Eötvös Loránd Tudományegyetem
Természettudományi Kar

Darida Sándor
Alkalmazott Matematikus MSc, Alkalmazott Analízis szakirány

A modell prediktív irányítás alkalmazása differenciálegyenletekre

Szakdolgozat

Témavezető: Simon Péter
Alkalmazott Analízis és Számításmatematikai Tanszék

Budapest, 2012

Tartalomjegyzék

1 Bevezetés **3**

2 Az algoritmus alapjai **5**

2.1 A *step response* modell 5

2.2 Az *impulse response* modell 7

2.3 A DMC algoritmus 8

3 A módszer alkalmazása differenciálegyenletekre **13**

3.1 Az egydimenziós eset 13

3.2 A többdimenziós eset 14

4 A módszer stabilitásának vizsgálata konstans trajektória mellett **17**

4.1 Az egydimenziós modell 17

4.1.1 A $\lambda = 0$ eset 18

4.1.2 A $\lambda > 0$ eset. 26

4.1.3 A nem konstans trajektória 29

4.2 Az n-dimenziós modell, $\lambda = 0$ mellett 32

5 Összefoglalás **39**

6 Függelék **40**

1 Bevezetés

A modell prediktív irányítás vagy *Model Predictive Control*(továbbiakban: MPC) egy igen fejlett numerikus optimalizáláson alapuló irányításelméleti módszer, melyet az 1980-as évek óta használnak az ipar különböző területein, többek közt a vegyiparban és olajfinomításban. A módszer diszkrét időben dolgozik. Az alapfeltevés, hogy adott rendszer egy paraméterét egy adott értékre akarjuk beállítani, s ehhez keresünk megfelelő bemeneti paramétert. A módszer ereje, hogy nem használ a rendszerről semmiféle információt, pusztán annyit, hogy bizonyos bemenetekre milyen kimeneti értékeket kapunk.

Az algoritmus megvalósítása során minden időlépésben végrehajtunk egy optimalizálást, ahol valamilyen előírt költségfüggvényt minimalizálunk. Ennek megvalósításához a folyamatot modelleznünk kell valamilyen módon, véges időhorizontra előre tekintve. Erre több elehetőség is van a szakdolgozatban az úgynevezett *step response* modellel fogunk dolgozni. E modellből jósoljuk meg hogy egy adott bemeneti értékekhez milyen kimeneti értékek várhatók. Így létrehozhatunk egy költségfüggvényt mely a bemenő kontroll-paraméterek függvényében a modellezett kimenet és a kívánt értékek közti eltérést számszerűsíti. Ez lehet például a kettő közti négyzetes eltérés, ahogy azt később látni fogjuk. Ezt minimalizálva megkapjuk az optimális bemeneti értékeket. Ezek közül azoban mindig csak a következő időlépésbeli értéket használjuk fel, utána újra végrehajtjuk az optimalizálást.

Az MPC nem egy konkrét algoritmust takar, hanem egy olyan módszercsaládot, mely a rendszer modelljéből közvetlenül nyeri a megfelelő bemenő paramétert, egy optimalizálással. Mi a már említett *step response* modellt fogjuk használni, az algoritmus pedig az úgynevezett *Dynamic Matrix Control*(továbbiakban: DMC) lesz.

Az MPC stratégiája igen hasonló ahhoz, amit egy autó vezetése közben alkalmazunk. Adott egy meghatározott trajektória, valamilyen véges időhorizonton, és ide akarjuk vezérelni az autót a különböző bemeneti jelek (gáz, fék, kormány) megfelelő megválasztásával. Meg kell jósolnunk, hogyan viselkedik az autó, ezen bemenő adatok függvényében, ugyanakkor annak felépítéséről mit sem tudunk, így modellezzük a jövőbeli kimenetet, letapogatjuk a rendszert valamilyen speciális bemenő jellel (jelen esetben jelekkel). Az ezekre adott kimeneti értékekből jósolunk a jövőre, például feltételezve, hogy kétszer annyi gázt adva, kétszer annyi lesz a sebesség. Így meg tudjuk jósolni milyen bemenő jelekkel leszünk a referencia tra-

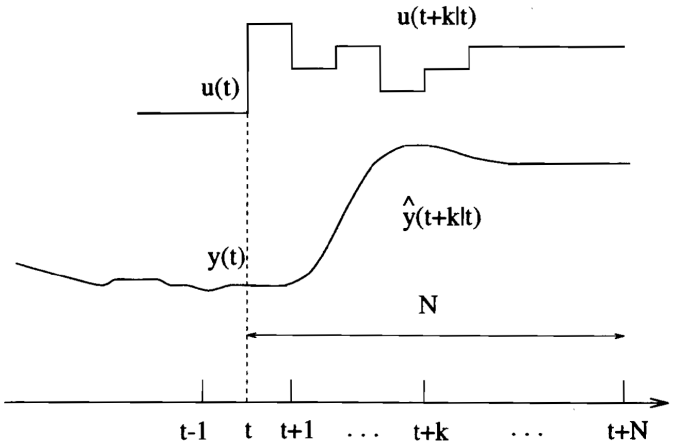
jektóriához a legközelebb, és ezeket a paramétereket választva eljutunk a következő időpillanatbeli valódi kimenethez. Ekkor újra "lefuttatjuk az algoritmust", s így (remélhetőleg) a kívánt trajektóriához tartok az idő előrehaladtával.

Bár a módszer felhasználása széleskörű, és gazdag múltra tekint vissza, keveset vizsgálták tisztán matematikai szempontból. A szakdolgozatban a rendszert differenciálegyenletek formájában adjuk meg, és e rendszerekhez adunk meg olyan kontrollfüggvényt, amellyel a referencia trajektóriára vezérelhetjük a célfüggvényünket. Egy nemrég megjelent cikkből [1] indulunk ki, melyet néhány általánosítással egészítünk ki. Először ismertetem a modell felépítését lineáris autonóm differenciálegyenletekre. Ezek közül az egy illetve a több dimenziós esetet is vizsgáljuk a modell felépítését, illetve hogy milyen feltételek mellett tart a rendszer a kívánt trajektóriához.

2 Az algoritmus alapjai

Ebben a fejezetben ismertetem, hogyan épül fel a *step response* modell, illetve a DMC algoritmus. Jegyezzük meg hogy nem a *step response* az egyetlen, mellyel jövőbeli kimenetet jósolhatunk, más modellek is szóba jöhetnek, például az *impulse response* modell. Ezen modelleket többnyire lineáris, diszkrét idejű, s időben invariáns rendszerekre alkalmazzák.

Legyen N az a lépésszám ameddig a rendszert vezéreljük. Feltesszük, hogy a t időpontig a kimenetekről pontos (mért) információnk van nevezetesen $y(t), y(t-1), \ldots$. Az adott modellel jósolt kimenetet jelölje $\hat{y}(t+k|t)$. Továbbá jelölje $u(t+1|t) \ldots u(t+k|t)$ a bemenő jeleket. Mind a függvény, mind a bemenő jelek argumentumában szerepel egy $|t$ feltétel. erre azért van szükség, mert az algoritmust minden t időpillanatban lefuttatom. A $\hat{y}(t+k|t)$ a t időpillanatban vett jósolt kimenet a $t+k$-adik időpontban. Az 1.1-es ábrán szemléltetjük ([3]. oldal 1.1-es ábra), hogyan is épül fel az algoritmus egy lépése a t időpillanatban.



1.1. ábra Az MPC ötlete

2.1 A *step response* modell

Ennek a modellnek a lényege, hogy a rendszert valamilyen módon letapogatva, próbalunk jóslást adni a jövőbeli kimeneti értékekre. Az úgynevezett *unit step response* modellben megmérjük milyen kimenet ad a rendszer, ha a kontrollt egységnek választjuk, majd azt feltételezzük, hogyha a kontrollt valamennyiszeresére megnöveljük két időpillanat között, akkor a rendszere kimenet is ugyanolyan arányban változik.

Vegyünk egy lineáris, diszkrét idejű, s időben invariáns rendszert, melynek a $t = 0, 1 \ldots$ időpontbeli bemenete $u(t)$, kimenete $y(t)$ valós számok. Továbbá tegyük fel, hogy $y(0) = 0$. Ilyen rendszerekre fogjuk az algoritmust alkalmazni. Mindenekelőtt felépítjük a *step response* modellt. Ehhez először szükség van a rendszer egységugrásra adott válaszára más néven *unit step response*-ra, vagyis hogy milyen kimeneteket kapunk, ha $u(t) \equiv 0$ ha $t \leq 0$ és $u(t) \equiv 1$ ha $t > 0$. Jelöljük az így kapott kimeneteket, $y(i) = g_i$-vel. Ennek segítségével építjük fel a *step response* modellt, nevezetesen a rendszer jósolt kimenete a t időpillanatban:

$$y(t)_{model} = \sum_{1}^{\infty} g_i \Delta u(t-i).$$

ahol $\Delta u(t) = u(t) - u(t-1)$. E modellben azt feltételezzük, hogy két időpillanat között y az egységugrásra adott válasz Δu -szorosával változik. Ezt a modellt fogjuk felhasználni, hogy t-nél nagyobb időpillanatokra is meg tudjuk határozni a kimenet értékét. Tegyük fel hogy k lépésre akarunk előre jósolni. Ekkor a t és annál kisebb időbeli értékekről mért adatunk van. A jósolt kimenet a k-adik lépésben

$$\hat{y}(t+k|t) = \sum_{i=1}^{\infty} g_i \Delta u(t+k-i) + \hat{n}(t+k|t),$$

ahol $\hat{n}$ a jósolt hiba a $t+k$-adik időpontban. Ez azt mutatja, mennyivel tér el a modell által jósolt kimenet a mért kimenttől. Más szóval $\hat{n}(t+k|t) = y_{model}(t+k) - y(t+k)$, ahol $y(t)$ mindig az eredeti rendszer által adott kimenet. Tegyük fel, hogy ez a hiba konstans. Ekkor $\hat{n}(t+k|t) = \hat{n}(t) = y_{model}(t) - y(t)$, vagyis a modell és mért kimenet különbségével korrigálok. Ne feledjük, hogy a t-edik lépésben vagyunk jelenleg, így $y(t)$ értékét ismerjük, az valamilyen mért adat. Így az előbbi összefüggést az alábbi alakba írhatjuk

$$\hat{y}(t+k|t) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + \sum_{i=k+1}^{\infty} g_i \Delta u(t+k-i) + y(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i). \quad (1)$$

Mi történik akkor, ha a t időpillanat után, nem változtatok a bemeneti értéken, vagyis magára hagyom a rendszert? Ekkor $\Delta u(t+k-i) = 0$, minden $i = 1..k$-ra. Jelöljük ezt a $t+k$ beli kimenetet $f(t+k)$-val. Nyilvánvalóan

$$f(t+k) = \sum_{i=k+1}^{\infty} g_i \Delta u(t+k-i) + y(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i), \quad (2)$$

az összegzést átindexelve

$$f(t+k) = y(t) + \sum_{i=1}^{\infty}(g_{k+i} - g_i)\Delta u(t-i).$$

Ezt hívjuk a rendszer szabad válaszának (*free response*), ez a jósolt kimenete a magára hagyott rendszernek. Más szóval a jósolt kimenet azon része, amely nem függ kontrol jövőbeli választásától. Az (1) összefüggésből, a jósolt kimenet az alábbi:

$$\hat{y}(t+k|t) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + f(t+k).$$

Ha feltesszük, hogy a rendszer stabil, vagyis létezik $\lim_{n\to\infty} y(t)$ létezik és véges, így elég nagy N-re $g_i \approx g_N$ minden $i > N$-re. Így megfelelő N-re számolhatunk az alábbi szabad válasszal,

$$f(t+k) = y(t) + \sum_{i=1}^{N}(g_{k+i} - g_i)\Delta u(t-i).$$

Az N megválasztása egyáltalán nem lehet önkényes, túl kicsi N esetén az algoritmus nem feltétlenül vezet megfelelő eredményre. Hogy ez mit jelent pontosan, később látni fogjuk.

