VILNIAUS UNIVERSITETAS
MATEMATIKOS INFORMATIKOS FAKULTETAS
DIFERENCIALINIŲ LYGČIŲ IR SKAIČIUOJAMOSIOS MATEMATIKOS KATEDRA

Mindaugas Dailyda

TIESINĖS DINAMINĖS SISTEMOS SU VALDOMOMIS PRADINĖMIS SĄLYGOMIS OPTIMIZAVIMAS

Magistro darbas

Darbo vadovas
doc. V. Daukšas

Vilnius
2006

Turinys

Įvadas

Darbe sprendžiamas tiesinės dinaminės sistemos optimizavimo uždavinys (tiesinis n-matis optimaliojo valdymo uždavinys su kintamomis pradinėmis sąlygomis). Daugelį netiesinių optimaliojo valdymo uždavinių galima apytiksliai pakeisti tiesinių sistemų optimizavimo uždaviniais (darbe neatsižvelgiama į perėjimą nuo vienos optimizavimo sistemos prie kitos). Darbe pasinaudota matematinio optimaliojo valdymo procesų teorijos pagrindiniais rezultatais. Būtina pastebėti, kad optimaliojo valdymo teorija remiasi dviem vystomomis kryptimis: maksimumo principu (Pontriagino) ir dinaminiu programavimu (R. Belmano). Šiame darbe nagrinėsime tiesinę dinaminę sistemą

$$c^T x(T) \to \max \tag{1}$$

$$\dot{x}(t) = Ax(t) + bu(t), t \in [0,T] \tag{2}$$

$$g_* \le Hx(T) \le g^* \tag{3}$$

$$u_* \le u(t) \le u^*, t \in [0,T]$$

su pradinėmis valdomomis sąlygomis

$$x(0) = x_0 + Dq\ ,\ q_* \le q \le q^*\ . \tag{4}$$

Sprendžiant šį uždavinį, reikia rasti optimalųjį valdymą $u^0(t) \in U$, optimalų vektorių q^0 ir juos atitinkančią optimaliąją trajektoriją $x^0(t) \in C[0..T]$, tenkinančią (2) lygčių sistemą, (3) kraštines sąlygas ir (4) pradines sąlygas . Optimalus valdymas ieškomas gabalais tolydžių funkcijų klasėje.

Darbo tikslas yra tiesinių dinaminių sistemų su valdomomis pradinėmis sąlygomis optimizavimas, tiesinio programavimo metodais – Simplekso metodu ir adaptyviniu metodu.

Pirmame skyriuje trumpai apžvelgiamos tiesinės dinaminės sistemos. Antrame skyriuje nagrinėjama tiesinė dinaminė sistema su valdomomis pradinėmis sąlygomis.

Sekančiame skyrelyje parodome kaip, n-matę tiesinę dinaminę sistemą su valdomomis pradinėmis sąlygomis, suvedame į tiesinio programavimo uždavinį. Ketvirtame skyriuje apžvelgiamas adaptyvinis metodas. Sekančiame skyriuje pateikiami tiesinės dinaminės sistemos sprendimo rezultatai, t.y. optimaliojo valdymo grafikai. Paskutiniame skyriuje pateikiama Simplekso ir adaptyvinio metodų palyginimas.

1. Tiesinės dinaminės sistemos

Šiame skyrelyje aptariamas vienas iš pagrindinių optimaliojo valdymo uždavinių - tiesinės dinaminės sistemos. Pateikiami optimaliojo valdymo teorijos pagrindai, kurie bus reikalingi tiesinės dinaminės sistemos sprendimui su pradinėmis valdomomis sąlygomis.

Tarkime, kad turime [2] tiesinę diferencialinių lygčių sistemą

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^{n} a_{ij}(t)x_j + b_i(u_1(t),...,u_r(t),t)$$

Pažymėkime $x = (x_1,...,x_n)^T$, $u = (u_1,...,u_r)$, $A = \{a_{ij}\}_1^n$, $b = (b_1,...,b_n)^T$.
Tuomet gausime lygtį

$$\frac{dx(t)}{dt} = A(t)x(t) + b(u(t),t) \qquad (1).$$

Diferencialinių lygčių sistemą (1), vadinsime tiesine diferencialinių lygčių sistema u(t) atžvilgiu, jei $b(u,t) = B(t)u(t)$ t.y., kai sistema (1) turi pavidalą

$$\frac{dx(t)}{dt} = A(t)x(t) + B(t)u(t). \qquad (2)$$

Sistema (2), kai ji yra tiesinė x(t) ir u(t) atžvilgiu, dar vadinama tiesine dinamine sistema.

Jeigu matricos $A(t)$, $B(t)$ nepriklauso nuo laiko t tai (2) sistemą vadinsime stacionariąja (toliau analizuojama tik stacionarios tiesinės dinaminės sistemos) [2] ir žymėsime

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t). \qquad (3)$$

Aišku [2], kad kiekvienai gabalais tolydžiai funkcijai $u(t) \in U$, $t \geq t_0$, egzistuoja vienintelis tolydus sprendinys $x(t)$, tenkinantis (1) diferencialinių lygčių ir pradines sąlygas

$$x(t_0) = x_0 \qquad (4)$$

kur x_0 n – matis vektorius, t_0 - bet kokia reikšmė.

Valdymą $u(t) \in U, t \geq t_0$ vadinsime leistinu [2], jeigu u(t) yra gabalais tolydi funkcija.

Dabar nagrinėkime homogeninę tiesinę diferencialinių lygčių sistemą [2]

$$\frac{dx(t)}{dt} = Ax(t) \tag{5}$$

Tegu $x^k(t)$, $k = 1,...,n$, $t \geq t_0$ yra tiesinės diferencialinių lygčių sistemos (4) su pradinėm sąlygomis

$$x_1^k(t_0) = 0, x_2^k(t_0) = 0..., x_{k-1}^k(t_0) = 0,\ x_k^k(t_0) = 1,\ x_{k+1}^k(t_0) = 0,..., x_m^k(t_0) = 0$$

sprendinys. Matricą F(t) sudaro vektoriai $x^k(t)$, $k = 1,...,n$. Matrica F(t) vadinama fundamentaliaja sprendinių matrica normuota taške $t = t_0$. Pagal apibrėžimą, matrica F(t) tenkina lygtį

$$\frac{dF(t)}{dt} = AF(t) \tag{6}$$

ir yra vienetinė matrica taške $t = t_0$. Kiekvieną lygties (3) sprendinį x(t), $t \geq t_0$, [2], atitinkantį gabalais tolydžią funkciją $u(t)$, $t \geq t_0$, ir tenkinantį pradines sąlygas $x(t_0) = x_0$, galima užrašyti pavidalu

$$x(t) = F(t)x(t_0) + \int_{t_0}^{t} F(t)F^{-1}(\tau)Bu(t)d\tau \tag{7}$$

kur F(t) fundamentalioji matrica tiesinių diferencialinių lygčių sistemai (5).

Parodysime, kad funkcija (7) yra lygties (3) sprendinys, tenkinantis (4) pradines sąlygas [2]. Kai $t = t_0$, turi būti tenkinama (7) lygtis iš čia turime, kad $x(t_0) = x_0$. Diferencijuojame (7) pagal laiką t

$$\frac{dx(t)}{dt} = \frac{dF(t)}{dt}F^{-1}(t_0)x(t_0) + \frac{dF(t)}{dt}\int_{t_0}^{t} F^{-1}(\tau)Bu(t)d\tau + F(t)F^{-1}(t)Bu(t)d\tau$$

Naudodamiesi tuo, kad $\frac{dF(t)}{dt} = AF(t)$ gauname

$$\frac{dx(t)}{dt} = AF(t)F^{-1}(t_0)x(t_0) + AF(t)\int_{t_0}^{t} F^{-1}(\tau)Bu(t)d\tau + Bu(t) = Ax(t) + Bu(t)$$

Iš to išplaukia, kad (7) funkcija yra lygties (3) sprendinys.

Yra įrodoma [2], kad stacionari sistema (3) yra valdoma, jeigu $rang\{b, Ab, .., A^{n-1}b\} = n$.

Tarkime, kad nagrinėjama stacionari diferencialinių lygčių sistema (3), tuomet turime, kad [1]

$$F(t) = e^{A(t-t_0)} = \sum_{i=0}^{\infty} \frac{(A(t-t_0))^i}{i!},$$

kai A yra pastovioji matrica t.y. nepriklauso nuo t. Turėdami sistemos $\dot{x}(t) = Ax(t)$ fundamentaliąją matricą F(t), galime užrašyti (3) sistemos sprendinį tenkinantį pradines sąlygas $x(t_0) = x_0$,

$$x(t) = F(t)x_0 + \int_0^t F(t)F^{-1}(\tau)Bu(t)d\tau \ , \ t \in [0, T].$$

Atlikę pertvarkymus gaunama, kad

$$x(t) = F(t)(x_0) + \int_0^t F(t)F^{-1}(\tau)Bu(t)d\tau = F(t)x_0 + F(t)\int_0^t F^{-1}(\tau)Bu(t)d\tau \ .$$

2. **Tiesinė dinaminė sistema su valdomomis pradinėmis sąlygomis**

Tarkime, kad duotas tiesinės dinaminės sistemos

$$\begin{aligned} & c^T x(T) \to \max \\ & \dot{x}(t) = Ax(t) + Bu(t), t \in [0,T] \\ & \qquad (1) \\ & g_* \le Hx(T) \le g^* \qquad (2) \\ & u_* \le u(t) \le u^*, t \in [0,T] \end{aligned}$$

su pradinėmis valdomomis sąlygomis

$$x(0) = x_0 + Dq\ ,\ q_* \le q \le q^* \qquad (3)$$

optimizavimo uždavinys. Kur A - $n \times n$ yra pastovioji matrica, H - $m \times n$ - pastovioji matrica, D - $n \times n$ - pastovioji matrica, B - $n \times r$ $x = (x_1,..,x_n)^T$, $c = (c_1,..,c_n)$, $q = (q_1,..,q_l)^T$, $q_* = (q_{*1},..,q_{*l})^T$, $q^* = (q_1^*,..,q_l^*)^T$, $g_* = (g_{*1},..,g_{*m})^T$, $g^* = (g_1^*,..,g_m^*)^T$ vektoriai. Šiame uždavinyje reikia rasti optimaliąją valdymo funkciją $u(t) = (u_r(t),...,u_r(t)) \in U$, optimalų vektorių $q^0 \in R^l$ ir juos atitinkančią tolydžią trajektoriją $x^0(t) \in X$, tenkinančią (1) lygčių sistemą ir (2) kraštines sąlygas, (3) pradines sąlygas.

