SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ
TECHNOLÓGIE

Radlinského 9, 812 37 Bratislava
Slovenská Republika

KATEDRA INFORMATIZÁCIE A RIADENIA PROCESOV

Bc. Štefan Ševčík

Dynamická optimalizácia procesov

Diplomová práca

Školiteľ: doc. Dr. Ing. Miroslav Fikar
Konzultant: Ing. Michal Čižniar
Bratislava 2006

SLOVENSKÁ TECHNICKÁ UNIVERZITA
v Bratislave
Slovak University of Technology in Bratislava

Fakulta chemickej a potravinárskej technológie
Radlinského 9, 812 37 Bratislava 1

Ústav: **Ústav informatizácie, automatizácie a matematiky**

Oddelenie: **Informatizácie a riadenia procesov**

Číslo: 5 /ÚIAM/2006

Vec: **Zadanie diplomovej práce**

Meno a priezvisko študenta: **Bc. Štefan Ševčík**

Meno a priezvisko vedúceho diplomovej práce: **doc. Dr. Ing. Miroslav Fikar**

Meno a priezvisko konzultanta diplomovej práce: Ing. Michal Čižniar

Názov diplomovej práce:

Dynamická optimalizácia procesov

Návrh dynamickej optimalizácie procesu chemickej technológie metódou ortogonálnej kolokácie na konečných prvkoch. Vytvorenie balíka v jazyku C založenom na kóde dynopt implementovaného v MATLABe.

Termín odovzdania diplomovej práce: **19.5.2006**

Diplomová práca sa odovzdáva v 3 exemplároch vedúcemu ústavu – oddelenia.

Bratislava, 20. februára 2006

doc. Ing. Dr. Miroslav Fikar
riaditeľ ústavu

Slovenská technická univerzita v Bratislave Fakulta chemickej a potravinárskej technológie dekanát - 5 -

prof. Ing. Dušan Bakoš, DrSc.
dekan

Chcel by som poďakovať vedúcemu Oddelenia informatizácie a riadenia procesov Fakulty chemickej a potravinárskej technológie STU v Bratislave a zároveň aj vedúcemu môjho projektu Doc. Dr. Ing. Miroslavovi Fikarovi za všetku pomoc, podporu a vedenie počas štúdia. Zároveň chcem poďakovať Ing. Michalovi Čižiarovi taktiež z Oddelenia informatizácie a riadenia procesov Fakulty chemickej a potravinárskej technológie STU v Bratislave za jeho trpezlivosť a ochotu pri tvorbe tejto práce.

Bratislava, 2006
Štefan Ševčík

Abstrakt

Táto práca sa zaoberá dynamickou optimalizáciou procesov. Pozostáva v hľadaní optimálnych profilov riadiacich a stavových veličín, ktoré optimalizujú danú účelovú funkciu vzhľadom k daným obmedzeniam. Bola vyvinutá metóda ortogonálnej kolokácie na konečných prvkoch, ktorá bola implementovaná v programovacom jazyku C. Pôvodné problémy dynamickej optimalizácie sú tak konvertované do NLP problémov, pre ktoré možno použiť ľubovoľný program na riešenie úlohy nelineárneho programovania.

Gradienty účelovej funkcie ako aj obmedzení potrebné pre NLP solver sú vypočítané analyticky.

Abstract

This work deals with dynamic optimization of processes. It consists in searching for optimal control and state profiles which optimize a given performance index under specified constraints. The method of orthogonal collocations on finite elements has been developed and implemented using C language. The original dynamic optimization problems are then converted into NLP problems which are solved using appropriate NLP solvers.

The gradients of the performance index as well as of the constraints needed in the NLP solver are analytically computed using formal calculus.

Obsah

Zoznam obrázkov

Zoznam použitých skratiek

NLP	Nonlinear programming – Nelineárne programovanie
ODE	Ordinary differential equations – Jednoduché diferenciálne rovnice
DAE	Differential algebraic equations – Diferenciálno algebraické rovnice
SQP	Successive quadratic programming – Postupné kvadratické programovanie
DP	Dynamic programming – Dynamické programovanie
PMP	Pontryagin's minimum principle – Pontrjaginov princíp minima
TPBVP	Two point boundary value problem – Problém dvoch hraníc
CVP	Control vector parameterization – Parametrizácia riadiaceho vektora
IDP	Iterative dynamic programming – Iteratívne dynamické programovanie

Kapitola 1

Úvod

Mohutný rozvoj informačných technológií v súčasnosti vytvára nové možnosti pre rýchle, presné a pohodlné riešenie úloh, ktoré sa museli v minulosti pracne a zdĺhavo vypracovávať. K týmto technológiám patrí aj optimálne riadenie dynamických systémov. Optimalizačné problémy sú všadeprítomné v matematickom modelovaní reálnych systémov a pokrývajú veľmi široké spektrum aplikácii. Tieto aplikácie sa objavujú vo všetkých odvetviach inžinierstva, počítačovej vedy, ekonomike, financiách, operačný výskum a veda manažmentu, chémii, materiálna veda, astronómia, fyzika, štruktúrna a molekulárna biológia a v medicíne.

Po druhej svetovej vojne nastal masívny rozvoj teórii a techník v optimalizácii po všetkých stránkach. Tento rozvoj zahŕňa: kombinatorická optimalizácia (Combinatorial Optimisation), doplnkové a variačné nerovnosti (Complementary and Variational Inequalities), obmedzené logické programovanie (Constraint Logic Programming), konvexná optimalizácia (Convex Optimisation), nehladká optimalizácia (Nonsmooth Optimisation), deterministická globálna optimalizácia (Deterministic Global Optimisation), stochastická globálna optimalizácia (Stochastic Global Optimisation), cielene programovanie (Goal Programming), viacúčelová optimalizácia (Multi–Objective Optimisation), polo–obmedzené programovanie (Semi–Infinite Programming), dvojstupňová a viacstupňová lineárna a nelineárna optimalizácia (Bilevel and Multilevel Linear and Nonlinear Optimisation), nelineárna optimalizácia s obmedzeniami a bez obmedzení (Nonlinear Unconstrained and Constrained Optimisation), kužeľová optimalizácia (Cone Programming), dynamické programovanie (Dynamic Programming), všeobecné geometrické programovanie (Generalized Geometric Programming), teória hier (Game Theory), intervalová analýza (Interval Analysis), finančná optimalizácia (Financial Optimisation), paralelná optimalizácia (Parallel Optimisation), dynamická optimalizácia (Dynamic Optimisation), optimálne riadenie (Optimal Control), zmiešaná–celočíselná nelineárna optimalizácia (Mixed–integer Nonlinear Optimisation), celočíselné programovanie (Integer Programming), lineárne programovanie (Linear Programming), semidefinitné programovanie (Semidefinite Programming), sieťová optimalizácia (Network Optimisation), robustná optimalizácia (Robust Optimisation), stochastické programovanie (Stochastic Programming), plánovanie, logistika, telekomunikácie, modelovanie systémov (Modeling Systems).

Avšak, riešenie optimalizačných problémov s diferenciálnymi a algebraickými rovnicami (DAE) stále zostáva zložitý problém. V súčasnosti, optimalizačné problémy s nelineárnymi algebraickými rovnicami môžu byť riešené priamo ako nelineárne programovanie (NLP). Na druhej strane, problémy bez obmedzení s diferenciálnymi

rovnicami môžu byť riešené variačným počtom. Avšak, modely, ktoré sú kombináciou obidvoch typov sú optimalizované určitým stupňom aproximácie tohto problému.

Diferenciálne a algebraické rovnice možno optimalizovať postupne rôznymi metódami. Postupné kvadratické programovanie (SQP) (Han, 1977; Powell, 1977) a redukované gradientové techniky (Murtagh a Saunders, 1978) sú dostupné pre narábanie s nelineárnymi algebraickými úlohami, pokým variačný počet (Bryson a Ho, 1975) môže byť použitý pre optimalizáciu diferenciálnych rovníc.

Štandardné metódy nelineárneho programovania (NLP) nemôžu byť použité bez využitia často zložitých numerických integračných schém. Navyše, existuje veľa alternatív (Sargent a Sullivan, 1977), ako aj v tomto prípade, pre obmedzenia nerovnosti na profile spojitých stavových premenných a pre narábanie s nespojitými riadiacimi profilmi. Metódy založené na teórii optimálneho riadenia, protikladne, ťažko narábajú aj s jednoduchými typmi algebraických rovníc. Ďalej, stredno-rozmerné (tri alebo viac stavových rovníc) nelineárne problémy s obmedzeniami, napríklad, zložené z nelineárnych diferenciálnych rovníc, môžu byť ťažko riešiteľné numerickou integráciou. Ďalšie komplexnosti ako nespojité profily, obmedzenia, a iné algebraické podmienky môžu spôsobiť, že problém sa stáva neriešiteľný použitím variačných výpočtov.

Mnoho chemicko – inžinierskych problémov vyžaduje optimalizáciu systému diferenciálnych a algebraických rovníc. Metóda prezentovaná v tejto práci rieši za istých podmienok, diferenciálne úlohy účinne a presne. Diferenciálne rovnice sú diskretizované použitím polynomickej aproximácie a ortogonálnej kolokácie na algebraické rovnice s neznámymi koeficientmi. Výsledné algebraické rovnice sú potom súčasťou nelineárneho programovania (NLP), ktoré sú riešené niektorou z optimalizačných techník.

Kapitola 2 pojednáva o dynamickej optimalizácii vo všeobecnosti. Kapitola začína definíciami niekoľkých optimalizačných problémov. V druhej časti je opísaných niekoľko prístupov k riešeniu optimalizačných problémov.

Kapitola 3 definuje formuláciu problému nelineárneho programovania.

Kapitola 4 popisuje funkciu *dynopt*, ako hlavnú funkciu kolekcie funkcií vytvorených v programovacom jazyku C, vrátane NLP solvera. Kapitola opisuje použitie funkcie *dynopt*, vrátane užívateľských funkcií pre zadanie procesu vo forme diferenciálnych rovníc, účelovú funkciu a obmedzenia typu rovností a nerovností ako aj popis vstupných a výstupných argumentov funkcie *dynopt*. Kapitola končí ukážkou použitia funkcie *dynopt* na jednoduchom príklade.

Kapitola 5 prezentuje a rozoberá problémy dynamickej optimalizácie z literatúry riešené pomocou *dynoptu*.

Kapitola 6 prezentuje závery a výsledky tejto práce.

Kapitola 2

Dynamická optimalizácia

Táto kapitola sa zaoberá dynamickou optimalizáciou vo všeobecnosti. Kapitola začína s definovaním niekoľkých problémov dynamickej optimalizácie. V druhej časti je opísaných niekoľko postupov pre hľadanie riešenia týchto problémov dynamickej optimalizácie. Nakoniec, táto časť končí formuláciou NLP problému.

2.1 Vyjadrenie optimalizačného problému

Cieľom dynamickej optimalizácie je určiť, v otvorenej riadiacej slučke, súbor dôležitých premenlivých časových profilov (tlak, teplota, prietok, elektrický prúd, tepelný výkon, ...) pre dynamické systémy, ktoré optimalizujú danú účelovú funkciu (hodnotu funkcionálu, optimalizačné kritérium) (cena, čas, energia selektivita, ...) vzhľadom na špecifické obmedzenia (bezpečnosť, environmentálne a operačné obmedzenia). Optimálne riadenie hovorí o určovaní najlepších časovo premenných profilov v uzavretej riadiacej slučke.

2.1.1 Účelová funkcia

Účelová funkcia (optimalizačné kritérium, hodnota funkcionálu), môže byť vo všeobecnosti napísaná v jednej z troch foriem nasledovne:

Bolzov tvar

$$J(u(t)) = G(x(t_f), t_f) + \int_{t_0}^{t_f} F(x(t), u(t), t)dt \tag{2.1}$$

Lagrangeov tvar

$$J(u(t)) = \int_{t_0}^{t_f} F(x(t), u(t), t)dt \tag{2.2}$$

Mayerov tvar

$$J(u(t)) = G(x(t_f), t_f) \tag{2.3}$$

kde

J – reprezentuje optimalizačné kritérium

G – reprezentuje časť účelovej funkcie vyčíslenej v koncových podmienkach

$\int_{t_0}^{t_f} F dt$ – reprezentuje časť účelovej funkcie vyčíslenej počas časového intervalu

$x(t)$ – reprezentuje vektor stavového profilu

$u(t)$ – reprezentuje vektor riadiaceho profilu

2.1.2 Opis modelu procesu

Správanie sa mnohých procesov možno vo všeobecnosti opísať súborom normálnych diferenciálnych rovníc (ODE) nasledovne:

$$\dot{x}(t) = f(x(t), u(t), t), \quad x(t_0) = x_0 \qquad t_0 \le t \le t_f \tag{2.4}$$

Tento systém ODE vytvára obmedzenia typu rovnosti v probléme optimálneho riadenia.

2.1.3 Obmedzenia

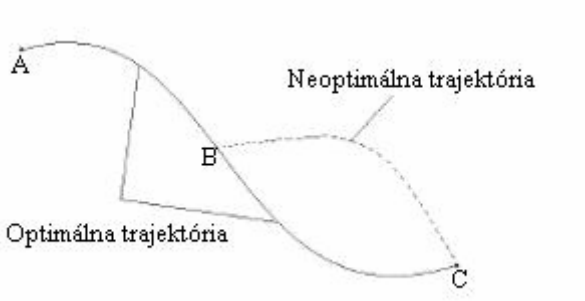
Obmedzenia, ktoré musia byť započítané, typicky zahrňujú rovnosť a nerovnosť nekonečne rozmerných, vnútorných a konečných obmedzení. Naviac, môžu byť napísané v nasledujúcej kanonickej forme, podobnej účelovej funkcii (2.1):

$$J_i(u(t)) = G_i(x(t_i), t_i) + \int_{t_0}^{t_i} F_i(x(t), u(t), t) dt \tag{2.5}$$

kde $t_i \le t_f, i = 1, ..., nc,$ a nc je počet obmedzení.

2.2 Riešenie problémov optimálneho riadenia

Existuje viacero prístupov, ktoré dokážu riešiť problémy optimálneho riadenia. Tieto môžu byť delené do analytických metód, ktoré boli dodnes používané a do numerických metód, ktoré sú v súčasnosti viac preferované.



Obr. 2.2.1 Grafická interpretácia Bellmanovho princípu optimality

2.2.1 Analytické metódy

Na riešenie problémov optimálneho riadenia možno použiť celú sériu metód. Najdôležitejšie z nich sú:

- Dynamické programovanie (Bellmanov princíp optimality)
- Pontrjaginov princíp minima (maxima)
- Variačný počet

2.2.1.1 Dynamické programovanie

Dynamické programovanie (DP) je veľmi všeobecná metóda pre riešenie rôznych optimalizačných problémov, často používaných pri analýze a návrhu automatických riadiacich systémov. Tato metóda je založená na princípe optimality, ktorý prvýkrát formuloval Bellman.

Bellmanov princíp optimality môžeme jednoducho formulovať nasledovne: “Ak existuje optimálna cesta z A do C, každá čiastková cesta z B do C je tiež optimálna”.

Uvažujme nasledujúci problém optimálneho riadenia v Bolzovom tvare:

$$J(u(t)) = G(x(t_f), t_f) + \int_{t_0}^{t_f} F(x(t), u(t), t)dt \qquad (2.6a)$$

$$\dot{x}(t) = f(x(t), u(t), t) \quad x(t_0) = x_0 \qquad (2.6b)$$

Predpokladá sa, že problém optimálneho riadenia (2.6) má riešenie. Uvažujme nasledujúcu funkciu, taktiež nazývanú Bellmanova funkcia, definovanú nasledovne:

$$v(x(t),t) = \min_{u(t)} \left[G(x(t_f),t_f) + \int_{t_0}^{t_f} F(x(t),u(t),t)dt \right] \tag{2.7}$$

Diferencovanie (2.7) vedie k Bellmanovej parciálnej diferenciálnej rovnici:

$$-\frac{\partial v}{\partial t} = \min_{u(t)} \left[F(x,u,t) + \left(\frac{\partial v}{\partial x}\right)^T f(x,u,t) \right] \tag{2.8}$$

ktorá musí vyhovovať hraničnej podmienke:

$$v(x_f,t_f) = G(x_f,t_f) \tag{2.9}$$

Bellmanova parciálna diferenciálna rovnica (2.8) spolu s hraničnou podmienkou (2.9) reprezentujú nevyhnutné podmienky pre získanie minima problému optimálneho riadenia (2.6).