2.2 Az *impulse response* modell

Egy másik lehetőség az előrejelzésre az *impulse response* modell. Ebben az esetben a rendszer nem egy konstans 1 értékű u-val tapogatjuk le, hanem egy olyan jellel, mely a 0 időpillanat- ban 0-ról egyre vált, majd a következő lépésben vissza nullára. Más szóval a rendszer egy egyszeri egység impulzust kap, és az ebből kapott adatokkal építjük fel a modellt az alábbi módon.

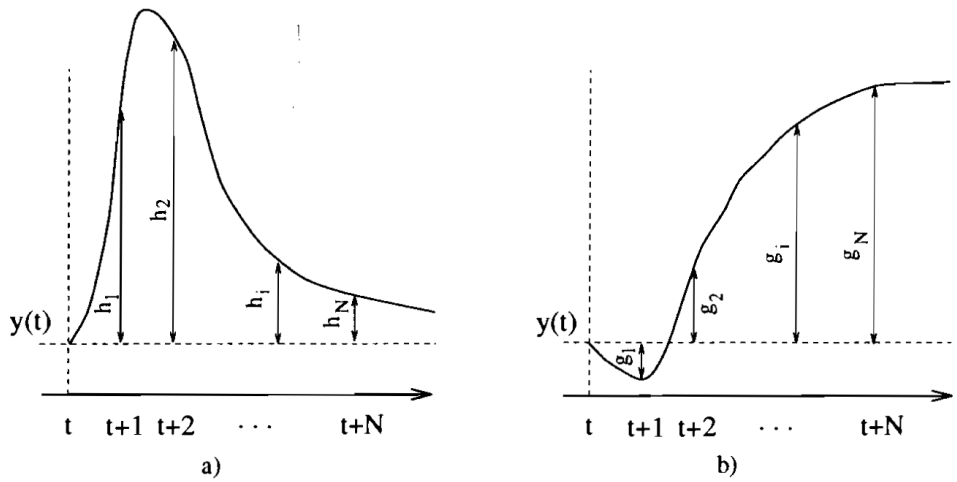
Működtessük a rendszert a fent említett u-val, azaz $u(t) = 1$ ha $0 < t \leq 1$ és minden más esetben $u(t) = 0$. Jelölje a rendszer kimenetét $t = k$-ban $y(k) = h_k$, ekkor az *impulse response* modell:

$$y_{model}(t) = \sum_{i=1}^{\infty} h_i u(t-i).$$

Az elképzelés a modell mögött egyszerűen az, hogy két időpillanat között az u-val arányosan nő az y értéke. Ha a rendszer stabil, $lim_{k\to\infty} h_k = 0$ így a végtelen szumma helyett dolgozhatunk az

$$y_{model}(t) = \sum_{i=1}^{N} h_i u(t-i)$$

alakkal, hiszen N-t megfelelően választva tetszőlegesen közel kerül a két összeg egymáshoz.



2.2. ábra. Az *impulse response* és *unit step response*

Az 2.2-es ábrán ([4] 15. oldal 2.1-es ábra) szemléltetjük a két megismert modell közti különbséget. Az ábrán g_i-k jelölik a *unit step response* kimeneteit, míg h_i-k az *implulse response* kimeneteit. A b, ábrán tehát egy olyan kontrollra adott választ látunk amely konstan egy, míg z a, ábrán egy olyan kontrollal dolgozunk ami a t és $t+1$ időpontok között 1 egyébként 0.

2.3 A DMC algoritmus

A következőekben memutatjuk, hogyan működik a DMC algoritmusa. A legegyszerűbb eset, ha a modellünkkel csak egy lépésre jósolunk előre kimenetet. Ekkor

$$\hat{y}(t+1|t) = g_1 \Delta u(t) + f(t+1).$$

Jelöljük a célfüggvényt $w(t)$-vel. Erre a trajektóriára akarjuk a rendszert vezérelni. Ekkor nyilván azt szeretnénk ha $\hat{y}(t+1|t)$ minél közelebb legyen $w(t+1)$-hez. Természetes ötlet, hogy vegyük a különbségük négyzetét, vagyis $(\hat{y}(t+1|t) - w(t+1))^2$-t. Ekkor az alábbi függvényt kell minimalizálnunk:

$$J = (g_1 \Delta u(t) + f(t+1) - w(t+1))^2,$$

ahol J az $\Delta u(t)$ egyváltozós függvénye. Ennek minimumhelye pont azt az impulzust adja vissza, amennyivel meg kell változtatnom $u(t-1)$-et (lehet pozitív és negatív is), s így megkapom $u(t)$-t. Egyszerű deriválás után adódik, hogy a minimumhelyét a

$$\Delta u(t) = 1/g_1(w(t+1) - f(t+1))$$

helyen veszi fel. Így $u(t) = u(t-1) + \Delta u(t)$, vagyis megvan a kívánt kontroll.

Az általános DMC algoritmus felépítése az alábbi. Tegyük fel, hogy m lépésre előre jósolunk *controll effort*-ot. Ekkor a jósolt kimentekre,

$$\hat{y}(t+1|t) = g_1 \Delta u(t) + f(t+1),$$

$$\hat{y}(t+2|t) = g_2 \Delta u(t) + g_1 \Delta u(t+1) + f(t+2),$$

$$\vdots$$

$$\hat{y}(t+m|t) = \sum_{i=1}^{m} g_i \Delta u(t+m-i) + f(t+m).$$

Ezt az alábbi mátrixos alakba írhatjuk,

$$\hat{y} = G\Delta u + f,$$

ahol,

$$\hat{y} = (\hat{y}(t+1|t), \ldots, (\hat{y}(t+m|t))^T,$$

$$\Delta u = (\Delta u(t), \ldots, \Delta u(t+m-1))^T,$$

$$f = (f(t+1), \ldots, f(t+m))^T,$$

illetve,

$G =$

$$\begin{pmatrix} g_1 & 0 & \dots & 0 \\ g_2 & g_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_m & g_{m-1} & \dots & g_1 \end{pmatrix}$$

(3)

a rendszer úgynevezett *dinamikus mátrixa*.

Emlékezzünk, hogy célunk továbbra is a megfelelő trajektóriára való vezérles, azonban most m különböző kimenetet kaptunk. Most a referencia trajektóriától való eltérést az alábbi kvadratikus függvénnyel reprezentáljuk:

$$J = \sum_{j=1}^{m} (\hat{y}(t+j|t) - w(t+j))^2.$$

Bevezetve a

$$w = (w(t+1), \dots, w(t+m))$$

vektort, ez nem más, mint

$$J = |G\Delta u + f - w|^2.$$

Vegyük észre, hogy mivel G alsóháromszög mátrix, és a főátlójában csupa nemnulla g_1-áll, létezik G^{-1}, s ekkor,

$$\Delta u = G^{-1}(f - w).$$

Jegyezzük meg, hogy e képlettel ugyan minden $t+k$ időpontra kapunk $\Delta u(t+k)$-re becslést, de csak a $\Delta u(t+1)$-re kapott értéket használjuk fel $y(t+1|t)$ kiszámításához. A következő időpillanatban újra végrehajtjuk az előző optimalizálást.

Természetes ötlet, hogy valamilyen módon szabályozzuk a kontroll paramétert. Ennek egyik módja Δu beépítése a minimalizálandó kvadratikus függvénybe, valamilyen $\lambda \geq 0$ paraméterrel. Ekkor

$$J = \sum_{j=1}^{m}(y(t+j|t)-w(t+j))^2+\sum_{j=1}^{m}\lambda\Delta(u(t+j-1))^2 = |G\Delta u+f-w|^2+\lambda|\Delta u|^2. \quad (4)$$

Ez egy kvadratikus pozitív definit függvény melynek minimumhelye ott lesz, ahol a Δu szerinti derivált 0. Deriválás után,

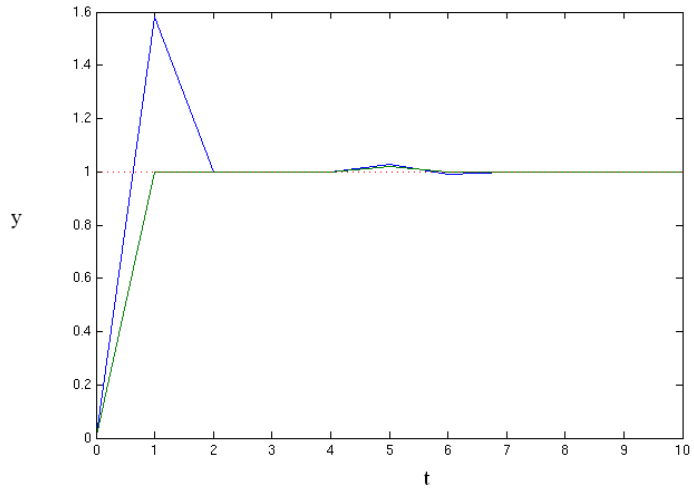
$$2G(G\Delta u + f - w) + 2\lambda\Delta u = 0$$

melyből az alábbi összefüggésre jutok:

$$\Delta u = (G^2 + \lambda I)^{-1}G(w - f). \quad (5)$$

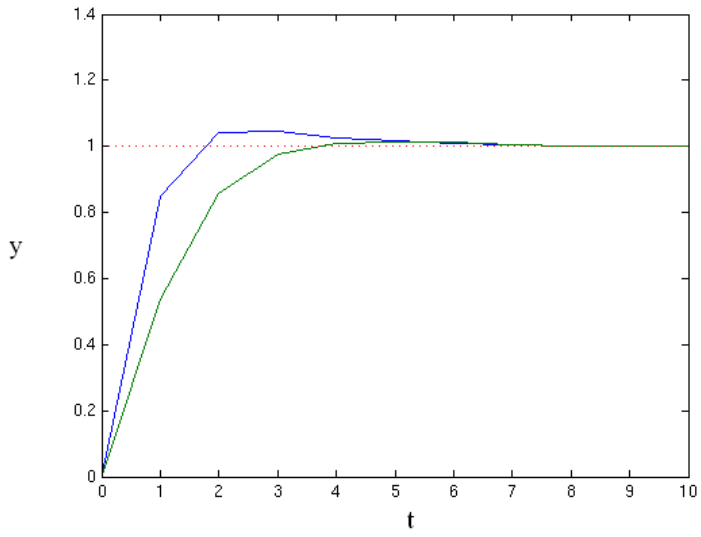
A λ paraméter azt fejezi ki, hogy mennyire büntetjük a kontroll gyors változását. Gyakorlati szempontból ennek igen nagy jelentősége van, hisz általában a kontroll egyfajta bemenő energiát fejez ki, ami erősen korlátos mennyiség.

