Numerički algoritam za suboptimalno upravljanje linearnim dinamičkim sustavima uz prisustvo poremećaja

Šegota, Karla Marija

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:065928

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Karla Marija Šegota

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Izv. prof. dr. sc. Vladimir Milić, mag. ing.

Student:

Karla Marija Šegota

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Ovim putem zahvaljujem se svom mentoru, izv. prof. dr. sc. Vladimiru Miliću na pomoći i korisnim savjetima tijekom izrade ovog rada.

Veliko hvala cijeloj mojoj obitelji na podršci i strpljenju tijekom cijelog studija.

Također se zahvaljujem svojim prijateljima koji su mi bili velika pomoć na studiju.

Na kraju se želim zahvaliti dragom Bogu koji mi je dao snage i ustrajnosti za učenje tijekom svih ovih godina.

Karla Marija Šegota

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Karla Marija Šegota** JMBAG: 0035214198

Naslov rada na hrvatskom jeziku: **Numerički algoritam za suboptimalno upravljanje linearnim dinamičkim sustavima uz prisustvo poremećaja**

Naslov rada na engleskom jeziku: **Numerical algorithm for suboptimal control of linear dynamic systems in the presence of disturbances**

Opis zadatka:

Tema ovog diplomskog rada je numerički algoritam za sintezu suboptimalnog zakona upravljanja linearnim dinamičkim sustavima koji će omogućiti zadržavanje utjecaja poremećaja ispod dozvoljene granice i asimptotski stabilizirati zatvoreni sustav. Sa stanovišta teorije diferencijalnih igara, vektor upravljanja može se promatrati kao varijabla koja minimizira kriterij optimalnosti, dok se vektor poremećaja može promatrati kao varijabla koja maksimizira kriterij optimalnosti. Zbog toga se u literaturi ovaj problem često naziva min-max optimalno upravljanje. Kod većine se poznatih pristupa raznim metodama pokušava riješiti pripadajuća Hamilton-Jacobi-Isaacsova jednadžba (ili Riccatijeva jednadžba) uz unaprijed pretpostavljeni iznos L_2 pojačanja sustava. Ovaj problem je poznat kao takozvani suboptimalni H_∞ problem upravljanja. Ovaj diplomski rad zasnovan je na direktnoj min-max optimizaciji pri čemu osnovnu strukturu algoritma čini unazadno računanje ulančanih derivacija.
U diplomskom radu je potrebno:

1. Provesti teorijska razmatranja, sa stanovišta teorije diferencijalnih igara (tzv. min-max pristup), o metodama optimalnog upravljanja linearnim sustavima uz H_∞ kriterij optimalnosti.
2. Postaviti problem sinteze suboptimalnog zakona upravljanja po svim varijablama stanja linearnog sustava uz prisustvo poremećaja koji će biti zasnovan na minimizaciji (s obzirom na upravljačke varijable) uz istovremenu maksimizaciju (s obzirom na poremećajne varijable) iste funkcije cilja.
3. Izvesti algoritam za rješavanje prethodno opisanog problema. Algoritam u osnovnoj strukturi treba biti temeljen na metodi konjugiranog gradijenta za istovremenu minimizaciju i maksimizaciju funkcije cilja pri čemu se gradijenti računaju unazadno primjenom pravila ulančanih derivacija.
4. Izvedeni algoritam napisati u nekom višem programskom jeziku kao što su npr. matematički programski alati MATLAB ili NumPy/SciPy u Pythonu.
5. Provesti analizu i usporedbu rezultata izvedenog algoritma s onima koja se dobivaju rješavanjem pripadajuće Hamilton-Jacobi-Isaacsova (odnosno Riccatijeve) jednadžbe.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. svibnja 2024.	11. srpnja 2024.	15. – 19. srpnja 2024.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Vladimir Milić Prof. dr. sc. Ivica Garašić