Substitúciou za optimálnu riadiacu veličinu u^* do Bellmanovej parciálnej diferenciálnej rovnice (2.8) vedie k formulácii rovnice, ktorá sa tiež nazýva Hamilton–Jacobi–Bellmanova parciálna diferenciálna rovnica:

$$-\frac{\partial v}{\partial t} = F(x,u^*,t) + \left(\frac{\partial v}{\partial x}\right)^T f(x,u^*,t) \tag{2.10}$$

Pri riešení problému optimálneho riadenia je vhodné definovať Hamiltonovu funkciu nasledovne:

$$H\left(x,u,\frac{\partial v}{\partial x},t\right) = F(x,u,t) + \left(\frac{\partial v}{\partial x}\right)^T f(x,u,t) \tag{2.11}$$

a po substitúcií (2.11) do (2.8), Bellmanova parciálna diferenciálna rovnica nadobudne tvar:

$$-\frac{\partial v}{\partial t} = \min_{u(t)} H\left(x,u,\frac{\partial v}{\partial x},t\right) \tag{2.12}$$

2.2.1.2 Pontrjaginov princíp minima

Ďalší veľmi efektívny prístup k riešeniu problémov optimálneho riadenia je opísaný pomocou Pontrjaginovho princípu minima (PMP). Medzi dynamickým programovaním (DP) a Pontrjaginovým princípom minima (PMP) je veľmi blízky vzťah, ktorý uvažuje dva absolútne rôzne prístupy k riešeniu problémov optimálneho riadenia.

Uvažujme už spomínaný problém optimálneho riadenia (2.6) a označme parciálnu deriváciu $\frac{\partial \nu}{\partial x}$ ako konjugovanú premennú $\lambda(t)$. Vhodná Hamiltonová funkcia nadobúda tvar:

$$H(x,u,p,t) = F(x,u,t) + \lambda^T f(x,u,t) \tag{2.13}$$

Bellmanová parciálna diferenciálna rovnica (2.12) nadobúda po substitúcii nasledovný tvar:

$$-\frac{\partial \nu}{\partial t} = \min_{u(t)} H(x,u,\lambda,t) \tag{2.14}$$

Diferencovaním ľavej a pravej strany výrazu $\frac{\partial \nu}{\partial x} = \lambda(t)$ podľa x oddelene dostaneme:

$$-\frac{\partial^2 \nu}{\partial x \partial t} = \frac{\partial H}{\partial x} + \frac{\partial^2 \nu}{\partial x^2}\frac{\partial H}{\partial \lambda} \tag{2.15a}$$

$$\dot{\lambda} = \frac{\partial^2 \nu}{\partial x^2}\dot{x} + \frac{\partial^2 \nu}{\partial t \partial x} \tag{2.15b}$$

Z uvedeného vyplývajú kanonické diferenciálne rovnice, reprezentujúce princíp minima:

$$\dot{x} = \frac{\partial H}{\partial \lambda} \tag{2.16a}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \tag{2.16b}$$

Nevyhnutné podmienky pre problém optimálneho riadenia (2.6) využívajúc Pontrjaginov princíp minima, môžu byť formulované nasledovne:

- podmienka optimality pre riadiacu veličinu:

$$0 = \frac{\partial H}{\partial u^T}, \qquad \forall t \in \left[t_0, t_f\right] \tag{2.17a}$$

- definovanie konjugovaných premenných

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x^T}, \quad \forall t \in \left[t_0, t_f\right] \tag{2.17b}$$

- konečné podmienky pre konjugované premenné

$$\lambda^T\left(t_f\right) = \frac{\partial G}{\partial x^T}|_{t_f} \tag{2.17c}$$

2.2.1.3 Variačný počet

Dynamické programovanie a Pontrjaginov princip minima sú viac všeobecnejšie a efektívnejšie ako klasický variačný počet. Elementárne vzťahy variačného počtu sú získané z Bellmanovej parciálnej diferenciálnej rovnice.

Klasický problém variačného počtu pre obmedzenie (2.4) vyplýva z Euler–Lagrangeovej diferenciálnej rovnice:

$$\frac{\partial \Gamma}{\partial x} - \frac{d}{dt}\left(\frac{\partial \Gamma}{\partial \dot{x}}\right) = 0 \tag{2.18}$$

kde Γ je Lagrangeova funkcia definovaná nasledovne:

$$\Gamma\left(x, \dot{x}, u, \lambda, t\right) = F\left(x, u, t\right) + \lambda^T\left[f\left(x, u, t\right) - \dot{x}\right] \tag{2.19}$$

Nevyhnutné podmienky pre problém optimálneho riadenia (2.6) využívajúc Euler–Lagrangeove diferenciálne rovnice, môžu byť formulované nasledovne:

- podmienka optimality pre riadiacu veličinu:

$$0 = \frac{\partial \Gamma}{\partial u^T}, \qquad \forall t \in \left[t_0, t_f\right] \tag{2.20a}$$

- definovanie konjugovaných premenných

$$\dot{\lambda}^T = -\frac{\partial \Gamma}{\partial x^T}, \quad \forall t \in \left[t_0, t_f\right] \tag{2.20b}$$

- konečné podmienky pre konjugované premenné

$$\lambda^T\left(t_f\right) = \frac{\partial G}{\partial x^T}|_{t_f} \tag{2.20c}$$

2.2.2 Numerické metódy

Podmienkami optimality spomenutými v sekcii 2.2.1 vo všeobecnom prípade nie sme schopní pre výsledný problém dvoch hraníc (Two Point Boundary Value Problem - TPBVP) zabezpečiť optimum, pretože tento problém alebo Bellmanova parciálna diferenciálna rovnica je ťažko riešiteľná. Počítačové metódy sú preto potrebné.

Predpokladáme, bez straty na všeobecnosti, že ODE systém je len jeden, pevné počiatočné podmienky existujú a účelová funkcia je v Mayerovom tvare. Inak, je jednoduché preformulovať problém do tejto formy. Problém môže byť riešený buď variačnými prístupmi alebo prijatím nejakého stupňa diskretizácie, ktoré konvertujú originálny spojitý problém na diskrétny problém. Skoršie stratégie na hľadanie riešenia známe ako nepriame metódy, boli zamerané na riešení klasických variačných podmienkach pre optimálnosť. Na druhej strane metódy, ktoré diskretizujú pôvodnú spojitú formuláciu, môžu byť rozdelené do dvoch kategórii podľa stupňa diskretizácie. Tu musíme rozlišovať medzi metódami ktoré diskretizujú iba vektor riadenia (čiastočná diskretizácia) a tie, ktoré diskretizujú stavový aj riadiaci profil (plná diskretizácia). V podstate čiastočne diskretizované problémy môžu byť riešené buď dynamickým programovaním alebo stratégiou nelineárneho programovania (NLP). Základnou charakteristikou týchto metód je to, že realizovateľné riešenie ODE systému, pre dané riadiace veličiny, je získané integráciou v každej iterácii NLP solvera. Hlavná výhoda tohto prístupu je generovanie menšieho diskrétneho problému ako u metód plnej diskretizácie.

Metódy, ktoré plne diskretizujú spojité problémy, taktiež prijímajú NLP stratégie na riešenie diskrétnych systémov a sú známe ako priamo – simultánne metódy. Tieto metódy môžu využívať rôzne NLP a diskretizačné techniky, ale základná charakteristika je to že riešia ODE systémy iba raz a to v optime. Navyše majú vyššiu stabilitu ako metódy čiastočnej diskretizácie, špeciálne pri existencii nestabilných dynamických modelov. Na druhej strane diskrétny problém je rozsiahlejší a vyžaduje veľkorozmerný NLP solver.

Ako už bolo spomenuté, numerické metódy môžeme rozdeliť do dvoch kategórií:

- **Nepriame metódy**
- **Priame metódy**

2.2.2.1 Nepriame metódy

Tieto metódy sú založené na podmienkach optimality zo state 2.2.1.

2.2.2.1.1 Iterácia hraničnej podmienky

Táto metóda sa snaží nájsť chýbajúcu hraničnú podmienku minimalizovaním chyby medzi hraničnými podmienkami, tak že rovnice (2.17b) a (2.4) môžu byť integrované priamo.

2.2.2.1.2 Iterácia vektoru riadenia

Stavové rovnice (2.4) sú integrované priamo s odhadom (nástrelom) riadiacich veličín a konjugované rovnice (2.17b) sú integrované spätne. Rovnice (2.17a) sú použité na výpočet odhadovaných (nastreľovaných) riadiacich veličín v každej iterácií a slúžia aj ako kritérium pre ukončenie iterácií. Hlavná myšlienka tejto metódy je spätná integrácia konjugovaných rovníc t.j. v smere ich hraničných podmienok. Avšak, tento prístup ma pomalú konvergenciu pre mnoho problémov.

2.2.2.2 Priame metódy

2.2.2.2.1 Sekvenčné metódy

S pomocou čiastočných diskretizačných metód (tiež nazývaných sekvenčné metódy alebo parametrizácia riadiaceho vektora – CVP) iba riadiace premenné sú disketizované. S danými počiatočnými podmienkami a parametrami riadenia ODE systém je riešený s diferenciálno – algebraickým solverom v každej iterácii. Týmto získame hodnotu účelovej funkcie, ktorú využíva NLP solver na nájdenie optimálnych parametrov v parametrizácii riadenia.

Sekvenčná metóda je spoľahlivá, keď je systém stabilný. Ak to nie je tento prípad, hľadanie realizovateľného riešenia pre sústavu riadiacich parametrov môže byť náročné. Časový horizont je rozdelený na časové úseky a v každom úseku sú riadiace premenné reprezentované ako po častiach konštantné, po častiach lineárne alebo polynomický aproximované (Feehery & Barton, 1998; Vassiliadis, 1993).

2.2.2.2.2 Simultánne metódy

V tejto formulácii spojitý problém je konvertovaný do NLP formulácie aproximovaním profilov riadenia aj stavov pomocou polynómov na konečných elementoch (prvkoch, úsekoch). Rôzne polynomické reprezentácie sú použité v literatúre, vrátane Lagrangeových interpolačných polynómov pre diferenciálne a algebraické profily (Cuthrell & Biegler, 1987). Betts (2001) využíva Hermite – Simpsonovu kolokáciu, kým Cervantes and Biegler (1998) a Tanartkit and Biegler (1995) používajú monomickú základnú reprezentáciu (Bader & Ascher, 1987) pre diferenciálne profily. Všetky tieto reprezentácie pochádzajú z Runge – Kutta formulácie a monomická reprezentácia je doporučená pre malý počet podmienok a vyššiu presnosť. Na druhej strane, riadiace a algebraické profily sú aproximované použitím Lagrangeových polynómov.

Diskretizácia použitím kolokačných formulácii vedie k veľkým NLP problémom, ktoré môžu byť efektívne riešené použitým veľkorozmerným NLP solverom. Biegler et al. (2002) dáva prehľad metód dynamickej optimalizácie s využitým simultánnych metód. Tieto metódy ponúkajú niekoľko výhod pre riešenie problémov dynamickej optimalizácie

Dynamická optimalizácia s použitým kolokačných metód bola aplikovaná na niekoľkých problémoch vrátane optimalizácie vsádzkových procesov (Bhatia & Biegler, 1996), modelov nelineárneho prediktívneho riadenia (Albuquerque et al., 1997) a na návrh reaktorov a syntézu (Lakshmanan, Rooney, & Biegler, 2000; Ierapetritou, 2001).

Metóda ortogonálnej kolokácie na konečných prvkoch bola použitá v tejto práci na získanie optimálnych profilov riadenia a stavov a podrobne je opísaná v nasledujúcej kapitole.

Kapitola 3

Formulácia nelineárneho programovania

V tejto práci je problém optimálneho riadenia riešený cestou kompletnej parametrizácie vektora riadenia a vektora stavov. To značí, že východzia spojitá trajektória riadenia a stavov je aproximovaná lineárnou kombináciou nejakých základných funkcií. Tu sa predpokladá, že základne funkcie sú známe a optimalizované sú koeficienty lineárnych kombinácii týchto funkcií. Navyše, každý segment sekvencie riadenia je definovaný na časovom intervale, ktorého dĺžka sama o sebe môže byť predmetom optimalizácie.

Predpokladá sa, že optimalizovaný dynamický model je opísaný systémom jednoduchých diferenciálnych rovníc.

Formulujme nasledovný základný problém pre $t \in \lfloor 0, t_f \rfloor$

$$\min_{u(t)} J\lfloor \Phi(t_f) \rfloor \tag{3.1}$$

tak, že platí:

$$\begin{aligned} \dot{x}(t) = f[t, x(t), u(t)], \quad x(0) = x_0 \\ h[t, x(t), u(t)] = 0 \\ g[t, x(t), u(t)] \le 0 \\ x(t)^L \le x(t) \le x(t)^U \\ u(t)^L \le u(t) \le u(t)^U \end{aligned}$$

kde

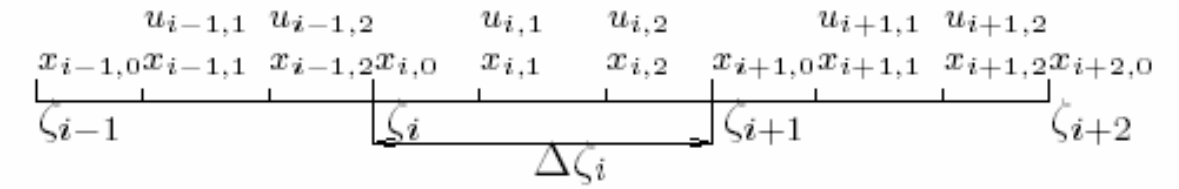
$J\lfloor \Phi(t_f) \rfloor$ – účelová funkcia, vypočítaná v konečných podmienkach,

f – vektorová funkcia opisujúca proces (pravé strany diferenciálnych rovníc),

h – vektor obmedzení rovnosti,

g – vektor obmedzení nerovnosti,

$x(t)$ – vektor stavových veličín,



Obr. 3.1.1: Kolokácia na konečných prvkoch

$u(t)$ – vektor riadiacich veličín,

x_0 – počiatočné podmienky pre stavový vektor,

$x(t)^L, x(t)^U$ – obmedzenia stavového profilu,

$u(t)^L, u(t)^U$ – obmedzenia riadiaceho profilu.

Vo formulácii nelineárneho programovania (NLP) sa použitím kolokácie na konečných prvkoch konvertujú diferenciálne rovnice na algebraické rovnice. A to tak, že na diskretizáciu diferenciálnych rovníc je použitá polynomická aproximácia a na vytvorenie základných rovníc, ktoré sú ďalej riešené ako sústava algebraických rovníc, je použitá ortogonálna kolokácia na konečných prvkoch. Tieto základne rovnice sa potom hodnotia v časoch zodpovedajúcim posunutým koreňom Legendrovho polynómu.

Procedúra je potom nasledovná: Uvažujme počiatočné hodnoty počas konečného elementu i v čase $t \in [\zeta_i, \zeta_{i+1}]$:

$$\dot{x} = f(t, x(t), u(t)) \qquad t \in \lfloor t_0, t_f \rfloor \tag{3.2}$$

pre stavový profil $x(t)$ a riadiaci profil $u(t)$. Na aproximáciu pre konečný prvok $i, \zeta_i \le t \le \zeta_{i+1}$ definujeme Lagrangeove polynómy:

$$X_{N+1}(t) = \sum_{j=0}^{N} x_{ij}\phi_j(t); \qquad \phi_j(t) = \prod_{k=0,j}^{N} \frac{(t - t_{ik})}{(t_{ij} - t_{ik})} \tag{3.3}$$

na elemente $i \quad i = 1,..., NE$

$$U_M(t) = \sum_{j=1}^{M} u_{ij}\theta_j(t); \qquad \theta_j(t) = \prod_{k=1,j}^{M} \frac{(t - t_{ik})}{(t_{ij} - t_{ik})} \tag{3.4}$$

na elemente $i \quad i = 1,..., NE$

kde $k = 0, j$ predstavuje $k \neq j$. Taktiež $X_{N+1}(t)$ je polynóm (*N+1*)-teho rádu (stupeň < *N+1*) a $U_M(t)$ je polynóm *M*-teho rádu (stupeň < *M*). Rozdiel v rádoch je zapríčinený existenciou počiatočných podmienok pre $x(t)$, pre každý element i a voľbou počtu kolokačných bodov pre stavové a riadiace veličiny.