Az alábbi ábrákon szemléltetem, hogyan változnak az algoritmus által adott $y(t)$-k illetve $u(t)$-k különböző λ-kra. A grafikonokon zöld pontozott vonal jelöli a referencia trajektóriát mely most konstans $w = 1$. Továbbá a kontrollfüggvényt($u(t)$) kék görbe illetve a kimenő jelet($y(t)$), az algoritmust 10 lépésig futtatom és $N = 4$ azaz 1-től 4-ig számolok különböző g_i-t, majd $i > 4$-re $g_i = g_4$.



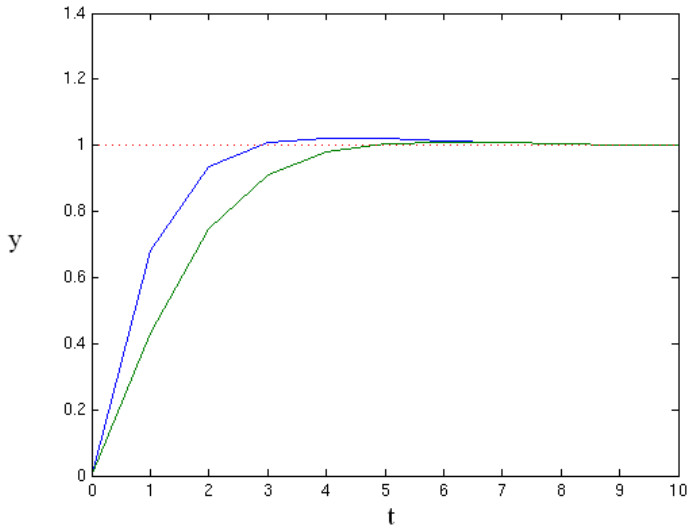
2.3. ábra. A $\lambda = 0$ eset

A 2.3-mas ábrán a $\lambda = 0$ esetet láthatjuk.



2.4. ábra. A $\lambda = 0.5$ eset

A 2.4-es ábrán az algoritmust a $\lambda = 0.5$ paraméterrel futtattuk.



2.5. ábra. A $\lambda = 1$ eset

Végül a 2.5-ös ábrán láthatjuk a $\lambda = 1$ esetet.

3 A módszer alkalmazása differenciálegyenletekre

Ebben a fejezetben azt az esetet vizsgáljuk, amikor az előzőekben definiált algoritmust egy differenciálegyenletre alkalmazzuk. Speciálisan egy lineáris rendszerre, amihez hozzáadódik a kontrollfüggvény. Az áttekinthetőség kedvéért először egydimenziós esettel foglalkozunk, s utána tárgyaljuk az általános rendszerek esetét. Vegyük észre, hogy az algoritmus felépítésekor nem használjuk a differenciálegyenlet paramétereit.

3.1 Az egydimenziós eset

Vegyük az alábbi differenciálegyenletet:

$$\dot{y}(t) = ay(t) + bu(t), \tag{6}$$

ahol $a, b \in \mathbb{R}$, és $u(t)$ a kontrollfüggvény. A célunk, hogy $y(t)$-t (illetve egy diszkretizációját) egy adott trajektóriára vezessük. Ezt a továbbiakban $w(t)$-vel jelöljük, ez lesz a referencia-trajektória. Tegyük fel, hogy $u(t) = u$ valamilyen konstans. Ekkor a megoldás, könnyen kiszámítható:

$$y(t) = e^{at}y_0 + \frac{b}{a}(e^{at} - 1)u. \tag{7}$$

Diszkretizáljuk (7)-t τ lépésközt választva. Keressünk olyan kontrollt, mely konstans minden $[\tau i, \tau(i+1)]$ szakaszon. Így ezeken a szakaszokon (7) érvényes. Jelöljük továbbá α-val $e^{a\tau}$-t. Ekkor az alábbi diszkrét rendszert kapjuk:

$$y_{k+1} = \alpha y_k + \frac{b}{a}(\alpha - 1)u_k. \tag{8}$$

Ahol a megfelelő k alsóindex azt jelenti hogy a $k\tau$ időpillanatban vagyunk.

A fenti rendszer (8) lineáris, idő invariáns és diszkrét, így alkalmazhatjuk rá a DMC algoritmust. Először is a *step response* modell megalkotásához ki kell számolnunk, hogyan reagál a rendszer egységnyi kontrollra, azaz ha a kontrollfüggvény a 0 időpillanat- ban vált konstans $u = 1$-re. A (8) képlet alapján

$$g_{k+1} = \alpha g_k + \frac{b}{a}(\alpha - 1).$$

Továbbá $g_1 = \frac{b}{a}(\alpha - 1)$, illetve $g_2 = \alpha\frac{b}{a}(\alpha - 1) + \frac{b}{a}(\alpha - 1) = \frac{b}{a}(\alpha^2 + 1)$. E gondolatmenetet folytatva g_k-t explicite is megadhatjuk,

$$g_k = \frac{b}{a}(\alpha^k - 1).$$

Vegyük észre, hogy g_1 pont u_k együtthatója (7)-ben, így azt

$$y_{k+1} = \alpha y_k + g_1 u_k$$

alakba írhatjuk. Továbbá a rendszer szabad válasza jelen jelölések mellett

$$f_k = (f(k\tau)...f((k+p)\tau)),$$

ahol $f(t)$ kiszámítását a (2) képletből ismerjük.

Az algoritmus tehát az alábbi lépésekből all,

1. Egytől n-ig, kiszámítom a g_k-k értékét, oly módon, hogy a rendszernek a k-adik lépésben mindig $g_{k-1} = y_{k-1}$ illetve $1 = u_{k-1}$ bemenő paramétert adok meg.

2. A kiszámított g_i-k-ből, (3) alapján felépítem a dinamikus mátrixot.

3. Tegyük fel, hogy y_k és u_{k-1} ismertek. Elvégzem az optimalizálást melyből a $\Delta u_k = G^{-1}(w - f_k)$ értékhez jutok. Így $u_k = u_{t-1} + \Delta u_k[1]$. (A Δu_k vektor is lehet, $\Delta u_k[1]$ az első koordinátát jelöli.)

4. A kapott u_k és a már ismert y_k értékeket bemenő paraméternek állítva, a (8) rendszerből megkapjuk y_{k+1}-et.

5. A $(3) - (4)$ lépéseket addig ismételem, amíg el nem jutunk addig az időpontig, ameddig a rendszert vezérelni akarjuk.

3.2 A többdimenziós eset

A fenti (6) rendszert könnyen általánosíthatjuk több dimenzióra. Induljunk ki az

$$\dot{x}(t) = Ax(t)$$

dinamikus rendszerből, ahol $A \in \mathbb{R}^{n\times n}$ és $x(t)$ n-dimenziós vektorfüggvény, amely folytonosan differenciálható minden koordinátájában, hogy a feladat értelmes legyen. A kontroll továbbra is egy $u(t)$ valós értékű függvény lesz, melyet egy $b \in \mathbb{R}^n$ vektorral szorozva a következő rendszert kapjuk:

$$\dot{x} = Ax + bu. \tag{9}$$

Továbbra is egydimenziós trajektóriára szeretnénk vezérelni a rendszert, így a kimenetet egy súlyvektorral skalárisan szorozva kapjuk az

$$y = c^T x \tag{10}$$

kimeneti függvényt. Jegyezzük meg, hogy mind c, mind b normálható, azaz feltehetjük, hogy $\|c\| = \|b\| = 1$. Ugyanis ha $\|c\| = \gamma$, akkor γ-val osztva a feladat csak annyiban változik, w helyett egy w/γ trajektóriára vezéreljük a feladatot. Másrészt, ha $\|b\| = \beta$, akkor b helyett b/β-t használva pusztán annyi a különbség, hogy u-helyett egy βu kontrollt kapunk.

Érdemes meggondolni, mit is jelentenek fizikailag a b és c vektorok. A b azt reprezentálja, hogy az adott koordinátában mennyire intenzív a kontroll hatása. A c vektor pedig azt fejezi ki, hogy mit is mérünk valójában. Példaként mérhetünk átlagos kimenetet, amikor minden c_i megegyezik, vagy adott i koordinátabeli kimenetet:

$$c = (0, \ldots, c_i, \ldots, 0).$$

Az egydimenziós esethez hasonlóan, konstans u-ra szintén megadhatjuk (9) pontos megoldását.

$$x(t) = e^{At}x(0) + (e^{At} - I)A^{-1}bu. \tag{11}$$

Hasonlóan járunk el, mint az egydimenzós esetben, olyan kontrollt keresünk, amely konstans $[\tau i, \tau(i+1)]$-en, valamilyen τ lépésközt választva. Így az alábbi diszkrét rendszerre jutunk:

$$x(t) = e^{A\tau}x(t-1) + (e^{A\tau} - I)A^{-1}bu(t-1). \tag{12}$$

A fenti (11) rendszer, diszkrét idejű dinamikai rendszer, melynek t-edik lépése az eredeti rendszerre nézve azt jelenti, hogy a $t\tau$ időpillanatban vagyunk. E rendszer lineáris, így a modell kimenetet becsülhetjük a *step response* modellel. Számitsuk ki az egységugrás- ra adott választ. Vegyük a (9) rendszert, $x(0) = 0$ és $u \equiv 1$ mellett, továbbá a szokásos $y_i = g_i$ jelöléssel:

$$g_i = c^T x(i) = c^T x(0) + c((e^{A\tau})^i - I)A^{-1}b = c^T((e^{A\tau})^i - I)A^{-1}b,$$

(13)

ha $i = 0, 1, \ldots, N$ és $g_i = g_N$ ha $i > N$. Így megadhatjuk a rendszer josólt kimenetét k-lépésre előre, ami az egydimenziós esettel teljesen egyező módon,

$$\hat{y}(t+k|t) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + f(t+k).$$

ahol f-et az (2) képlet adja.