Paėmus kokią nors konkrečią q reikšmę gaunamos pradinės sąlygos

$$x(0) = x_0 + Dq = x_{01}\ .$$

Todėl, kad ir kokį q beparinktume trajektoriją x(t) galime išreikšti 1.7. Koši formule.

Užrašome (2) sistemos sprendinį , tenkinantį pradines sąlygas $x(0) = x_0 + Dq$

$$x(t) = F(t)(x_0 + Dq) + \int_0^t F(t)F^{-1}(\tau)Bu(t)d\tau\ ,$$

kai t=T

$$x(T) = F(T)(x_0 + Dq) + \int_0^T F(T)F^{-1}(\tau)Bu(t)d\tau\ .$$

Atlikę pertvarkymus matome, kad

$$x(t) = F(t)(x_0 + Dq) + \int_0^t F(t)F^{-1}(\tau)Bu(t)d\tau = F(t)(x_0 + Dq) + F(t)\int_0^t F^{-1}(\tau)Bu(t)d\tau .$$

Tarkime, kad nagrinėjama stacionari diferencialinių lygčių sistema (1). Sudarykime Hamiltono funkciją stacionariai sistemai (1) $H(x,\psi,u) = \psi'(Ax + Bu)$. Turime, kad $\dot{\psi} = -A^T\psi$ ir remiantis maksimumo principu [2] gauname, kad optimalus valdymas yra lygus

$$u_i(t)^0 = \begin{cases} u_i^*, jeigu [\psi^{0^T} B]_i > 0, \\ u_i^*, jeigu [\psi^{0^T} B]_i < 0 \end{cases}.$$

Vadinasi, optimalus valdymas $u(t)^0 \in U$ bus iš gabalais pastovių funkcijų klasės.

Tarkime, kad $u(t) \in U \subset R$ vienmatėė funkcija ir $B = b \in R^n$. Tuomet remiantis Hamiltono principu [2] optimalus valdymas $u(t)^0 \in U$ įgyja reikšmes tik u_*, u^*. Uždavinio sprendimas susiveda į valdymo funkcijos persijungimo taškų $(\tau_1,...,\tau_r)$, $\tau_j \in [0...T]$ paiešką. Persijungimo taškuose optimalaus valdymo funkcijos $u(t)^0 \in U$ reikšmės yra u_* , u^* ir keičia viena kitą . Norint surasti persijungimo taškų vietą intervale mes suskaidome intervalą $[0,T]$ į smulkesnius intervalus $[0,T_1)$, $[T_1,T_2)$,..., $[T_{j-1},T_j)$,..., $[T_{r-1},T_r)$, $[T_{N-1},..,T]$ ir $u(t) = u_j = const$, kai $t \in [T_{j-1},...,T_j]$. Tiesinę dinaminę sistemą pakeičiame tiesinio programavimo uždaviniu.

Integralas $\int_0^T F^{-1}(\tau)b(u(\tau),\tau)d\tau$ pakeičiamas integralų suma ir bus lygus

$\sum_{j=0}^{N-1}\int_{T_j}^{T_{j+1}} F^{-1}(\tau)bu_j d\tau$, kai u_j, $u_* \le u_j \le u^*$ yra ieškomi dydžiai. Atliekami pertvarkymai ir gaunama, kad $\sum_{j=0}^{N-1}\int_{T_j}^{T_{j+1}} C(\tau)d\tau u_j$, kai $C(\tau) = F^{-1}(\tau)b$. Lieka apskaičiuoti integralus

$c_j = \int_{T_{j-1}}^{T_j} C(\tau)d\tau$, $j = 0,...,N$.

Atlikę tiesinio dinaminio uždavinio diskretizaciją, turime tiesinio programavimo uždavinį:

$$\sum_{i=1}^{N} c^T \int_{T_{i-1}}^{T_i} F(T)F^{-1}(\tau)bd\tau u_i + c^T F(T)Dq \to \max$$

$$g_* \le \sum_{i=1}^{N} H \int_{T_{i-1}}^{T_i} F(T)F^{-1}(\tau)bd\tau u_i + HF(T)Dq + x_0 HF(t) \le g^*$$

$$q_{i*} \le q_i \le q_i^*,\ i = 1,..,l$$

$$u_{i*} \le u_i \le u_i^*,\ i = 1,..,N$$

Šiame uždavinyje nežinomieji yra $u_1,..u_N$, $q_1,..q_l$. Išsprendus tiesinio programavimo uždavinį, randamas dinaminės sistemos optimalus valdymas $u(t)^0 \in U$ ir optimalus vektorius q^0. Jeigu kuris nors u_i nelygus u_{i*} arba u_i^*, tai persijungimo taškas patenka į intervalo $[T_{i-1}, T_i)$ vidų. Mūsų pateikta diskretizacija yra apytikslė. Dinaminės sistemos diskretizacija ir jos sprendimas atliekamas „Maple“ programa, kurios programinis kodas pateikiamas 9.3. priede.

3. Tiesinės n-matės dinaminės sistemos su valdomomis pradinėmis sąlygomis optimizavimas

Tarkime, kad intervale $t \in [0,..,T]$ turime uždavinį: rasti valdymo funkciją $u(t) \in U$ iš gabalais pastovių funkcijų klasės, kad $X(T) - > \max$,

$$X^{(n)}(t) = u(t),$$

$$X(0) = q_1,\ \dot{X}(0) = \ddot{X}(0) = ... = X^{(n-1)}(0) = 0,$$

$$\dot{X}(T) \le 1, \ddot{X}(T) \le 1,.., X^{(n-1)}(T) \le 1,$$

$$-1 \le \dot{X}(T), -1 \le \ddot{X}(T),.., -1 \le X^{(n-1)}(T),$$

$$-1 \le u(t) \le 1,\ -1 \le q_1 \le 1,\ t \in [0,T].$$

Įvedame ketinį $x_1 = X,..,x_n = X^{(n-1)}$ ir tuomet gauname teisinę dinaminę sistemą

$$\begin{aligned} & c^T x(T) \to \max \\ & \dot{x}(t) = Ax(t) + Bu(t), t \in [0,T] \\ & \qquad (1) \\ & g_* \le Hx(T) \le g^* \\ & u_* \le u(t) \le u^*, t \in [0,T] \end{aligned}$$

su pradinėmis valdomomis sąlygomis

$$x(0) = x_0 + Dq,\ q_* \le q \le q^*. \qquad (2)$$

A - $n \times n$ yra pastovioji matrica, H - $n-1 \times n$ - pastovioji matrica, D - $n \times n$ - pastovioji matrica, B - $n \times 1$ $x = (x_1,..,x_n)^T$, $c = (c_1,..,c_n)$, $b = (b_1,..,b_n)^T$, $q = (q_1,..,q_n)^T$, $q_* = (q_{*1},..,q_{*_n})^T$, $q^* = (q_1^*,..,q_n^*)^T$, $g_* = (g_{*1},..,g_{*_{n-1}})^T$, $g^* = (g_1^*,..,g_{n-1}^*)^T$ vektoriai (tiesinės dinaminės sistemos valdomas pradines sąlygas (2), šiuo atveju, galima užrašyti ir paprastesniu pavidalu). Turime, kad

$$A=\begin{pmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & .. \\ 0 & 0 & 0 & 0 & ... & 1 \\ 0 & 0 & 0 & 0 & ... & 0 \end{pmatrix},$$

$$D=\begin{pmatrix} 1 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & .. \\ 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & ... & 0 \end{pmatrix},$$

$$H=\begin{pmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0.. \\ ... & ... & ... & ... & ... & .. \\ 0 & 0 & 0 & 0 & ... & 1 \end{pmatrix},$$

$x_0 = (0,..,0)^T$, $B = (0,..,0,1)^T$, $-1 \le u(t) \le 1$, $-1 \le q_1 \le 1$, $g^* = (1,..,1)^T$,

$g_* = (-1,..,-1)^T$, $c = (1,..,1)^T$. Randam sistemos (1) fundamentaliąją matricą (

$$F(t) = e^{A(t-t_0)} = \sum_{i=0}^{\infty} \frac{(A(t-t_0))^i}{i!})$$

$$F(t)=\begin{pmatrix} 1 & t & t^2/2 & t^3/6 & ... & t^{n-1}/(n-1)! \\ 0 & 1 & t & t^2/2 & ... & t^{n-2}/(n-2)! \\ 0 & 0 & 1 & t/2 & ... & t^{n-3}/(n-3)! \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & t \\ 0 & 0 & 0 & 0 & ... & 1 \end{pmatrix},$$

ir jos atvirkštinė bus lygi

$$F^{-1}(t)=\begin{pmatrix} 1 & -t & t^2/2 & -t^3/6 & ... & (-1)^{2n}t^{n-1}/(n-1)! \\ 0 & 1 & -t & t^2/2 & ... & (-1)^{2n-1}t^{n-2}/(n-2)! \\ 0 & 0 & 1 & -t & ... & (-1)^{2n}t^{n-3}/(n-3)! \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & -t \\ 0 & 0 & 0 & 0 & ... & 1 \end{pmatrix}..$$

ir dabar galima užrašyti tiesinės dinaminės sistemos sprendinį

$$x(t)=F(t)(x_0+Dq)+\int_0^t F(t)F^{-1}(\tau)Bu(t)d\tau = F(t)(x_0+Dq)+F(t)\int_0^t F^{-1}(\tau)Bu(t)d\tau$$

tenkinantį pradines valdomas sąlygas. Ir kaip ankstesniame skyrelyje diskretizuoti, suskaidant intervalą į mažesnius intervaliukus ir gaunant tiesinio programavimo uždavinį

$$\sum_{i=1}^{N} c^T \int_{T_{i-1}}^{T_i} F(T)F^{-1}(\tau)bd\tau u_i + c^T F(T)Dq \to \max,$$

$$\begin{pmatrix} -1 \\ -1 \\ -1 \\ ... \\ -1 \\ -1 \end{pmatrix} \le \begin{pmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & 1 \\ 0 & 0 & 0 & 0 & ... & 0 \end{pmatrix}.. x(T) \le \begin{pmatrix} 1 \\ 1 \\ 1 \\ ... \\ 1 \\ 1 \end{pmatrix}, \; -1 \le q_1 \le 1, \; -1 \le u_1,..,u_N \le 1,$$

kurį jau galima spręsti tiesinio programavimo metodais.

