbf{z}$ a omezení lineární funkcí, dostáváme

$$\begin{aligned}\mathcal{L}(\mathbf{z}^*,\boldsymbol{\lambda}) &= \left[f(\mathbf{z})+\boldsymbol{\lambda}^T\mathbf{g}(\mathbf{z})\right]+\left[\nabla f(\mathbf{z})+\boldsymbol{\lambda}^T\nabla\mathbf{g}(\mathbf{z})\right]^T\mathbf{h}+\frac{1}{2}\mathbf{h}^T\mathbf{W}\mathbf{h}= \\ &= f(\mathbf{z})+\nabla f(\mathbf{z})^T\mathbf{h}+\frac{1}{2}\mathbf{h}^T\mathbf{W}\mathbf{h}\end{aligned} \tag{6.24}$$

kde $\mathbf{W}$ je matice druhých derivací $\mathcal{L}(\mathbf{z},\boldsymbol{\lambda})$ podle $\mathbf{z}$, neboť $\mathbf{g}(\mathbf{z})+\nabla\mathbf{g}(\mathbf{z})^T\mathbf{h}=\mathbf{g}(\mathbf{z}^*)=\mathbf{0}$. Výpočet pak probíhá jako posloupnost problémů kvadratického programování

$$\begin{aligned}&f(\mathbf{z}_k)+\nabla f(\mathbf{z}_k)^T\mathbf{h}+\frac{1}{2}\mathbf{h}^T\mathbf{W}\mathbf{h}\rightarrow\min_{\mathbf{h}} \\ &\nabla\mathbf{g}^T\mathbf{h}=\mathbf{0}.\end{aligned} \tag{6.25}$$

Nový bod je určen jako $\mathbf{z}_{k+1}=\mathbf{z}_k+\mathbf{h}$ a jako nový odhad $\boldsymbol{\lambda}$ je použit vektor Lagrangeových multiplikátorů podproblému (6.25). Získaný bod se většinou využije jako směr pro jednorozměrové hledání. Popsaný algoritmus je považován za velmi efektivní, avšak vyžaduje druhé parciální derivace J. Existuje však i quasi-Newtonova varianta, která pracuje pouze s gradienty.

Zajímavou alternativu představují rozšířená kritéria

$$J'(\mathbf{x})=J(\mathbf{x})+\sum_i\sigma_i|g_i(\mathbf{x})| \tag{6.26}$$

$$J'(\mathbf{x})=J(\mathbf{x})+\max_i\left\{\sigma_i|g_i(\mathbf{x})|\right\}. \tag{6.27}$$

V obou těchto případech se jedná o spojité, ale nehladké funkce, pro jejichž efektivní minimalizaci nelze použít výše uvedené algoritmy. Tento přístup je však atraktivní tím, že pro dostatečně vysoké pokutové konstanty σ_i se řešení problémů (6.26), (6.27) přesně shodují s řešením původního problému s omezeními [11] (v případě hladkých pokutových členů je shoda pouze v limitním případě $i\rightarrow\infty$).

Pro minimalizaci nehladkých funkcí ve speciálním tvaru (6.26), (6.27) existují účinné algoritmy, které jsou do značné míry podobné přímým metodám řešení problému (6.1), (6.2) [11]. Efektivní a robustní univerzální algoritmy pro nehladké úlohy však dosud nejsou k dispozici. V tomto případě je však možné rovněž použít algoritmy nevyužívající derivací, jako je např. metoda flexibilního simplexu [26], stručně popsaná dále v kapitole Heuristické algoritmy. Využití nehladkých pokutových funkcí se rovněž jeví jako perspektivní v souvislosti s výpočtem trajektorií v prostoru s překážkami.

6.3. Nalezení globálního minima

V předchozích kapitolách byly popsány metody, které na základě zadaného počátečního bodu $\mathbf{x}_0$ jsou schopné nalézt lokální minimum kritéria. Kritérium však může mít na uvažované množině parametrů více lokálních minim, jejichž hodnoty se mohou značně lišit.

Tento případ může nastat i u úloh bez omezení. V případě nahrazení trajektorie praktické zkušenosti ukazují, že počáteční odhad $\mathbf{z}_0=\mathbf{0}$ při ortogonálním rozkladu soustavy okrajových podmínek zpravidla vede rychle ke kvalitnímu řešení. V tomto případě je pravděpodobně možné se spokojit s lokální optimalizací.

V případě dodatečných omezení a rovněž v případě nahrazení řízení je třeba počítat s tím, že pro získání globálního minima, nebo alespoň jemu blízkého řešení z hlediska hodnoty kritéria, může být třeba vyzkoušet více počátečních odhadů. Tím však dojde k znásobení doby výpočtu, která i pro jednorázový výpočet není zanedbatelná.

Pro efektivnější výpočet existuje řada tzv. metod globální optimalizace. Těchto metod existuje velké množství a mohla být vyzkoušena pouze malá část. Prakticky testované algoritmy lze rozdělit do tří skupin:

a) lokální optimalizace z mnoha počátečních bodů
b) heuristické algoritmy
c) hybridní metody

Toto rozdělení se však týká pouze algoritmů využitých v rámci této práce a není nijak jednotné s existující literaturou v oblasti globální optimalizace.

6.4. Lokální optimalizace z mnoha počátečních bodů

Metody z této skupiny v principu pracují ve třech krocích:

1. určení odhadu umístění lokálních minim v zadané oblasti
2. nalezení lokálních minim v určených podoblastech některou z běžných metod
3. výběr minima

Elementárním postupem z této kategorie je zmíněný opakovaný lokální optimalizační výpočet z náhodně generovaných počátečních odhadů, který je v literatuře uváděn pod označením „Multistart" [21]. Tento algoritmus je však velmi neefektivní, neboť zřejmě mnoho optimalizačních výpočtů může směřovat do stejného lokálního minima.

Dokonalejší techniky jsou založeny na odhadu umístění lokálních minim v zadané oblasti a následné optimalizaci pouze v těchto podoblastech. Odhad umístění lokálních minim je proveden většinou náhodným generováním bodů v dané oblasti a následným zpracováním. Jednou skupinou možných postupů je „shlukování" (Clustering), které je obdobou metod shlukové analýzy z umělé inteligence [21]. Jsou hledány body (středy shluků), které mají nejmenší hodnotu v okolí dané velikosti. Se zvyšujícím se počtem vygenerovaných bodů se velikost shluků zmenšuje. V literatuře lze nalézt řadu variant uvedeného postupu. Ze středů shluků je provedeno lokální hledání minima.

Jinou variantou je tzv. topografická metoda [22], která určuje odhady minim jako takové body, jejichž hodnota je nejmenší z hodnot jim nejbližších M bodů, kde M je zvolená konstanta. Předností tohoto postupu oproti shlukování je, že odpadá výpočet velikosti shluku na základě počtu vygenerovaných bodů.

Třebaže pro některé ze zmíněných postupů existují důkazy konvergence ke globálnímu minimu, v praxi se ukazuje, že tyto metody jsou málo účinné pro vyšší počet optimalizovaných parametrů. V případě, že prostor parametrů má dimenzi větší než 10, je pro vytvoření reprezentativních odhadů minim třeba nereálně vysoký počet vygenerovaných bodů.

Prakticky byla testována upravená topografická metoda a upravená metoda Multistart. Nejdůležitější úprava spočívá ve vyloučení neperspektivních minim. Jelikož samotný výpočet lokálního minima je tomto případě časově náročnou operací, není možné provést úplný výpočet ze všech počátečních odhadů. Místo toho je výpočet ze všech bodů prováděn paralelně a čas je těmto bodům přidělován úměrně jejich úspěšnosti vzhledem k počtu vykonaných iteračních kroků. Na základě určitých kritérií pak může být rozhodnuto o „zmrazení" určitých bodů.

Třebaže je uvedený postup poměrně sofistikovaný a komplikovaný z hlediska implementace, jsou výsledky horší, než by bylo možné očekávat. Algoritmus je funkční pouze v případě, kdy má kritérium malý počet lokálních minim, která lze dobře lokalizovat malým počtem náhodně vygenerovaných bodů. Pro typický případ výpočtu optimálních trajektoriím kde $n_z \approx 30$, není

bohužel možné očekávat spolehlivé nalezení globálního minima tímto způsobem. U některých jiných typů problémů tyto algoritmy mohou být úspěšnější, avšak pro uvažovanou třídu problémů, kdy kritérium je deformováno pokutovými funkcemi a tvoří úzká údolí, jsou patrně nevhodné. Přitom jejich výpočetní náročnost je značná a i přes různá vylepšení zde jednoznačně dochází k plýtvání výpočetním výkonem.

6.5. Heuristické algoritmy

Druhá skupina testovaných algoritmů globální optimalizace je založená na různých, většinou heuristických principech. Některé z těchto metod jsou inspirovány procesy v přírodě.

Na základě praktických zkušeností autora lze říci, že většina těchto metod trpí stejnými problémy jako algoritmy z první skupiny. Pro $n_z < 10$ se často chovají uspokojivě, avšak pokud bychom uvažovali pouze ty algoritmy, které účinně řeší problémy pro $n_z > 20$, jejich počet se výrazně sníží. Vývoj v této oblasti je často podceňován, neboť u mnohých algoritmů není zaručena konvergence ke globálnímu minimu. Důraz je spíše kladen na metody, kde příslušné matematické důkazy existují. Bohužel, algoritmy s těmito vlastnostmi jsou v praxi často neúčinné.