1. Egytől N-ig, kiszámítom a g_k-k értékét, oly módon, hogy a rendszernek a k-adik lépésben mindig $g_{k-1} = y_{k-1}$ illetve $u = 1b$ bemenő paramétert adok meg.

2. A kiszámított g_k-k segítségével, (3) alapján felépítem a G dinamikus mátrixot.

3. Tegyük fel, hogy $y_k = cx(t)$ és $u(t-1)$ ismertek. Elvégzem az optimalizálást melyből az $G^{-1}(w - f(t)) = \Delta u(t)$ értékhez jutunkk. Így $u(t) = u(t-1) + \Delta u(t)$[1].

4. A $bu(t)$ és a már ismert $y(t)$ értékeket bemenő paraméternek állítva, a (12) rendszerből megkapjuk $y(t+1)$-et

5. A $(3) - (4)$ lépéseket addig ismételem, amíg el nem jutunk addig az időpontig, ameddig a rendszert vezérelni akarjuk.

4 A módszer stabilitásának vizsgálata konstans trajektória mellett

Ebben a fejezetben megmutatjuk, hogy a módszer milyen feltételek mellett használható, vagyis az algoritmus által adott $u(t)$-kkel kapott $y(t)$ kimenet mikor tarta a $w(t)$ célfüggvényhez. Vizsgált rendszernek az előző fejezetben differenciálegyenletből kapott diszkrét rendszert tekintjük. Először látni fogjuk, hogy az egydimenziós esetben minden stabil rendszerhez építhetünk stabil algoritmust. Utána megvizsgáljuk milyen plusz feltételek garantálhatju a stabilitást több dimenzióban.

Az alapfeltevésünk, hogy $w(t) = w$ konstans, de látjuk, hogy a módszer stabilitása valójában nem függ $w(t)$ változásától, így ez az eset általánosnak tekinthető. Ugyanakkor, ha $w(t)$ nem konstans, nyilvánvalóan nem érhetünk el tetszőleges pontosságot.

A rendszer hibamátrixának vizsgálatakor, annak sajátértékeire adunk felső becslést. Mivel mi norma szerinti konvergenciát akarunk belátni így felhasználjuk az alábbi állítást.

Állítás 4.0.1. *Legyen Q $k \times k$-as mátrix. Ekkor $\lim_{j\to\infty} Q^j v = 0$ minden v-re, akkor és csak akkor teljesül, ha $\rho(Q) < 1, ahol \rho(Q)$ a mátrix spektruma. Továbbá bármely rögzített $\|.\|$ normára és $\epsilon > 0$-ra létezik j_0 úgy, hogy minden $j > j_0$-ra*

$$\|Q^j v\| \leq (\rho(Q) + \epsilon)^j \|v\|$$

4.1 Az egydimenziós modell

Ebben a fejezetben azt vizsgáljuk, hogy a (6) rendszer mely feltételei esetén alkalmazhatjuk sikerrel a DMC algoritmust.

Ez a fejezet leginkább a már említett [1] cikkre támaszkodik. Látni fogjuk, hogy a hiba változását az idő függvényében egy mátrix-szorzással reprezentáljuk. E mátrix sajátértékeinek abszolút értékére felső becslést adunk, s mivel (bizonyos feltételek mellett) e sajátértékek az egységkörön belül helyezkednek el így megfogalmazhatunk egy hiba csökkenéséről szóló tételt, mi több annak ütemére is kapunk becslést.

4.1.1 A $\lambda = 0$ eset

Ebben az esetben $w(t) = w \in \mathbb{R}$ és $\lambda = 0$. Célunk meghatározni, hogyan változik $y_{k+1} - w$, $y_k - w$-hez képest tetszőleges k-ra. Ha azt kapnánk, hogy valamilyen 1-nél kisebb számszorosa, az garantálmá, hogy a k-ben vett limesz 0-hoz tart. Ezt az ütemet fogjuk meghatározni.

Először u_k-t határozzuk meg, méghozzá az algoritmus segítségével. Emlékezzünk, hogy számítottuk ki a kontrollt,

$$\Delta u_k = G^{-1}(w - f_k).$$

A fenti képletben Δu_k, w és f mind m dimenziós vektorok, de mivel nekünk csak Δu első koordinátája kell és G^{-1} alsó háromszög mátrix, ezért azt könnyen megadhatjuk,

$$u_k - u_{k-1} = \Delta u_k = \frac{1}{g_1}(w - f_{k+1}).$$

Emlékezzünk f definíciójára

$$f_{k+1} = y_k + \sum_{i=1}^{N-1}(g_{i+1} - g_i)\Delta u_{k-i},$$

így

$$u_k = \frac{1}{g_1}(w - y_k) + \sum_{i=1}^{N} \frac{2g_i - g_{i-1} - g_{i+1}}{g_1} u_{k-i}.$$

Ezzel az a gond, hogy szerepel benne $y(t)$, amit azonban kifejezhetünk a dinamikus rendszer diszkrét alakját felhasználva, vagyis, hogy

$$y_k = \alpha y_{k-1} + g_1 u_{k-1} \tag{14}$$

Ezt behelyettesítve, az alábbi összefüggésre jutunk:

$$u_k = -\frac{\alpha}{g_1}(y_{k-1}) + \frac{g_1 - g_2}{g_1} u_{k-1} + \sum_{i=1}^{N} \frac{2g_i - g_{i-1} - g_{i+1}}{g_1} u_{k-i} + \frac{w}{g_1}.$$

Azt tudjuk hogy y_k-nek w-hez kell tartania, mihez tart ekkor u_k? Ha k időpillanatban w-ben vagyunk, $k+1$-ben is ott akarunk lenni így, a rendszer diszkrét alakjából a,

$$w = \alpha w + g_1 u_k$$

összefüggésre jutunk. Tehát ha a rendszert az adott w-re vezéreljük, akkor az u értéke az $(1-\alpha)w/g_1$ -ben stabilizálódik. Így a stabilitáshoz az $y_k - w$ mellett $u_k - (1-\alpha)w/g_1$ értékeket célszerű vizsgálni az idő előrehaladtával.

A (14) mindkét oldalából kivonva w-t,

$$y_k - w = \alpha(y_{k-1} - w) + g_1(u_{k-1} - (1-\alpha)\frac{w}{g_1}).$$

Másrészt $(u_k - (1-\alpha)w/g_1)$-et a_k-vel jelölve,

$$a_k = -\frac{\alpha}{g_1}(y_{k-1} - w) + \frac{g_1 - g_2}{g_1}a_{k-1} + \sum_{i=1}^{N} \frac{2g_i - g_{i-1} - g_{i+1}}{g_1}a_{k-i}.$$

Az egészet írjuk mátrixos alakba. Vezessük be az alábbi vektort:

$$v_k = (y_k - w, u_k - (1-\alpha)w/g_1, \ldots, u_{k-N+1} - (1-\alpha)w/g_1).$$

Így az alábbi alakra jutunk:

$$v_k = M_N v_{k-1},$$

ahol M_N $N+1 \times N+1$-es mátrix, $M_N =$

$$\begin{pmatrix} \alpha & g_1 & 0 & \ldots & 0 & 0 \\ -\frac{\alpha}{g_1} & \frac{g_1-g_2}{g_1} & \frac{2g_2-g_1-g_3}{g_1} & \ldots & \frac{2g_{N-1}-g_{N-2}-g_N}{g_1} & \frac{2g_N-g_{N-1}-g_{N+1}}{g_1} \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0. \end{pmatrix}$$

Mivel, ha $\|M_N\| < 1$, akkor $\lim_{n\to\infty} \|M_N^n v\| = 0$, így ez azt jelenti hogy a módszerünk konvergens bármely v-re. Más szóval, ha $\|M_N\| = R$, akkor a referencia trajektóriától való eltérés az algoritmus n-edik lépésében kisebb, mint az eredeti különbség R^n-szerese. Formalizálva,

$$|y_{k+n} - w| \leq R^n |y_k - w|.$$

Így a konvergencia nem függ sem a kezdeti állapottól, sem az elérendő értéktől. Ezért a továbbiakban azt fogjuk vizsgálni, hogy mely esetekben teljesül

$$\|M_N\| < 1.$$

Ehhez elég, hogy $\rho(M_N) < 1$, vagyis a spektrálsugár 1-nél kisebb.

A sajátértékekre közvetlenül a mátrix karakterisztikus polinomjából adunk becslést. A karakterisztikus polinom kiszámításához az alábbi lemma [3] lesz segítségünkre.

Lemma 4.1.1. *Az alábbi $k \times k$-as mátrix,*

$$\begin{pmatrix} c_{k-1} & c_{k-2} & \dots & c_1 & c_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

karakterisztikus polinomja,

$$(-1)^k(x^k - c_{k-1}x^{k-1} - \dots - c_1 x - c_0).$$

A karakterisztikus polinom gyökeinek becsléséhez szükségünk lesz egy komplex függvénytanból ismert eredményre, a Rouché-tételre [4] .

Lemma 4.1.2. *Legyen Ω a komplex számsík egy korlátos tartománya, és $\gamma \subset \Omega$ Jordan-görbe. Tegyük fel, hogy f_0 illetve f_1 holomorf függvények, melyekre $|f_0(z) - f_1(z)| < |f_0(z)|$, minden $z \in \gamma$ esetén. Ekkor f_0-nak és f_1-nek multiplicitással számolva ugyanannyi gyöke van γ belsejében.*

Tétel 4.1.3. *Ha az eredeti rendszer stabil volt, azaz (6)-ben $a < 0$ akkor létezik N úgy, hogy $\rho(M_N) < 1$ teljesül így a DMC módszerrel a rendszer a kívánt trajektóriához tart.*

Bizonyítás.