4. Adaptyvinis metodas

Daugelis taikomųjų programų leidžia spręsti tiesinio programavimo uždavinius suvedus juos į kanoninį tiesinio programavimo uždavinį. Bet suvedus tiesinio programavimo uždavinį į kanoninį pavidalą prarandama labai svarbi informacija. Praktika parodė [7], kad metodo efektyvumas priklauso nuo uždavinio struktūros ir pradinės informacijos apie modelį. Šiame skyriuje trumpai apžvelgsime adaptyvinį metodą, kuris paimtas iš [7] knygos. Adaptyvinis metodas realizuojamas „Maple“ programa 9.1. priede.

Spręskime tiesinio programavimo uždavinį pavidalu:

$$c^T x \to \max$$
$$b_* \le Ax \le b^* \qquad (1)$$
$$d_* \le x \le d^* \ .$$

Kur x, c, d_*, d^* - n- mačiai vektoriai, b^*, b_* m-mačiai vektoriai, A - $m \times n$ - matrica. Reikia rasti optimalų x^0 tokį, kad tenkintų nelygybių $b_* \le Ax^0 \le b^*$, $d_* \le x^0 \le d^*$ sistemas ir $c^T x^0 \to \max$ įgytų didžiausią reikšmę.

Uždavinio (1) atrama vadinsime neelementariąją matricą

$$A_{AT} = A(I_{AT}, J_{AT})$$

sudarytą iš elementų a_{ij}, $i \in I_{AT}$, $j \in J_{AT}$, matricos $A = A(I,J)$, $I = (1,..m)$, $J = (1,..n)$. Porą $\{x, A_{AT}\}$ su pradiniu planu ir atrama vadinsime uždavinio (1) atraminiu planu [7]. Pažymėkime neatraminių indeksų aibes $I_H = I \setminus I_{AT}$, $J_H = J \setminus J_{AT}$.

Tegu $\{x, A_{AT}\}$ yra pradinis atraminis planas ir apibrėžkime vektorius

$$w^* = w^*(x) = b^* - Ax,$$
$$w_* = w_*(x) = b_* - Ax,$$
$$\Delta^T = \Delta^T(J) = u^T A(I_{AT}, J) - c^T$$

, ir potencialo vektorių

$$u^T = u^T(I_{AT}) = c_{AT} A_{AT}^{-1}.$$

Atraminį planą $\{x, A_{AT}\}$ laikysime optimaliu jeigu:

$$u(I_{*AT}) \le 0,\ u(I^*_{AT}) \ge 0,\ u(I^{\sim}_{AT}) = 0,\ \Delta(J_{*H}) \ge 0,\ \Delta(J^*_H) \le 0,\ \Delta(J^{\sim}_H) = 0,$$

čia $I_* = I_*(x) = \{i \in I : a_i^T x = b_{*i}\}$, $I^* = I^*(x) = \{i \in I : a_i^T x = b_i^*\}$, $I^{\sim} = I \setminus (I^* \cup I_*)$, $J^* = J^*(x) = \{j \in J : x_j = d_j^*\}$, $J_* = J_*(x) = \{j \in J : x_j = d_{*j}\}$, $J^{\sim} = J \setminus (J^* \cup J_*)$, $I_{*AT} = I_* \cap I_{AT}$, $J_{*H} = J_* \cap J_H$, kai $a_i = A(i, J)$. Šios optimalumo sąlygos įrodomos [7] knygoje.

Jeigu pasirinktas atraminis planas netenkina optimalumo sąlygų turime pereiti prie naujo plano ir naujos atramos. Naujasis planas $\bar{x}$ bus lygus

$$\bar{x} = x + \Theta l,$$

l-leistinoji kryptis, Θ-didžiausias maksimalus žingsnis.

Vektorius $l = \{l(J_{AT}, J_H)\}$ bus sudarytas iš:

$$l_j = d_{*j} - x_j,$$

jeigu $\Delta_j > 0$; $l_j = d_j^* - x_j$, jeigu $\Delta_j < 0$; $l_j = 0$, jeigu $\Delta_j = 0$, $j \in H$;

$$l(J_{AT}) = A_{AT}^{-1} w(I_{AT}) - A_{AT}^{-1} A(I_{AT}, J_H) l(J_H),$$

kai $w(I_{AT}) = \{w_i, i \in I_{AT}\}$, $w_i = w_{*i}$, jeigu $u_i < 0$; $w_i = w_i^*$, jeigu $u_i > 0$; $w_i = 0$, jeigu $u_i = 0$, $i \in I_{AT}$.

Leistiną žingsnį apskaičiuosime taip:

$$\Theta = \min\{1, \Theta_{i_0}, \Theta_{j_0}\},$$

kur $\Theta_{j_0} = \min(\Theta_j), j \in J_{AT}$; $\Theta_j = (d_{*j} - x_j)/l_j$, jeigu $l_j < 0$; $\Theta_j = (d_j^* - x_j)/l_j$, jeigu $l_j > 0$; $\Theta_j = \infty$, jeigu $l_j = 0$;

$\Theta_{i_0} = \min(\Theta_i), i \in I_H$; $\Theta_i = w_{*i} / a_i^T l$, jeigu $a_i^T l < 0$; $\Theta_i = w_i^* / a_i^T l$, jeigu $a_i^T l > 0$; $\Theta_i = \infty$, jeigu $a_i^T l = 0$ [7].

Dabar sukonstruosime naują atramą $\bar{A}_{AT}$. Pirma apibrėžkime naujus dydžius. Pažymėkime, kad

$$\sigma_i = \begin{cases} -u_i / z_i, jegu - u_i z_i < 0 \\ 0, jeigu \quad u_i = 0, z_i > 0, i \notin I^*, arba \quad u_i = 0, z_i < 0, i \in I^*, i \in I_{AT} \\ \infty, visais \quad kitais \quad atvejais \end{cases}$$

$$\sigma_j = \begin{cases} -\Delta_j / z_j, jegu - \Delta_j z_j < 0 \\ 0, jeigu \quad \Delta_i = 0, x_j \neq d_{*j}, z_j > 0, arba \quad \Delta_i = 0, z_i < 0, x_j \neq d_j^* \\ \infty, visais \quad kitais \quad atvejais \end{cases}$$

Apibrėžkime naują dydį

$$\sigma = \min(\sigma_{i*}, \sigma_{j*}),$$

$$\sigma_{i*} = \min(\sigma_i), i \in I_{AT},$$

$$\sigma_{j*} = \min(\sigma_j), j \in J_H.$$

Dabar galimi du atvejai. Tegu $\Theta = \Theta_{i_0} < 1$, tai

$$z^T(J_H, I_{AT}) = ke_{i_0}^T \{A(I_H, J_H) - A(I_H, J_{AT}) A_{AT}^{-1} A(I_{AT}, J_H), -A(I_H, J_{AT}) A_{AT}^{-1}\},$$

kai k=1, jeigu $a_{i_0}^T \bar{x} = b_{i_0}^*$, k=-1, jeigu $a_{i_0}^T \bar{x} = b_{*i_0}$. Dabar $\Theta = \Theta_{j_0} < 1$, tai

$$z^T(J_H, I_{AT}) = ke_{j_0}^T \{A_{AT}^{-1} A(I_{AT}, J_H), A_{AT}^{-1}\},$$

kai k=1, jeigu $\bar{x}_{j_0} = d_{*j_0}$, k=-1, jeigu $\bar{x}_{j_0} = d_{j_0}^*$.

Dabar keisime atramą, tuomet galimi keturi variantai.

1. Tegu $\Theta = \Theta_{i_0} < 1$, $\sigma = \sigma_{i*}$, tada naujoji atraminių indeksų aibė bus lygi

   $\bar{I}_{AT} = (I_{AT} \setminus i_*) \cup i_0$, $\bar{J}_{AT} = J_{AT}$.

2. $\Theta = \Theta_{i_0} < 1$, $\sigma = \sigma_{j*}$, tai $\bar{I}_{AT} = I_{AT} \cup i_0$, $\bar{J}_{AT} = J_{AT} \cup j_0$

3. $\Theta = \Theta_{j_0} < 1$, $\sigma = \sigma_{i*}$, tai $\bar{I}_{AT} = I_{AT} \setminus i_*$, $\bar{J}_{AT} = J_{AT} \setminus j_0$

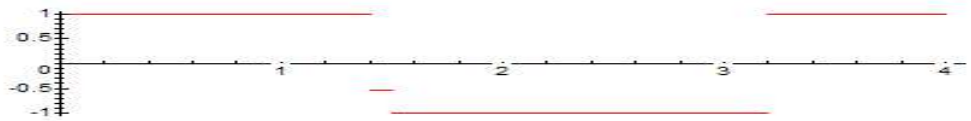
4. $\Theta = \Theta_{j_0} < 1$, $\sigma = \sigma_{j*}$, tai $\bar{I}_{AT} = I_{AT}$, $\bar{J}_{AT} = (J_{AT} \setminus j_0) \cup j_*$.[3]

Šioje vietoje tepateikiama atvirkštinės atramos pakeitimas nauja atvirkštine atrama. Pastebime, kad atramą galime keisti, ne tik keisdami atramos stulpelį, bet ir sumažinti, padidinti atramos matmenis. Atrama gali būti sudaryta net iš 1×1 matricos ir iš $m \times m$ matricos, o pradinio plano elementai gali visi kartu kisti, vykstant iteracijai. Tuo adaptyvinis metodas labiausiai skiriasi nuo Simplekso, ir yra bendresnis atvejis.

Priede 9.1. pateikiama programa automatiškai apdorojanti parinktą tiesinio programavimo uždavinį ir suskaičiuoja, kiek iteracijų atliekama.