Na druhé straně však existují heuristické algoritmy, které řeší komplikované úlohy i pro $n_z > 30$ a vyšší. Pomocí algoritmů z této skupiny byla v minulých letech úspěšně řešena řada problémů, které byly dříve pokládány za neřešitelné. Do této kategorie pravděpodobně patří i problém výpočtu optimálních trajektorií v prostoru s překážkami.

Významnou výhodou heuristických metod je, že zpravidla nevyžadují hladkost funkcí J a $\mathbf{g}$. Je často dokonce možné nahradit omezení (6.2) výrazně obecnější podmínkou

$$\mathbf{z} \in \Omega \tag{6.28}$$

kde Ω je kompaktní množina, která je definována pouze funkcí příslušnosti

$$\begin{aligned} \omega(\mathbf{z}) &= 1 \text{ pokud } \mathbf{z} \in \Omega \\ \omega(\mathbf{z}) &= 0 \text{ pokud } \mathbf{z} \notin \Omega \end{aligned} \tag{6.29}$$

Omezení tvaru (6.2) lze zřejmě snadno převést na tvar (6.28), avšak nikoliv naopak.

V případě výpočtu trajektorií v prostoru s překážkami je vyhodnocení (6.28) výrazně efektivnější i algoritmicky jednodušší než konstrukce pokutové funkce odpovídající omezení ve tvaru (6.2), neboť stačí zjistit jediný bod, ve kterém dochází ke kolizi. Ukazuje se však, že většina algoritmů přesto konverguje rychleji a spolehlivěji k řešení, jsou-li omezení reprezentována pokutovým členem kritéria. Příčinou tohoto jevu pravděpodobně je, že v případě pokutových členů je možné přibližovat se k minimu i ze strany omezení. Jelikož hladkost účelové funkce není vyžadována, je v tomto případě patrně nejvýhodnější použít nehladké pokutové funkce (6.26), (6.27). Konstrukce hladkých a nehladkých pokutových funkcí pro výpočet trajektorií v prostoru s překážkami pro konkrétní typ modelů těles byla detailně popsána v předchozí části práce.

Charakteristickou vlastností většiny heuristických algoritmů je, že v průběhu výpočtu dochází k hustšímu generování nových bodů v okolí bodů s nejnižší dosaženou hodnotou. Toto je nutnou podmínkou praktické konvergence k minimu v prostorech s vyšší dimenzí n_z, třebaže teoreticky je konvergence k minimu zaručena např. i při rovnoměrném náhodném generování bodů. Tím ale na druhé straně v průběhu výpočtu dochází ke snižování schopnosti pokrýt celou množinu Ω, takže algoritmus postupně přechází z globálního režimu do režimu lokálního. V případě, že okolí globálního extrému je poměrně malé, je pravděpodobné, že toto okolí nebude nalezeno v počátečních iteracích a výše popsaný efekt způsobí, že tato pravděpodobnost se bude dále snižovat.

Z toho však vyplývá, že tyto algoritmy je třeba chápat pouze jako jakýsi nejasný kompromis mezi schopností nalézt globální minimum a účinností. Je ale obtížné měřit kvalitu tohoto kompromisu a proto zpravidla nelze porovnat vlastnosti heuristických metod pomocí analytických nástrojů. Pro konkrétní typy problémů je patrně možné najít obzvláště efektivní algoritmy. K tomu je však třeba hlubší znalost problému a tvaru účelové funkce, než která je obvykle k dispozici.

Pro konkrétní typ problému je však možné provést experimentální srovnání s různými počátečními odhady a parametry. Takto je možné u každé metody najít přibližně optimální nastavení a nejlepší výsledky různých metod mezi sebou porovnat. Často je však třeba nalézt algoritmy do značné míry univerzální. V tom případě je nutné zvolit určitou množinu reprezentativních testovacích problémů, což však znásobí množství potřebné experimentální práce. Výhodou je, že tyto experimenty je možné naprogramovat na počítači. Takto lze jistě získat kvalitní představu o účinnosti jednotlivých algoritmů. Tento automatizovaný způsob porovnání účinnosti algoritmů však dosud nebyl realizován.

Za kvalitní algoritmus lze považovat ten, který:

- pro různé počáteční odhady je schopen nalézt pro každou testovací úlohu téměř vždy globální minimum v akceptovatelném čase

- umožňuje nalézt takové konstantní nastavení parametrů algoritmu, že předchozí bod platí pro každou z testovacích funkcí

Druhý bod není samozřejmostí, neboť pro různé problémy může konkrétní algoritmus vyžadovat jiné nastavení parametrů, a to zpravidla není známé. Proto je druhý bod jedním z významných kritérií kvality algoritmu.

Testované heuristické algoritmy lze rozdělit do dvou hlavních skupin:

a) jednobodové metody
b) vícebodové metody

Jednobodové metody je možné definovat tak, že udržují v paměti mezi iteračními kroky pozici a hodnotu kritéria jednoho bodu v prostoru parametrů, obdobně jako lokální metody. Za jednobodové však rovněž považujme algoritmy, které uchovávají hodnoty více bodů současně, ale nedochází k interakci mezi těmito body.

Naopak vícebodové metody pracují s množinou bodů, které se navzájem ovlivňují. Ve všech zde uvažovaných případech se počet těchto bodů nemění v průběhu výpočtu.

K jednobodovým metodám patří:

1. Metoda simulovaného ochlazování [24]. Tento algoritmus je inspirován fyzikálními procesy probíhajícími při ochlazování rozžhaveného materiálu. V případě, že ochlazování je dostatečně pomalé, je vytvořena struktura, která odpovídá stavu s minimální energií. Typickým jevem v tomto případě je, že jsou s určitou pravděpodobností akceptovány změny stavu i směrem k vyšší energii, zatímco změny směrem k nižší energii jsou akceptovány vždy. Pravděpodobnost přijetí stavu s vyšší energií exponenciálně klesá s teplotou k nule. Energii odpovídá minimalizovaná funkce. Algoritmus generuje náhodné změny pozice aktuálního bodu a akceptuje body s nižší hodnotou kritéria a s klesající pravděpodobností i body s vyšší hodnotou. Velikost změny zpravidla rovněž klesá s teplotou. Po provedení daného počtu kroků se uloží nejnižší hodnota bodu dosažená při průchodu jako výsledek iterace a sníží se parametr odpovídající teplotě.

Algoritmus se pro řešení uvažovaného problému neosvědčil. Důvodem patrně je, že kritérium díky omezením formuje úzká údolí a pro vyšší n_z je velmi malá pravděpodobnost, že bude

vygenerován náhodný směr, který v těchto údolích odpovídá poklesu funkční hodnoty. Nebyly však testovány všechny varianty, které lze nalézt v literatuře.

2. Algoritmus hledání podél náhodné úsečky se spline-interpolací. Algoritmus spočívá v generování úsečky délky L náhodného směru se středem v bodě $\mathbf{z}_i$ a rozdělení úsečky na stejných $2k$ dílů. V uzlových bodech $\mathbf{p}_j$, $j=\{-k,...,k\}$, jsou vypočítány hodnoty kritéria a je jimi proložena spline-funkce. Je analyticky vypočítán bod minima spline-funkce $\mathbf{p}_s$. Bod $\mathbf{z}_{i+1}$ je pak určen jako

$$\mathbf{z}_{i+1} = \arg\left\{\min\left\{J(\mathbf{p}_s),\ J(\mathbf{p}_j),\ j=-k,...,k\right\}\right\}. \tag{6.30}$$

V základní verzi popsané v [7] je délka L volena tak, že úsečka obsáhne celou množinu Ω. V tom případě je sice zachována schopnost prohledat celý prostor parametrů z libovolného bodu $\mathbf{z}_i$, avšak metoda velmi pomalu konverguje v lokálním okolí extrému. Dokonce lze snadno sestrojit případ, kdy nekonverguje ani k lokálnímu minimu. Z tohoto důvodu je dle názoru autora výhodnější ponechat délku L jako náhodnou veličinu, jejíž střední hodnota je úměrná velikosti množiny Ω.

Tento postup byl první metodou globální optimalizace, která byla použita pro výpočet optimálních trajektorií. Je možné konstatovat, že i přes některá vylepšení oproti autorovi doporučené základní verzi a přes poměrně komplikovanou praktickou realizaci se jedná o algoritmus nekvalitní, který je funkční pouze pro hodnoty $n_z < 10$.

Třebaže se může zdát, že předností tohoto algoritmu oproti jiným metodám je zachování schopnosti nalézt okolí globálního extrému při rostoucím počtu iteračních kroků, ve skutečnosti při klesající hodnotě $J(\mathbf{z}_i)$ se rovněž snižuje pravděpodobnost, že bude vygenerován směr takový, že úsečka prochází oblastí s menší hodnotou. Tato pravděpodobnost sice teoreticky zůstává nenulová pokud $J(\mathbf{z}_i) > J(\mathbf{z}_{\min})$, pro vyšší n_z však může být zanedbatelně malá.