Írjuk fel M_N karakterisztikus polinomját, vagyis $\det(M_N - xI)$-t. Alkalmazzuk a determinánsokról szóló kifejtési tételt az első oszlopra. Így a keresett polinom a (4.1.1) lemma alapján,

$$p_N(x) = (-1)^N(\alpha - x)(x^N - \frac{g_1 - g_2}{g_1}x^{N-1} - \sum_{i=2}^{N} \frac{2g_i - g_{i-1} - g_{i+1}}{g_1}x^{N-i}).$$

Vegyük észre, hogy g_i-k definíciója alapján

$$\frac{g_1 - g_2}{g_1} = \frac{\alpha - \alpha^2}{\alpha - 1} = -\alpha,$$

továbbá

$$\frac{g_N - g_{N-1}}{g_1} = \frac{\alpha^N - \alpha^{N-1}}{\alpha - 1} = \alpha^{N-1}.$$

Így a karakterisztikus ponlinomot az alábbi egyszerű alakra hozhatjuk:

$$p_N = (-1)^N(\alpha - x)(x^N + \alpha x^{N-1} + \sum_{i=2}^{N-1}(\alpha^i - \alpha^{i-1})x^{N-i} - \alpha^{N-1} + (-1)^{N-1}\alpha x^{N-1}),$$

(15)

azaz

$$p_N = (-1)^{N+1}(x^{N+1} - \alpha^N x + \alpha^N).$$

Azt szeretnénk, hogy p_N gyökei az egységkörön belül legyenek. Ennél pontosabban is meghatározzuk hollétüket, nevezetesen, hogy a

$$R = 3^{\frac{1}{N+1}}|\alpha|^{\frac{N}{N+1}}$$

sugarú körben helyezkednek el. Ehhez felhasznájuk a Rouche-tételt (4.1.2. Lemma) $f_0(z) = (-1)^{N+1}z^{N+1}$, illetve $f_1(z) = p_N(z)$ választással, a $\gamma = |z| = R$ görbén. Mivel $z \in \gamma$, ezért szükségképpen $|z| < 2$, tehát

$$|f_0(z) - f_1(z)| = \alpha^N|z - 1| < 3|\alpha|^N = |z|^{N+1} = |f_0(z)|.$$

Következésképp p_N-nek γ-n belül ugyanannyi gyöke van, mint z^{N+1}-nek, amelynek minden gyöke 0, ezért p_N-nek minden gyöke kisebb, mint 1.

Így ha a rendszer stabil, $a < 0$, így $e^{a\tau} = \alpha < 1$ a gyökök, tehát M_N sajátértékei kisebbek mint 1, vagyis a módszer konvergens. Ezzel a konvergencia sebességére is

becslést adunk, e_0-lal jelölve a kezdeti hibát felhasználva a 4.0.1 állítást, az n-edik lépésben az eltérés

$$e_n = (3^{\frac{1}{N+1}}|\alpha|^{\frac{N}{N+1}} + \epsilon)^n e_0.$$

Ezzel az állítást beláttuk.

Ha a rendszer nem stabil, a módszer sem feltétlenül adja a kívánt eredményt. Ennek fő okát könnyen láthatjuk, ha egy pillantást vetünk a *step sesponse modell*re. Emlékezzünk, hogy a modellben a rendszer úgynevezett szabad válasza

$$f_{k+n} = y_k + \sum_{i=1}^{\infty}(g_{k+i} - g_i)\Delta u_{k-i}.$$

Ehelyett ennek egy csonkított változatával számolunk azaz,

$$f_{k+n} = y_k + \sum_{i=1}^{N}(g_{k+i} - g_i)\Delta u_{k-i}.$$

Ezzel nem követünk el nagy hibát, hisz elég nagy i-re $g_{k+i} - g_i$ tetszőlegesen kicsi, amennyiben a rendszer stabil volt. Ugyanis, mint láttuk ekkor a g_i sorozat konvergens. Így, ha a rendszer nem stabil, ezt a csonkítást nem végezhetjük el. Ekkor a k-adik lépésben, ha a modellel 1 lépésre jósulunk,

$$\Delta u_k = g_1^{-1}(w - f_{k+1}).$$

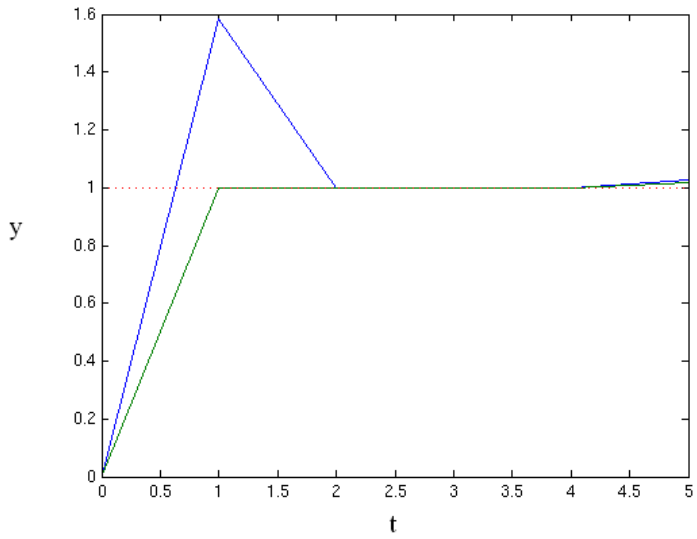
Ami lényeges különbség hogy f most egy végtelen szumma, vagy legalábbis k-ig kell összegeznünk. Nyilván k-ig elég hisz , ha $i > k$ akkor $\Delta u(i) = 0$. Így ha a rendszert M lépésig futtatom ugyanennyi g_i-t kellene kiszámolnunk. Ezt nyilván nem tehetjük meg mert a rendszert tetszőleges idejig szeretnénk vezérelni. Ebben az esetben megoldást jelenthet g_i-ket valamely N -ig kiszámítani, majd a továbbiakat extrapolációval közelítjük.

A 4.1-es ábrán a stabil

$$\dot{y}(t) = -y(t) + u(t)$$

differenciálegyenlethez felépített algoritmus által adott eredményeket mutatjuk be. A lépésköz $\tau = 1$, a predikciós horizont 5, illetve $N = 4$ vagyis g_4 az utolsó melyet

kiszámolunk a továbbiakban $g_4 = g_5 = ...$ Láthatjuk hogy $k = 4$-ig a megoldás elég pontos $k = 4$ után pedig fellép egy kis perturbáció.

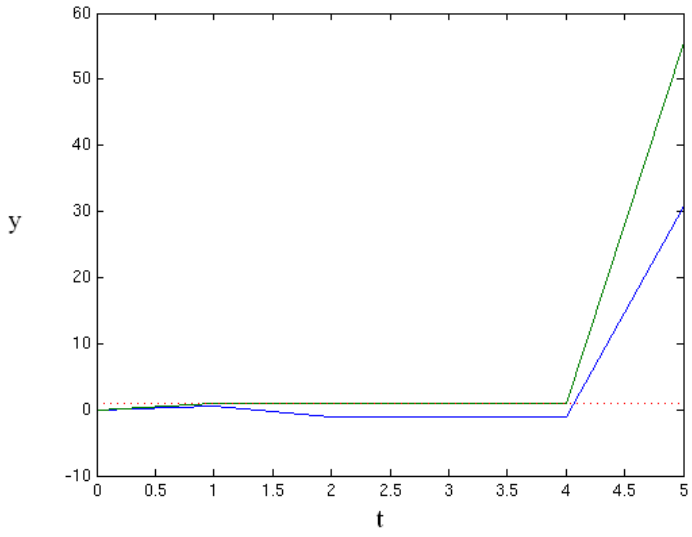


4.1. ábra. Az algoritmus első 4 lépése stabil esetben

Tekintsük most az

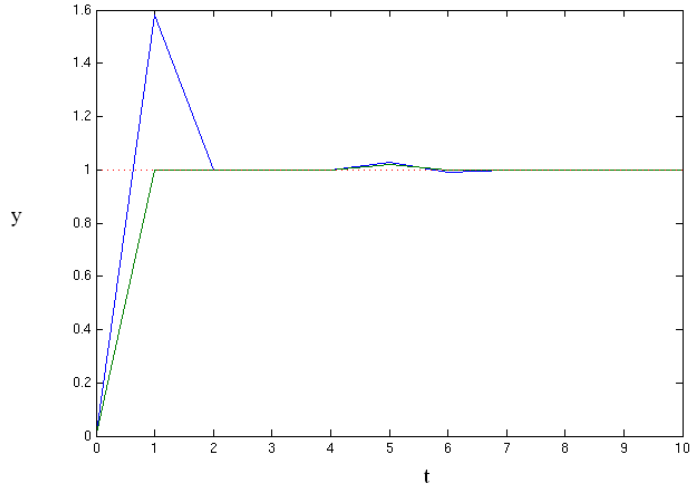
$$\dot{y}(t) = y(t) + u(t)$$

instabil rendszert. Építsük fel az algoritmust az előbbi paraméterekkel. Láthatjuk a 4.2-es ábrán, hogy itt is fellép $k = 4$ után a perturbáció, de az sokkal erőteljesebb, mint a stabil esetben.



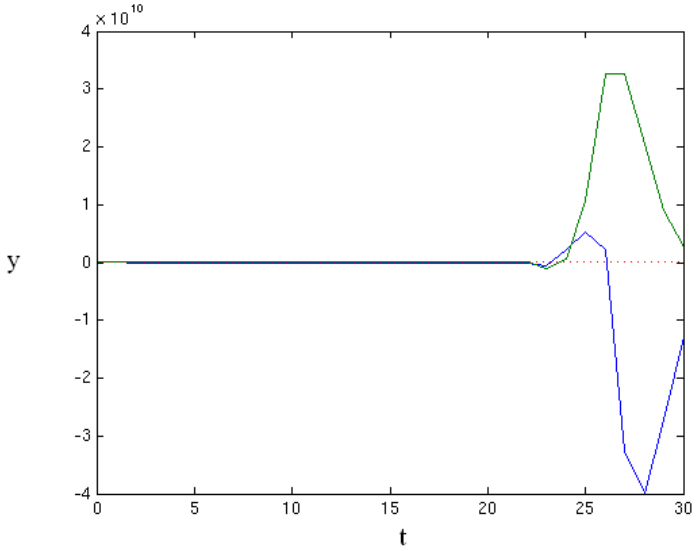
4.2. ábra. Az algoritmus első 4 lépése instabil esetben

De az igazi különbséget akkor látjuk ha tovább futtajuk az algoritmust, legyen most a predikciós horizont 10.



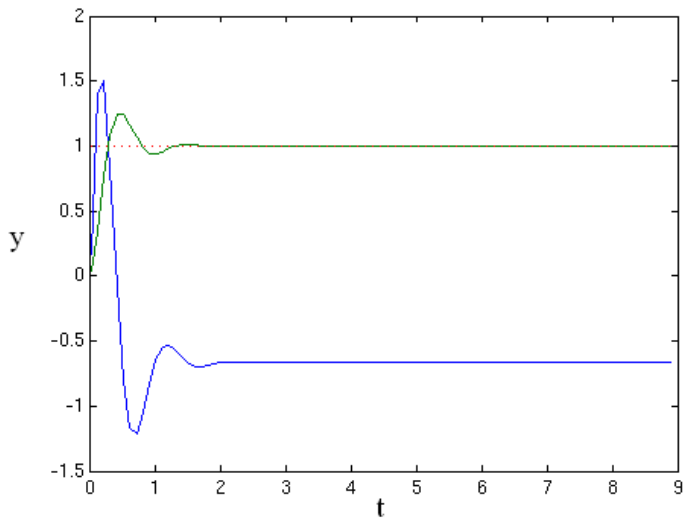
4.3. ábra. Az algoritmus első 10 lépése stabil esetben

A 4.3-mas ábrán látható, hogy a stabil differenciálegyenlet esetén, az algoritmus a kis perturbáció után, újra visszatalál a keresett trajektóriára, míg instabil esetben (4.4 ábra) $k = 4$ után a rendszer teljesen elszáll.