5. Skaitinis eksperimentas

Parinkime, kad n=3 ir diskretizuokime trečiame skyriuje nagrinėtą tiesinį optimaliojo valdymo uždavinį. Fiksuojame, kad T=4 (intervalo ilgis) ir intervalą skaidome į N=40 lygių dalių (tiesinės dinaminės sistemos diskretizacija buvo pateikta antrame skyriuje). Išsprendus tiesinio programavimo uždavinį (9.2. priede pateikiama šios diskretizacijos rezultatas) turime, kad valdymo funkcija turi pavidalą

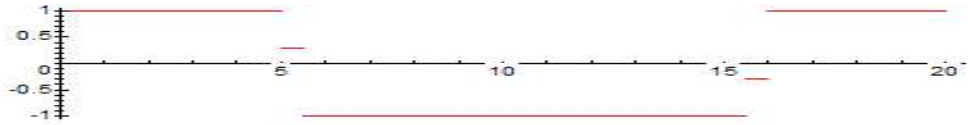


Optimalus $q_1^0 = -1$.

Matome, kad valdymo funkcija turi du persijungimo taškus. Padidinkime intervalą iki T=10, tuomet valdymo funkcija yra

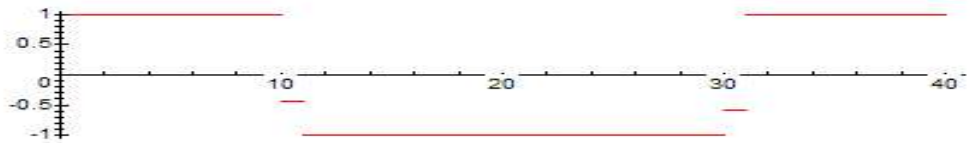
Optimalus $q_1^0 = -1$.

Ir toliau, kai T=20:



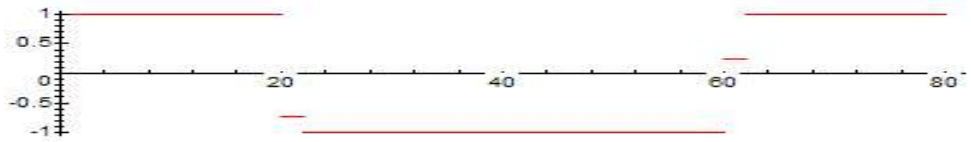
Optimalus $q_1^0 = -1$.

T=40:



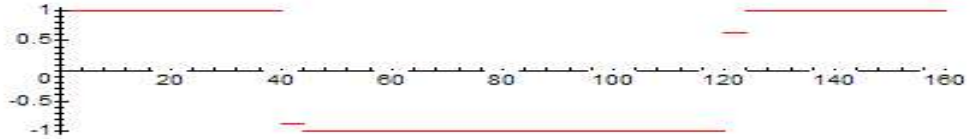
Optimalus $q_1^0 = -1$.

T=80:



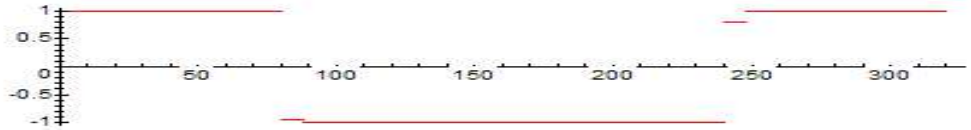
Optimalus $q_1^0 = -1$.

T=160:



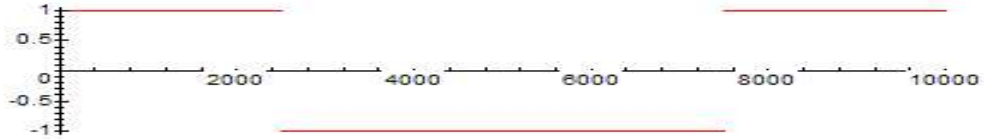
Optimalus $q_1^0 = -1$.

T=320:



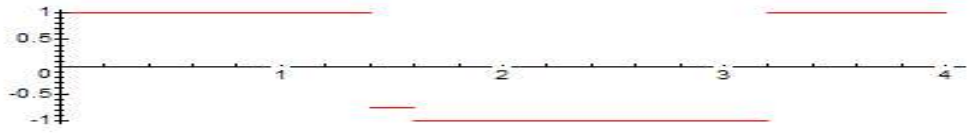
Optimalus $q_1^0 = -1$.

O kai T=10000 tai turime, kad

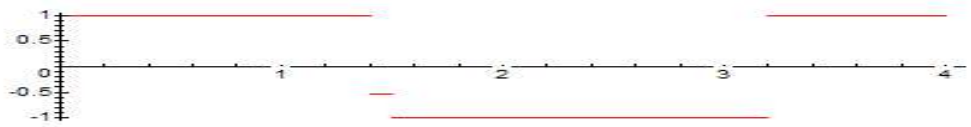


Iš gautų rezultatų galime daryti išvadą, kad turime du persijungimo taškus ir jie juda ilgėjant intervalui link pirmo ketvirčio ir trečio ketvirčio.

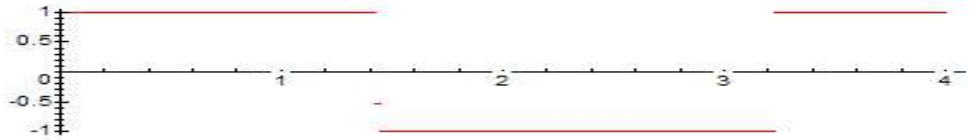
Dabar įsitikinsime, kad persijungimo taškai "lieka" savo vietoje, keičiant intervalo skaidymo žingsnį. Fiksuokime, kad intervalą T=4 skaidome į 20 lygių dalių, tai valdumo funkcija



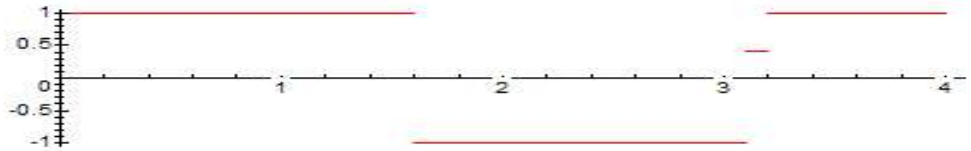
Dabar į 40 dalių:



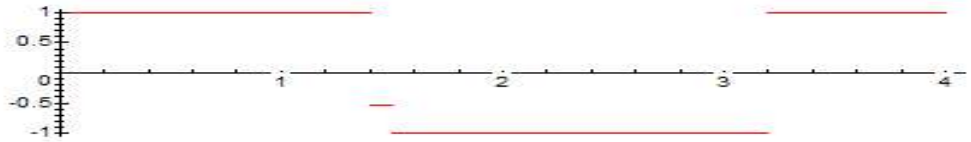
ir 160 dalių:



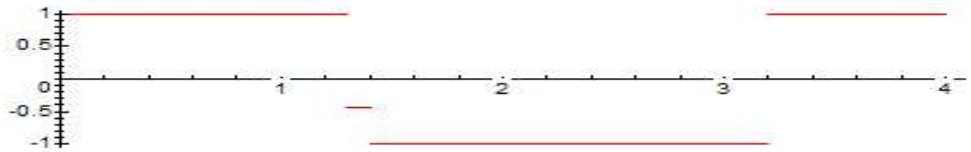
Optimizuokime dinaminę sistemą, kai T=4,o intervalą skaidome į 40 lygių dalių. Keiskime kraštines sąlygas g^*, g_*. Kai $g^* = (2,2,2)^T$, $g_* = (-2,-2,-2)^T$ gausime, kad valdymo funkcija



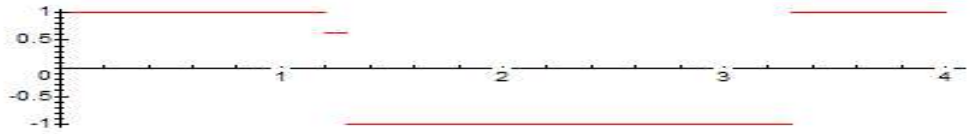
Kai $g^* = (1,1,1)^T$, $g_* = (-1,-1,-1)^T$ turime



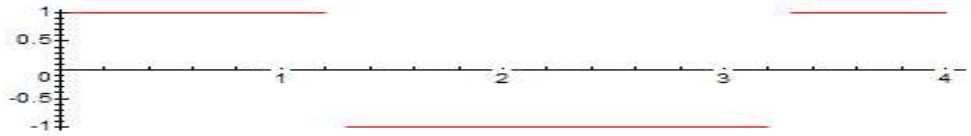
kai $g^* = (0.5,0.5,0.5)^T$, $g_* = (-0.5,-0.5,-0.5)^T$



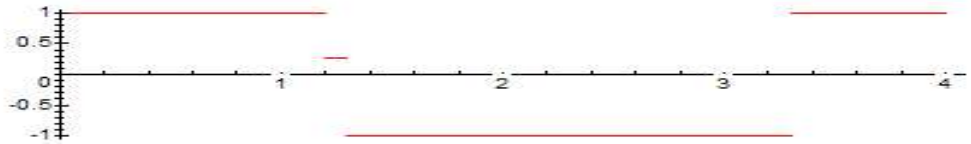
kai $g^* = (0.1,0.1,0.1)^T$, $g_* = (-0.1,-0.1,-0.1)^T$



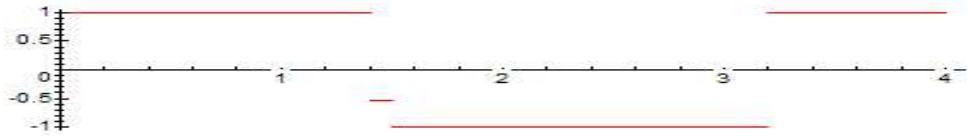
kai $g^* = (0.001,000.1,000.1)^T$, $g_* = (-0.0001,-0.0001,-0.0001)^T$



Liko dar netikrinta, kokią įtaką turi kraštinė valdoma sąlyga. Jeigu laikysime, kad T=4 ir, kad intervalą skaidome į 40 lygių dalių ir $g^* = (1,1,1)^T$, $g_* = (-1,-1,-1)^T$, $0 \le q(t) \le 0$ turime, kad valdymo funkcija:

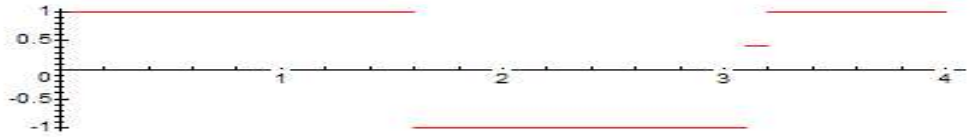


$-1 \le q_1 \le 1$



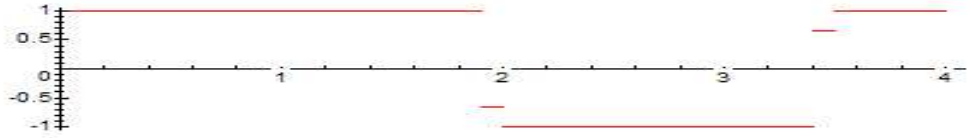
Optimalus $q_1^0 = -1$.