Do skupiny vícebodových metod patří např. následující:

3. Rozšířená metoda flexibilního simplexu [27]. Klasická lokální metoda flexibilního simplexu [26] používá lineárně nezávislých n_z+1 bodů, které se přesouvají směrem k poklesu hodnoty kritéria. Základním krokem je výběr bodu s nejvyšším ohodnocením a jeho přesun za protilehlou stěnu simplexu (tzv. reflexe). Kromě toho se provádí další operace, které mají přizpůsobit tvar simplexu tvaru kritéria jako expanze a redukce. Tato metoda je méně efektivní, než algoritmy s výpočtem derivací - zejména pro vyšší n_z. Výhodou však je, že lze použít i pro minimalizaci nehladkých funkcí a je možné ji poměrně snadno upravit i pro přímé řešení problémů s omezeními v obecném tvaru (6.28).

Možné rozšíření metody flexibilního simplexu pro globální optimalizaci spočívá v tom, že se pracuje s vyšším počtem bodů (typicky několikanásobek n_z). V každém kroku je z těchto bodů náhodně vybráno n_z+1 bodů a s těmito body je proveden jeden krok metody flexibilního simplexu. Tento algoritmus se však neosvědčil. Pro vyšší počet bodů konverguje příliš pomalu, a přitom ani v tomto případě často není schopen nalézt globální minimum. Nebyly dosud testovány některé varianty.

4. Stochastické evoluční algoritmy. Tyto metody jsou založeny na generování náhodných bodů uvnitř dané oblasti. Na počátku je vygenerováno v množině Ω rovnoměrně M nebo více náhodných bodů. Množina M bodů s nejlepší hodnotou je uchovávána v paměti a z tvaru této množiny jsou odvozeny parametry generátoru nových bodů. Většinou je použito normální rozdělení se střední hodnotou v bodě $\mathbf{z}_{\min}$. Parametr σ může být v každém směru určen např. jako

$$\sigma_k = \alpha \max_{i,j}\left\{(\mathbf{z}_i)_k - (\mathbf{z}_j)_k\right\} + \varepsilon \tag{6.31}$$

$$\sigma_k = \alpha \left| (\mathbf{z}_r)_k - (\mathbf{z}_s)_k \right| + \varepsilon \tag{6.32}$$

kde $(\mathbf{z}_i)_k$ je k-tá složka vektoru $\mathbf{z}_i$, $\mathbf{z}_r, \mathbf{z}_s$ jsou náhodné body z populace různé od $\mathbf{z}_{\min}$ a α je zvolený parametr. Heuristiky (6.31), (6.32) byly převzaty z [25].

Prakticky bylo ověřeno, že tento jednoduchý algoritmus může být efektivní pro určité typy problémů při správné volbě parametru α. Pokud funkce tvoří úzká údolí, je však v prostorech vyšší dimenze velmi malá pravděpodobnost, že bude vygenerován bod uvnitř tohoto údolí. Patrně proto se tento algoritmus pro návrh optimálních trajektorií rovněž neosvědčil.

5. Genetické algoritmy [23]. Tyto metody jsou inspirovány vývojem živých organismů směrem k dokonalejším druhům a jeho realizací na úrovni chromozómů. Chromozóm je řetězec bitů, který odpovídá vektoru **z**. V populaci, tj. konečné množině bodů, jsou náhodně vybírány dvojice prvků a mezi jejich chromozomy jsou provedeny operace podobné těm, které probíhají na buněčné úrovni u živých organismů, tj. přesmyk, křížení a mutace. Výsledkem je bod, který je určitou kombinací svých rodičů. Současně probíhá proces vymírání nejslabších jedinců, tj. odstraňování bodů s nejvyšší hodnotou.

Genetické algoritmy společně s algoritmy simulovaného ochlazování přinesly v určitém smyslu průlom v oblasti optimalizace. Svou povahou jsou však blízké spíše kombinatorickým problémům. Už při hledání minima v diskrétní oblasti, tj. kdy složky vektoru **z** jsou celá čísla, je sice možné chromozom sestavit z binárního vyjádření složek, avšak jednotlivé bity se dle rozsahu hodnot mohou velmi výrazně lišit svým vlivem na hodnotu kritéria. To má negativní vliv na rychlost konvergence k minimu. V případě reálných čísel, kdy binární reprezentace je delší a obsahuje i exponent, je významový rozdíl mezi jednotlivými bity obvykle ještě mnohem větší. Proto s implementací těchto algoritmů pro optimalizaci ve spojité oblasti je spojena řada problémů.

6. Algoritmy diferenciální evoluce [28], [29] byly původně navrženy jako jednoduchá náhrada genetických algoritmů pro optimalizaci ve spojité oblasti, kdy se z výše uvedených důvodů nevyužívají chromozomy jako řetězce bitů jednotlivých složek vektoru **z**, ale místo toho se pracuje s těmito složkami přímo v reálné podobě. To je umožněno náhradou genetických operátorů.

Z mnoha variant diferenciální evoluce se velmi dobře osvědčily tyto heuristiky:

$$\mathbf{z}_n = \mathbf{z}_1 + \mathcal{Z}_C\left(F(\mathbf{z}_2 - \mathbf{z}_3)\right) \tag{6.33}$$

$$\mathbf{z}_n = \mathbf{z}_{\min} + \mathcal{Z}_C\left(F(\mathbf{z}_1 - \mathbf{z}_2 + \mathbf{z}_3 - \mathbf{z}_4)\right). \tag{6.34}$$

Zde $\mathbf{z}_n$ je nově generovaný bod, $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4$ jsou různé náhodně vybrané body z populace (u varianty (6.34) různé od $\mathbf{z}_{\min}$). F je konstanta a $\mathcal{Z}_C(.): \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ je operátor poruchy, který s danou pravděpodobností C nuluje složky svého vektorového argumentu (ovšem tak, aby zůstala alespoň jedna složka nenulová). Tento operátor odpovídá mutaci u genetických algoritmů.

Algoritmy diferenciální evoluce se z popsaných metod při vhodném nastavení parametrů, tj. konstant F a C, osvědčily zdaleka nejlépe. Varianta (6.33) se zdá být spolehlivější pro mimořádně komplikované funkce. Příkladem jsou funkce, kde počet lokálních extrémů závisí exponenciálně na n_z. V ostatních případech je obvykle mnohem účinnější varianta (6.34).

Úspěšnost těchto metod je patrně dána tím, že jsou schopny se dobře přizpůsobit tvaru účelové funkce. Na rozdíl např. od metody flexibilního simplexu jsou však akceptovány i body s vyšší hodnotou než $\mathbf{z}_{\min}$ (jsou-li lepší než $\mathbf{z}_{\max}$), což zaručuje globální chování algoritmu. Ačkoliv se může zdát, že varianta (6.34) není vhodná pro problémy s mnoha extrémy, bylo prakticky ověřeno, že je schopna poměrně spolehlivě nalézt globální minimum velmi složitých funkcí v prostoru vysoké dimenze (např. $n_z = 100$).

Počet bodů M byl většinou zvolen jako násobek n_z. Pro méně složité funkce obvykle stačí $M = 4n_z - 6n_z$. Pro funkce s mnoha lokálními extrémy to může být i $10n_z$ a více. Není jisté, jestli je korektní uvažovat lineární závislost počtu bodů na n_z, ale tato volba se poměrně dobře osvědčila.

Ve snaze dále urychlit algoritmy diferenciální evoluce byly testovány modifikace, které heuristiky (6.33) a (6.34) určitým způsobem kombinují s hledáním minima podél úsečky s využitím spline-interpolace. Bod $\mathbf{z}_n$ v předchozím algoritmu je v podstatě využit jako směr pro hledání. Tyto metody někdy mohou být účinnější než diferenciální evoluce, ale jednoznačné závěry nejsou k dispozici.

6.6. Hybridní metody

Možností, jak urychlit výpočet, je použít heuristický algoritmus pouze pro získání odhadu globálního minima. Získaný odhad pak upřesnit efektivní lokální metodou. Předpokladem je, že účelová funkce je hladká. Tento postup může být relativně úsporný z hlediska spotřebovaného výpočetního výkonu. Na druhé straně však nezaručuje nalezení globálního minima, neboť to zpravidla nezaručuje ani použitý heuristický algoritmus. Kromě toho může být problém rozhodnout, kdy se jedná o správný odhad globálního minima a je možné ukončit první fázi výpočtu. Výjimkou je případ, kdy jsou předem k dispozici nějaké informace o globálním extrému – např. jeho přibližná hodnota. Jinou možností je nějakým způsobem odhadnout iterační krok, kdy heuristický algoritmus přešel definitivně do lokální fáze (tj. všechny následující generované body směřují k jedinému minimu). U vícebodových metod je např. možné použít aktuální velikost shluku bodů a apriorní informaci o minimální vzdálenosti lokálních minim.