Jednou z vlastnosti Lagrangeovho polynómu je (napr. pre $X_{N+1}(t)$)

$$X_{N+1}(t_{ij}) = x_{ij} \tag{3.5}$$

pretože $\phi_k(t_j) = \delta_{kj}$, kde δ_{kj} je Kroneckerov operátor delta. Tento tvar polynómu umožňuje priamo definovať obmedzenia na stavové a riadiace veličiny.

Po použití ortogonálnej kolokácie na konečných prvkoch s N prvkami pre stavové veličiny a *M* prvkami pre riadiace veličiny ako ukazuje obrázok 3.1 (v tomto prípade *N=M*), a po definovaní základných funkcií tak, že tieto sú normalizované na každom prvku $\Delta\zeta_i(\tau \in [0,1])$, možno základné rovnice formulovať nasledovne:

$$\Delta\zeta_i r(t_{ik}) = \sum_{j=0}^{N} x_{ij}\dot{\phi}_j(\tau_k) - \Delta\zeta_i \dot{F}(t_{ik}, x_{ik}, u_{ik}) \tag{3.6}$$
$$i = 1,..., NE$$
$$j = 0,..., N$$
$$k = 1,..., N$$

kde funkcia $\dot{\phi}_j(\tau_k) = d\phi_j / d\tau$, ktorá môže byť vypočítaná off-line. Tiež treba mať na pamäti, že $t_{ij} = \zeta_i + \Delta\zeta_i \tau_k$. Dĺžky konečných intervalov sa používajú k nájdeniu možných bodov nespojitosti riadiaceho profilu a taktiež na ubezpečenie sa, že presnosť integrácie je v číselnej tolerancii. V koncových bodoch je nutná spojitosť stavových profilov, zatiaľ čo riadiaci profil môže byť v týchto bodoch nespojitým, takže dostávame dodatočné rovnosti:

$$X^{i}_{N+1}(\zeta_i) = X^{i-1}_{N+1}(\zeta_i) \tag{3.7}$$
$$i = 2,..., NE$$

alebo

$$x_{i0} = \sum_{j=0}^{N} x_{i-1,j}\phi_j(\tau = 1) \tag{3.8}$$
$$i = 2,..., NE$$

$$j = 0,..., N$$

Tieto body teda zabezpečujú počiatočné podmienky pre stavové profily ďalšieho elementu. Treba mať však na pamäti, že základné funkcie $(\phi(\tau))$ a ich derivácie $(\dot{\phi}(\tau))$ sa počítajú dopredu, potom sú len funkciou umiestnenia koreňov Legendrovho polynómu.

Taktiež je vhodné si uvedomiť, že Lagrangeove polynómy pre riadiacu veličinu používajú *M* koeficientov na každý element a sú iného rádu než Lagrangeove polynómy pre stavovú veličinu. Táto rozdielnosť zapríčiňuje nutnosť počítať funkciu Lagrangeovu funkciu pre riadenie v kolokačných bodoch pre stavové veličiny.

Riadiace veličiny sú ohraničené len v kolokačných bodoch. Obmedzenia pre stavovú veličinu sú vyriešené obmedzením – ohraničením hodnôt každého polynómu na oboch koncoch elementu. To možno dosiahnuť nasledovnými rovnicami:

$$u_i^L \le U_M^i(\zeta_i) \le u_i^U, \quad i = 1,..., NE \tag{3.9}$$

$$u_i^L \le U_M^i(\zeta_{i+1}) \le u_i^U, \quad i = 1,..., NE \tag{3.10}$$

Pretože koeficienty riadiacej veličiny existujú iba v kolokačných bodoch, a teda presadenie týchto ohraničení možno dosiahnuť iba extrapolovaním Lagrangeovho polynómu ku hraničným bodom každého elementu nasledovným spôsobom:

$$U_M^i(\zeta_i) = \sum_{j=1}^{M} u_{ij}\theta_j(\tau = 0) \qquad i = 1,..., NE \tag{3.11}$$

$$U_M^i(\zeta_{i+1}) = \sum_{j=1}^{M} u_{ij}\theta_j(\tau = 1) \qquad i = 1,..., NE \tag{3.12}$$

Dôvodom použitia týchto obmedzení je udržať hodnoty profilu riadiacej veličiny v rozsahu $\left[u_i^L, u_i^U\right]$.

Po použití ortogonálnej kolokácie na konečné prvky nadobúda NLP formulácia nasledovný tvar:

$$\min_{x_{ij}, u_{ij}, \Delta\zeta_i} J(x_f) \tag{3.13}$$

tak, že platí

$$X_{10} - x_0 = 0$$

$$t_f - \sum_{i=1}^{NE} \Delta\zeta_i = 0$$

$$\Delta\zeta_i r(t_{ij}) = \dot{X}_{N+1}(t_{ij}) - \Delta\zeta_i \dot{F}(t_{ij}, x_{ij}, u_{ij}) = 0,$$

$$i = 1,..., NE \qquad j = 1,..., N$$
$$x_{i0} - X_{N+1}^{i-1}(\zeta_i) = 0, \qquad i = 2,..., NE$$
$$x_f - X_{N+1}^{NE}(\zeta_{NE+1}) = 0,$$
$$u_i^L \le U_M^i(\zeta_i) \le u_i^U, \qquad i = 1,..., NE$$
$$u_i^L \le U_M^i(\zeta_{i+1}) \le u_i^U, \quad i = 1,..., NE$$
$$\Delta\zeta_i^L \le \Delta\zeta_i \le \Delta\zeta_i^U, \qquad i = 1,..., NE$$
$$\sum_{i=1}^{NE} \Delta\zeta_i = \zeta_{total}$$
$$h(t_{ij}, x_{ij}, u_{ij}) = 0,$$
$$g(t_{ij}, x_{ij}, u_{ij}) \le 0$$
$$x_{ij}^L \le X_{N+1}(t_{ij}) \le x_{ij}^U, \quad i = 1,..., NE, \qquad j = 0,..., N$$
$$u_{ij}^L \le U_M(t_{ij}) \le u_{ij}^U, \qquad i = 1,..., NE, \qquad j = 1,..., N$$

kde

i – odkazuje na element,

j – odkazuje na kolokačný bod,

$\Delta\zeta_i$ – sú šírky konečných intervalov (elementov) pre $i = 1,..., NE$,

x_f – je hodnota stavovej veličiny v koncovom čase $t = t_f$,

h – obmedzenia rovnosti vyčíslené v čase t_{ij} ,

g – obmedzenia nerovnosti vyčíslené v čase t_{ij} ,

x_{ij}, u_{ij} – koeficienty Lagrangeových funkcií pre stavový a riadiaci profil

Na riešenie takto formulovaného NLP problému možno použiť ľubovoľný program na riešenie úlohy nelineárneho programovania. Metódu je možné rozšíriť pre optimalizáciu časovo nezávislých parametrov, čím v NLP formulácii pribudnú derivácie jednotlivých rovníc podľa parametrov a rozšíri sa vektor optimalizovaných koeficientov o samotné parametre.

Kapitola 4

Funkcia *dynopt*

Táto kapitola je venovaná opisu funkcie *dynopt* ako hlavnej funkcie kolekcie funkcii naprogramovaných v jazyku C pre dynamickú optimalizáciu (ODE systémy). Program bol vytvorený vo vývojovom prostredí *Microsoft Visual C++ 6.0*, ako Win32 konzolová aplikácia, teda nie je spustiteľný v prostredí MS DOS. Súčasťou *dynoptu* je aj solver nelineárneho programovania, ktorý bol taktiež naprogramovaný v C (Pernille Brock, Kaj Madsen, Hans Bruun Nielsen 2004). Primárnou úlohou bolo vytvorenie rozhrania medzi solverom nelineárneho programovania a metódou ortogonálnej kolokácie na konečných prvkoch, ktoré bude ďalej bližšie opísané.

Kapitola začína opisom vstupov a výstupov funkcie *dynopt*, pokračuje opisom funkcie a končí praktickou ukážkou na jednoduchom príklade.

Všetko v tejto kapitole čo sa bude týkať optimalizácie časovo nezávislých parametrov nie je v programe úplne implementované (**parametre sa v programe nedajú použiť i keď budú spomínané**).

4.1 Argumenty funkcie *dynopt*

Deklarácia funkcie *dynopt* v C vyzerá nasledovne:

```
int dynopt(param *pp, double t, double *ksi_s, double *u_s, double *p_s, double *luu, double *lxu,
        void (*process)(double*,double,double*,double*,double*,int),
        void (*objf)(double*,double,double*,double*,double*),
        void (*eqq)(double*,double,double*,double*,double*,int),
        void (*inn)(double*,double,double*,double*,double*,int),
        double eps, int maxfcn);
```

4.1.1 Opis vstupných parametrov

pp smerník na štruktúru *param*, ktorej definícia vyzerá nasledovne:

```
typedef struct
{
        matrix *taux,*tauu,*fx,*fu,*dfx,*source;
        int nint,ncolx,ncolu,nx,nu,eq,in,ob_eq,ob_in,obu,obx,nparam;
        int n,l,leq,iw;
}param;
```

Definícia štruktúry *matrix* je nasledovná:

```
typedef struct
{
        double *mat;
        int rows,cols;
}matrix;
```

a slúži na uloženie čísel matice vo forme vektora.

Užívateľ musí inicializovať položky uvedené v tabuľke pred spustením funkcie *dynopt*. Štruktúry *matrix* a hodnoty *n*, *l*, *leq*, *iw* užívateľ neinicializuje (tieto nastavuje funkcia *dynopt* automaticky).

Položka	Popis
nint	počet intervalov
ncolx	počet kolokačných bodov pre stavové veličiny
ncolu	počet kolokačných bodov pre riadiace veličiny
nx	počet stavových veličín
nu	počet riadiacich veličín
eq	počet obmedzení typu rovnosti, viazané s položkou ***ob_eq***
in	počet obmedzení typu nerovnosti, viazané s položkou ***ob_in***
ob_eq	umiestnenie obmedzení typu rovnosti (1 – obmedzenia platia na začiatku, 2 – obmedzenia platia vo všetkých intervaloch, 3 – obmedzenia platia iba na konci), viazané s položkou ***eq***
ob_in	umiestnenie obmedzení typu nerovnosti (1 – obmedzenia platia na začiatku, 2 – obmedzenia platia vo všetkých intervaloch, 3 – obmedzenia platia iba na konci), viazané s položkou ***in***
obx	prítomnosť obmedzení na stavové veličiny (0 – bez obmedzenia na stavové veličiny, 1 – s obmedzením na stavové veličiny), viazané s parametrom ***lxu*** vo funkcii ***dynopt***
obu	prítomnosť obmedzení na riadiace veličiny (0 – bez obmedzenia na riadiace veličiny, 1 – s obmedzením na riadiace veličiny), viazané s parametrom ***luu*** vo funkcii ***dynopt***
nparam	počet optimalizovaných časovo nezávislých parametrov, viazané s parametrom ***p_s*** vo funkcii ***dynopt***

Tab. 4.1.1 Položky štruktúry *param* vypĺňané užívateľom

t konečný čas, keď nastavené na 0, čas je optimalizovaný.

ksi_s vektor *ksi_s[nint]* počiatočných dĺžok intervalov.

u_s matica *u_s[nint][nu]* počiatočných hodnôt riadiacich veličín pre jednotlivé intervaly.

p_s vektor *p_s[nparam]* počiatočných hodnôt parametrov.

luu matica *luu[2][nu]* obmedzení na riadiace veličiny.

lxu matica *lxu[2][nx]* obmedzení na stavové veličiny.

process smerník na funkciu, v ktorej je nadefinovaní proces s počiatočnými podmienkami a príslušnými deriváciami v tvare:

```
void proces(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
                case 1:
                        sys[i++]=...;   //pravé strany diferenciálnych rovníc
                                        //f1.....fn
                        sys[i++]=...;   //derivácie podľa stavových veličín
                                        //df1/dx1.....df1/dxn
                                        //df2/dx1.....df2/dxn
                                        //.....
                        sys[i++]=...;   //derivácie podľa riadiacich veličín
                                        //df1/du1.....df1/dun
                                        //df2/du1.....df2/dun
                                        //.....
                        sys[i++]=...;   //derivácie podľa času
                                        //df1/dt.....dfn/dt
                        sys[i++]=...;   //derivácie podľa parametrov
                                        //df1/dp1.....df1/dpn
                                        //df2/dp1.....df2/dpn
                                        //.....
                        break;
                case 2:
                        sys[i++]=...;   //počiatočne podmienky
                                        //f1(0).....fn(0)
                        sys[i++]=...;   //derivácie podľa parametrov
                                        //df1(0)/dp1.....df1(0)/dpn
                                        //df2(0)/dp1.....df2(0)/dpn
                                        //.....
                        break;
                default:
```

```
                    break;
        }
}
```

kde do premennej *sys* sa ukladajú údaje ako je vyššie uvedené. Premenná *t* predstavuje čas, *x* vektor stavových veličín, *u* vektor riadiacich veličín a *p* vektor časovo nezávislých optimalizovaných parametrov. Premennú *opt* nastavuje funkcia ktorá volá funkciu *proces*, a preto by nemala byť menená. Funkcia *proces* nemá žiadnu návratovú hodnotu.

objf smerník na funkciu v ktorej je nadefinovaná účelová funkcia s príslušnými deriváciami v tvare:

```
void obj(double *sys, double t, double *x, double *u, double *p)
{
        int i = 0;
        sys[i++] = ...;         //účelová funkcia
                                //J
        sys[i++] = ...;         //derivácie podľa stavových veličín
                                //dJ/dx1.....dJ/dxn
        sys[i++] = ...;         //derivácie podľa riadiacich veličín
                                //dJ/du1.....dJ/dun
        sys[i++] = ...;         //derivácia podľa času
                                //dJ/dt
        sys[i++] = ...;         //derivácie podľa parametrov
                                //dJ/dp1.....dJ/dpn
}
```

kde význam jednotlivých parametrov funkcie *obj* je rovnaký ako vo funkcii *proces*. Funkcia *obj* nemá návratovú hodnotu.

eqq smerník na funkciu v ktorej sú nadefinované obmedzenia typu rovnosti s príslušnými deriváciami v tvare:

```
void econstr(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
                case 1:
                        sys[i++]=...;   //obmedzenia typu rovnosti
                                        //econ1.....econn
                                        //decon1/dx1.....decon1/dxn
                                        //decon2/dx1.....decon2/dxn
                                        //.....
                                        //decon1/du1.....decon1/dun
                                        //decon2/du1.....decon2/dun
                                        //.....
```

```
                                        //decon1/dt.....deconn/dt
                                        //decon1/dp1.....decon1/dpn
                                        //decon2/dp1.....decon2/dpn
                                        //.....
                        break;
                case 2:
                        sys[i++]=...;   //ako v prípade case 1
                        break;
                case 3:
                        sys[i++]=...;   //ako v prípade case 1
                        break;
                default:
                        break;
        }
}
```

kde význam jednotlivých parametrov funkcie *econstr* je rovnaký ako vo funkcii *proces*. Do návestia *case 1* sa vpisujú obmedzenia, ktoré platia na začiatku. Do návestia *case 2* sa vpisujú obmedzenia, ktoré platia vo všetkých časových intervaloch a do návestia *case 3* obmedzenie, ktoré platia na konci. Funkcia *econstr* nemá návratovú hodnotu.

inn smerník na funkciu v ktorej sú nadefinované obmedzenia typu nerovnosti s príslušnými deriváciami v tvare:

```
void inconstr(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
                case 1:
                        sys[i++]=...;   //obmedzenia typu rovnosti
                                        //incon1.....inconn
                                        //dincon1/dx1.....dincon1/dxn
                                        //dincon2/dx1.....dincon2/dxn
                                        //.....
                                        //dincon1/du1.....dincon1/dun
                                        //dincon2/du1.....dincon2/dun
                                        //.....
                                        //dincon1/dt.....dinconn/dt
                                        //dincon1/dp1.....dincon1/dpn
                                        //dincon2/dp1.....dincon2/dpn
                                        //.....
                        break;
                case 2:
                        sys[i++]=...;   //ako v prípade case 1
                        break;
                case 3:
                        sys[i++]=...;   //ako v prípade case 1
```

```
                    break;
                default:
                    break;
            }
        }
```

kde význam jednotlivých parametrov funkcie *inconstr* je rovnaký ako vo funkcii *proces*. Do návestia *case 1* sa vpisujú obmedzenia, ktoré platia na začiatku. Do návestia *case 2* sa vpisujú obmedzenia, ktoré platia vo všetkých časových intervaloch a do návestia *case 3* obmedzenie, ktoré platia na konci. Funkcia *inconstr* nemá návratovú hodnotu.

eps — výpočtová presnosť, zadáva sa ako desatinné číslo, napr.: 0,001 – riešenie s presnosťou na tri desatinné čísla.

maxfcn — maximálny počet vyčíslení účelovej funkcie, po prekročení tejto hodnoty solver skončí s hľadaním riešenia a vráti hodnoty z poslednej iterácie.