$-2 \le q_1 \le 2$



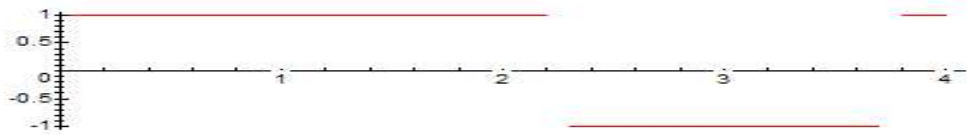
Optimalus $q_1^0 = -2$

$-3 \le q_1 \le 3$



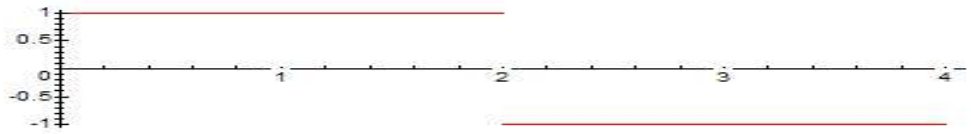
Optimalus $q_1^0 = -3$

$-4 \le q_1 \le 4$



Optimalus $q_1^0 = -4$

Kai turime, kad T=4, $-100 \le q_1 \le 100$ ir $g^* = (0,0,0)^T$, $g_* = (0,0,0)^T$ tai valdymo funkcija



Optimalus $q_1^0 = -4$.

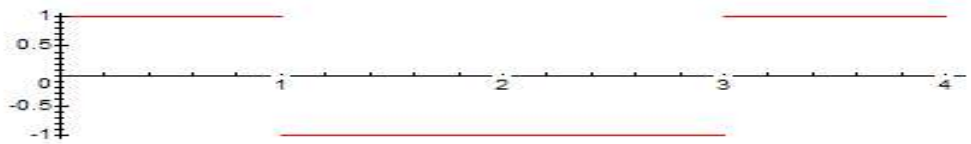
Dabar fiksuokime, kad $-10000 \le q_1 \le 10000$ ir $g^* = (0,0,0)^T$, $g^* = (0,0,0)^T$, bet keiskome intervalo ilgį nuo T=1 iki T=64. Ir šiam kitimui sudarome lentelę.

Intervalo ilgis T	Optomalus q_1^0, kai $g^* = (0,0,0)^T$, $g^* = (0,0,0)^T$	Optomalus q_1^0, kai $g^* = (1,1,1)^T$, $g_* = (-1,-1,-1)^T$
1	-0,25	-0,5
2	-1	-0,75
3	-9/4	-1999/800
4	-4	-19/4
5	-25/4	-15/2
6	-9	-2149/200
8	-16	-937/50
16	-64	-1768/25
32	-256	-1353/5
64	-1024	-5271/5

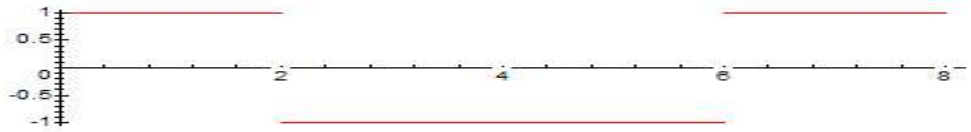
Matme, kad galioja optimaliam q_1^0 sąryšis $q_1^0 = \frac{1}{4}T^2$, kai $g^* = (0,0,0)^T$, $g^* = (0,0,0)^T$.

Vadinasi keičiant pradinių sąlygų apribojimus matome, kad persijungimo taškai juda link intervalo galo. Galime daryti išvadą, kad persijungimo taškų vieta intervale priklauso nuo intervalo ilgio, pradinių ir kraštinių sąlygų parinkimo, bet nepriklauso nuo žingsnio, kuriuo diskretizavome tiesinę dinaminę sistemą su valdomomis pradinėmis sąlygomis.

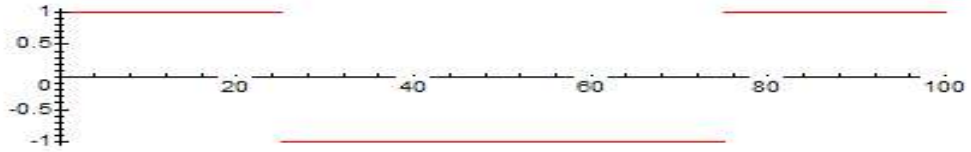
Kitas įdomus atvejis šiame tiesinio dinaminio valdymo uždavinyje, kai fiksuojame, kad $0 \le q_1 \le 0$ ir $g^* = (0,0,0)^T$, $g_* = (0,0,0)^T$, T=4, N=40, tuomet turime, kad valdymo funkcija atrodo:



Toliau padidinkime žingsnį iki T=8, tuomet valdymo funkcija:



Galime dar praplėsti intervalą T=100 ir tuomet turėsime, kad



Matome, kad šiuo atveju persijungimo taškai nejuda santykinai intervalo ilgiui ir yra ties pirmu ir trečiu ketvirčiais.

Tarkime, kad n=6 ir iš trečio skyrelio turime, kad

$$. \ A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

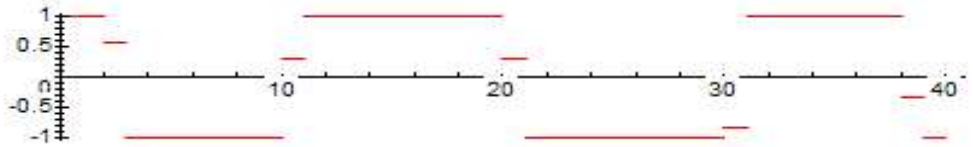
$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

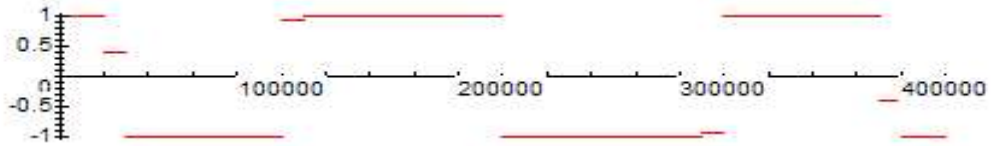
$x_0 = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$, $b = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1)^T$, $-1 \le u(t) \le 1$, $-1 \le q_1 \le 1$,

$g^* = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)^T$, $g_* = (-1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1)^T$, $c = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$.

Tuomet skaidome intervalą į 40 lygių dalių. Kai T=40 turime valdymo funkciją



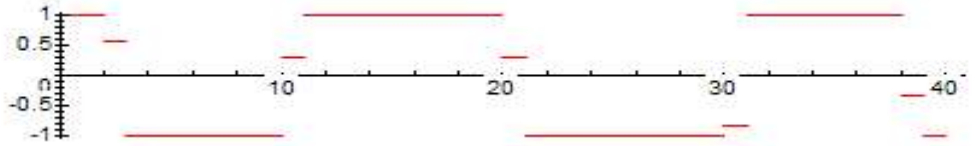
Jei T=400000, tai



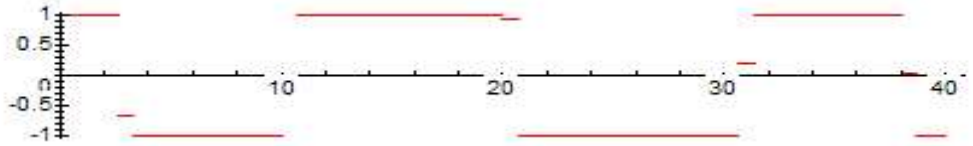
.

Iš grafikų matome, kad didėjant intervalui persijungimo taškų vieta intervale keičiasi labai negreitai jeigu intervalo ilgis pakankamai didelis lyginant su kraštinėmis sąlygomis. Matome, kad turime penkis persijungimo taškus.

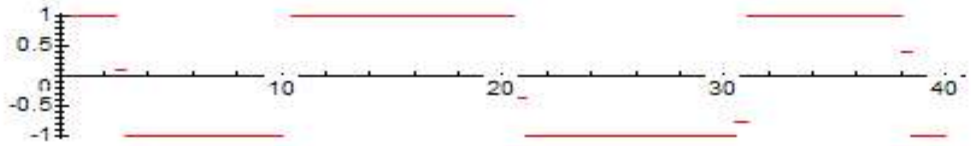
Patiktinsime, ar priklauso intervalo persijungimo taškų vieta intervale nuo skaidymo žingsnio. Imkime, kad T=40 ir $0 \le q_1 \le 0$, N=40,



N=60



N=80



Pastebime, kad persijungimo taškai išlaiko savo vietą intervale, ir nekeičia vietos, keičiant intervalo skaidinių skaičių. Detaliai šeštos eilės tiesinės dinaminės sistemos analizei reikėtų pateikti daugiau grafikų ir tirti atsižvelgiant į pradines valdomas sąlygas ir kraštinius apribojimus. Bet remiantis ankščiau išspręstu pavyzdžiu galima daryti išvadas, kad persijungimo taškų vieta intervale priklauso nuo kraštinių apribojimų, valdomų pradinių sąlygų ir nuo intervalo ilgio. Bet, pakeitus kraštines sąlygas iš nelygybių į lygybes, gauname, kad persijungimo taškų vieta intervale lieka pastovi santykiškai intervalo ilgiui. Šie visi skaičiavimai buvo atlikti „Maple“ programa, kurios programinis kodas yra pateiktas 9.3. priede.