Pokud nelze takto rozhodnout o ukončení globální fáze, je možné tento postup upravit tak, že obě fáze probíhají současně. Uvažujme, že heuristický algoritmus generuje posloupnost odhadů globálního minima $\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2, \ldots$. Položme $\mathbf{z}_1 = \tilde{\mathbf{z}}_1$ a z tohoto bodu startujme hledání lokálního minima. Algoritmus lokální optimalizace generuje v následujících krocích posloupnost $\mathbf{z}_1, \mathbf{z}_2, \ldots$, která je zpřesněním odhadu $\tilde{\mathbf{z}}_1$. Kroky hledání $\tilde{\mathbf{z}}_i, \mathbf{z}_i$ jsou vykonávány pro stejné i ihned po sobě. Jelikož předpokládáme, že lokální algoritmus konverguje rychleji, rozdíl $J(\tilde{\mathbf{z}}_i) - J(\mathbf{z}_i)$ je zpočátku kladný a jeho hodnota se zvyšuje. Postupně se však hodnota vyrovná a jestliže body $\mathbf{z}_1, \mathbf{z}_2, \ldots$ jsou zpřesněním odhadu pouze lokálního minima, pro určité i bude tento rozdíl záporný. V tom okamžiku končí fáze zpřesnění odhadu $\tilde{\mathbf{z}}_1$ a lokální algoritmus se restartuje pro zpřesnění bodu $\tilde{\mathbf{z}}_i$.

Předpokladem konvergence tohoto hybridního postupu je pouze schopnost použitého heuristického algoritmu poskytnout po konečném počtu kroků dostatečně přesný odhad globálního minima. Lokální fáze na konvergenci ke globálnímu extrému nemá žádný vliv. Příklad průběhu hodnoty kritéria při současně probíhající globální a lokální fázi je na Obr. 6.2.

Je zřejmé, že popsaný hybridní postup je účinný zejména v případě, kdy je poměrně malý počet lokálních minim s výrazně rozdílnými hodnotami. Jinak se schopnost nalezení globálního minima nesníží (tu garantuje dle předpokladu použitý heuristický algoritmus), ale dojde často k restartování lokální optimalizace a tím k plýtvání výpočetním výkonem pro zpřesnění nesprávných lokálních minim. Z Obr. 6.2 je zřejmé, že při ukončení výpočtu po jakémkoliv počtu iterací je výsledná hodnota lepší než v případě samotného heuristického algoritmu.

Obr. 6.2: Průběh hodnoty kritéria při současně probíhající globální i lokální fázi

V některých případech je možné postup dále urychlit využitím výsledků z lokální fáze v globálním algoritmu, tj. přepsáním odhadu globálního minima (zpravidla bodu s nejmenší dosaženou hodnotou) $\tilde{\mathbf{z}}_i$ bodem $\mathbf{z}_i$, jestliže $J(\mathbf{z}_i) < J(\tilde{\mathbf{z}}_i)$. Tato možnost závisí na typu heuristického algoritmu. V některých případech tato úprava může negativně ovlivnit schopnost nalezení globálního minima.

Např. v případě metody založené na generování úsečky náhodného směru bylo řečeno, že schopnost nalézt globální minimum se zhoršuje s klesající hodnotou aktuálního bodu (pokud není přímo v okolí globálního minima). U vícebodových metod může mít výrazný posun pozice bodu s nejmenší hodnotou za následek deformaci tvaru shluku bodů. Algoritmus pak může rychleji ztrácet schopnost prohledat celou množinu parametrů.

V případě, že obě fáze probíhají současně, musí nutně musí dojít k plýtvání s výpočetním výkonem, avšak rozdělíme-li výpočetní čas mezi oba typy kroků např. v poměru 1:1, ztráta výkonu v globální fázi výpočtu je právě 50% (jestliže globální algoritmus nevyužívá výsledky lokální fáze), obdobně v blízkosti minima bude výpočet o 50% pomalejší než samotná lokální metoda. Toto rozdělení výpočetního času lze provést tak, že body $\tilde{\mathbf{z}}_i$, resp. $\mathbf{z}_i$ jsou chápány jako výsledky určité dávky iterací globální, resp. lokální metody, které jsou vykonávány souvisle. V některých případech může být výhodnější poměr měnit v průběhu výpočtu, aby např. v počátečních krocích bylo více času přiděleno globální fázi.

Hybridní metody představují jinou možnost, jak využít potenciálu efektivních lokálních algoritmů optimalizace, než představuje hledání minima z více počátečních bodů. Takto je možno do jisté míry využít výhod obou skupin algoritmů – zachovat schopnost nalezení globálního minima a zvýšit účinnost v blízkosti lokálních minim.

Již v rámci disertační práce byl realizován hybridní algoritmus využívající výše popsané generování úsečky náhodného směru se spline aproximací a pro lokální fázi metodu PARTAN. V pozdější době byl tento algoritmus překonán hybridními metodami na bázi diferenciální evoluce a PARTAN nebo BFGS, které využívaly lokální algoritmus pro přiblížení bodu s nejmenší hodnotou z populace k minimu. Podle posledních zkušeností je však samotná diferenciální evoluce, zejména varianta (6.34), většinou efektivnější. Tyto závěry však nejsou konečné a dle názoru autora jistě existují účinnější kombinace algoritmů, než které byly dosud testovány.

6.7. Využití výkonu více procesorů

Pro urychlení výpočtu je možné výpočet paralelizovat pro současnou práci více procesorů v rámci víceprocesorového počítače, popř. několika počítačů na lokální síti. Obecnými podmínkami je aby platilo:

- mezi dvěma kroky algoritmu je možné problém rozdělit na $n > 1$ elementárních podproblémů, které lze zpracovat nezávisle
- každý elementární podproblém je časově relativně náročnou operací

V případě, že není splněna druhá podmínka, může být režie předávání dat mezi procesy významná a paralelizace se nevyplatí.

Obě podmínky jsou většinou splněny při výpočtu optimální trajektorie v prostoru s překážkami nějakou metodou globální optimalizace. V tomto případě je zpravidla možné v určitém úseku algoritmu vypočítat hodnoty více bodů nezávisle na sobě. Přitom výpočet hodnoty bodu odpovídá integrálu kritéria s vyšetřením kolize podél trajektorie a je tedy časově náročnou operací.

Paralelní výpočet probíhá tak, že hlavní proces vygeneruje v určitém bodě algoritmu globální frontu úloh (1 úloha = 1 výpočet hodnoty bodu) a spouští pracovní procesy, které úlohy ve frontě postupně zpracují. Ty mohou běžet buďto na stejném počítači, nebo na jiných počítačích lokální sítě. Po zpracování všech úloh hlavní proces zpracuje výsledky, dokončí smyčku a pokračuje dalším iteračním krokem.

Tento postup autor v minulých letech prakticky realizoval univerzálně pro paralelní výpočet na jednom víceprocesorovém počítači nebo na lokální síti s využitím distribuovaných vlastností jádra operačních systémů řady Microsoft Windows NT. Rozbor řešení a základní kroky praktické realizace byly popsány v [A14] přímo pro globální optimalizaci algoritmem evolučního typu.

Jednodušší a současně i účinnější je však paralelizace výpočtu pouze v rámci jednoho víceprocesorového počítače. Na dvouprocesorových počítačích, které jsou dnes běžně dostupné, je tímto způsobem možné téměř zdvojnásobit rychlost výpočtu.

6.8. Závěr

V této části byl prezentován přehled algoritmů optimalizace, které byly testovány, popř. využity pro řešení daného problému, a stručně byly zmíněny i některé jiné perspektivní postupy.

V případě úloh bez překážek je situace poměrně jasná. S využitím známých algoritmů, jako jsou metody konjugovaných směrů nebo quasi-Newtonovy metody, lze dosáhnout zpravidla velmi dobrých výsledků. Pro řešení úloh, kde se vyskytují omezení typu rovností, je možné rozšířit kritérium o pokutové členy. Je však známo, že v tomto případě jsou většinou účinnější přímé metody, které však dosud nebyly v rámci této práce prakticky realizovány.

V případě úloh s překážkami se jedná o komplikovaný problém globální optimalizace a většina algoritmů popsaných v literatuře selhává. Na druhé straně však existují metody, které tyto úlohy řeší poměrně úspěšně. Vesměs se však jedná o postupy, které nezaručují nalezení globálního minima, ale jsou pouze určitým kompromisem mezi schopností prohledat celou množinu parametrů a přiblížit se efektivně k řešení. Jako perspektivní se autorovi jeví zejména tzv. hybridní metody, které kombinují vlastnosti heuristických algoritmů a metod lokální optimalizace. Dosud bylo však lepších výsledků dosaženo pomocí diferenciální evoluce (viz též řešené úlohy v 5. části práce).

Část 7.

Příloha - Matematické modely mechanických systémů

V této části jsou popsány čtyři matematické modely mechanických systémů, které jsou využity v řešených příkladech částí 3-5.

U prvních dvou modelů, které byly převzaty z práce [1], chybí detailní odvození pohybových rovnic a je uveden pouze výsledný tvar. Tyto modely jsou typickým příkladem reálných systémů regulárních vzhledem k řízení a byly od počátku využívány jako velmi dobrý materiál pro praktické testování vlastností navržených výpočetních postupů.