4.1.2 Opis výstupov funkcie *dynopt*

Návratová hodnota funkcie *dynopt* je celé číslo a udáva či bolo nájdené optimálne riešenie. Hodnota 0 znamená, že optimalizácia prebehla úspešne so súčasným výpisom výsledkov optimalizácie na obrazovku, iná hodnota znamená zlyhanie optimalizácie so súčasným výpisom chybovej správy. Program po prebehnutí optimalizácie vypíše na obrazovku získaný výsledok t.j. jednotlivé optimalizované koeficienty $\Delta\zeta_i, x_{ij}, u_{ij}, p_i$, optimálnu hodnotu účelovej funkcie a počet vyčíslený účelovej funkcie. Zároveň program vygeneruje dva výstupne súbory s menom *result.dat* a *fresult.txt*. Súbor *result.dat* obsahuje číselne výsledky optimalizácie v tvare:

počet intervalov
počet kolokačných bodov pre stavové veličiny
počet kolokačných bodov pre riadiace veličiny
počet stavových veličín
počet riadiacich veličín
hodnoty intervalov
hodnoty koeficientov pre riadiace veličiny
hodnoty koeficientov pre stavové veličiny
hodnoty parametrov

Súbor *fresult.txt* obsahuje predošle uvedené údaje s popisom v textovom formáte aj s hodnotou účelovej funkcie a počtom vyčíslení účelovej funkcie.

4.2 Opis funkcie *dynopt*

Súčasná verzia *dynoptu* je schopná riešiť problémy dynamickej optimalizácie s účelovou funkciou v Mayerovom tvare. Formulácia problému s ODE systémom môže byť nasledovná:

$$\min_{u(t)} G\left(x\left(t_f\right), t_f\right) \tag{4.1}$$

tak, že platí:

$$\begin{gathered}
\dot{x}(t) = f\left[t, x(t), u(t)\right] \\
x(0) = x_0 \\
h\left[t, x(t), u(t)\right] = 0 \\
g\left[t, x(t), u(t)\right] \le 0 \\
x(t)^L \le x(t) \le x(t)^U \\
u(t)^L \le u(t) \le u(t)^U
\end{gathered}$$

kde

$G\left(x\left(t_f\right), t_f\right)$ – účelová funkcia vyčíslená v koncových podmienkach

f – pravé strany diferenciálnych rovníc opisujúce proces

h – vektor obmedzení typu rovnosti

g – vektor obmedzení typu nerovnosti

$x(t)$ – vektor stavových veličín

$u(t)$ – vektor riadiacich veličín

$x(0)$ – počiatočné podmienky pre diferenciálne rovnice

$x(t)^L, x(t)^U$ – obmedzenia na stavové veličiny

$u(t)^L, u(t)^U$ – obmedzenia na riadiace veličiny

Na riešenie úlohy optimálneho riadenia bol použitý solver nelineárneho programovania MI1CF (Pernille Brock, Kaj Madsen, Hans Bruun Nielsen 2004), ktorý hľadá minimum nelineárnej účelovej funkcie vzhľadom na nelineárne obmedzenia typu rovnosti a nerovnosti. Algoritmus využívaný funkciou MI1CF je iteratívny. Je založený na postupnom aproximovaní nelineárneho problému na kvadratický t.j. v každej iterácií objektívna funkcia je aproximovaná kvadratickou funkciou a obmedzenia sú aproximované lineárnou funkciou.

V ďalšej sekcii bude uvedené na jednoduchom príklade, ako uvedený problém zapísať do užívateľských funkcií uvedených v stati 4.1.1.

4.3 Praktické použitie funkcie *dynopt*

Táto sekcia znázorňuje použitie funkcie *dynopt* na optimalizačnom probléme, definíciu vstupných funkcií *obj*, *proces*, *econstr*, *inconstr*, definíciu vstupných parametrov funkcie *dynopt*, riešenie tejto optimalizačnej úlohy a jej grafické zobrazenie.

4.3.1 Problém optimálneho riadenia automobilu

Problém optimálneho riadenia automobilu spočíva v rozbehnutí a zastavení auta v minimálnom čase, pričom auto musí prejsť fixnú vzdialenosť. Stavmi systému sú x_1 (aktuálna rýchlosť v ms^{-1}) a x_2 (prejdená vzdialenosť v m). Zadaná vzdialenosť je $x_2(t_f) = 300m$. Riadeným vstupom je zrýchlenie u v ms^{-2}. Ďalej uvažujme obmedzenia na maximálnu akceleráciu $(u \le 1ms^{-2})$ a na maximálne brzdenie $(u \ge -2ms^{-2})$, maximálnu $(x_1 \le 30ms^{-1})$ a minimálnu $(x_1 \ge 0ms^{-1})$ rýchlosť, maximálnu $(x_2 \le 300m)$ a minimálnu $(x_2 \ge 0m)$ prejdenú dráhu. Takto zadefinovaný problém dynamickej optimalizácie môže byť matematicky formulovaný nasledovne:

$$\min_{u(t)} J = t_f \tag{4.2}$$

pričom

$$\begin{aligned} \dot{x}_1 &= u,\ x_1(0) = 0ms^{-1} \\ \dot{x}_2 &= x_1,\ \ x_2(0) = 0m \\ x_1(t_f) &= 0ms^{-1} \\ x_2(t_f) &= 300m \\ -2 \le u &\le 1ms^{-2} \\ 0 \le x_1 &\le 30ms^{-1} \end{aligned}$$

$$0 \le x_2 \le 300m$$

kde

x_1 – rýchlosť

x_2 – prejdená vzdialenosť

u – zrýchlenie

4.3.2 Definícia funkcií *obj*, *proces*, *econstr*, *inconstr*

Problém optimálneho riadenia automobilu je opísaný dvoma diferenciálnymi rovnicami, ktoré spolu s počiatočnými podmienkami sú definované v užívateľskej funkcii *proces*. Gradienty funkcií sa nepočítajú numericky, a preto je potrebné ich zadať analytický. Do premennej *sys* vo funkcii *proces* sa ukladajú pravé strany diferenciálnych rovníc a ich príslušne derivácie. Vo funkcii *obj* sa do premennej *sys* ukladá hodnota účelovej funkcie a jej príslušne derivácie. Vo funkciách *econstr* a *inconstr* sa do tejto premennej ukladajú hodnoty obmedzení, taktiež aj ich príslušne derivácie. Premenná t je skalárna veličina a predstavuje čas. Premenné *x* (stavová veličina), *u* (riadiaca veličina) a *p* (časovo nezávislé parametre) môžu byť skalárne, ale aj vektorové veličiny.

Definícia funkcií *obj*, *proces*, *econstr*, *inconstr* potom vyzerá nasledovne:

Funkcia *proces*

Funkcia *proces* slúži na zadanie diferenciálnych rovníc, ktoré reprezentujú daný proces aj s príslušnými deriváciami a počiatočnými podmienkami:

```
void proces(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
        case 1:
                sys[i++]=u1;      //f1
                sys[i++]=x1;      //f2
                sys[i++]=0.0;     //df1/dx1
                sys[i++]=0.0;     //df1/dx2
                sys[i++]=1.0;     //df2/dx1
                sys[i++]=0.0;     //df2/dx2
                sys[i++]=1.0;     //df1/du
                sys[i++]=0.0;     //df2/du
                sys[i++]=0.0;     //df1/dt
                sys[i++]=0.0;     //df2/dt
```

```
            break;
        case 2:
            sys[i++]=0.0;     //f1(0)
            sys[i++]=0.0;     //f2(0)
            break;
        default:
            break;
        }
}
```

Funkcia *obj*

Funkcia *obj* slúži na zadanie účelovej funkcie, ktorá ma byť minimalizovaná aj s príslušnými deriváciami:

```
void obj(double *sys, double t, double *x, double *u, double *p)
{
        int i = 0;
        sys[i++] = t;             //J
        sys[i++] = 0.0;           //dJ/dx1
        sys[i++] = 0.0;           //dJ/dx2
        sys[i++] = 0.0;           //dJ/du
        sys[i++] = 1.0;           //dJ/dt
}
```

Funkcia *econstr*

Do funkcie *econstr* sa definujú obmedzenia typu rovnosti s príslušnými deriváciami:

```
void econstr(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
        case 1:
            break;
        case 2:
            break;
        case 3:
            sys[i++]=x1;          //econ1
            sys[i++]=x2-300;      //econ2
            sys[i++]=1.0;         //decon1/dx1
            sys[i++]=0.0;         //decon1/dx2
            sys[i++]=0.0;         //decon2/dx1
            sys[i++]=1.0;         //decon2/dx2
            sys[i++]=0.0;         //decon1/du
            sys[i++]=0.0;         //decon2/du
            sys[i++]=0.0;         //decon1/dt
```

```
                sys[i++]=0.0;                   //decon2/dt
                break;
        default:
                break;
        }
}
```

Funkcia *inconstr*

Do funkcie *inconstr* sa definujú obmedzenia typu nerovnosti s príslušnými deriváciami:

```
void inconstr(double *sys, double t, double *x, double *u, double *p, int opt)
{
        int i=0;
        switch(opt)
        {
        case 1:
                break;
        case 2:
                break;
        case 3:
                break;
        default:
                break;
        }
}
```

Po definovaní funkcií *obj*, *proces*, *econstr*, *inconstr* užívateľ nadefinuje vstupne parametre pre funkciu *dynopt* vo funkcii *main* nasledovne:

```
param init;                                     //štruktúra param
double ksi_s[2] = {20,20};                      //počiatočné hodnoty intervalov
double u_s[2][1] = {{0.8},{-1}};                //počiatočné hodnoty riadiacej veličiny
                                                //v jednotlivých intervaloch
double luu[2][1] = {{-2},{1}};                  //dolné a horné obmedzenia pre riadiacu
                                                //veličinu
double lxu[2][2] = {{-1e-2,-1e-2},{30.01,300.01}};  //dolné a horné obmedzenia pre stavové
                                                //veličiny
double eps = 1e-3;                              //výpočtová presnosť
int maxfcn = 1000;                              //maximálny počet vyčíslení účelovej
                                                //funkcie
init.nint = 2;                                  //počet intervalov
init.nx = 2;                                    //počet stavových veličín
init.nu = 1;                                    //počet riadiacich veličín
init.ncolx = 5;                                 //počet kolokačných bodov pre stavové
                                                //veličiny
init.ncolu = 2;                                 //počet kolokačných bodov pre riadiacu
                                                //veličinu
```

```
init.eq = 2;                                    //počet obmedzení typu rovnosti
init.in = 0;                                    //počet obmedzení typu nerovnosti
init.ob_eq = 3;                                 //umiestnenie obmedzení typu rovnosti
init.ob_in = 0;                                 //umiestnenie obmedzení typu nerovnosti
init.obu = 1;                                   //prítomnosť obmedzenia na riadiacu
                                                //veličinu
init.obx = 1;                                   //prítomnosť obmedzenia na stavové
                                                //veličiny
init.nparam = 0;                                //počet parametrov (zatiaľ nefunkčné,
                                                //treba nastaviť nulu)
return dynopt(&init, 0.0, ksi_s, u_s, NULL, luu, lxu, proces, obj, econstr, inconstr, eps, maxfcn);
                                                //volanie funkcie dynopt s nastavenými
                                                //parametrami
```

4.3.3 Riešenie optimalizačného problému

Aplikovaním 5 kolokačných bodov pre stavové veličiny a 2 kolokačných bodov pre riadiacu veličinu na dvoch intervaloch, sa tieto hodnoty nastavia do jednotlivých položiek štruktúry *param*. Každý interval ma počiatočnú dĺžku 20. Konečný čas bol optimalizovaný, preto je druhý parameter funkcie *dynopt* nastavený na 0,0. Počiatočné hodnoty riadiacej veličiny boli nastavené na 0,8 pre prvý interval a –1 pre druhý interval. Obmedzenie pre riadiacu veličinu je nastavené v rozsahu –2 až 1. Pre prvú stavovú veličinu od –0,01 až 30,01 a pre druhú od –0,01 až 300,01 (stotiny sú kvôli numerickým výpočtom solvera). Presnosť riešenia je nastavená na 0,001 a maximálny počet vyčíslenia účelovej funkcie je 1000. Ďalej je nastavený počet obmedzení a ich umiestnenie, tieto hodnoty sa taktiež nastavujú do štruktúry *param*.

Po 16 iteráciach je riešenie nasledovne (obsah súboru fresult.txt):

```
Riešenie optimalizačného problému

Hodnoty intervalov:
                    20.000
                    10.000

Hodnoty riadiacich veličín:
        1. interval
                1. riadiaca veličina
                        1.000
                        1.000
        2. interval
                1. riadiaca veličina
                        -2.000
                        -2.000

Hodnoty stavových veličín:

        1. interval
```