6. Simplekso ir adaptyvinio metodų palyginimas

Šiame paragrafe palyginsime adaptyvinį metodą su Simplekso metodu, sprendžiant tiesinio programavimo uždavinius, gautus diskretizavus ankščiau spręstas tiesines dinamines sistemas su valdomomis pradinėmis sąlygomis. Simplekso metodo rezultatai (iteracijų skaičius) bus gauti iš Excel programos, panaudojus solver modulį. Tarkime, kad intervalą skaidome į keturias ir dešimt lygių dalių, kai $0 \le q(t) \le 0$.

	Simplekso metodas	Adaptyvinis metodas
N=4, iteracijų sk.	6	4
N=10 iteracijų sk.	12	7
N=20 iteracijų sk.	27	13
N=30 iteracijų sk.	43	21
N=40 iteracijų sk.	64	24
N=50 iteracijų sk.	79	30

Galima daryti išvadą, kad sprendžiant tiesinio programavimo uždavinius, gautus dikretizuojant tiesines dinamines sistemas, pranašesnis yra adaptyvinis metodas. Adaptyviniame metode iteracijos metu, atrama gali didėti, mažėti, likti tokia pat. Pradinio taško komponentės iteracijos metu keičiasi visos, o Simplekso metode tik bazinės. Adaptyviniame metode galima išnaudoti konkretaus uždavinio pradinę informaciją. Taip pat, darant šias išvadas, galima remtis [7] knygoje pateikiama statistika apie iteracijų skaičių lyginant Simplekso metodą su adaptyviniu.

Dar geresnių sprendimo rezultatų galima pasiekti adaptyviniame metode tinkamai parinkus pradinį tašką, arba panaudojus prieš tai spręsto uždavinio rezultatus.

7. IŠVADOS

Sprendėme tiesinį dinaminę sistemą su valdomomis pradinėmis sąlygomis. Šio uždavinio sprendimas susiveda į valdymo funkcijos persijungimo taškų suradimą. O valdymo funkcija, remiantis Pontriagino principu, buvo ieškojama gabalais pastovių funkcijų klasėje.

Iš pateiktų skaičiavimų ir rezultatų galima daryti išvadą, kad persijungimo taškų vieta intervale (santykis su intervalo ilgiu) priklauso nuo intervalo ilgio, pasirinktų kraštinių apribojimų, pradinių valdomų sąlygų, bet visiškai nepriklauso nuo intervalo suskaidymo žingsnio. Bet, pakeitus kraštines sąlygas iš nelygybių į lygybes, gauname, kad persijungimo taškų vieta intervale lieka pastovi santykiškai intervalo ilgiui.

Darbe palyginta Simplekso ir adaptyvinio metodų efektyvumas, sprendžiant tiesines dinamines sistemas su valdomomis kraštinėmis sąlygomis. Galima daryti išvadą, kad adaptyvinis metodas yra efektyvesnis. Tai galima paaiškinti tuo, kad sprendžiant tiesinio programavimo uždavinį adaptyviu metodu nebūtina jį susivesti į kanoninį tiesinio programavimo uždavinį. Dėl šios priežasties sumažėja duomenų struktūros ir išlaikoma pradinė informacija apie modelį.

8. Summary (Linear dynamic system optimisation with guided initial conditions)

The linear dinamic operating problem with guided initial conditions was solved. The solution of this problem consists of finding the guided function of switching points. Guiding function, according principle of Pontragin, was searched in uniform functions class by piecemeal.

The conclusion can be drawn from results and showed calculations that the place in interval of switching points depends on the length of interval, chosen marginal confines, initial guided conditions, but does not depend at all on the step of interval separation. But if we change marginal conditions from inequality to equality, we will see that the place of switched points in the interval does not depend on the length of interval.

There were compared the efficiency of Simplex method and adapted method in solving linear dynamic systems with guided marginal conditions. So, we can draw the conclusion that the adapted method is much more effective than Simplex one. This can be explained by a fact that when you solve linear programming problem by adapted method it is not necessary to transform it into canonic linear programming problem. That's why the structure of data decreases and the initial information about the model is kept.