Poslední dva modely, které jsou popsány včetně odvození pohybových rovnic, jsou zde uvedeny pro demonstraci problému s netriviální koncovou podmínkou a jako příklad mechanického systému, který není regulární vzhledem k řízení.

7.1. Robot pracující v cylindrickém souřadném systému

Obr. 7.1 znázorňuje průmyslový robot pracující v cylindrickém souřadném systému a na Obr. 7.2 je odpovídající kinematické schéma. Tato struktura je poměrně jednoduchá, ale v současné době se v praxi vyskytuje zřídka.

Systém má tři stupně volnosti :

- rotace kolem svislé osy
- vertikální posuv
- horizontální posuv

Parametry modelu jsou:

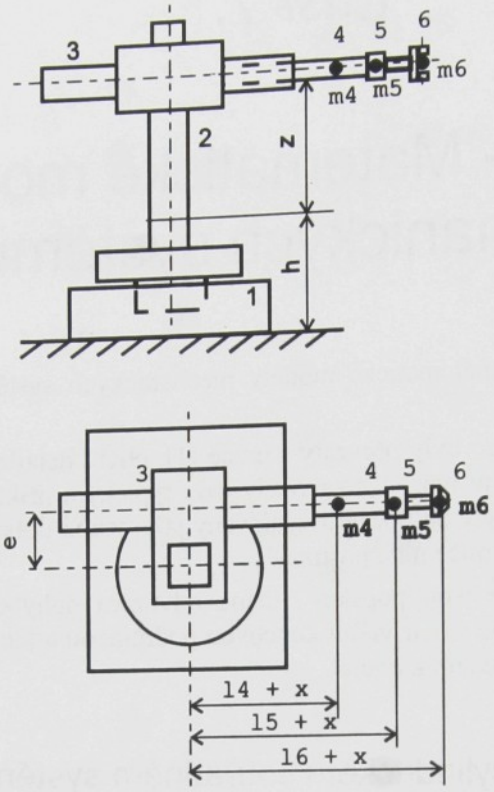
- m_3, m_4, m_5, m_6 hmotnosti členů 3, 4, 5 a břemene
- I_2, I_3, I_4 momenty setrvačnosti k ose rotace a členu 4 vzhledem k těžišti T_4
- l_4, l_5, l_6 polohy těžiště členů 4, 5, 6 při zasunutém rameni
- h, e rozměrové údaje

Vektor zobecněných souřadnic je

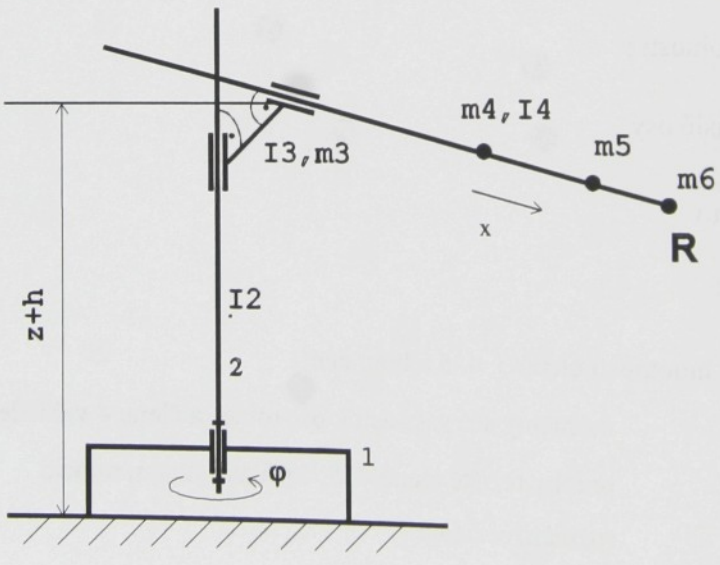
$$\mathbf{y} = (z, \varphi, x)^T . \tag{7.1}$$

Vektor působících zobecněných sil je

$$\mathbf{u} = (F_z, M_\varphi, F_x)^T . \tag{7.2}$$



Obr. 7.1: Robot v cylindrickém souřadném systému



Obr. 7.2: Kinematické schéma robota cylindrického typu

Zavedeme-li označení

$$A=\sum_{i=4}^{6} m_i,\ B=\sum_{i=4}^{6} l_i m_i,$$
$$C=\sum_{i=2}^{4} I_i+\sum_{i=4}^{6} m_i(e^2+l_i^2),\ D=\sum_{i=3}^{6} m_i \tag{7.3}$$

je možné získat s využitím Lagrangeových rovnic soustavu s řízením na pravé straně:

$$\begin{aligned}&\ddot{z}D+gD=F_z\\&\ddot{\varphi}(Ax^2+2Bx+C)+2\dot{\varphi}\dot{x}(Ax+B)-\ddot{x}eA=M_\varphi\\&\ddot{x}A-\dot{\varphi}^2(Ax+B)+\ddot{\varphi}eA=F_x\end{aligned} \tag{7.4}$$

Řídicí veličiny u_i ve skutečnosti nejsou přímo F_x, M_φ, F_z, ale jsou přímo úměrné těmto zobecněným silám.

Poloha referenčního (koncového) bodu robota R je popsána rovnicemi

$$\begin{aligned}X_R&=(l_6+z)\cos\varphi-e\sin\varphi\\Y_R&=(l_6+z)\sin\varphi+e\cos\varphi\\Z_R&=h+z\end{aligned} \tag{7.5}$$

7.2. Robot pracující v angulárním souřadném systému

Tato struktura průmyslových robotů je v současné době v praxi pravděpodobně nejrozšířenější (Obr. 7.3). Na druhé straně je však značně komplikovaná z hlediska dynamiky. Systém má tři stupně volnosti reprezentované třemi rotačními dvojicemi. Moment setrvačnosti J_2 členu 2 je vzhledem k ostatním členům robota zanedbán. K členu 3 náleží hmotné body m_1, m_2. m_4 vyjadřuje hmotnost zápěstí robota a hmotnost m_5 nahrazuje zátěž.

Parametry modelu:

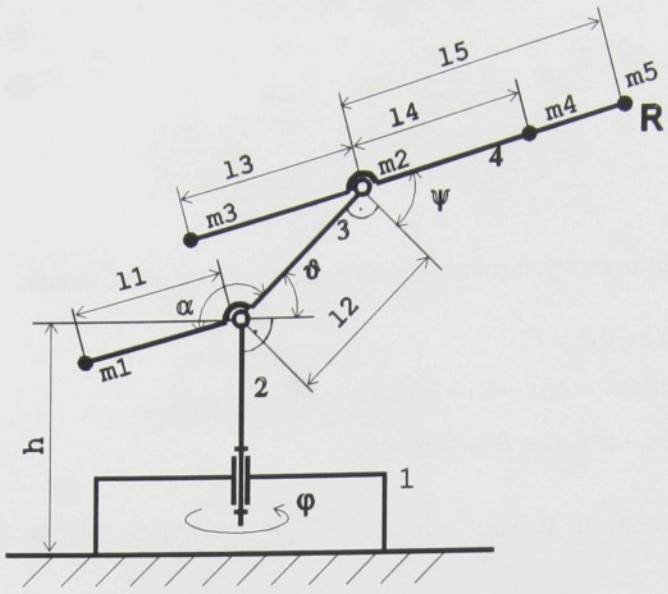
- m_1, m_2, m_3, m_4, m_5 redukované hmotnosti (hmotné body)
- l_1, l_2, l_3, l_4, l_5 vzdálenosti hmotných bodů od středů rotace
- h, e, α rozměrové údaje

Vektor zobecněných souřadnic je

$$\mathbf{y}=(\varphi,\vartheta,\psi)^T. \tag{7.6}$$

Vektor působících zobecněných sil je

$$\mathbf{u}=(M_\varphi,M_\vartheta,M_\psi)^T. \tag{7.7}$$



Obr. 7.3: Robot angulárního typu

Označme

$$C_1 = m_1 l_1^2,\ C_2 = l_2^2 \sum_{i=2}^{5} m_i,\ C_3 = l_2 \sum_{i=3}^{5} m_i l_i$$
$$C_4 = \sum_{i=3}^{5} m_i l_i^2,\ C_5 = C_1 + C_2 \tag{7.8}$$

Pohybové rovnice mají tvar :

$$\begin{aligned} M_\varphi = {} & \ddot{\varphi}\left[C_1 \cos^2(\vartheta+\alpha) + C_2 \cos^2\vartheta + 2C_3 \cos\vartheta.\sin(\psi+\vartheta) + C_4 \sin^2(\psi+\vartheta)\right] - \\ & - \dot{\varphi}\dot{\vartheta}\left[C_1 \sin(2\vartheta+2\alpha) + C_2 \sin 2\vartheta - 2C_3 \cos(\psi+2\vartheta) - C_4 \sin(2\psi+2\vartheta)\right] + \\ & + \dot{\varphi}\dot{\psi}\left[2C_3 \cos\vartheta.\cos(\psi+\vartheta) + C_4 \sin(2\psi+2\vartheta)\right] \end{aligned}$$

$$\begin{aligned} M_\vartheta = {} & \ddot{\vartheta}(2C_3 \sin\varphi + C_4 + C_5) + \ddot{\psi}(C_3 \sin\varphi + C_4) + \dot{\psi}(\dot{\psi}+2\dot{\vartheta})C_3 \cos\psi + \\ & + \tfrac{1}{2}\dot{\varphi}^2\left[C_1 \sin(2\vartheta+2\alpha) + C_2 \sin 2\vartheta - 2C_3 \cos(\psi+2\vartheta) - C_4 \sin(2\psi+2\vartheta)\right] + \\ & + g\left[\tfrac{C_1}{l_1}\cos(\vartheta+\alpha) + \tfrac{C_2}{l_2}\cos\vartheta + \tfrac{C_3}{l_2}\sin(\psi+\vartheta)\right] \end{aligned} \tag{7.9}$$