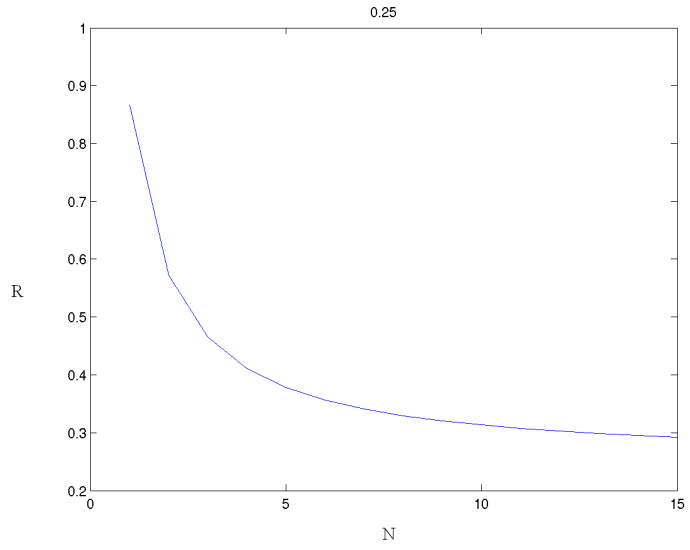
4.4. ábra. Az algoritmus viselkedése hosszútávon instabil esetben

Azonban, ha g_i-ket pontosan kiszámítjuk az egész horizontra nézve, akkor instabil esetben is a célfüggvényhez tartanak az algoritmus által adott y_k értékek, amint az a 4.5 ábrán látható.



4.5. ábra. Az algoritmus viselkedése instabil esetben teljes letapogatás mellett

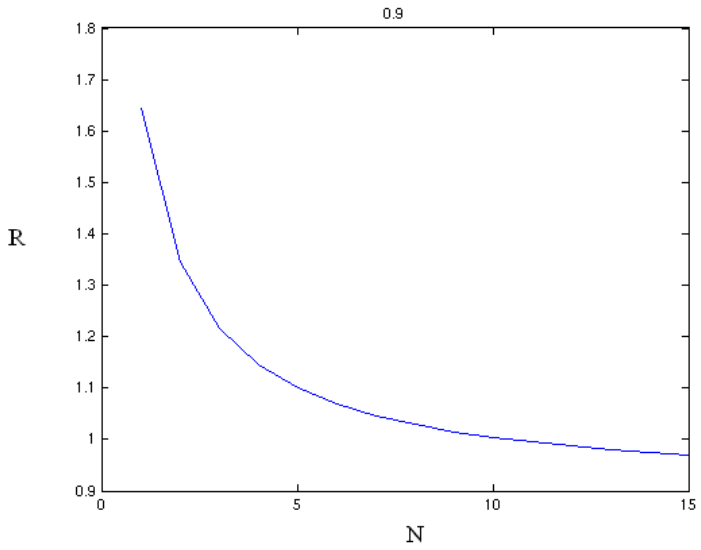
Vegyük észre hogy rögzített α mellett a hiba szorzója függ az N-től is, méghozzá annak monoton csökkenő függvénye. Az alábbi ábrán szemléltetem, hogyan változik az R néhány speciális α esetén.



4.6. ábra. A sajátértékek változása $\alpha = 0.25$ mellett

Nyilvánvalóan $\lim_{N\to\infty} R = \alpha$. Vagyis a DMC konvergenciájának sebessége nagyságrendileg megegyezik azzal a sebességgel, ahogy a kontroll nélküli rendszer a 0-hoz tart. Vegyük észre továbbá, hogy 1-hez közeli α esetén viszonylag nagy N-t kell

választanunk, hogy egyáltalán $R < 1$ teljesüljön. S ekkor a rendszer lassabban stabilizálódik.



4.7. ábra. A sajátértékek változása $\alpha = 0.9$ mellett

Ennek ellenére N értékét mégsem érdemes a nagyon nagynak választani, hiszen g_i-k értékét kiszámolni növeli a műveletigényt. Így meg kell gondolnunk, hogy az adott konvergencia sebességért mennyi műveletet vagyunk hajlandóak "áldozni".

4.1.2 A $\lambda > 0$ eset.

Ennek az esetnek speciális esetével foglalkozunk, nevezetesen amikor a *dinamikus mátrix* 1×1-es, vagyis $p = m = 1$. A $\lambda = 0$ esetben ennek nem volt jelentősége, mivel minden $p = m$ esetén (vagyis amikor a G mátrix négyzetes) ugyanaz az összefüggés adódik Δu első koordinátájára.

A *DMC* algoritmus $\lambda > 0$ esetéből (5) tudjuk, hogy

$$u_k - u_{k-1} = (G^2 + \lambda I)^{-1} G(w - f_{k+1}).$$

Mivel $G = [g_1]$, mint 1×1-es mátrix, így

$$u_k - u_{k-1} = \frac{g_1}{g_1^2 + \lambda}(w - f_{k+1}).$$

Vezessük be az alábbi paramétert,

$$\mu = \frac{g_1^2}{g_1^2 + \lambda}.$$

Így

$$u_k - u_{k-1} = \frac{\mu}{\gamma_1}(w - f_{k+1}).$$

Tudjuk, hogy

$$f_{k+1} = y_k + \sum_{i=1}^{N-1}(g_{i+1} - g_i)\Delta u_{k-i}.$$

Felhasználva, hogy

$$y_k = \alpha y_{k-1} + (\alpha - 1)u_{k-1},$$

az alábbi összefüggést írhatjuk fel:

$$u_k = -\frac{\mu}{g_1}\alpha y_{k-1} + (\mu\frac{g_1 - g_2}{g_1} + 1 - \mu) + \sum_{i=2}^{N}\mu\frac{2g_i - g_{i-1} - g_{i+1}}{g_1}u_{k-i}.$$

Illetve

$$y_k = \alpha y_{k-1} + g_1 u_k.$$

A fenti összefüggés mátrixos alakba irható nevezetesen

$$\hat{v}_{k+1} = M_N \hat{v}_k$$

ahol $\hat{v}_k = (y_k, u_k, \ldots, u_{k-N+1})^T$ és

$$M_N = \begin{pmatrix} \alpha & g_1 & 0 & \ldots & 0 & 0 \\ -\frac{\alpha}{g_1} & \mu\frac{g_1-g_2}{g_1} + 1 - \mu & \mu\frac{2g_2-g_1-g_3}{g_1} & \ldots & \mu\frac{2g_{N-1}-g_{N-2}-g_N}{g_1} & \mu\frac{2g_N-g_{N-1}-g_{N+1}}{g_1} \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{pmatrix}$$

a szokásos jelölésekkel, azaz $\alpha = e^{a\tau}$, továbbá $\mu = \frac{g_1^2}{g_1^2+\lambda}$.

Vegyük észre ha egy olyan s vektorral dolgozunk amelynek megfelelő koordinátái helyére a stabil állapotokat írom azaz, $y_k = w$ és $u_k = (1-\alpha)w/g_1$ mindent kívánt t-re, akkor az egyenlőség nyilván fennáll,

$$s = M_N s$$

.

Így $v = \hat{v} - s$-sel jelölve megkapjuk a hibavektort, mely az M_N mátrixszal szorzódik minden időlépésben. Így ha belátjuk, hogy e mátrix normája kisebb, mint 1 akkor a módszer által adott u_k-kel a rendszer a kívánt trajektóriához tart.

Állítás 4.1.4. *Az $\alpha > 0$ esetben M_N karakterisztikus polinomja*

$$p_N(x) = (-1)^{N+1}(x^{N+1} - (1-\mu)(\alpha+1)x^N + (1-\mu)\alpha x^{N-1} - \mu\alpha^N x + \mu\alpha^N).$$

Bizonyítás. Fejtsük ki a mátrixot az első oszlop szerint, továbbá hasznájuk fel hogy $(g_i - g_{i-1}/g_1 = \alpha^{i-1}$. Ekkor a (4.1.1) lemmát felhasználva a karakterisztikus polinom

$$(-1)^N(\alpha - x)(x^N + (\mu\alpha + 1 + \mu)x^{N-1} + \sum_{i=2}^{N-1} \mu(\alpha^i - \alpha^{i-1})x^{N-i} - \mu\alpha^{N-1}) + (-)^{N-1}\mu\alpha x^{N-1}.$$

Melyből a megfelelő tagok összevonása után adódik az állítás.

Állítás 4.1.5. *Minden $\alpha < 1$-hez létezik elég nagy N, illetve $0 \leq \mu^* < 1$ úgy, hogy minden $\mu^* \leq \mu \leq 1$-re p_N gyökei a $|z| = R$ körön belül vannak, ahol*

$$R = 3^{\frac{1}{N+1}}|\alpha|^{\frac{N}{N+1}}.$$

Bizonyítás. A $\lambda = 0$ esethez hasonlóan megkeressük azt a komplex kört, melyen a $f_0(z) = (-1)^{N+1}z^{N+1}$ és $f_1(z) = p_N(z)$ polinomokra alkalmazható a 4.1.2 lemma. Legyen R a kör sugara. Azt kell belátnunk, hogy $z = |R|$-re,

$$(1-\mu)|(\alpha+1)z^N - \alpha z - \alpha^N z + \alpha^N| < |z^{N+1} - \alpha^N z + \alpha^N|.$$

A jobb oldalt alulról, a bal oldalt felülről becsüljük. Mivel $|\alpha| < 1$, illetve $|z| = R < 1$, így

$$|z^{N+1} - \alpha^N z + \alpha^N| \geq |\alpha|^N,$$

illetve

$$(1-\mu)|(\alpha+1)z^N - \alpha z - \alpha^N z + \alpha^N \leq (1-\mu)5R^N,$$

következésképp a

$$(1-\mu)5R^N < |\alpha|^N$$

egyenlőtlenség teljesülése elég. A $\lambda = 0$ esetben használt R-rel a

$$1-\mu < \frac{1}{5}3^{-\frac{N}{N+1}}|\alpha|^{\frac{1}{N+1}}$$

feltételt kapjuk, melyhez valóban létezik $\mu*$ úgy, hogy az minden $\mu* < \mu$-re fennáll.

Tétel 4.1.6. *Ha az eredeti egydimenziós rendszer stabil volt, akkor létezik N úgy, hogy $\rho(M_N) < 1$ teljesül így a DMC módszerrel a rendszer a kívánt trajektóriához tart.*

Bizonyítás. Láttuk, hogy az algoritmus felépíthető úgy, hogy a hibamátrix spektrálsugara $R < 1$. Ekkor a 4.0.1 tétel alapján, a kezdeti hibát e_0-lal jelölve érvényes a

$$e_n < (R+\epsilon)^n e_0$$

becslés. Így beláttuk, hogy a hiba 0-hoz tart az idő előrehaladtával.