9. Priedai

9.1. Adaptyvio metodo programinis kodas

```
> with(linalg):
> A:=array([[1,0,3,4,-1],[2,1,0,1,3]]);
> bbbb:=array([-1,-1]);
> bbb:=array([1,1]);
> dddd:=array([-1,-1,-1,-1,-1]);
> ddd:=array([1,1,1,1,1]);
> c:=([1,1,1,1,1]);
> x:=([1/2,0,0,0,0]);
> J:={1,2,3,4,5};
> II:={1,2};
> JOP:={1,2};
> IOP:={1,2};
> pp:=1;
> for kk from 1 to 10 do
> pp:=pp+1;
> JH:=J minus JOP;
> IH:=II minus IOP;
> AIOPJOP:=array(1..nops(IOP),1..nops(JOP));
> for i from 1 to nops(IOP) do
> for j from 1 to nops(JOP) do
> AIOPJOP[i,j]:=A[IOP[i],JOP[j]];
> od;od;
> AIHJH:=array(1..nops(IH),1..nops(JH));
> for i from 1 to nops(IH) do
> for j from 1 to nops(JH) do
> AIHJH[i,j]:=A[IH[i],JH[j]];
> od;od;
> AIHJOP:=array(1..nops(IH),1..nops(JOP));
> for i from 1 to nops(IH) do
> for j from 1 to nops(JOP) do
> AIHJOP[i,j]:=A[IH[i],JOP[j]];
> od;od;
> AIOPJH:=array(1..nops(IOP),1..nops(JH));
> for i from 1 to nops(IOP) do
> for j from 1 to nops(JH) do
> AIOPJH[i,j]:=A[IOP[i],JH[j]];
> od;od;
> AIOPJ:=array(1..nops(IOP),1..nops(J));
> for i from 1 to nops(IOP) do
> for j from 1 to nops(J) do
> AIOPJ[i,j]:=A[IOP[i],J[j]];
```

```
> od;od;
> cJOP:=array(1..nops(JOP));
> for i from 1 to nops(JOP) do
> cJOP[i]:=c[JOP[i]];
> od;
> cJH:=array(1..nops(JH));
> for i from 1 to nops(JH) do
> cJH[i]:=c[JH[i]];
> od;
> xH:=array(1..nops(JH));
> for i from 1 to nops(JH) do
> xH[i]:=x[JH[i]];
> od;
> xOP:=array(1..nops(JOP));
> for i from 1 to nops(JOP) do
> xOP[i]:=x[JOP[i]];
> od;
> wwww:=array(1..nops(II));
> wwww:=matadd(bbbb,-multiply(A,x));
> www:=array(1..nops(II));
> www:=matadd(bbb,-multiply(A,x));
> uIOP:=array(1..nops(IOP));
> uIOP:=multiply(transpose(cJOP),inverse(AIOPJOP));
> deltaJ:=matadd(multiply(transpose(uIOP),AIOPJ),-c);
> uIOP:=transpose(uIOP);
> wIOP:=array(1..nops(IOP));
> for i from 1 to nops(IOP) do
> if uIOP[i]<0 then wIOP[i]:=wwww[IOP[i]]; fi;
> if uIOP[i]>0 then wIOP[i]:=www[IOP[i]]; fi;
> if uIOP[i]=0 then wIOP[i]:=0; fi;
> od;
> f1 := proc(a,b) local i,k,p;
> if nops(b)=0 then k:=true else
> k:=true;
> for i from 1 to nops(a) do
> if a[i]>0 then k:=false; fi;
> od;
> fi;
> k;
> end:
> f2 := proc(a,b) local i,k,p;
> if nops(b)=0 then k:=true; else
> k:=true;
> for i from 1 to nops(a) do
> if a[i]<0 then k:=false; fi;
> od;
> fi;
```

```
> k;
> end:
> f3 := proc(a,b) local i,k,p;
> if nops(b)=0 then k:=true ; else
> k:=true;
> for i from 1 to nops(a) do
> if a[i]<>0 then k:=false; fi;
> od;
> fi;
> k;
> end:
> IIII:={};
> III:={};
> for i from 1 to nops(IOP) do if multiply(row(A,i),x)=bbbb[i] then IIII:=IIII union {i}; fi;
od;
> for i from 1 to nops(IOP) do if multiply(row(A,i),x)=bbb[i] then III:=III union {i}; fi;
od;
> JJJJ:={};
> JJJ:={};
> for i from 1 to nops(J) do if x[i]=dddd[i] then JJJJ:=JJJJ union {i}; fi;  od;
> for i from 1 to nops(J) do if x[i]=ddd[i] then JJJ:=JJJ union {i}; fi;  od;
> I1:=II minus (III union IIII);
> J1:=J minus (JJJ union JJJJ);
> IIIIOP:=IIII intersect IOP;
> IOP;
> III;
> IIIOP:=III intersect IOP;
> JJJJH:=JJJJ intersect JH;
> JJJH:=JJJ intersect JH;
> I1OP:=I1 intersect IOP;
> J1H:=J1 intersect JH;
> nops(IIIOP);
> IIIOP;
> uIIIIOP:=array(1..nops(IIIIOP));
> uIIIOP:=array(1..nops(IIIOP));
> uI1:=array(1..nops(I1OP));
> deltaJJJJH:=array(1..nops(JJJJH));
> deltaJJJH:=array(1..nops(JJJH));
> deltaJ1:=array(1..nops(J1));
> for i from 1 to nops(IIIIOP) do
> uIIIIOP[i]:=uIOP[IIIIOP[i]];
> od;
> I1OP;
> for i from 1 to nops(I1OP) do
> uI1[i]:=uIOP[i];
> od;
> IOP;
```

```
> IIIOP;
> print(uIOP);
> for i from 1 to nops(IIIOP) do
> uIIIOP[i]:=uIOP[i];
> od;
> for i from 1 to nops(JJJJH) do
> deltaJJJJH[i]:=deltaJ[JJJJH[i]];
> od;
> for i from 1 to nops(JJJH) do
> deltaJJJH[i]:=deltaJ[JJJH[i]];
> print(deltaJJJH[i]);
> od;
> for i from 1 to nops(J1H) do
> deltaJ1[i]:=deltaJ[J1H[i]];
> od;
> if f1(uIIIIOP,IIIIOP)=true and f2(uIIIOP,IIIOP)=true and f3(uI1,I1OP)=true and f2
(deltaJJJJH,JJJJH)=true and f1(deltaJJJH,JJJH)=true and f3(deltaJ1,J1H)=true then
optimalu:=true else optimalu:=false fi;optimalu;
> if optimalu=true then ppp:=kk; break fi;
> lJOPJH:=array(1..nops(J));
> lJOP:=array(1..nops(JOP));
> lJH:=array(1..nops(JH));
> for i from 1 to nops(JH) do
> if deltaJ[JH[i]]<0 then lJH[i]:=ddd[JH[i]]-x[JH[i]]; fi;
> if deltaJ[JH[i]]>0 then lJH[i]:=dddd[JH[i]]-x[JH[i]]; fi;
> if deltaJ[JH[i]]=0 then lJH[i]:=0; fi;
> od;
>              lJOP:=matadd(multiply(inverse(AIOPJOP),wIOP),-multiply(multiply(inverse
(AIOPJOP),AIOPJH),lJH));
> for i from 1 to nops(JOP) do
> lJOPJH[JOP[i]]:=lJOP[i];
> od;
> for i from 1 to nops(JH) do
> lJOPJH[JH[i]]:=lJH[i];
> od;
> lJOPJH;
> tetaJ:=array(1..nops(J));
> for i from 1 to nops(J) do
> if lJOPJH[i]<0 then tetaJ[i]:=(dddd[i]-x[i])/lJOPJH[i]; fi;
> if lJOPJH[i]>0 then tetaJ[i]:=(ddd[i]-x[i])/lJOPJH[i]; fi;
> if lJOPJH[i]=0 then tetaJ[i]:=infinity; fi;
> od;
> tetaI:=array(1..nops(II));
> for i from 1 to nops(II) do
> if multiply(transpose(row(A,i)),lJOPJH)<0 then tetaI[i]:=wwww[i]/multiply(transpose
(row(A,i)),lJOPJH); fi;
```

```
> if multiply(transpose(row(A,i)),lJOPJH)>0 then tetaI[i]:=www[i]/multiply(transpose
(row(A,i)),lJOPJH); fi;
> if multiply(transpose(row(A,i)),lJOPJH)=0 then tetaI[i]:=infimity; fi;
>
> multiply(transpose(row(A,i)),lJOPJH);
> od;
> print(tetaI);
> tetajo:=tetaJ[JOP[1]]:
> jo:=JOP[1]:
> for i from 1 to nops(JOP)-1 do
> if tetajo>tetaJ[JOP[i+1]] then tetajo:=tetaJ[JOP[i+1]]; jo:=JOP[i+1]; fi;
> od;
>
> if IH<>{} then
> io:=IH[1];
> tetaio:=tetaI[IH[1]];
> for i from 1 to nops(IH)-1 do
> if tetaio>tetaI[IH[i+1]] then tetaio:=tetaI[IH[i+1]]; io:=IH[i+1]; fi;
> od;
> else tetaio:=infinity ; io:={};
> fi;
> TT:=proc(tetaio,tejajo) local tt;
> tt:=1;
> if tt>tetaio then tt:=tetaio; fi;
> if tt>tetajo then tt:=tetajo; fi;
> tt;
> end:
> print(tetajo,tetaio);
> tt:=TT(tetaio,tejajo);
> x:=matadd(x,tt*lJOPJH);
> if tt=tetaio then
> zJH:= matadd(AIHJH,-multiply(AIHJOP,multiply(transpose(AIOPJOP),AIOPJH)));
> zIOP:=-multiply(AIHJOP,transpose(AIOPJOP));
> zJHIOP:=concat(zJH,zIOP);
> row(A,io);
> if multiply(row(A,io),x)=bbbb[io] then k:=-1 fi;
> if multiply(row(A,io),x)=bbb[io] then k:=1 fi;
> e:=array(1..nops(II));
> eio:=array(1..nops(IH));
> for i from 1 to nops(II) do e[i]:=0; od;
> e[io]:=1;
> for i from 1 to nops(IH) do if io=IH[i] then eio[i]:=1 else eio[i]:=0; fi; od;
> zzJHIOP:=scalarmul(multiply(eio,zJHIOP),k);
> zzJH:=scalarmul(multiply(eio,zJH),k);
> zzIOP:=scalarmul(multiply(eio,zIOP),k);
> sigmaj:=array(1..nops(JH));
> for i from 1 to nops(JH) do
```

```
> sigmaj[i]:=infinity; od;
> for i from  1 to nops(JH) do
> if deltaJ[JH[i]]*zzJH[i]<0 then sigmaj[i]:=-deltaJ[JH[i]]/zzJH[i]; fi;
> if deltaJ[JH[i]]=0 and zzJH[i]>0 and x[JH[i]]<>dddd[JH[i]] then sigmaj[i]:=0; fi;
> if deltaJ[JH[i]]=0 and zzJH[i]>0 and x[JH[i]]<>ddd[JH[i]] then sigmaj[i]:=0; fi;
> od;
> sigmai:=array(1..nops(IOP));
> for i from  1 to nops(IOP) do
> sigmai[i]:=infinity; od;
> for i from  1 to nops(IOP) do
> if -uIOP[i]*zzIOP[i]<0 then sigmai[i]:=-uIOP[i]/zzIOP[i]; fi;
> if (uIOP[i]=0 and zzIOP[i]>0) or (uIOP[i]=0 and zzIOP[i]<0 ) then sigmai[i]:=0; fi;
> od;
> sigmaio:=infinity;
> for i from 1 to nops(IOP) do if sigmai[i]<sigmaio then sigmaio:=sigmai[i]; sio:=IOP[i];
fi; od;
> sigmajo:=infinity;
> print(sigmaj);
> JH;
> for i from 1 to nops(JH) do if sigmaj[i]<=sigmajo then sigmajo:=sigmaj[i]; sjo:=JH[i];
print(sjo); fi; od;
> sigma:=min(sigmaio,sigmajo);
> if sigma=sigmaio then IOP:=(IOP minus {sio}) union {io}; JOP:=JOP; fi;
> if sigma=sigmajo then IOP:=IOP union {io} ; JOP:=JOP union {sjo}; fi;
> fi;
> if tt=tetajo then zJH:=multiply(transpose(AIOPJOP),AIOPJH);
> zJH:=multiply(transpose(AIOPJOP),AIOPJH);
> zIOP:=scalarmul(transpose(AIOPJOP),-1);
> zJHIOP:=concat(zJH,zIOP);
> if x[jo]=dddd[jo] then k:=1 fi;
> if x[jo]=ddd[jo] then k:=-1 fi;
> nops(JOP);
> ejo:=array(1..nops(JOP));
> for i from 1 to nops(JOP) do if jo=JOP[i] then ejo[i]:=1; else ejo[i]:=0; fi; od;
> zJHIOP:=scalarmul(multiply(ejo,zJHIOP),k);
> zJH:=scalarmul(multiply(ejo,zJH),k);
> zIOP:=scalarmul(multiply(ejo,zIOP),k);
> sigmaj:=array(1..nops(JH));
> for i from  1 to nops(JH) do
> sigmaj[i]:=infinity; od;
> for i from  1 to nops(JH) do
> if deltaJ[JH[i]]*zJH[i]<0 then sigmaj[i]:=-deltaJ[JH[i]]/zJH[i]; fi;
> if (delta[JH[i]]=0 and zJH[i]>0 and x[JH[i]]<>dddd[JH[i]]) or (delta[JH[i]]=0 and zJH
[i]>0 and x[JH[i]]<>ddd[JH[i]]) then sigmaj[i]:=0; fi;
> od;
> sigmai:=array(1..nops(IOP));
> for i from  1 to nops(IOP) do
```

```
> sigmai[i]:=infinity; od;
> for i from  1 to nops(IOP) do
> if -uIOP[i]*zIOP[i]<0 then sigmai[i]:=-uIOP[i]/zIOP[i]; fi;
> if (uIOP[i]=0 and zIOP[i]>0) or (uIOP[i]=0 and zIOP[i]<0 ) then sigmai[i]:=0; fi;
> od;
> sigmaio:=infinity;
> for i from 1 to nops(IOP) do if sigmai[i]<sigmaio then sigmaio:=sigmai[i]; sio:=IOP[i];
fi; od;
> sigmajo:=infinity;
> for i from 1 to nops(JH) do if sigmaj[i]<sigmajo then sigmajo:=sigmaj[i] ; sjo:=JH[i];
fi; od;
> sigma:=min(sigmaio,sigmajo);
> if sigma=sigmaio then IOP:=IOP minus {sio}; JOP:=JOP minus {jo} fi;
> if sigma=sigmajo then IOP:=IOP ; JOP:=(JOP minus {jo}) union {sjo}; fi;
> fi;
> JOP;
> IOP;
> od;
```

9.2. Tiesinio programavio uždavinys

3871/6000*u[16]+4081/6000*u[15]+3667/6000*u[17]+3469/6000*u[18]
+3277/6000*u[19]+3091/6000*u[20]+5977/6000*u[7]+5719/6000*u[8]+5467/6000*u
[9]+5221/6000*u[10]+4981/6000*u[11]+4747/6000*u[12]+4519/6000*u[13]
+4297/6000*u[14]+7651/6000*u[1]+7357/6000*u[2]+7069/6000*u[3]+6787/6000*u[4]
+6511/6000*u[5]+6241/6000*u[6]+q[1]+1219/6000*u[33]+697/6000*u[39]
+631/6000*u[40]+769/6000*u[38]+1021/6000*u[35]+931/6000*u[36]+847/6000*u[37]
+1117/6000*u[34]+2569/6000*u[23]+1441/6000*u[31]+1327/6000*u[32]+1687/6000*u
[29]+1561/6000*u[30]+1819/6000*u[28]+1957/6000*u[27]+2101/6000*u[26]
+2407/6000*u[24]+2251/6000*u[25]+2911/6000*u[21]+2737/6000*u[22]->max