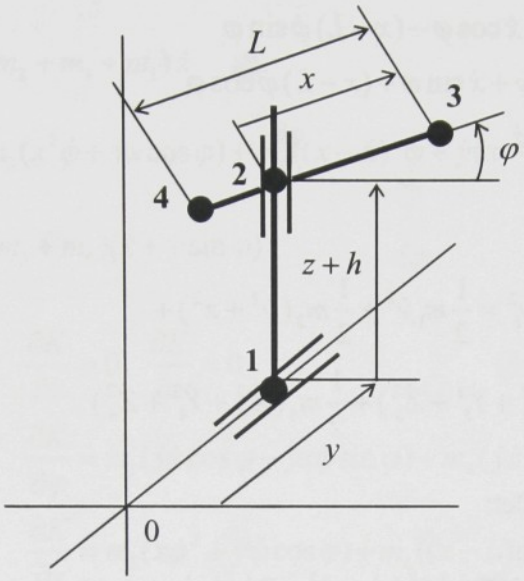
$$\begin{aligned} M_\psi = {} & C_4\ddot{\psi} + \ddot{\vartheta}(C_3 \sin\psi + C_4) - \dot{\vartheta}^2 C_3 \cos\psi - \\ & - \dot{\varphi}^2\left[C_3 \cos\vartheta.\cos(\psi+\vartheta) + \tfrac{1}{2}C_4 \sin(2\psi+2\vartheta)\right] + g\tfrac{C_3}{l_2}\sin(\psi+\vartheta) \end{aligned}$$

Poloha referenčního (koncového) bodu robota R je popsána rovnicemi

$$\begin{aligned} X_R &= \left[(l_2 \cos\vartheta + l_5 \cos(\psi - (\tfrac{\pi}{2} - \vartheta))\right]\cos\varphi \\ Y_R &= \left[(l_2 \cos\vartheta + l_5 \cos(\psi - (\tfrac{\pi}{2} - \vartheta))\right]\sin\varphi \\ Z_R &= h + l_2 \sin\vartheta + l_5 \sin(\psi - (\tfrac{\pi}{2} - \vartheta)) \end{aligned} \tag{7.10}$$

7.3. Zjednodušený model robota cylindrického typu rozšířeného o pojezd

Uvažujme zjednodušený model robota cylindrického typu, který je však navíc umístěn na pojezdu ve směru osy y (Obr. 7.4).



Obr. 7.4: Robot cylindrického typu s pojezdem

Setrvačné vlastnosti jsou reprezentovány pouze hmotnými body **1** – **4** s hmotnostmi m_1 až m_4.

Konkrétní hodnoty mechanických konstant byly zvoleny následovně:

$$\begin{aligned} m_1 &= 20 \\ m_2 &= 10 \\ m_3 &= m_4 = 5 \\ h &= 0.75 \\ L &= 2 \end{aligned} \tag{7.11}$$

Souřadnice bodů **3** a **4** v prostoru v závislosti na souřadnicích systému $\mathbf{y} = (y, z, \varphi, x)^T$ jsou

$$\begin{aligned} X_3 &= x\cos\varphi \\ Y_3 &= y + x\sin\varphi \\ Z_3 &= z + h \end{aligned} \tag{7.12}$$

$$\begin{aligned} X_4 &= (x - L)\cos\varphi \\ Y_4 &= y + (x - L)\sin\varphi \\ Z_4 &= z + h \end{aligned} \tag{7.13}$$

Složky vektoru rychlosti v_3 bodu **3** a rychlosti v_4 bodu **4** jsou

$$\begin{aligned} \dot{X}_3 &= \dot{x}\cos\varphi - x\dot{\varphi}\sin\varphi \\ \dot{Y}_3 &= \dot{y} + \dot{x}\sin\varphi + x\dot{\varphi}\cos\varphi \\ \dot{Z}_3 &= \dot{z} \end{aligned} \tag{7.14}$$

$$\begin{aligned} \dot{X}_4 &= \dot{x}\cos\varphi - (x-L)\dot{\varphi}\sin\varphi \\ \dot{Y}_4 &= \dot{y} + \dot{x}\sin\varphi + (x-L)\dot{\varphi}\cos\varphi \\ \dot{Z}_4 &= \dot{z} \end{aligned} \tag{7.15}$$

Kinetická energie soustavy je ve tvaru

$$\begin{aligned} K = \frac{1}{2}\sum_{i=1}^{4} m_i v_i^2 = \frac{1}{2}m_1\dot{y}^2 + \frac{1}{2}m_2(\dot{y}^2+\dot{z}^2) + \\ + \frac{1}{2}m_3(\dot{X}_3^2+\dot{Y}_3^2+\dot{Z}_3^2) + \frac{1}{2}m_4(\dot{X}_4^2+\dot{Y}_4^2+\dot{Z}_4^2) \end{aligned} \tag{7.16}$$

Vyjádříme-li členy $\dot{X}_3^2$, $\dot{Y}_3^2$ a jejich součet:

$$\begin{aligned} \dot{X}_3^2 &= \dot{x}^2(\cos\varphi)^2 + x^2\dot{\varphi}^2(\sin\varphi)^2 - 2\dot{x}x\dot{\varphi}\sin\varphi\cos\varphi \\ \dot{Y}_3^2 &= \dot{y}^2 + \dot{x}^2(\sin\varphi)^2 + x^2\dot{\varphi}^2(\cos\varphi)^2 + 2\dot{y}\dot{x}\sin\varphi + \\ &\quad + 2\dot{y}x\dot{\varphi}\cos\varphi + 2\dot{x}x\dot{\varphi}\sin\varphi\cos\varphi \end{aligned} \tag{7.17}$$

$$\dot{X}_3^2 + \dot{Y}_3^2 = \dot{x}^2 + \dot{y}^2 + x^2\dot{\varphi}^2 + 2\dot{y}(\dot{x}\sin\varphi + x\dot{\varphi}\cos\varphi)\,. \tag{7.18}$$

Obdobně pro $\dot{X}_4^2$, $\dot{Y}_4^2$:

$$\dot{X}_4^2 + \dot{Y}_4^2 = \dot{x}^2 + \dot{y}^2 + (x-L)^2\dot{\varphi}^2 + 2\dot{y}[\dot{x}\sin\varphi + (x-L)\dot{\varphi}\cos\varphi]\,. \tag{7.19}$$

Po dosazení do (7.16):

$$\begin{aligned} K = \frac{1}{2}m_1\dot{y}^2 + \frac{1}{2}(m_2+m_3+m_4)(\dot{z}^2+\dot{y}^2) + \frac{1}{2}(m_3+m_4)\dot{x}^2 + \\ + \frac{1}{2}m_3\left[x^2\dot{\varphi}^2 + 2\dot{y}(\dot{x}\sin\varphi + x\dot{\varphi}\cos\varphi)\right] + \\ + \frac{1}{2}m_4\left[(x-L)^2\dot{\varphi}^2 + 2\dot{y}(\dot{x}\sin\varphi + (x-L)\dot{\varphi}\cos\varphi)\right] \end{aligned} \tag{7.20}$$

Potenciální energie je rovna

$$P = (m_2+m_3+m_3)g\,z \tag{7.21}$$

kde g je tíhové zrychlení. Lagrangeovy rovnice mají tvar:

$$\frac{d}{dt}\frac{\partial K}{\partial \dot{\mathbf{y}}} - \frac{\partial K}{\partial \mathbf{y}} + \frac{\partial P}{\partial \mathbf{y}} = \mathbf{F} \tag{7.22}$$

kde $\mathbf{F} = (F_y, F_z, M_\varphi, F_x)^T$ je vektor zobecněných sil. Vyjádříme-li parciální derivace K podle $\mathbf{y}, \dot{\mathbf{y}}$:

$$\begin{aligned}
\frac{\partial K}{\partial \dot{y}} &= (m_1 + m_2 + m_3 + m_3)\,\dot{y} + m_3(\dot{x}\sin\varphi + x\dot{\varphi}\cos\varphi) + m_4(\dot{x}\sin\varphi + (x-L)\dot{\varphi}\cos\varphi) \\
\frac{\partial K}{\partial \dot{z}} &= (m_2 + m_3 + m_3)\,\dot{z} \\
\frac{\partial K}{\partial \dot{\varphi}} &= m_3(x^2\dot{\varphi} + \dot{y}x\cos\varphi) + m_4((x-L)^2\dot{\varphi} + \dot{y}(x-L)\cos\varphi) \\
\frac{\partial K}{\partial \dot{x}} &= (m_3 + m_4)(\dot{x} + \dot{y}\sin\varphi)
\end{aligned} \tag{7.23}$$

$$\begin{aligned}
&\frac{\partial K}{\partial y} = 0, \quad \frac{\partial K}{\partial z} = 0 \\
&\frac{\partial K}{\partial \varphi} = m_3(\dot{y}\dot{x}\cos\varphi - \dot{y}x\dot{\varphi}\sin\varphi) + m_4(\dot{y}\dot{x}\cos\varphi - \dot{y}(x-L)\dot{\varphi}\sin\varphi) \\
&\frac{\partial K}{\partial x} = m_3(x\dot{\varphi}^2 + \dot{y}\dot{\varphi}\cos\varphi) + m_4((x-L)\dot{\varphi}^2 + \dot{y}\dot{\varphi}\cos\varphi)
\end{aligned} \tag{7.24}$$