1. stavová veličina

0.000
0.938
4.615
10.000
15.385
19.062

2. stavová veličina

0.000
0.440
10.651
50.000
118.344
181.676

2. interval

1. stavová veličina

20.000
19.062
15.385
10.000
4.615
0.938

2. stavová veličina

200.000
209.162
240.828
275.000
294.675
299.780

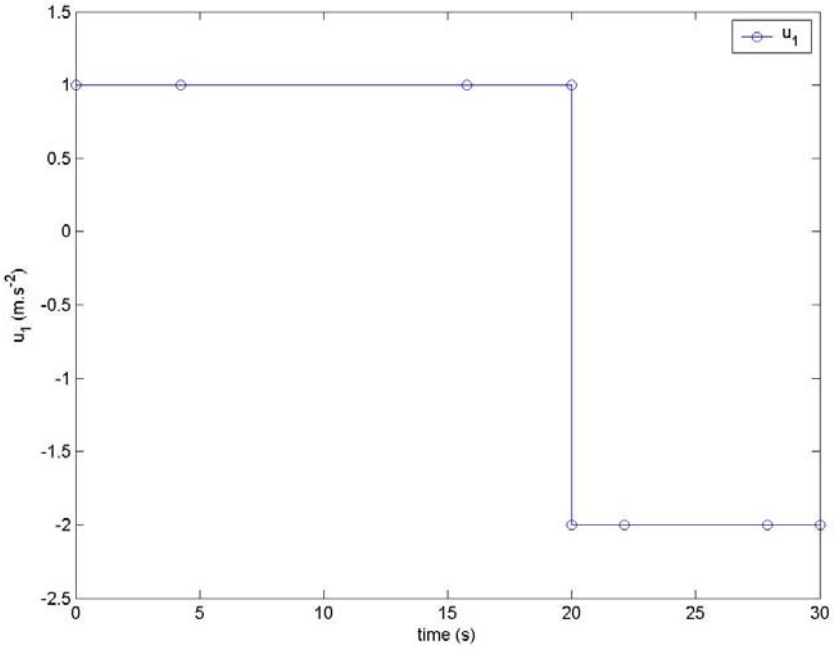
Počet volaní funkcie fdfcdc: 16

Hodnota účelovej funkcie v optime: 30.000

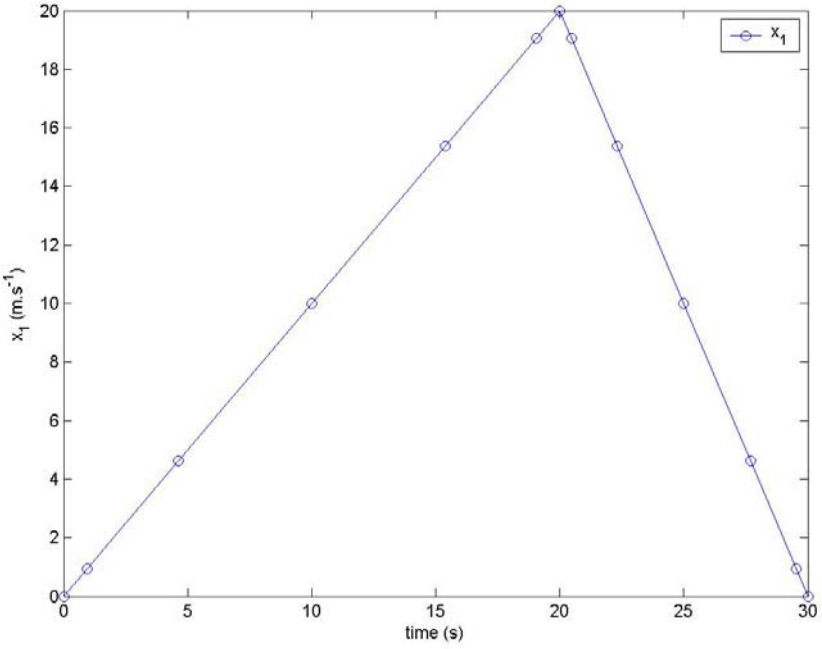
4.3.4 Grafická interpretácia riešenia

Všetky grafy v tejto práci boli vytvorené pomocou MATLABu. Súbor result.dat generovaný *dynoptom* je v ascii formáte a obsahuje údaje zo sekcie 4.3.3 bez komentárov. Na začiatku je uvedený aj počet intervalov, počet kolokačných bodov pre stavové aj pre riadiace veličiny a počet stavových a riadiacich veličín, vďaka čomu sa dali jednotlivé údaje roztriediť a vykresliť do grafov. Po 16 iteráciach, optimálna hodnota účelovej funkcie je 30s. Na priebehu riadiacej veličiny obr. 4.3.1 je vidieť jej nespojitosť a rozdelenie na dva intervaly. Priebeh stavových veličín je spojitý obr. 4.3.2 – 4.3.3, ako je spomenuté v kapitole 3 čo je napokon vidieť aj z uvedených grafov. Z grafov, je aj vidieť splnenie obmedzení na stavové a riadiace veličiny a splnenie koncových obmedzení.

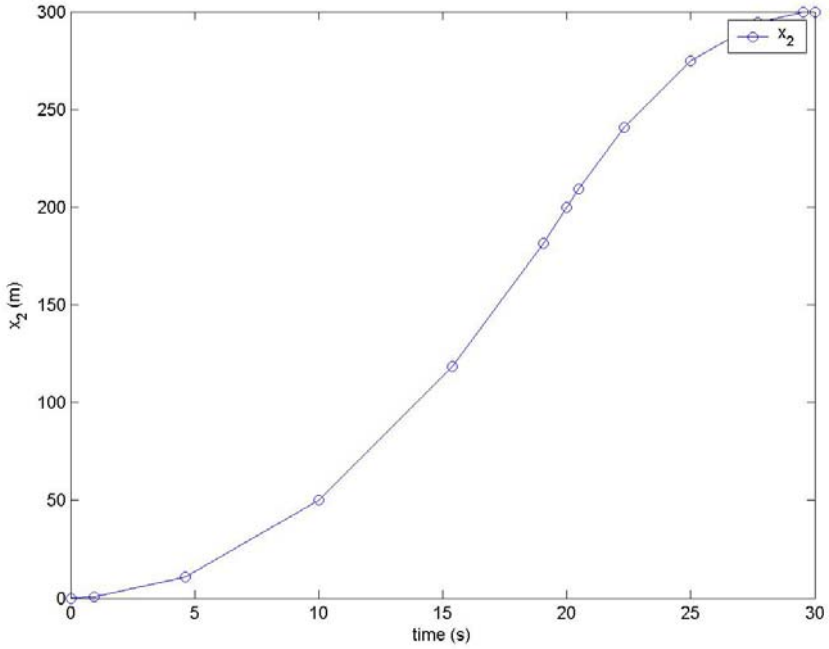
Grafické riešenie pre problém optimálneho riadenia automobilu je nasledovné:



Obr. 4.3.1: Profil priebehu zrýchlenia



Obr. 4.3.2: Rýchlostný profil



Obr. 4.3.3: Dráhový profil

Kapitola 5

Praktické aplikácie

5.1 Problém 1

Uvažujme vsádzkový reaktor s nasledujúcimi reakciami: $A \rightarrow B \rightarrow C$

$$\min_{u(t)} J = -x_2(t_f) \tag{5.1}$$

pričom

$$\begin{aligned} \dot{x}_1 &= -k_1 x_1^2 \qquad x_1(0) = 1 \\ \dot{x}_2 &= k_1 x_1^2 - k_2 x_2 \qquad x_2(0) = 0 \\ k_1 &= 4000 e^{\left(-\frac{2500}{T}\right)} \\ k_2 &= 620000 e^{\left(-\frac{5000}{T}\right)} \\ 298 &\le T \le 398 \\ t_f &= 1 \end{aligned}$$

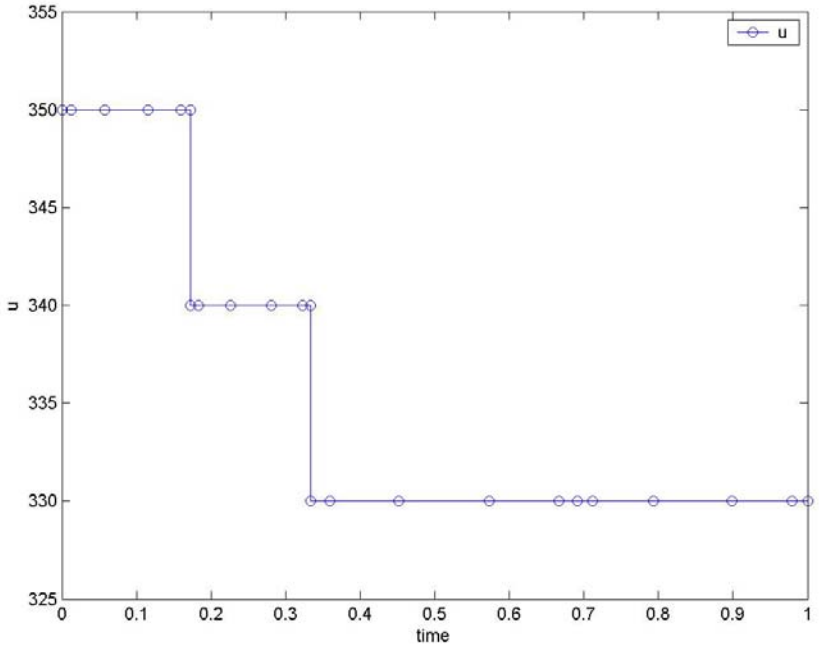
kde

$x_1(t)$ – koncentrácia látky A,

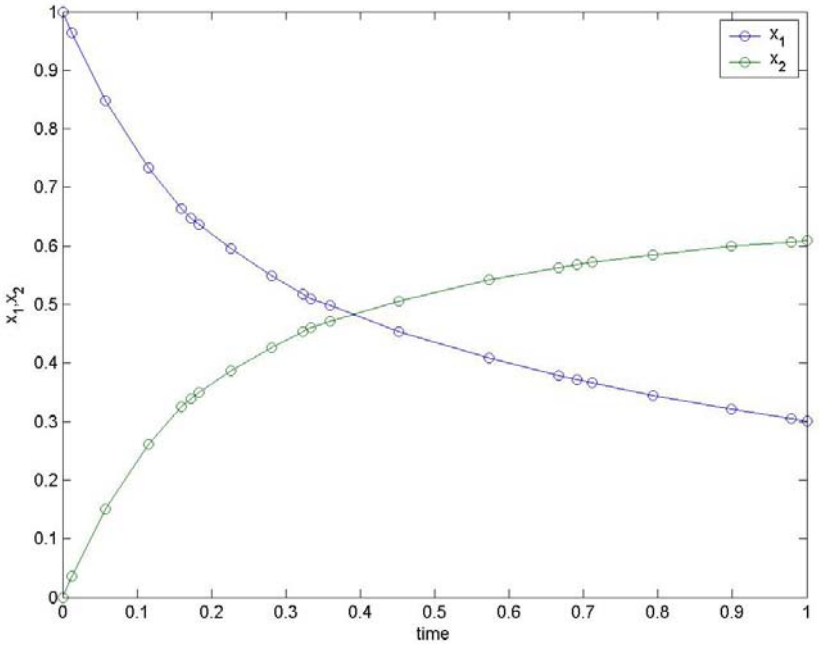
$x_2(t)$ – koncentrácia látky B,

T – teplota.

Cieľom problému (5.1) je získať optimálny teplotný profil, ktorý maximalizuje koncentráciu látky B na konci špecifikovaného času. Optimalizačný problém bol riešený Logsdonom a Bieglerom (1989) a získané optimum bolo 0,610775. S použitím *dynoptu* pre tento optimalizačný problém na 4 intervaloch, 4 kolokačných bodov pre stavové veličiny a 4 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,609. Počet vyčíslení účelovej funkcie bol 9. Na Obr. 5.1.1 – 5.1.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.1.1 Teplotný profil pre príklad 1



Obr. 5.1.2 Koncentračný profil pre príklad 1

5.2 Problém 2

Uvažujme rúrkový reaktor s nasledujúcimi paralelnými reakciami:

$A \rightarrow B$

$A \rightarrow C$

$$\min_{u(t)} J = -x_2\left(t_f\right) \tag{5.2}$$

pričom

$$\begin{gathered} \dot{x}_1 = -\left[u + 0{,}5u^2\right]x_1 \qquad x_1(0) = 1 \\ \dot{x}_2 = ux_1 \qquad\quad x_2(0) = 0 \\ 0 \le u \le 5 \\ t_f = 1 \end{gathered}$$

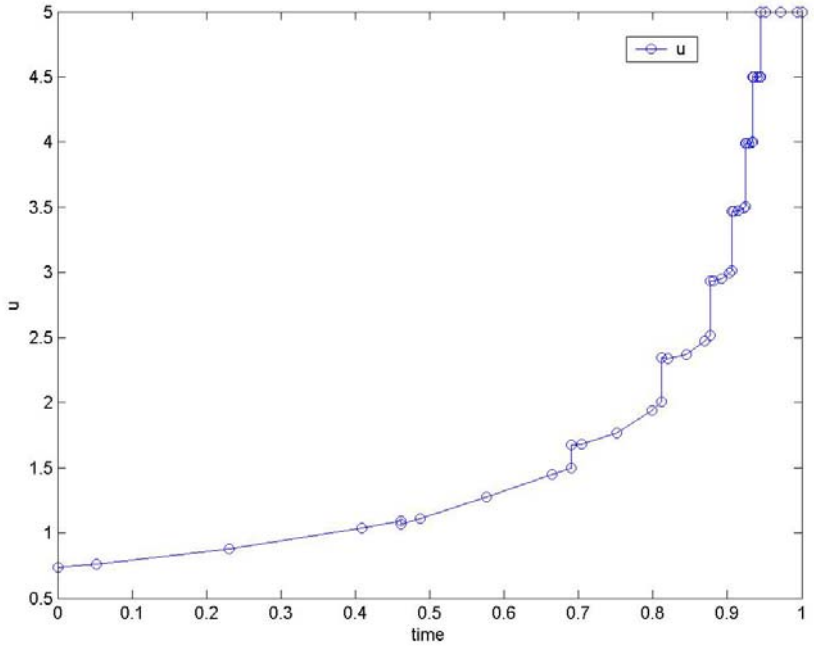
kde

$x_1(t)$ – koncentrácia látky A,

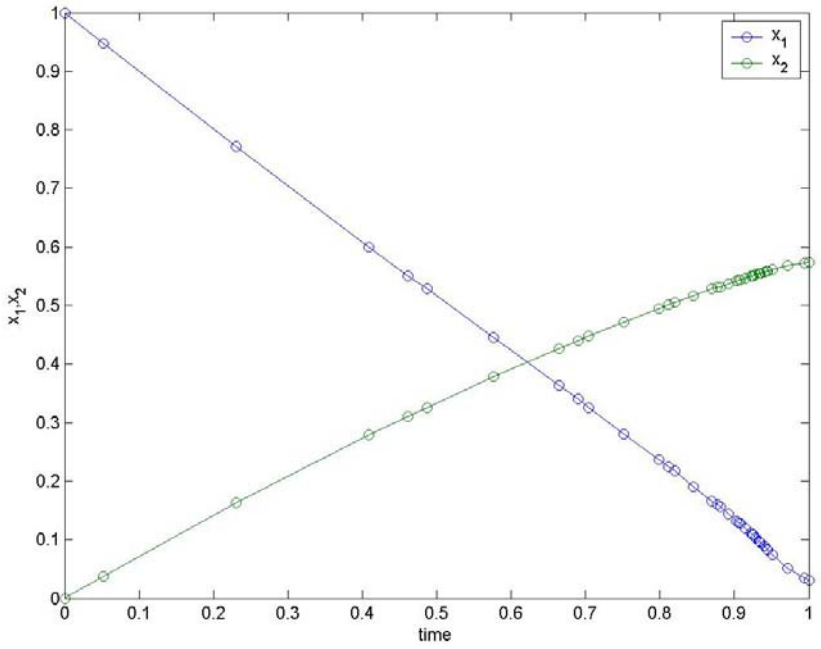
$x_2(t)$ – koncentrácia látky B,

u – riadiaca veličina.