-1<=1/10*u[16]+1/10*u[15]+1/10*u[17]+1/10*u[18]+1/10*u[19]+1/10*u[20]+
1/10*u[7]+1/10*u[8]+1/10*u[9]+1/10*u[10]+1/10*u[11]+1/10*u[12]+1/10*u[13]
+1/10*u[14]+1/10*u[1]+1/10*u[2]+1/10*u[3]+1/10*u[4]+1/10*u[5]+1/10*u[6]+1/10*u
[33]+1/10*u[39]+1/10*u[40]+1/10*u[38]+1/10*u[35]+1/10*u[36]+1/10*u[37]+1/10*u
[34]+1/10*u[23]+1/10*u[31]+1/10*u[32]+1/10*u[29]+1/10*u[30]+1/10*u[28]+1/10*u
[27]+1/10*u[26]+1/10*u[24]+1/10*u[25]+1/10*u[21]+1/10*u[22], 49/200*u[16]
+51/200*u[15]+47/200*u[17]+9/40*u[18]+43/200*u[19]+41/200*u[20]+67/200*u[7]
+13/40*u[8]+63/200*u[9]+61/200*u[10]+59/200*u[11]+57/200*u[12]+11/40*u[13]
+53/200*u[14]+79/200*u[1]+77/200*u[2]+3/8*u[3]+73/200*u[4]+71/200*u[5]
+69/200*u[6]+q[1]+3/40*u[33]+3/200*u[39]+1/200*u[40]+1/40*u[38]+11/200*u[35]
+9/200*u[36]+7/200*u[37]+13/200*u[34]+7/40*u[23]+19/200*u[31]+17/200*u[32]
+23/200*u[29]+21/200*u[30]+1/8*u[28]+27/200*u[27]+29/200*u[26]+33/200*u[24]
+31/200*u[25]+39/200*u[21]+37/200*u[22] <= 1, -1 <= 49/200*u[16]+51/200*u[15]
+47/200*u[17]+9/40*u[18]+43/200*u[19]+41/200*u[20]+67/200*u[7]+13/40*u[8]
+63/200*u[9]+61/200*u[10]+59/200*u[11]+57/200*u[12]+11/40*u[13]+53/200*u[14]
+79/200*u[1]+77/200*u[2]+3/8*u[3]+73/200*u[4]+71/200*u[5]+69/200*u[6]+q[1]
+3/40*u[33]+3/200*u[39]+1/200*u[40]+1/40*u[38]+11/200*u[35]+9/200*u[36]
+7/200*u[37]+13/200*u[34]+7/40*u[23]+19/200*u[31]+17/200*u[32]+23/200*u[29]
+21/200*u[30]+1/8*u[28]+27/200*u[27]+29/200*u[26]+33/200*u[24]+31/200*u[25]
+39/200*u[21]+37/200*u[22], 1/10*u[16]+1/10*u[15]+1/10*u[17]+1/10*u[18]+1/10*u

[19]+1/10*u[20]+1/10*u[7]+1/10*u[8]+1/10*u[9]+1/10*u[10]+1/10*u[11]+1/10*u[12]
+1/10*u[13]+1/10*u[14]+1/10*u[1]+1/10*u[2]+1/10*u[3]+1/10*u[4]+1/10*u[5]+1/10*u
[6]+1/10*u[33]+1/10*u[39]+1/10*u[40]+1/10*u[38]+1/10*u[35]+1/10*u[36]+1/10*u
[37]+1/10*u[34]+1/10*u[23]+1/10*u[31]+1/10*u[32]+1/10*u[29]+1/10*u[30]+1/10*u
[28]+1/10*u[27]+1/10*u[26]+1/10*u[24]+1/10*u[25]+1/10*u[21]+1/10*u[22] <= 1,
u[2] <= 1, -1 <= u[2], u[1] <= 1, -1 <= u[1], -1 <= u[14], u[12] <= 1, -1 <= u[12], u[11]
<= 1, -1 <= u[11], u[10] <= 1, -1 <= u[10], u[9] <= 1, -1 <= u[9], u[8] <= 1, -1 <= u[8], u
[7] <= 1, -1 <= u[7], u[6] <= 1, u[5] <= 1, -1 <= u[5], -1 <= u[4], u[4] <= 1, u[3] <= 1, -1
<= u[3], u[17] <= 1, -1 <= u[17], u[16] <= 1, -1 <= u[16], -1 <= u[6], u[15] <= 1, -1 <= u
[15], u[14] <= 1, u[13] <= 1, -1 <= u[13], u[18] <= 1, -1 <= u[18], -1 <= u[19], u[19] <=
1, -1 <= u[20], u[20] <= 1, -1 <= u[25], u[24] <= 1, -1 <= u[24], u[23] <= 1, -1 <= u[23],
u[22] <= 1, -1 <= u[22], u[21] <= 1, -1 <= u[21], u[30] <= 1, -1 <= u[30], u[29] <= 1, -1
<= u[29], u[28] <= 1, -1 <= u[28], u[27] <= 1, -1 <= u[27], u[26] <= 1, -1 <= u[26], u
[25] <= 1, -1 <= u[36], u[35] <= 1, -1 <= u[35], u[34] <= 1, -1 <= u[34], u[33] <= 1, -1
<= u[33], u[32] <= 1, -1 <= u[32], u[31] <= 1, -1 <= u[31], u[40] <= 1, -1 <= u[40], u
[39] <= 1, -1 <= u[39], u[38] <= 1, -1 <= u[38], u[37] <= 1, -1 <= u[37], u[36] <= 1,
-1 <= q[3], -1 <= q[2], q[1] <= 1, -1 <= q[1], q[3] <= 1, q[2] <= 1.

Išspręndus šį uždavinį gauname rezultatą

u[9] = 1, u[7] = 1, u[8] = 1, u[11] = 1, u[31] = -1, u[29] = -1, u[30] = -1, u[28] = -1, u[27]
= -1, u[12] = 1, u[10] = 1, u[1] = 1, u[2] = 1, u[3] = 1, u[4] = 1, u[5] = 1, u[6] = 1, u[26]
= -1, u[32] = -1, u[13] = 1, u[14] = 1, u[24] = -1, u[25] = -1, u[21] = -1, u[22] = -1, u[23]
= -1, u[34] = 1, u[36] = 1, u[37] = 1, u[20] = -1, u[33] = 1, q[1] = -1, u[40] = 1, u[39] = 1,
u[35] = 1, u[38] = 1, u[17] = -1, u[19] = -1, u[18] = -1, q[2] = 1, q[3] = 1, u[16] = -1, u
[15] = -9/17

9.3.Tiesinio dinaminio uždavio programinis kodas

```
> restart;
> with(linalg):
> MM:=6:
> A:=band([0,0,1], MM):
> H:=band([0,0,1], MM):
> DDD:=band([0,0,0], MM):
> DDD[1,1]:=1:
> qq:=0:
> with(linalg):  b:=array(1..MM):
> c:=array(1..MM):
> #DDD:=array( [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]);
> g2:=array(1..MM):
> g1:=array(1..MM):
> xo:=array(1..MM):
> for i from 1 by 1 to MM do c[i]:=1; xo[i]:=0; g1[i]:=1;g2[i]:=-1;  b[i]:=0: od:
> b[MM]:=1:
> n:=40:
> T:=20:
> F:=exponential(A,tt):
> FT:=exponential(A,T):
> x:=0:
> Ftt:=inverse(F):
> mmm:=MM: x:=array(1..n): y:=array(1..n):
> for i from 1 by 1 to n do x[i]:=(T/n)*(i-1): y[i]:= (T/n)*(i): oo:=map(int, Ftt,tt=x[i]..y
[i]):   ooo:=multiply(FT,oo): row(R,i):= multiply(ooo,b): od:
> CC := array(1..n):
> for i from 1 by 1 to n do CC[i]:=multiply(row(R,i),c): od:
> AA := array(1..mmm,1..n):
> for j from 1 by 1 to mmm do
> for i from 1 by 1 to n do AA[j,i]:=multiply(row(H,j),(row(R,i))):     od:
> od:
> #gg:=multiply(FT,xo ):
> #bb:=matadd(g,-gg):
> u:=array(1..n):
> q:=array(1..mmm):aa:=array(1..mmm):
> a:=multiply(AA,u): multiply(FT,DDD):
> aa:=multiply(q,FT,DDD): DD:=multiply(H,DDD):
> u:=array(1..n):
> j:={}:print(aa[2]):
> for i from 1 by 1 to mmm do if (a[i]=0 and aa[i]=0) then g:={}: gg:={} else  g:={g1[i]
>=a[i]+aa[i]}:gg:={g2[i]<=a[i]+aa[i]}:  fi: j:= j union g union gg:   od:
> R:=multiply(FT,DDD): k:=multiply(u,CC)+ multiply(c,q):
> m:={}:
> for i from 1 by 1 to n do u0:={u[i]<=1,u[i]>=-1}: m:= m union u0:  od:
```

```
> mm:={}:
> for i from 1 by 1 to mmm do u0:={q[i]<=qq,q[i]>=(-1)*qq}: mm:= mm union u0:  od:
> with (simplex):
> KK:=(maximize(k,j union m union mm)):
> k:
> j:
> uu:=array(1..n):
> for j from 1 by 1 to n  do
> for k from 1 by 1 to n+mmm do if (has(KK[k],u[j])=true) then ll:=cterm(KK[k]):uu[j]:
=ll:    fi: od :od:
> uuu:=array(1..n):
> for i from 1 by 1 to n  do uuu[i]:=[[x[i],uu[i]],[y[i],uu[i]]]: od:
> sss:=seq(uuu[i],i=1..n):
> plot([sss],color=red);
```

10. Literatūros sąrašas

1. P. Golokvosčius. *Diferencialinės lygtys*. Vilnius 2000 TEV leidykla 511 p.

2. Р.Ф. ГАБАСОВ. Ф.М. КИРИЛОВА. *Оптимизация линейных систем*. Минск 1973.

3. Р.Ф. ГАБАСОВ. Ф.М. КИРИЛОВА. *Метод оптимизацсии.* Минск 1973.

4. A. Apynis. *Optimizavimo metodai*. Vilnius 1988.

5. A. Apynis, E. Stankus. *Matematika*. Vilnius 2001.

6.Р.Ф. ГАБАСОВ. *Линейного программирования*. Минск 1973.

7. Р.Ф. ГАБАСОВ. Ф.М. КИРИЛОВА. *Методы линейного программирования част 3.* Минск 1980.