Parciální derivace P jsou nulové kromě

$$\frac{\partial P}{\partial z} = (m_2 + m_3 + m_4)g\ . \tag{7.25}$$

Dosadíme-li do (7.22) získáme pohybové rovnice systému:

$$\begin{aligned}
&(m_1 + m_2 + m_3 + m_4)\,\ddot{y} + (m_3 + m_4)(\ddot{x}\sin\varphi + 2\dot{x}\dot{\varphi}\cos\varphi) + m_3(x\ddot{\varphi}\cos\varphi - x\dot{\varphi}^2\sin\varphi) + \\
&+ m_4((x-L)\ddot{\varphi}\cos\varphi - (x-L)\dot{\varphi}^2\sin\varphi) = F_y
\end{aligned} \tag{7.26}$$

$$(m_2 + m_3 + m_4)\,\ddot{z} + (m_2 + m_3 + m_4)g = F_z \tag{7.27}$$

$$m_3(2x\dot{x}\dot{\varphi} + x^2\ddot{\varphi} + \ddot{y}x\cos\varphi) + m_4[2(x-L)\dot{x}\dot{\varphi} + (x-L)^2\ddot{\varphi} + \ddot{y}(x-L)\cos\varphi) = M_\varphi \tag{7.28}$$

$$(m_3 + m_4)(\ddot{x} + \ddot{y}\sin\varphi) - m_3x\dot{\varphi}^2 - m_4(x-L)\dot{\varphi}^2 = F_x \tag{7.29}$$

Zamezíme-li pohybu v ose y, je $\dot{y} = \ddot{y} = 0$ a z rovnic (7.26) - (7.29) dostáváme jednodušší soustavu

$$(m_2 + m_3 + m_4)\,\ddot{z} + (m_2 + m_3 + m_4)g = F_z \tag{7.30}$$

$$m_3(2x\dot{x}\dot{\varphi} + x^2\ddot{\varphi}) + m_4[2(x-L)\dot{x}\dot{\varphi} + (x-L)^2\ddot{\varphi}] = M_\varphi \tag{7.31}$$

$$(m_3 + m_4)\ddot{x} - [m_3x + m_4(x-L)]\dot{\varphi}^2 = F_x \tag{7.32}$$

která odpovídá základnímu modelu robota v cylindrickém souřadném systému bez pojezdu.

Poloha koncového bodu robota je dána rovnicemi (7.12).

7.4. Model systému neregulárního vzhledem k řízení

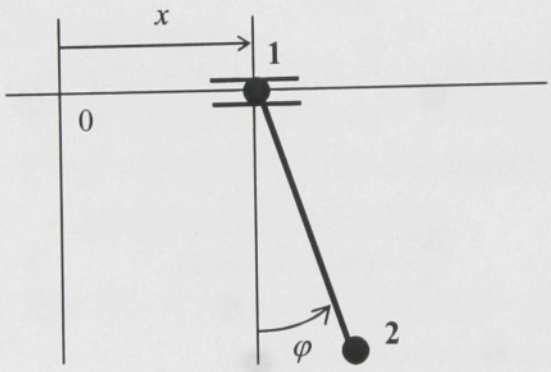
Uvažujme jednoduchý mechanický systém, který se skládá z kyvadla volně zavěšeného na posuvném rámu (Obr. 7.5). Ve směru posuvu působí síla F_x. Hmotnosti rámu a kyvadla jsou opět pro jednoduchost soustředěné do hmotných bodů **1** a **2**. Ve směru x je uvažováno viskózní tlumení. Výsledná síla působící na soustavu ve směru x je tedy

$$F_x - \beta \dot{x} \tag{7.33}$$

kde β je koeficient tlumení.

Konkrétní hodnoty mechanických konstant byly zvoleny následovně:

$$\begin{aligned} m_1 &= 2 \\ m_2 &= 1 \\ L &= 1 \\ \beta &= 0.1 \end{aligned} \tag{7.34}$$



Obr. 7.5: Kyvadlo na posuvném rámu

Zobecněné souřadnice systému jsou $\mathbf{y} = (x, \varphi)^T$. Pro polohu bodu **2** v prostoru v závislosti na souřadnicích **y** platí:

$$\begin{aligned} X &= x + L \sin \varphi \\ Z &= -L \cos \varphi \end{aligned} \tag{7.35}$$

Složky vektoru rychlosti v_2 bodu **2** jsou

$$\begin{aligned} \dot{X} &= \dot{x} + L\dot{\varphi} \cos \varphi \\ \dot{Z} &= L\dot{\varphi} \sin \varphi \end{aligned} \tag{7.36}$$

Kinetická energie soustavy je ve tvaru

$$K = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 \dot{x}^2 + \frac{1}{2} m_2 (\dot{X}^2 + \dot{Z}^2) . \tag{7.37}$$

Vyjádříme-li členy $\dot{X}^2$, $\dot{Z}^2$ a jejich součet:

$$\begin{aligned} \dot{X}^2 &= \dot{x}^2 + L^2 \dot{\varphi}^2 (\cos\varphi)^2 + 2L\dot{x}\dot{\varphi}\cos\varphi \\ \dot{Z}^2 &= L^2 \dot{\varphi}^2 (\sin\varphi)^2 \\ \dot{X}^2 + \dot{Z}^2 &= \dot{x}^2 + L^2 \dot{\varphi}^2 + 2L\dot{x}\dot{\varphi}\cos\varphi \end{aligned} \tag{7.38}$$

Po dosazení do (7.37):

$$K = \frac{1}{2}(m_1 + m_2)\dot{x}^2 + \frac{1}{2} m_2 (L^2 \dot{\varphi}^2 + 2L\dot{x}\dot{\varphi}\cos\varphi) . \tag{7.39}$$

Potenciální energie je rovna

$$P = m_2 g \cos\varphi . \tag{7.40}$$

Parciální derivace K a P podle $\mathbf{y}, \dot{\mathbf{y}}$ jsou:

$$\begin{aligned} \frac{\partial K}{\partial \dot{x}} &= (m_1 + m_2)\dot{x} + m_2 L\dot{\varphi}\cos\varphi \\ \frac{\partial K}{\partial \dot{\varphi}} &= m_2 L^2 \dot{\varphi} + m_2 L\dot{x}\cos\varphi \\ \frac{\partial K}{\partial x} &= 0 \\ \frac{\partial K}{\partial \varphi} &= -m_2 L\dot{x}\dot{\varphi}\sin\varphi \end{aligned} \tag{7.41}$$

$$\begin{aligned} \frac{\partial P}{\partial x} &= 0 \\ \frac{\partial P}{\partial \varphi} &= -m_2 g \sin\varphi \end{aligned} \tag{7.42}$$

Pohybové rovnice získáme po dosazení do Lagrangeových rovnic (7.22):

$$\begin{aligned} &(m_1 + m_2)\ddot{x} + m_2 L(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2 \sin\varphi) + \beta\dot{x} = F_x \\ &m_2 L^2 \ddot{\varphi} + m_2 L\ddot{x}\cos\varphi - m_2 g \sin\varphi = 0. \end{aligned} \tag{7.43}$$

Systém není regulární vzhledem k řízení, neboť na pravé straně (7.43) se vyskytuje nula, a ve vektoru řízení tedy chybí jedna složka.

Závěr

Jednotlivé části této zprávy již obsahují dílčí závěry. Dle názoru autora bylo v řadě směrů dosaženo kvalitních výsledků. Snad největším úspěchem je konečná verze metody nahrazení trajektorie využívající ortogonální rozklad soustavy okrajových podmínek a urychlený výpočet gradientu. Zajímavé výsledky však byly popsány i v případě nahrazení funkce řízení.

Nejméně jednoznačné a uzavřené jsou závěry v oblasti výpočtu optimálních trajektorií v prostoru s překážkami. Na druhé straně, právě zde bylo ve skutečnosti vykonáno největší množství práce. Pro složitější scény je možné výpočet patrně dále urychlit pomocí sofistikovanějších algoritmů vyšetření kolize, které lze nalézt v literatuře.

Otázka algoritmů globální optimalizace pro větší počet parametrů je všeobecně považována za otevřený problém, na který neexistuje jednoznačná odpověď. Při pohybu v prostoru s překážkami je bohužel pro aproximaci optimální trajektorie zpravidla třeba většího počtu parametrů (např. 10 nebo více na jednu osu). Přesto existují heuristické algoritmy, které dávají poměrně uspokojivé výsledky a je možné předpokládat, že v budoucnu se podaří objevit ještě účinnější postupy.