Problém (5.2) je príklad rúrkového reaktora, kde stavová veličina $x_2(t)$ v koncovom čase ma byť maximalizovaná. Optimálna hodnota účelovej funkcie získaná Longstonom a Bieglerom je 0,57353. S použitím *dynoptu* pre tento optimalizačný problém na 9 intervaloch, 3 kolokačných bodov pre stavové veličiny a 3 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,573. Počet vyčíslení účelovej funkcie bol 24. Na Obr. 5.2.1 – 5.2.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.2.1 Profil riadiacej veličiny pre príklad 2



Obr. 5.2.2 Profil stavových veličín pre príklad 2

5.3 Problém 3

Uvažujme katalytický izotermický prietokový reaktor s dvoma katalyzátormi pozdĺž reaktora s nasledujúcimi reakciami:

$A \rightarrow B \leftrightarrow C$

$$\max_{u(t)} J = 1 - x_1(t_f) - x_2(t_f) \tag{5.3}$$

pričom

$$\begin{gathered} \dot{x}_1 = u[10x_2 - x_1] \qquad x_1(0) = 1 \\ \dot{x}_2 = -u[10x_2 - x_1] - [1-u]x_2 \quad x_2(0) = 0 \\ 0 \le u \le 1 \\ t_f = 12 \end{gathered}$$

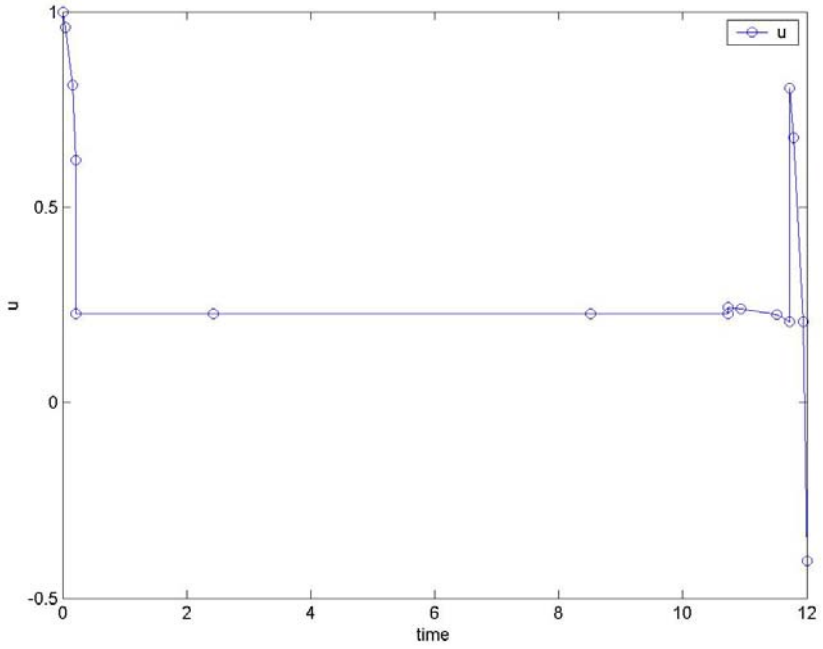
kde

$x_1(t)$ – mólová frakcia látky A,

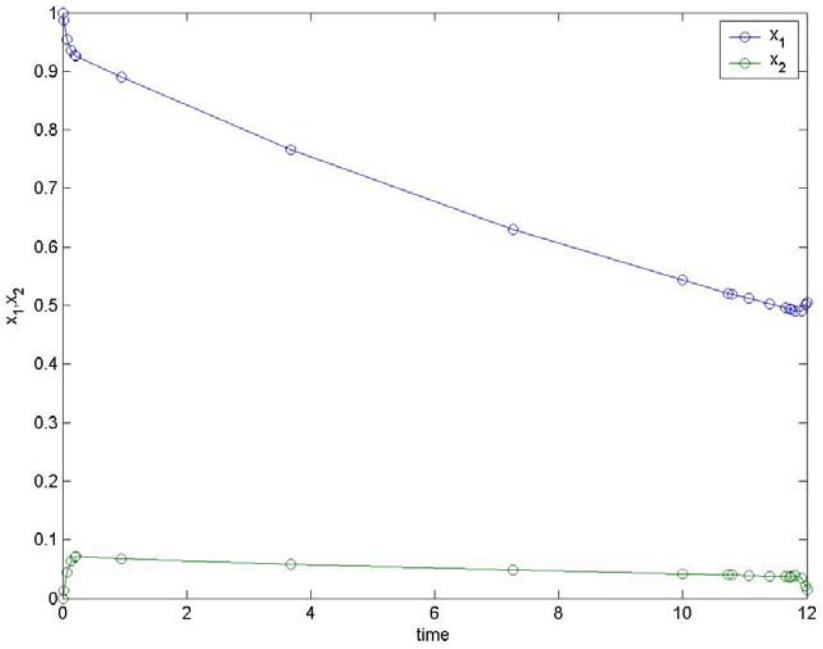
$x_2(t)$ – mólová frakcia látky B,

u – frakcia katalyzátora.

Optimalizačný problém (5.3) bol tiež analyzovaný. Tento problém bol riešený Longstonom a Bieglerom a získali hodnotu účelovej funkcie 0,476946. S použitím *dynoptu* pre tento optimalizačný problém na 4 intervaloch, 4 kolokačných bodov pre stavové veličiny a 2 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,478. Počet vyčíslení účelovej funkcie bol 50. Na Obr. 5.3.1 – 5.3.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.3.1 Profil riadiacej veličiny pre príklad 3



Obr. 5.3.2 Profil stavových veličín pre príklad 3

5.4 Problém 4

Uvažujme nelineárny prietokový miešaný reaktor:

$$\max_{u(t)} J = x_8\left(t_f\right) \tag{5.4}$$

pričom

$$\begin{aligned}
\dot{x}_1 &= u_4 - qx_1 - 17{,}6x_1x_2 - 23x_1x_6u_3 \qquad x_1(0) = 0{,}1883\\
\dot{x}_2 &= u_1 - qx_2 - 17{,}6x_1x_2 - 146x_2x_3 \qquad x_2(0) = 0{,}2507\\
\dot{x}_3 &= u_2 - qx_3 - 73x_2x_3 \qquad x_3(0) = 0{,}0467\\
\dot{x}_4 &= -qx_4 + 35{,}2x_1x_2 - 51{,}3x_4x_5 \qquad x_4(0) = 0{,}0899\\
\dot{x}_5 &= -qx_5 + 219x_2x_3 - 51{,}3x_4x_5 \qquad x_5(0) = 0{,}1804\\
\dot{x}_6 &= -qx_6 + 102{,}6x_4x_5 - 23x_1x_6u_3 \qquad x_6(0) = 0{,}1394\\
\dot{x}_7 &= -qx_7 + 46x_1x_6u_3 \quad x_7(0) = 0{,}1046\\
\dot{x}_8 &= 5{,}8\left(qx_1 - u_4\right) - 3{,}7u_1 - 4{,}1u_2 + q\left(23x_4 + 11x_5 + 28x_6 + 35x_7\right) - 5u_3^2 - 0{,}099 \qquad x_8(0) = 0
\end{aligned}$$

$$q = u_1 + u_2 + u_4$$

$$0 \le u_1 \le 20$$

$$0 \le u_2 \le 6$$

$$0 \le u_3 \le 4$$

$$0 \le u_4 \le 20$$

$$t_f = 0{,}2$$

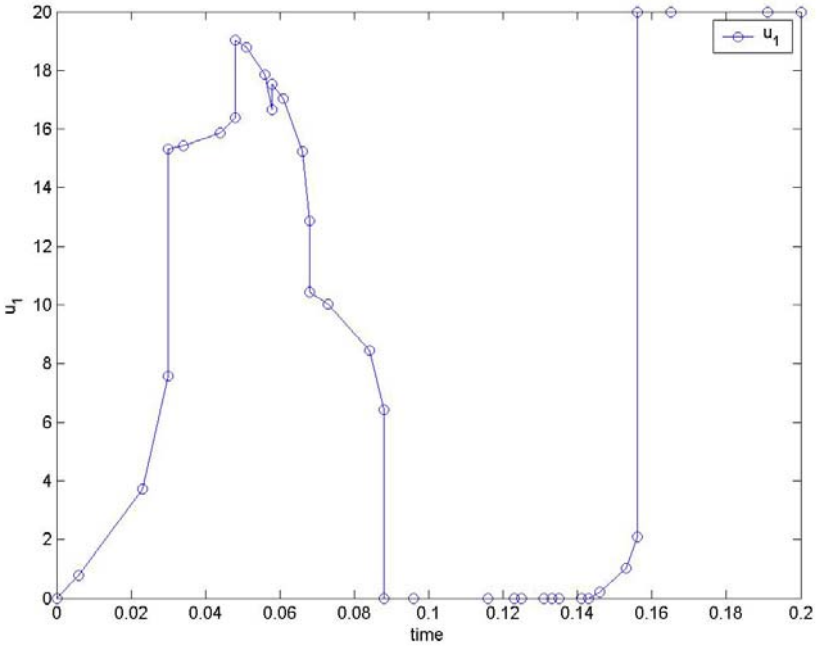
kde

$x_1(t) - x_8(t)$ – vektor stavových veličín

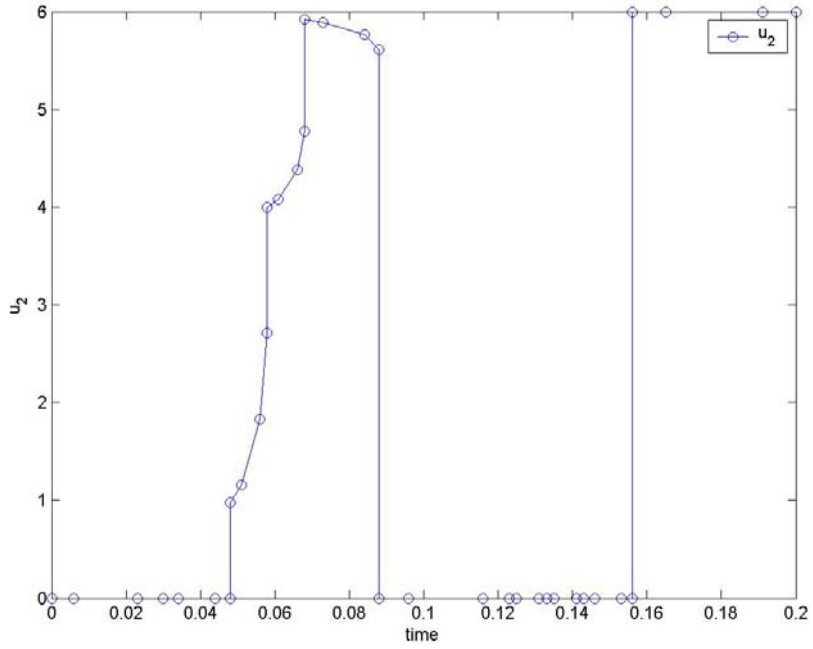
$u_1(t) - u_4(t)$ – vektor riadiacich veličín

Problém (5.4) bol prvýkrát uvedený Jensenom (1964) a spočíva v hľadaní štyroch optimálnych riadiacich veličín pre chemický reaktor, aby sa dosiahol maximálny ekonomický profit. Dynamika systému popisuje 4 simultánne chemické reakcie, ktoré prebiehajú v izotermickom prietokovom miešanom reaktore. Riadiace veličiny sú 3 prietoky vstupných prúdov a spotreba elektrickej energie pre fotochemickú podporu reakcie. Luus (1990) a Bojkov *et al*. (1993) uvádzajú optimálnu hodnotu účelovej funkcie 21,757 a 21,744. S použitím *dynoptu* pre tento optimalizačný problém na 10 intervaloch,

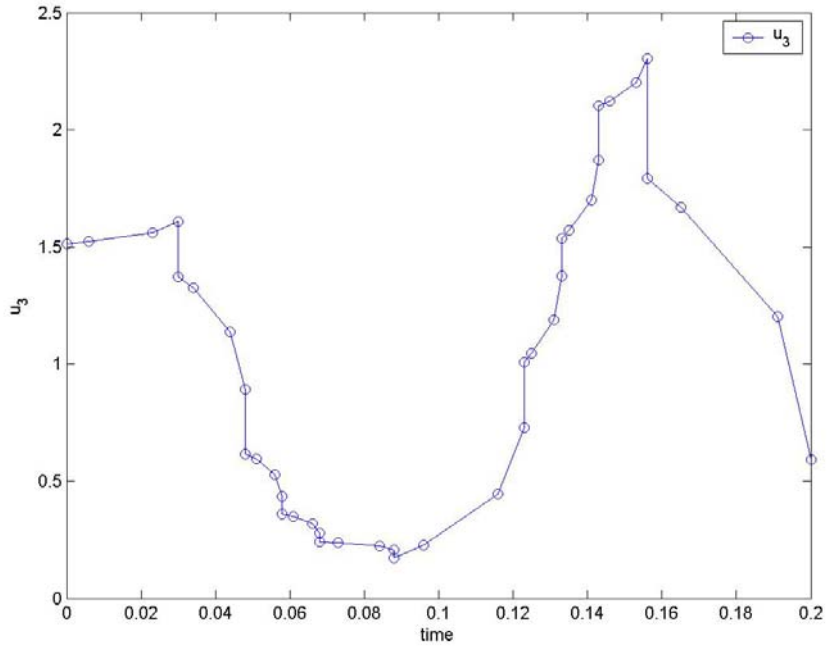
4 kolokačných bodov pre stavové veličiny a 2 kolokačné body pre riadiace veličiny bola získaná optimálna hodnota účelovej funkcie 21,757. Počet vyčíslení účelovej funkcie bol 227. Na Obr. 5.4.1 – 5.4.5 je zobrazený optimálny profil riadiacich a stavových veličín.



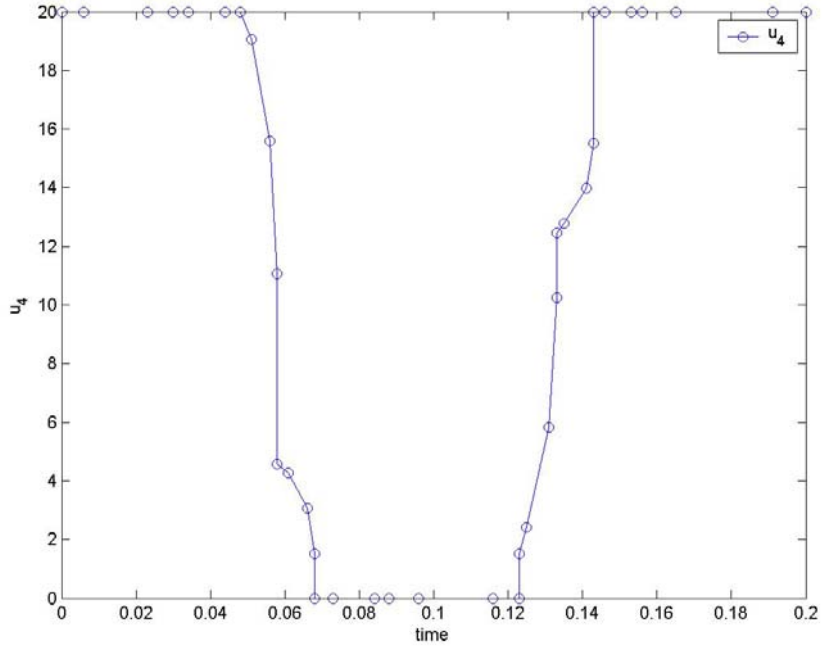
Obr. 5.4.1 Profil prvej riadiacej veličiny pre problém 4



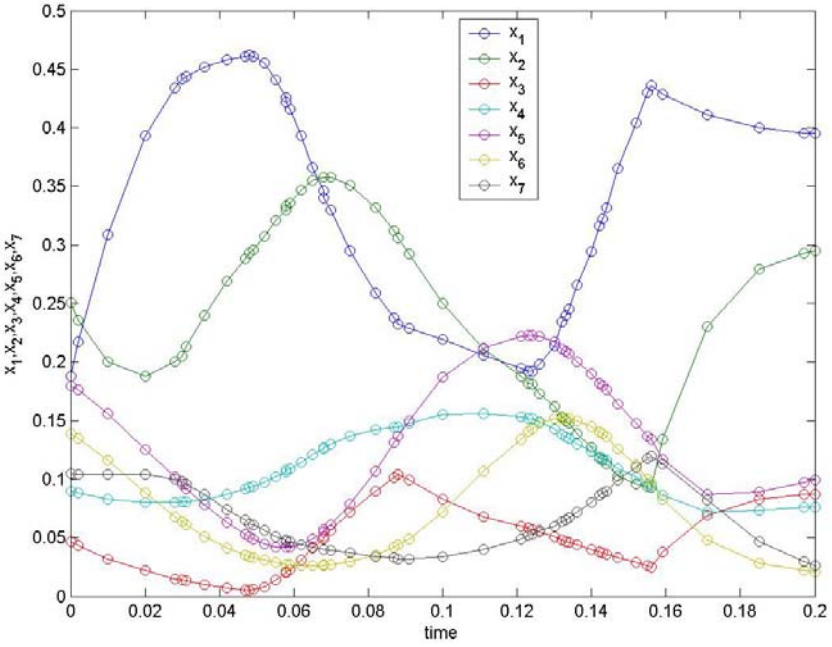
Obr. 5.4.2 Profil druhej riadiacej veličiny pre problém 4