4.1.3 A nem konstans trajektória

Gondoljuk meg, mi történik ha a trajektória nem konstans, továbbá hogy a G dinamikus mátrix 1×1-es vagyis egyszerűen g_1 A hibabecslő képletben alapvetően w_{k+1} szerepel, aminek persze nincs jelentőssége, ha w konstans. Így a becslés,

$$|y_{k+1} - w_{k+1}| \leq R|y_k - w_{k+1}|.$$

Nekünk azonban $|y_k - w_k|$-függvényében kellene becslést adni. Legyen $\Delta w_{k+1} = w_{k+1} - w_k$. Ekkor,

$$R|y_k - w_k + \Delta w_{k+1}| \geq |y_{k+1} - w_{k+1}|.$$

Kihasználva a háromszög egyenlőtlenséget,

$$|y_{k+1} - w_{k+1}| \leq R|y_k - w_k| + |R\Delta w_{k+1}|.$$

Ez az összefüggés nyilván fennál a $k+2$ időpontban is így

$$|y_{k+2} - w_{k+2}| \leq R|y_{k+1} - w_{k+1}| + |R\Delta w_{k+2}|,$$

így a második lépésre a

$$|y_{k+2} - w_{k+2}| \leq R^2|y_k - w_k| + |R^2\Delta w_{k+2}| + R|\Delta w_{k+1}|.$$

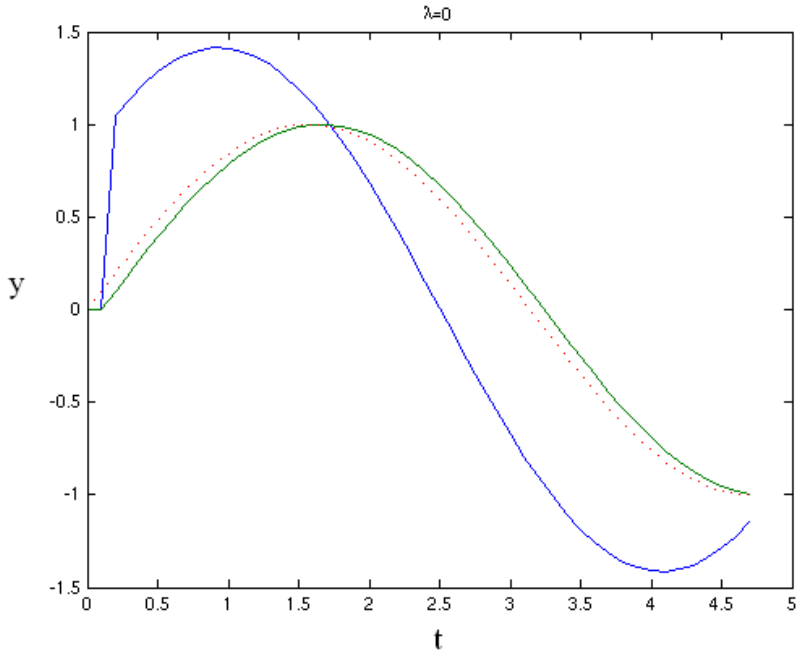
Általánosan a n-adik lépésre

$$|y_{k+n} - w_{k+n}| \leq R^n|y_k - w_k| + R^n|\Delta w_{k+1}| + \ldots + R^2|\Delta w_{k+n-1}| + R|\Delta w_{k+n}|.$$

Tegyük fel hogy létezik valamilyen L_w konstans, úgy, hogy minden k-re $|\Delta w_k| \leq C$. Ha w_k Lipschitz tulajdonságú függvény, akkor annak Lipschitz konstansa szorozva a lépésközzel megfelelő lesz. Ekkor a mértani sor összegképlete alapján a,

$$|y_{k+n} - w_{k+n}| \leq R^n|y_k - w_k| + L_w\tau|R\frac{R^{n+1}-1}{R-1}| \leq R^n|y_k - w_k| + L_w\tau|\frac{1}{1-R}|.$$

Rögzített α és N esetén $|\frac{1}{1-R}|$ konstans, így a hiba $o(\tau)$-val növekszik. Ezt figyelhetjük meg, egy folyamatosan változó trajektória esetén.



4.8. ábra. Az algoritmus viselkedése változó trajektóriára

A *3.2.1*-es ábrán $w(t) = sint$. Kiválóan megfigyelhető, a lépésközzel megegyező nagyságrendű eltolódás a célfüggvény és az algoritmus által adott kimenet között.

4.2 Az n-dimenziós modell, $\lambda = 0$ mellett

Tekintsük az n dimenziós rendszer diszkretizált alakját azaz,

$$x(t+1) = e^{A\tau}x(t) + (e^{A\tau} - I)A^{-1}bu(t),$$

(16)

$$y(t+1) = cx(t+1) \tag{17}$$

Jelen esetben a függvények argumentuma függ a diszkretizálástól. Tehát ha $x(t)$-t írok az azt jelenti, hogy a rendszer τt időpillanatban van.

Az egyszerűség kedvéért tegyük fel, hogy A diagonális. Ha A diagonalizálható ekkor létezik P mátrix amelyre $P^{-1}AP = \widetilde{A}$ ahol $\widetilde{A}$, diagonális mátrix, melynek a főátlóban lévő elemei A sajátértékei. Ekkor a rendszer a következőképpen módosul:

$$P^{-1}\dot{x} = (P^{-1}AP)(P^{-1}x) + (P^{-1}b)u$$

$$y = (c^T P)(P^{-1}x).$$

Így a fenti rendszerrel dolgozhatunk.

Számítsuk ki, hogy a $y(t) = w$ stabil kimenethez, milyen u és x értékek tartoznak. Ha a rendszer már beállt a stabil állapotra akkor,

$$x = e^{A\tau}x + (e^{A\tau} - I)A^{-1}bu.$$

Átrendezve,

$$-(e^{A\tau} - I)x = (e^{A\tau} - I)A^{-1}bu.$$

Ha az A diagonális mátrix főátlójában nemnulla elemek vannak, akkor $(e^{A\tau} - I)$ főátlójában is, így az invertálható. Balról $(e^{A\tau} - I)^{-1}$-el szorozva az

$$x = A^{-1}bu,$$

összefüggésre jutunk. Szorozzunk be balról a c sorvektorral, figyelembe véve, hogy $cx = y = w$.

$$w = cx = cA^{-1}bu,$$

A $cA^{-1}b$ mátrix szorzat egy szám lesz, így azzal leoszthatunk. A stabil állapotok, adott w mellett tehát:

$$u = -\frac{w}{cA^{-1}b},$$

$$x = \frac{A^{-1}bw}{cA^{-1}b}.$$

Most vizsgáljuk a hibát vagyis $x(t) - \frac{A^{-1}bw}{cA^{-1}b}$-t. Egyrészt $x(t)$-t kifejezhetem $x(t-1)$ és $u(t-1)$ segítségével. Másrészt a diszkrét képlétbe a stabil értékeket helyettesítve,

$$\frac{A^{-1}bw}{cA^{-1}b} = e^{A\tau}\frac{A^{-1}bw}{cA^{-1}b} - (e^{A\tau} - I)A^{-1}b\frac{w}{cA^{-1}b},$$

Legyen

$$\tilde{x}(t) = x(t) - \frac{A^{-1}bw}{cA^{-1}b},$$

$$\tilde{u}(t) = u(t) + -\frac{w}{cA^{-1}b}.$$

Így

$$\tilde{x}(t) = e^{A\tau}\tilde{x}(t-1) + (e^{A\tau} - I)A^{-1}b\tilde{u}(t-1).$$

Az $u(t) = u(t-1) + \Delta u(t) = u(t-1) + 1/g_1(w - f)$ képletből pedig,

$$\tilde{u}(t) = \frac{ce^{A\tau}}{g_1}\tilde{x}(t-1) + \frac{g_1 - g_2}{g_1}\tilde{u}(t-1) + \sum_{i=2}^{N}\frac{2g_i - g_{i-1} - g_{i+1}}{g_1}.$$

Így mátrixos alakba rendsezhetjük a fenti összefüggéseket.

$$e(t) = M_N e(t-1).$$

Ahol

$$e(t) = (\tilde{\mathbf{x}}(t), \tilde{u}(t), \ldots, \tilde{(}u(t-N+1))).$$

illetve $M_N =$

$$\begin{pmatrix} e^{A\tau} & (e^{A\tau}-I)A^{-1}b & 0 & \dots & 0 & 0 \\ -\frac{ce^{A\tau}}{g_1} & \frac{g_1-g_2}{g_1} & \frac{2g_2-g_1-g_3}{g_1} & \dots & \frac{2g_{N-1}-g_{N-2}-g_N}{g_1} & \frac{2g_N-g_{N-1}-g_{N+1}}{g_1} \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

A továbbiakban e mátrix, sajátértékeire próbálunk becslést adni.

Első lépésként a mátrix karakterisztikus polinomját írjuk fel. A karakterisztikus polinomot nem közvetlenül számoljuk, az eggyel kisebb dimenziós mátrix karakterisztikus polinomjából fejezzük ki. Korábban láttuk, hogy feltehetjük, hogy az A-mátrixról feltehetjük, hogy diagonális. Tekintsük az alábbi n-dimenziós rendszer mátrixát és jelölje annak karakterisztikus polinomját p_N^n, továbbá tekintsük azt az $\hat{M}_N$ $n-1 \times n-1$-es mátrixot, mely az eredeti mátrixból úgy adódik, hogy annak első sorát, és oszlopát elhagyjuk, illetve ennek karatkterisztikus polinomja legyen p_N^{n-1}

Állítás 4.2.1. *Az M_N és $\hat{M}_N$ mátrixokhoz tartozó karakterisztikus polinomok között az alábbi összefüggés áll fenn:*

$$p_N^n(x) = (\alpha_1 - x)p_N^{n-1} - (-1)^{n+1}\frac{c_1 b_1 \alpha_1(\alpha_1 - 1)}{g_1 \lambda_1}\prod_{i=2}^{n}(\alpha_i - x)(-x)^{N-1}.$$

Bizonyítás.
Jelölje $e^{A\tau - I}A^{-1}b$ vektor i-edik koordinátáját β_i, és hasonlóan $(-ce^{A\tau})/g_1$ vektor i-edik koordinátáját γ_i. Vessünk egy pillantást $\hat{M}_N - Ix$-re,

$$\left(\begin{array}{c|ccccccc}
\alpha_1 - x & 0 & \dots & 0 & \beta_1 & 0 & \dots & 0 \\
0 & \alpha_2 - x & \dots & 0 & \beta_2 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \dots & \vdots \\
0 & 0 & \dots & \alpha_n - x & \beta_n & 0 & \dots & 0 \\
\gamma_1 & \gamma_1 & \dots & \gamma_n & \frac{g_1 - g_2}{g_1} & \frac{2g_2 - g_1 - g_3}{1} & \dots & \frac{2g_N - g_{N-1} - g_{n+1}}{g_1} \\
0 & 0 & \dots & 0 & 1 & -x & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \dots & 0 & \dots & 0 & 1 & -x
\end{array}\right).$$