Seznam použité literatury

[1] Záda, V.: Optimalizace řízení robotů užitím spline-funkcí. Výzkumná zpráva, KTK-0169, VŠST Liberec, 1988

[2] Kolmogorov, A. N., Fomin, S. V.: Základy teorie funkcí a funkcionální analýzy. SNTL, Praha, 1975

[3] Alexejev, V. M., Tichomorov, V. M., Fomin, S. V.: Matematická teorie optimálních procesů. Academia, Praha, 1991

[4] Bryson, A. E., Ho, Y.-C.: Applied Optimal Control. Hemisphere Corp., New York, 1975

[5] Polak, E.: Computational Methods In Optimization. Academic Press, New York, 1971

[6] Rektorys, K.: Variační metody v inženýrských problémech a problémech matematické fyziky. SNTL, Praha, 1974

[7] Záda, V.: Globální optimalizace se spline-aproximací. Automatizace, 38 (1995), č.10

[8] Lubojacký, O. a kol.: Základy robotiky. Učební text, VŠST Liberec, 1990

[9] Rektorys, K. a kol.: Přehled užité matematiky. SNTL, Praha, 1981

[10] Jarník, V.: Diferenciální počet II. Academia, Praha, 1984

[11] Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons Ltd. 2nd edition, New York, 1987

[12] Gill, P. E., Murray, W.: Numerical Methods for Constrained Optimization. Academic Press, London, 1974

[13] Hanuš, B. a kol.: Teorie automatického řízení II. Učební text, VŠST Liberec, 1985

[14] Hamming, R. W.: Numerical Methods for Scientists and Engineers. McGraw-Hill, New York, 1973

[15] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.: Numerical Recipes in C. Cambridge University Press, 1992

[16] Arimoto, S.: Control Theory of Non-linear Mechanical Systems. Clarendon Press, Oxford, 1996

[17] Latombe, J. C.: Robot Motion Planning. Kluwer Academic Publishers, 1991.

[18] Amato, N., Wu, Y.: A randomized roadmap method for path and manipulation planning. IEEE International Conference on Robotics and Automation, 1996.

[19] Hubbard, P. M.: Approximating Polyhedra with Spheres for Time-Critical Collision Detection. ACM Transactions on Graphics,Vol.15, No. 3, 1996, str.179–210.

[20] Gilbert, E. G., Johnson, D. W., Keerthi, S. S.: A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal of Robotics and Automation 4(2):193–203, 1988.

[21] Törn, A., Žilinskas, A.: Global Optimization, Springer-Verlag, Berlin, 1989

[22] Törn, A., Viitanen, S.: Topographical global optimization using pre-sampled points. Journal of Global Optimization 5, 1994, 267-276

[23] Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Adison-Wesley, 1989

[24] Kirkpatrick, S., Gelatt Jr., C. D., Vecchi, M. P.: Optimization by Simulated Annealing. Science 220, 1983

[25] Tvrdík, J.: Algoritmus řízeného náhodného prohledávání a alternující heuristiky. Automa č. 1, 2002.

[26] Nelder, J.A., Meed, R.: A Simplex Method for Function Minimization. Computer Journal 7, Vol. 1, p. 308-313, 1964

[27] Price, W. L.: A Controlled Random Search Procedure for Global Optimization. Computer Journal 20, Vol. 4, p. 367-370, 1976

[28] Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Heuristics for Global Optimization over Continuous Spaces. Journal of Global Optimization 11. Kluwer Academic Publishers, p. 341-359, 1997

[29] Webovská stránka diferenciální evoluce - www.icsi.berkeley.edu/~storn/code.html

Seznam všech publikací autora

[A1] Cvejn, J.: Optimální řízení robotů v prostoru s překážkami. Diplomová práce, VŠST v Liberci, Fakulta strojní, 1996

[A2] Cvejn, J.: Metoda optimálního řízení dynamických systémů. Aplikace pro řízení průmyslových robotů v prostoru s překážkami. Disertační práce, Technická univerzita v Liberci, Fakulta mechatroniky a mezioborových inženýrských studií, 2000

[A3] Cvejn, J.: Wonderware InControl - software pro přímé řízení pomocí PC. Automatizace 10/97

[A4] Cvejn, J., Červenka, Z.: Využití operačních systémů MS Windows NT a Windows CE pro průmyslové řízení v reálném čase. Automa 9-10/98

[A5] Cvejn, J.: An algorithm of optimal cutting glass raw material. Electronics, Control, Measurements and Signals. Liberec, červen 1999.

[A6] Cvejn, J.: Optimal Control of Non-linear Systems with Real Constraints. 13th International Conference on Process Control, Štrbské Pleso, 2001, pp.42. ISBN 80-227-1542-5

[A7] Cvejn, J.: A Method of Solving for the Optimal Trajectories of Dynamic Systems using Splines. 16th International Conference on Production Research, Prague, July 2001, vol. 3+4, pp.206. ISBN 80-02-01438-3

[A8] Cvejn, J.: The finite elements approach to the optimal process control. 5-th International Scientific-Technical Conference on Process Control Říp 2002. Kouty nad Desnou, June 2002. pp.83. ISBN 80-7149-452-1

[A9] Cvejn, J.: The optimal control of industrial robots in space with obstacles. 5-th International Scientific-Technical Conference on Process Control Říp 2002. Kouty nad Desnou, June 2002. pp.84. ISBN 80-7149-452-1

[A10] Cvejn, J.: The optimal movement of industrial robots in space with constraints and obstacles. 12-th International Conference on Flexible Automation and Intelligent Manufacturing, Dresden, Germany 2002. pp 951. ISBN 3-486-27036-2

[A11] Cvejn, J.: Numerical Method of Optimal Control Using Compact Support Bases. 2nd IFAC Conference Control System Design '03. Bratislava, 2003. ISBN 0-08-044175-0 (Elsevier, 2004)

[A12] Cvejn, J.: Two Unconventional Numerical Methods of Setting up Optimal Regulator for Continuous Linear Systems. 14th International Conference on Process Control. Štrbské Pleso, June 2003, pp.147. ISBN 80-227-1902-1

[A13] Cvejn, J.: Fast Algorithm of Detecting Collisions. 14th International Conference on Process Control. Štrbské Pleso, June 2003, pp.135. ISBN 80-227-1902-1

[A14] Cvejn, J.: Distributed Computation under Microsoft Windows NT. ECMS 2003. 6th International Workshop on Electronics, Control, Measurement and Signals. Liberec, 2003. pp.195. ISBN 80-7083-708-X

[A15] Cvejn, J.: Fast Algorithm of Detecting Collisions. ECMS 2003. 6th International Workshop on Electronics, Control, Measurement and Signals. Liberec, 2003. pp. 190. ISBN 80-7083-708-X

[A16] Cvejn, J.: Software pro optimální výrobu rovinných tvarů ze skla. XI. mezinárodní konference Sklářské stroje. Liberec, 2003. str.111. ISBN 80-7083-732-2

[A17] Cvejn, J.: Numerical Determination of Optimal Regulator for Continuous Linear Systems Using direct Minimization. 6-th International Scientific-Technical Conference Process Control 2004. Kouty nad Desnou, June 2004. pp.62. ISBN 80-7194-662-1

[A18] Cvejn, J.: Numerical Determination of LQ-optimal Regulator for Constrained Discrete Systems. 6-th International Scientific-Technical Conference Process Control 2004. Kouty nad Desnou, June 2004. pp.61. ISBN 80-7194-662-1

[A19] Cvejn, J.: Effective Way of Overriding C++ Operators for Matrix Operations. Acta Electrotechnica et Informatica No.2, Vol.4, 2004. ISSN 1335-8243

[A20] Cvejn, J.: Numerical Determination of a Quadratic-optimal Regulator for Linear Systems with Constraints. 10th IEEE International Conference on Methods and Models in Automation and Robotics. Miezyzdroje, Poland, 2004. ISBN 83-88764-04-7

[A21] Cvejn, J.: Optimizing the Trajectories of Robotic Systems in the Space with Obstacles. 10th IEEE International Conference on Methods and Models in Automation and Robotics. Miezyzdroje, Poland, 2004. ISBN 83-88764-04-7

[A22] Cvejn, J.: Výpočet kvadraticky optimálního regulátoru pro lineární systém s omezeními. Automatizace č.6, červen 2004. ISSN 0005-125X

[A23] Cvejn, J.: Efektivní způsob implementace operátorů v jazyku C++ pro třídy s dynamickou alokací dat. K7 vědecko populární časopis Fakulty mechatroniky TU v Liberci, č.1, 2004. ISSN 1214-7370.

[A24] (Cvejn, J.: Numerical Determination of Quadratic-Optimal Regulator for Linear Systems with Constraints. Archives of Control Sciences. ISSN 0004-072X. Přijato k tisku v r. 2005)

[A25] (Cvejn, J.: Numerical Determination of a Continuous Constrained LQ-Regulator. WSEAS Transactions on Systems. ISSN 1109-2777. Přijato k tisku v r. 2005)