Obr. 5.4.3 Profil tretej riadiacej veličiny pre problém 4



Obr. 5.4.4 Profil štvrtej riadiacej veličiny pre problém 4



Obr. 5.4.5 Profil stavových veličín pre problém 4

5.5 Problém 5

Uvažujme nasledujúci problém bez obmedzení:

$$\min_{u(t)} J = x_2\left(t_f\right) \tag{5.5}$$

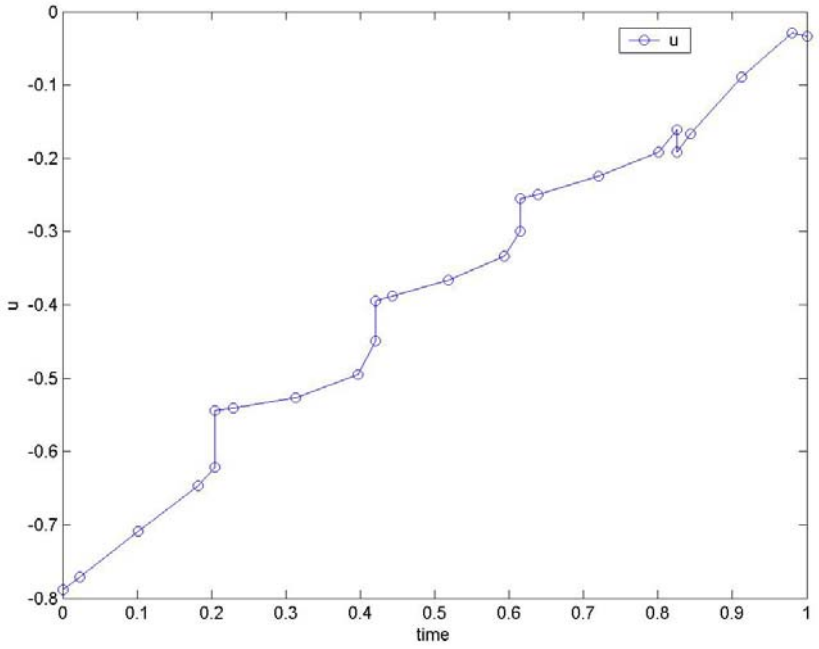
pričom

$$\begin{aligned} \dot{x}_1 &= u \quad x_1(0) = 1 \\ \dot{x}_2 &= x_1^2 + u^2 \quad x_2(0) = 0 \\ t_f &= 1 \end{aligned}$$

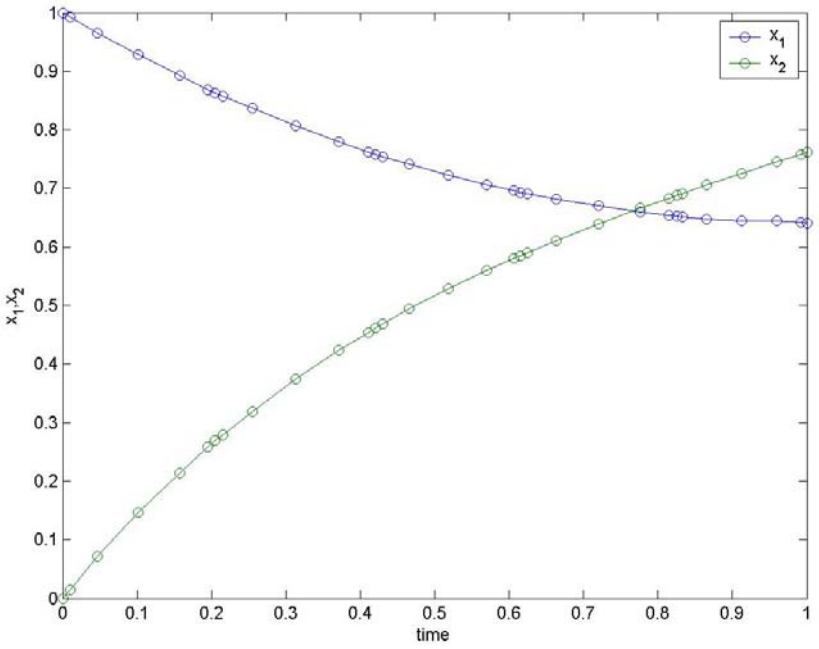
$x_1 - x_2$ – vektor stavových veličín

u – riadiaca veličina

Problém (5.5) nemá žiadne obmedzenia, a pre tento problém hodnota účelovej funkcie získaná Luusom je 0,76519. S použitím *dynoptu* pre tento optimalizačný problém na 5 intervaloch, 5 kolokačných bodov pre stavové veličiny a 3 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,762. Počet vyčíslení účelovej funkcie bol 4. Na Obr. 5.5.1 – 5.5.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.5.1 Profil riadiacej veličiny pre problém 5



Obr. 5.5.2 Profil stavových veličín pre problém 5

5.6 Problém 6

Uvažujme nasledujúci problém s obmedzením:

$$\min_{u(t)} J = x_2\left(t_f\right) \tag{5.6}$$

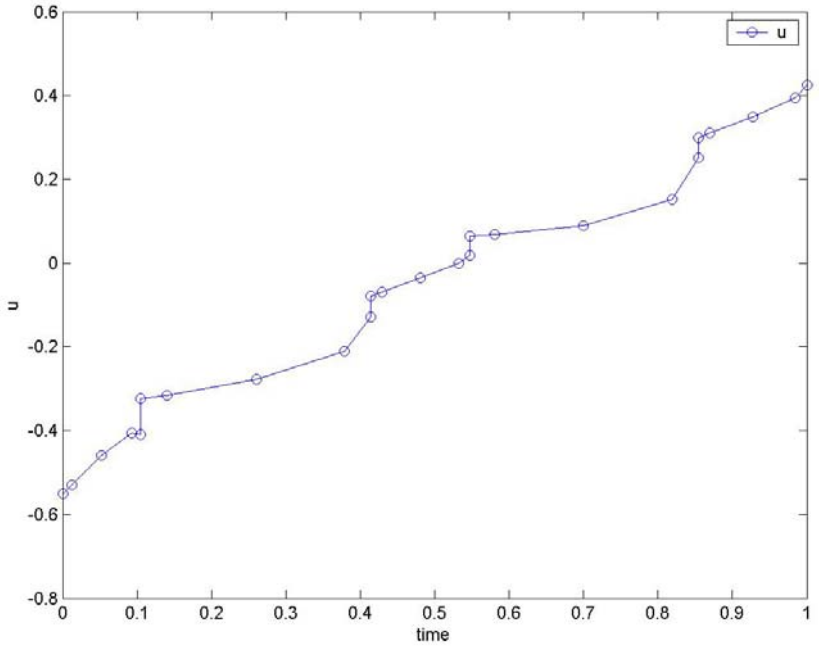
pričom

$$\begin{gathered}\dot{x}_1 = u \quad x_1(0) = 1 \\ \dot{x}_2 = x_1^2 + u^2 \quad x_2(0) = 0 \\ x_1(1) = 1 \\ t_f = 1\end{gathered}$$

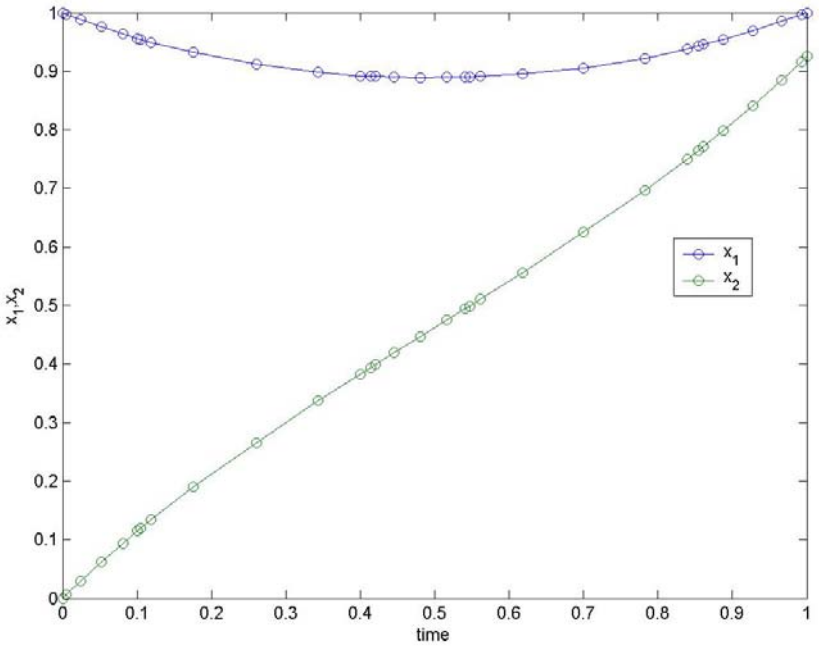
$x_1 - x_2$ – vektor stavových veličín

u – riadiaca veličina

Problém (5.6) sa od predošlého problému líši koncovým obmedzením. Pre tento prípad optimálna hodnota účelovej funkcie získaná Luusom je 0,92518. S použitím *dynoptu* pre tento optimalizačný problém na 5 intervaloch, 5 kolokačných bodov pre stavové veličiny a 3 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,925. Počet vyčíslení účelovej funkcie bol 7. Na Obr. 5.6.1 – 5.6.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.6.1 Profil riadiacej veličiny pre problém 6



Obr. 5.6.2 Profil stavových veličín pre problém 6

5.7 Problém 7

Uvažujme nasledujúci nelineárny problém bez obmedzení:

$$\min_{u(t)} J = x_4\left(t_f\right) \tag{5.7}$$

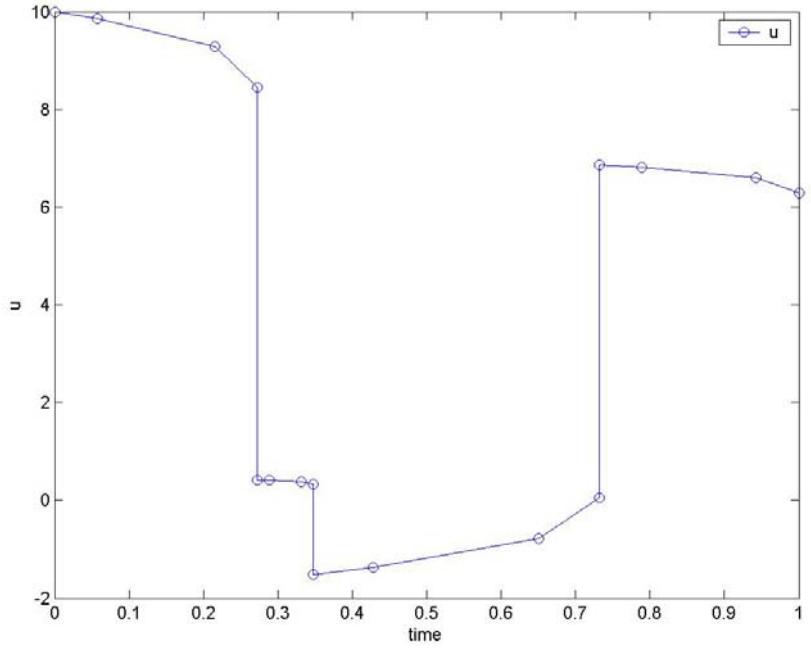
pričom

$$\begin{aligned}
\dot{x}_1 &= x_2 & x_1(0) &= 0 \\
\dot{x}_2 &= -x_3 u + 16t - 8 & x_2(0) &= -1 \\
\dot{x}_3 &= u & x_3(0) &= -\sqrt{5} \\
\dot{x}_4 &= x_1^2 + x_2^2 + 0{,}0005\left(x_2 + 16t - 8 - 0{,}1 x_3 u^2\right)^2 & x_4(0) &= 0 \\
& -4 \le u \le 10 & & \\
& t_f = 1 & &
\end{aligned}$$

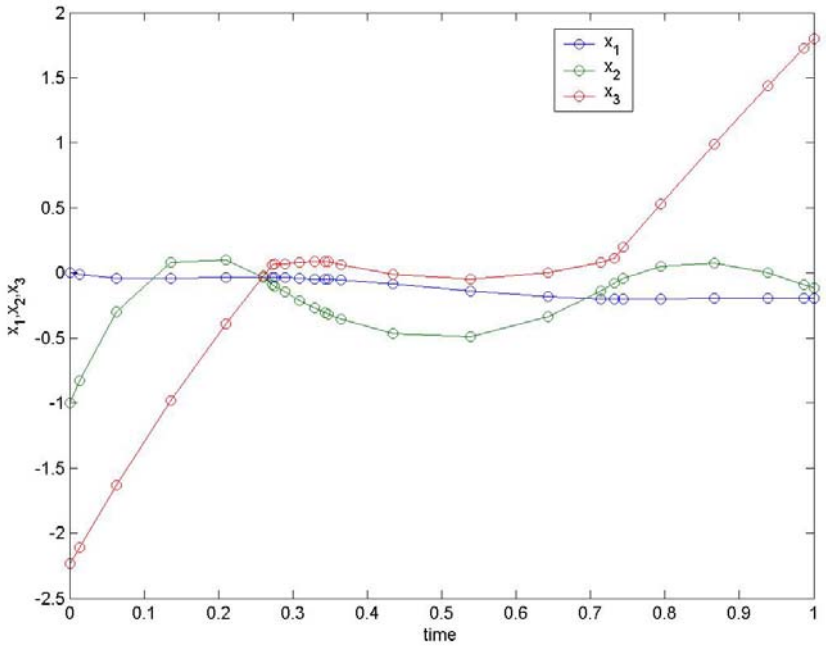
$x_1 - x_4$ – vektor stavových veličín

u – riadiaca veličina

Problém (5.7) je systém so štyrmi stavmi a bol riešený Luusom. Pre štvrtú stavovú veličinu je hodnota minima 0,12011 získaná Luusom. S použitím *dynoptu* pre tento optimalizačný problém na 4 intervaloch, 5 kolokačných bodov pre stavové veličiny a 2 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,121. Počet vyčíslení účelovej funkcie bol 67. Na Obr. 5.7.1 – 5.7.2 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.7.1 Profil riadiacej veličiny pre problém 7



Obr. 5.7.2 Profil stavových veličín pre problém 7

5.8 Problém 8

Uvažujme nasledujúci problém s obmedzením:

$$\min_{u(t)} J = x_3\left(t_f\right) \tag{5.8}$$

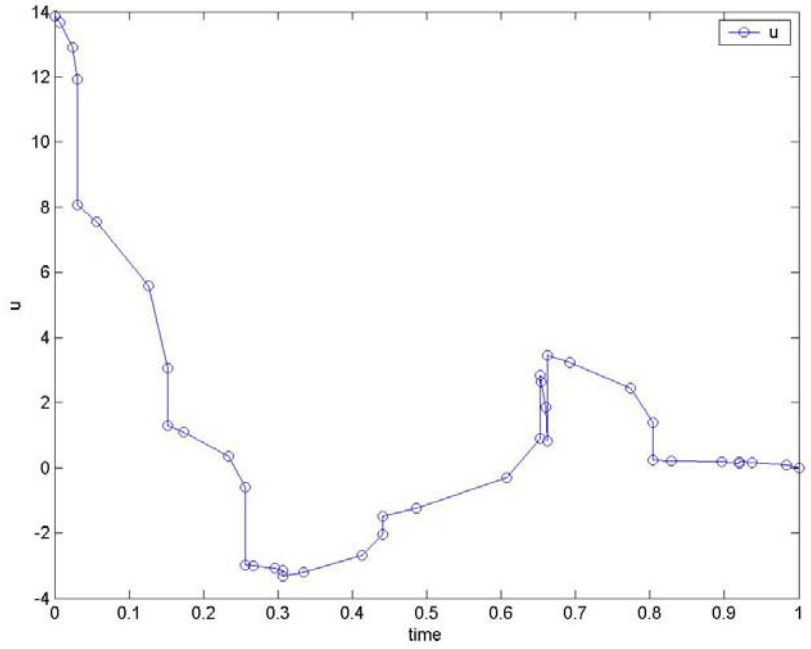
pričom

$$\begin{aligned} \dot{x}_1 &= x_2 & x_1(0) &= 0 \\ \dot{x}_2 &= -x_2 + u & x_2(0) &= -1 \\ \dot{x}_3 &= x_1^2 + x_2^2 + 0{,}005u^2 & x_3(0) &= 0 \\ x_2 - 8(t-0{,}5)^2 + 0{,}5 &\le 0 & t &\in [0,1] \\ t_f &= 1 \end{aligned}$$