Alkalmazzuk a determinánsokról szóló kifejtési tételt az első oszlopra. Így a determináns első sor első oszlop eleme által kifejtett rész azaz $(\alpha_{n+1} - x)p_N^n$. Ezen kívül még egy nem nulla elem van az első sorban mégpedig γ_1. Ekkor az aldeterminánshoz tartozó mátrix:

$$\begin{pmatrix}
0 & \dots & 0 & \beta_1 & 0 & \dots & 0 \\
\alpha_1 - x & \dots & 0 & \beta_2 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \dots & \vdots \\
0 & \dots & \alpha_{n-1} - x & \beta_{n-1} & 0 & \dots & 0 \\
0 & \dots & 0 & 1 & -x & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \dots & 0 & \dots & 0 & 1 & -x
\end{pmatrix}.$$

Melyet az első sor szerint fejtünk ki, melynek csak a β_1 a nemulla eleme. Így az alábbi mátrix karakterisztikus polinomját számoljuk:

$$\begin{pmatrix}
\alpha_1 - x & \dots & 0 & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \dots & \vdots \\
0 & \dots & \alpha_{n-1} - x & 0 & \dots & 0 \\
0 & \dots & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
0 & \dots & 0 & \dots & 1 & -x
\end{pmatrix}.$$

Ez egy úgynevezett blokk-diagnális mátrix, melyről ismeretes, hogy karakterisztikus polinomja a blokkok karakte- risztikus polinomjainak szorzata. Az $e^{A\tau}$ mátrix karakterisztikus polinomja

$$\prod_{i=1}^{n}(\alpha_i - x),$$

mivel $e^{A\tau}$ diagonális. A másik mátrix az alábbi:

$$\begin{pmatrix} -x & 0 & \dots & 0 & 0 \\ 1 & -x & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -x & 0 \\ 0 & 0 & \dots & 1 & -x \end{pmatrix}$$

Mely egy Jordan blokk melynek sajátértekei a főátlóban lévő elemek. Így ennek karakterisztikus polinomja $(-x)^{N-1}$. Így az összeg második tagja, $\hat{\beta}_1\hat{\gamma}_1(-x)^{N-1}\prod_{i=1}^{n}(\alpha_i - x)$. Vagyis a karakterisztikus polinom

$$p_N^n(x) = (\alpha_1 - x)p_N^n - \gamma_1\beta_1\prod_{i=2}^{n}(\alpha_i - x)(-x)^{N-1}.$$

Állítás 4.2.2. *Ha a (16)-(17) rendszerre $b = e_i$ vagy $c = e_i$, továbbá a rendszer stabil, akkor a hibamátrix sajátértékeire érvényes a* $\max(\rho(A), R_1)$ *felső becslés, ahol R_1 a p_N^1 maximális abszolútértékű gyökének hossza.*

Bizonyítás. Feltehetjük, hogy $i = n$, hiszen a koordináták felcserélésével a rendszeren nem változtatok. A 4.2.1 állításból, felhasználva, hogy vagy c_1=0, vagy $b_1 = 0$

$$p_N^n(x) = (\alpha_1 - x)p_N^n.$$

Ezt a gondolatmenetet folytatva

$$p_N^n = \prod_{i=2}^{n}(\alpha - x)p_N^1.$$

A fenti polinom gyökei, $\alpha_2 \dots \alpha_N$ vagy valamilyen R_1-nél kisebb abszolút értékű komplex szám.

Így ebben a speciális esetben a feladatot az alábbi mátrix sajátértékeinek kiszámítására vezettük vissza:

$$\begin{pmatrix} \alpha_n & g_1 & 0 & \dots & 0 & 0 \\ -\frac{c_1\alpha_n}{g_1} & \frac{g_1-g_2}{g_1} & \frac{2g_2-g_1-g_3}{g_1} & \dots & \frac{2g_{N-1}-g_{N-2}-g_N}{g_1} & \frac{2g_N-g_{N-1}-g_{N+1}}{g_1} \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0. \end{pmatrix}$$

Tudjuk, hogy a b és c vektor konstansszorosának használata nem befolyásolja az algoritmust, így feltehetjük, hogy $b_n = c_n = 1$. Felhasználva a 4.1.3 bizonyításában használtakat, a fenti mátrix karakterisztikus polinomja

$$p_N^1(x) = (-1)^N(\alpha_n - x)(x^N - \frac{g_1 - g_2}{g_1}x^{N-1} - \sum_{i=2}^{N} \frac{2g_i - g_{i-1} - g_{i+1}}{g_1}x^{N-i}).$$

Ebben a speciális esetben g_i a (13) képlet alapján,

$$g_i = \frac{\alpha_n^i - 1}{\lambda_n}.$$

Így

$$\frac{g_i - g_{i-1}}{g_1} = \alpha_n^{i-1}.$$

Ezért a karakterisztikus polinom, felhasználva a (15) összefüggést

$$p_N = (-1)^{N+1}(x^{N+1} - \alpha_n^N x + \alpha_n^N)$$

alakra egyszerűsödik. Erről pedig tudjuk a 4.1.3 eredményéből, hogy a gyökei abszolútértékben kisebbek mint R_1, ahol

$$R_1 = 3^{\frac{1}{N+1}}|\alpha|^{\frac{N}{N+1}}$$

Így az alábbi következtetést vonhatjuk le

Tétel 4.2.3. *Ha a (16)-(17) (már diagonalizált) rendszerre $b = e_i$ vagy $c = e_i$, továbbá a rendszer stabil, akkor létezik N, úgy hogy a DMC algoritmus által adott $y(t)$ értékek a w konstans trajektóriához tartanak.*

Bizonyítás. A 4.2.2 állításból tudjuk, hogy létezik N, úgy hogy a hibamátrix sajátértékei valamilyen $R = \max_{i=1\ldots n}(R_1, \alpha_i) < 1$ sugarú komplex körben helyezkednek el. Ekkor felhasználva a (4.0.1) állítást, az alábbi becslésre jutunk:

$$\|M_N^j v\| \leq (R + \epsilon)^j \|v\|,$$

ahol megfelelő j mellett ϵ-t tetszőlegesen kicsire választhatom, speciálisan úgy, hogy $R + \epsilon < 1$ még teljesüljön. Ekkor láthatjuk, a hiba 0-hoz tart.

5 Összefoglalás

A szakdolgozatban megismerkedtünk, a Model Perdictive Control alapelveivel, annak előnyeivel, nevezetesen, hogy a rendszert annak belső ismerete nélkül vezérelhetjük. Egy MPC elveken alapuló konkrét algoritmust a dinamikus mátrix kontrollt teszteltük, olyan rendszerekre, melyeket differenciálegyenlettel adtunk meg. Azt tapasztaltuk, hogy stabil rendszerre a predikciós horizont egész kicsi részének letapogatásával előállíthatunk olyan algoritmust, mely a megfelelő bemeneti jelet, vagy kontrollfüggvényt adja ahhoz, hogy a kívánt trajektóriára vezéreljük a rendszert. Bevezettünk egy hibamátrixot, mellyel két időpillanatbeli hiba kapcsolatát tudtuk reprezentálni. Beláttuk hogy egy dimenzióban, stabil rendszerre, a mátrix felépíthető úgy, hogy sajátértékei kisebbek legyenek mint egy, amiből a norma nagyságára is következtettünk. Továbbá láttuk, hogy a trajektória változását az algoritmus $o(\tau)$ hibával tudja követni, ahol τ volt a diszkretizált rendszer lépésköze.

Végül általánosításként tekintettünk egy n-dimenziós differenciálegyenletrendszert, és ehhez is elkészítettük az algoritmust. Elkészítettük a hibamátrixot, és azt tapasztaltuk, hogy annak karakterisztikus polinomja kifejezhető, az eggyel kisebb dimenziós mátrixhoz tartozó karakterisztikus polinomból. Ennek segítsével elégséges feltételt adunk az algoritmus használhatóságára egy speciális n-dimenzós rendszer esetén.

6 Függelék

A DMC algoritmust meg is valósítottam Matlabban, a forráskód az alábbi. E program bármely olyan rendszerrel képes dolgozni, mely egy $\tau - 1$ időpillanatbeli $u(\tau-1)$ és $y(\tau-1)$ bemenetből meghatározza az $y(\tau)$ kimenetet. Érdekessége, hogy bár a módszer kidolgozása a lineáris differenciálegyenletekre szorítkozik, a program bizonyos nemlineáris esetre is használható eredményt ad.

A forráskódbeli jelölések a következők. A *lambda* a Δu csökkentésére használt paraméter, *tau* a lépésköz, illetve N jelöli a határt amíg az algoritmust futtatjuk. A lenti esetben a dinamikus mátrix 4×4-es, és $i = 1$-től 4-ig számoljuk ki a g_i értékeket,a további g_i értékekre $g_i = g_4$.

```
function DMC
lambda=.2;
tau=.1;
N=30;
y=zeros(N,1); f=zeros(4,1); U=zeros(2*N,1); w=1*ones(N+3,1);
g(1)=0;
for i=1:4 g(i+1)=rendszer1(tau,g(i),1); end
for i=5:N+4 g(i)=g(5); end
G=spdiags([ones(4,1)*g(5) ones(4,1)*g(4) ones(4,1)*g(3) ones(4,1)*g(2)], -3:0,
4,4);
for i=1:N
for k=1:4
S=0; for j=1:N-1 S=S+(g(k+j)-g(j))*(U(N+i-j)-U(N+i-j-1)); end
f(k)=y(i)+S; end
w1=[w(i), w(i+1), w(i+2), w(i+3)];
du=(G'*(w1'-f))'/(G'*G+lambda*eye(4));
U(N+i)=U(N+i-1)+du(1);
y(i+1)=rendszer1(tau,y(i),U(N+i)); end
U=U(N:2*N);
w=w(1:N+1);
x=0:.1:N/10;
plot(x(1:N-2),U(1:N-2),x(1:N-2),y(1:N-2),x(1:N-2),w(1:N-2),':')
```

```
e=w-y;
end
```

Hivatkozások

[1] Ádám Besenyei, Péter Simon: *Asymptotic output controllability via dynamic matrix control.*

[2] Eduardo F. Camacho, Carlos Bordons: *Modell Predictive Controll*, Springer-Verlag, London, 2004.

[3] R. A. Horn, Ch. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1990.

[4] Petruska György: *Komplex függvénytan*