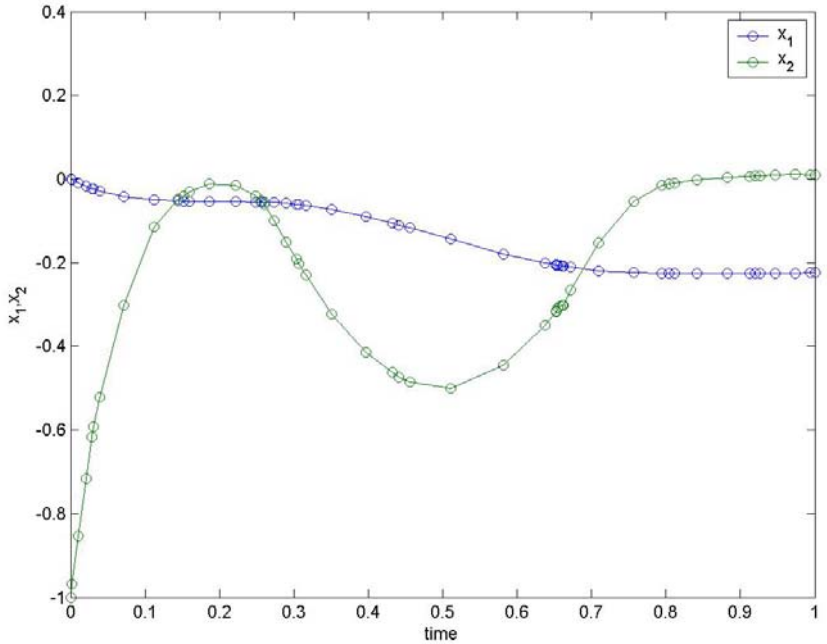
$x_1 - x_3$ – vektor stavových veličín

u – riadiaca veličina

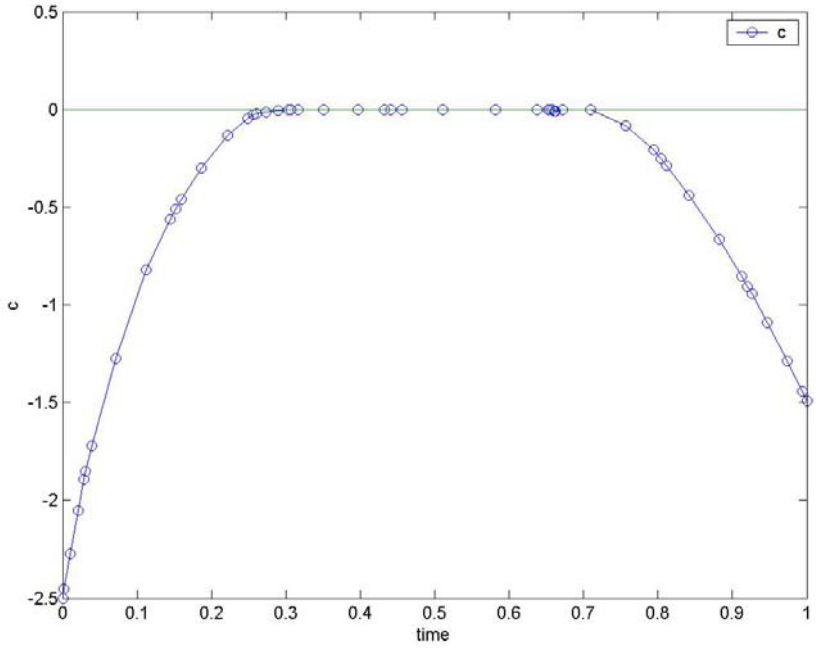
S použitým *dynoptu* pre tento optimalizačný problém na 10 intervaloch, 4 kolokačných bodov pre stavové veličiny a 2 kolokačné body pre riadiacu veličinu bola získaná optimálna hodnota účelovej funkcie 0,171. Počet vyčíslení účelovej funkcie bol 55. Na Obr. 5.8.1 – 5.8.2 je zobrazený optimálny profil riadiacej a stavových veličín. Na Obr. 5.8.3 je znázornený profil obmedzenia typu nerovnosť. Z obrázku je vidieť platnosť obmedzenia počas celého časového intervalu.



Obr. 5.8.1 Profil riadiacej veličiny pre problém 8



Obr. 5.8.2 Profil stavových veličín pre problém 8



Obr. 5.8.3 Profil priebehu obmedzenia pre problém 8

5.9 Problém 9

Uvažujme nasledujúci lineárny systém s obmedzeniami:

$$\min_{u(t)} J = x_1^2(t_f) + x_2^2(t_f) + x_3^2(t_f) + x_4^2(t_f) \tag{5.9}$$

pričom

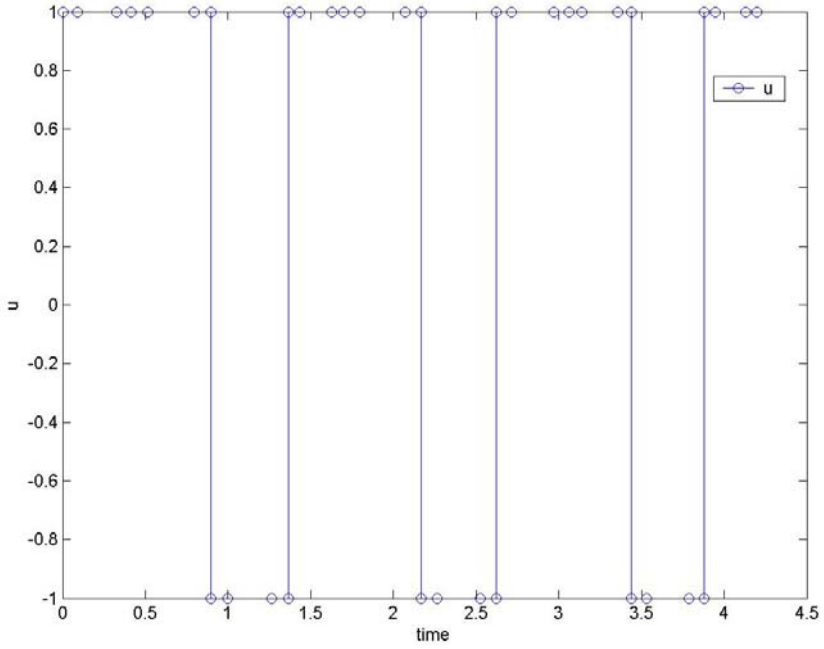
$$\begin{aligned}
\dot{x}_1 &= -0{,}5x_1 + 5x_2 & x_1(0) &= 10 \\
\dot{x}_2 &= -5x_1 - 0{,}5x_2 + u & x_2(0) &= 10 \\
\dot{x}_3 &= -0{,}6x_3 + 10x_4 & x_3(0) &= 10 \\
\dot{x}_4 &= -10x_3 - 0{,}6x_4 + u & x_4(0) &= 10 \\
-1 &\le u \le 1 \\
x_1(t_f) &\le 1 \\
x_2(t_f) &\le 1 \\
x_3(t_f) &\le 1 \\
x_4(t_f) &\le 1 \\
t_f &= 4{,}2
\end{aligned}$$

$x_1 - x_4$ – vektor stavových veličín

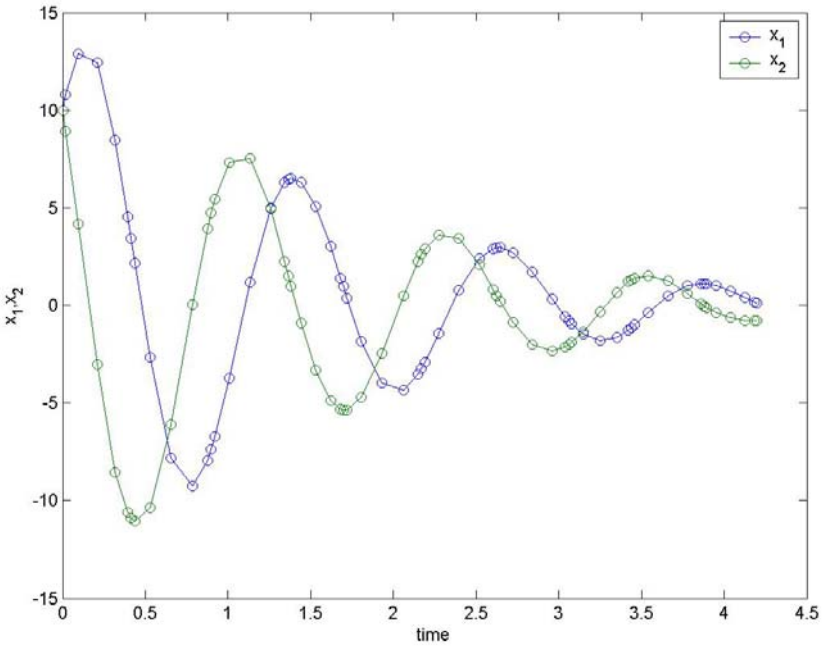
u – riadiaca veličina

Optimalizačný problém (5.9) bol riešený Nishidom *et al*. (1976), Jacobsonom (1968), Plantom a Athansom (1966). Problém spočíva v posune z počiatočnej polohy $x_1(0) = 10, x_2(0) = 10, x_3(0) = 10, x_4(0) = 10$ na polohu v konečnom čase vo vnútri jednotkovej gule umiestnenej v začiatku súradnicovej osi. Úlohou je minimalizovať finálnu polohu.

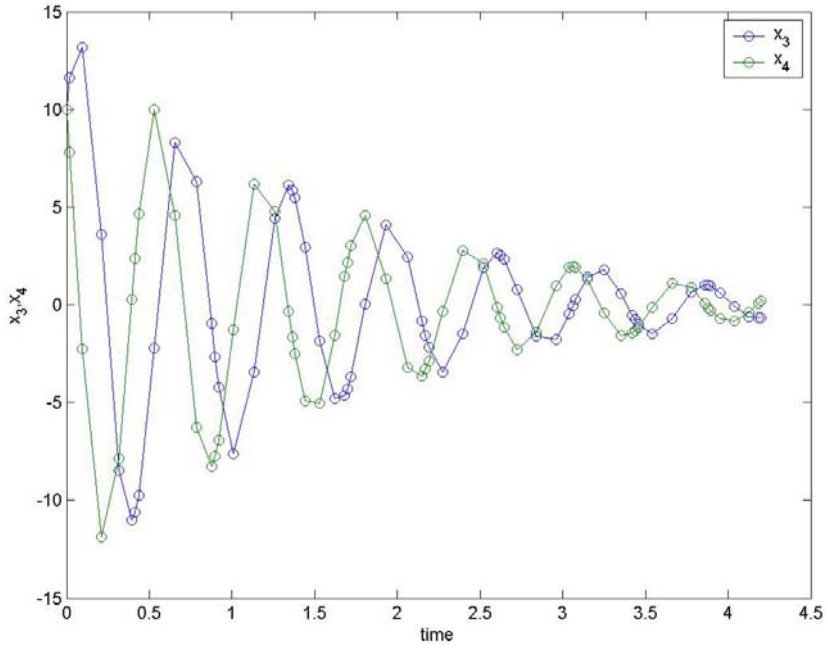
Nishida *et al*. uvádza optimálnu hodnotu účelovej funkcie 0,9952. Optimálna hodnota účelovej funkcie získaná Hindmarshom (1980) je 1,0067. S použitým *dynoptu* pre tento optimalizačný problém na 10 intervaloch, 5 kolokačných bodov pre stavové veličiny a 2 kolokačné body pre riadiacu veličinu bola získaná optimálná hodnota účelovej funkcie 1,035. Počet vyčíslení účelovej funkcie bol 113. Na Obr. 5.9.1 – 5.9.3 je zobrazený optimálny profil riadiacej a stavových veličín.



Obr. 5.9.1 Profil riadiacej veličiny pre príklad 9



Obr. 5.9.2 Profil prvej a druhej stavovej veličiny pre príklad 9



Obr. 5.9.3 Profil tretej a štvrtej stavovej veličiny pre príklad 9

Kapitola 6

Záver

Metóda prezentovaná v tejto práci predstavuje jednu z techník ako získať optimálne profily riadenia a stavov na riadenie procesu opísaného diferenciálnymi rovnicami. Diferenciálne rovnice sú diskretizované a aproximované použitím Lagrangeových polynómov a ortogonálnej kolokácie. Algebraické rovnice získané aproximáciou sa použijú na definovanie úlohy nelineárneho programovania.

Hlavným cieľom tejto práce bolo vytvorenie rozhrania na dynamickú optimalizáciu založenú na ortogonálnej kolokácii na konečných prvkoch. Celá metóda bola implementovaná v programovacom jazyku C ako funkcia *dynopt* na hľadanie optimálnych profilov pre zadaný proces vzhľadom na účelovú funkciu a obmedzení typu rovnosti a nerovnosti.

V práci bol prezentovaný jeden vzorový príklad, ktorého výsledky sú na jednotlivých obrázkoch, na ktorých vidieť štruktúru kolokačných bodov. Štruktúru intervalov je najlepšie vidieť z priebehu riadiacich veličín, ktoré sú na okrajoch intervalov nespojité. Stavové veličiny majú spojitý priebeh, čo napokon vidieť aj z obrázkov.

Funkcia bola testovaná na 9 problémoch z literatúry. Problémy boli volené tak, aby ukázali schopnosť *dynoptu* riešiť problémy rôznej zložitosti.

Treba spomenúť, že všetky získané optimálne hodnoty sú len lokálne.

Literatúra

Bryson, A. E. a Ho, Y. Applied Optimal Control. Blaisdell Publishing Company, Massachussetts, 1969.

Cuthrell, J. E. a Biegler, L. T. On the Optimization of Differential - Algebraic Process Systems. AIChE Journal, 33, 1257 – 1270, 1987.

Han, S. P. A Globally Convergent Method for Nonlinear Programming. 22, 297, 1977.

Logsdon, J. S. a Biegler, L. T. Accurate Solution of Differential – Algebraic Optimization Problems. Chem. Eng. Sci., (28), 1628 – 1639, 1989.

Logsdon, J. S. a Biegler, L. T. Decomposition Strategies for Large – Scale Dynamic Optimization Problems. Chem. Eng. Sci., 47(4), 851 – 864, 1992.

Luus, R. Application of Iterative Dynamic to State Constrained Optimal Control Problems. Hung. J. Ind. Chem., 19, 245 – 254, 1991.

Powell, M. J. D. Nonlinear Programming 3, kapitola The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations. Academic Press, 1987.

Rajesh, J., Gupta, K., Kusumakar, H., Jayaraman, V. a Kulkarni, B. Dynamic Optimization of Chemical Processes using Ant Colony Framework. Computers and Chemistry, 25, 583 – 595, 2001.

Sargent, R. W. H. a Sullivan, G. R. The Development of an Efficient Optimal Control Package. 1977.

Villadsen, J. a Michelsen, M. L. Solution of Differential Equation Models by Polynomial Approximation. Prentice Hall, 1978.

Čižniar, M., Fikar, M. a Latifi, M. User's Guide for Matlab Dynamic Optimisation Code DYNOPT, 2005.

A. Víteček a M. Vítečková. Optimální systémy řízení. VŠT – Technická Univerzita Ostrava, 2002

J. E. Cuthrell a L. T. Biegler. On the optimization and solution methods for batch reactor control profiles. Computers chem.. Engng., 13(1/2), 49 – 62, 1989.

S. Storen a T. Hertzberg. The Sequential Linear Quadratic Programming Algorithm for Solving Dynamic Optimization Problems – a review. Computers chem.. Engng., 19, S495 – S500, 1995.

P. Tanarkit a L. T. Biegler. A nested, simultaneous approach for dynamic optimization problems – II: the outer problem. Computers chem.. Engng., 21, 1365 – 1388, 1997.

E. Balsa-Canto, J. R. Banga, A. A. Alonso, V. S. Vassiliadis. Dynamic optimization of chemical and biochemical processes using restricted second – order information. Computers and Chemical Engineering, 25, 539 – 546, 2001.

C. J. Goh a K. L. Teo. Control parametrization: A unified approach to optimal control problems with general constraints. Automatica, 24, 3 – 18, 1988.

J. Mikleš a V. Hutla. Teória automatického riadenia. Alfa, 1986.

R. Luus. Optimal control by dynamic programming using systematic reduction in grid size. Int. J. Control, 51, 995 – 1013, 1990.

S. A. Dadeo a K. B. McAuley. Dynamic optimization of constrained chemical engineering problems using dynamic programming. Computers chem.. Engng., 19, 513 – 525, 1995.

R. Luus. Application of dynamic programming to high – dimensional non – linear optimal control problems. Int. J. Control, 52(1), 239 – 250, 1990.

L. T Biegler a I. E. Grossmann. Retrospective on optimization. Computers and Chemical Engineering, 28, 1169 – 1192, 2004.

P. Brock, K. Madesn, H. B. Nielsen. Robust C subroutines for non – linear optimization, Technical University of Denmark, IMM-Technical Report, 21, 2004.