SADRŽAJ

SADRŽAJ I
POPIS SLIKA II
POPIS TABLICA III
POPIS OZNAKA IV
SAŽETAK VI
SUMMARY VII
1. UVOD 1
2. H_∞ UPRAVLJANJE DINAMIČKIM LINEARNIM SUSTAVIMA 3
2.1. Linearni dinamički sustavi 3
2.1.1. Dinamički model linearnih sustava 3
2.1.2. Upravljivost 4
2.2. H_∞ kriterij optimalnosti 6
2.2.1. L_p stabilnost 6
2.2.2. H_∞ upravljanje 7
2.3. Primjena diferencijalnih igara na problem H_∞ upravljanja 9
2.3.1. Diferencijalne igre i minimaks princip 9
2.3.2. H_∞ upravljanje 10
3. NUMERIČKI ALGORITAM ZA H_∞ UPRAVLJANJE LINEARNIM DINAMIČKIM SUSTAVIMA 13
3.1. Formulacija problema 13
3.2. Vremenska diskretizacija Eulerovom metodom 14
3.3. Metoda konjugiranog gradijenta 16
3.3.1. Parametar β konjugirane gradijentne metode 17
3.3.2. SuperSAB metoda 18
3.3.3. Računanje gradijenta za zadani problem 19
3.4. Sumiranje algoritma 23
4. SIMULACIJSKI PRIMJER 25
4.1. Dinamički model linearnog sustava 25
4.2. Rezultati simulacije 26
4.3. Prilagođavanje ulaznih parametara 31
4.4. Usporedba 37
4.4.1. Usporedba s rješenjem standardnog gradijentnog algoritma 37
4.4.2. Usporedba s rješenjem Riccatijeve jednadžbe 38
5. ZAKLJUČAK 40
LITERATURA 41
PRILOZI 42
MATLAB KOD 43

POPIS SLIKA

Slika 3.1. Matlab kod za Eulerovu diskretizaciju 15
Slika 3.2. Matlab kod za izračun funkcije f(i) 15
Slika 3.3. Matlab kod SuperSAB algoritma 20
Slika 3.4. Matlab kod za računanje gradijenta 23
Slika 4.1. Varijable stanja u ovisnosti o vremenu 28
Slika 4.2. Upravljačka varijabla *u* i varijabla poremećaja *d* u ovisnosti o vremenu 29
Slika 4.3. Funkcija troška u ovisnosti s brojem iteracija 29
Slika 4.4. Dai-Yuan metoda izračuna parametra β 31
Slika 4.5. Funkcija troška standardna gradijentne metode u ovisnosti o broju iteracija 38
Slika 4.6. Ovisnost matrice pojačanja K o broju iteracija 39
Slika 4.7. Ovisnost matrice pojačanja W o broju iteracija 39

POPIS TABLICA

Tablica 3.1. Suma algoritma po koracima 23
Tablica 4.1. Usporedba različitih optimizacijskih metoda 30
Tablica 4.2. Utjecaj parametra β_{max} 32
Tablica 4.3. Utjecaj kriterija zaustavljanja ε 33
Tablica 4.4. Utjecaj početne stope konvergencije η_0 34
Tablica 4.5. Utjecaj duljine vremenskog koraka τ 35
Tablica 4.6. Utjecaj dilatacijskih parametara 36

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\boldsymbol{A}$	-	Matrica koeficijenata sustava
$\boldsymbol{B}$	-	Matrica ulaza sustava
$\boldsymbol{C}$	-	Matrica izlaza sustava
$\boldsymbol{D}$	-	Matrica prijenosa sustava
d^+, d^-	-	Dilatacijski parametri
d	-	Vektor smjera pretraživanja
$\boldsymbol{E}$	-	Matrica upravljivosti
$\boldsymbol{g(i)}$	-	Matrica upravljačke varijable i poremećaja
$G(s)$	-	Prijenosna funkcija sustava
$\boldsymbol{I}$	-	Jedinična matrica
j	-	Imaginarna jedinica
J	-	Funkcija troška
$\boldsymbol{K}$	-	Matrica pojačanja upravljačke varijable
$\boldsymbol{Q}$	-	Matrica težina stanja
$\boldsymbol{S}$	-	Matrica pretraživanja
t_f	-	Konačan broj točaka
x	-	Vektor stanja sustava
$\dot{x}$	-	Derivacija vektora stanja
u	-	Vektor upravljanja
$u(s)$	-	Laplaceove transformacija ulaza
$V(x)$	-	Lyapunovljeva funkcija stabilnosti
$\dot{V}(x)$	-	Derivacija Lyapunovljeve funkcije
$\boldsymbol{W}$	-	Matrica pojačanja poremećaja
$z(t)$	-	Kaznena varijabla
$y(t)$	-	Vektor izlaza
$y(s)$	-	Laplaceove transformacija izlaza
γ	-	Ograničenje H_∞ norme
$\boldsymbol{Z}$	-	Matrica pojačanja
α	-	Parametra veličine koraka
β	-	Parametar ažuriranja
$\boldsymbol{\lambda}$	-	Korak učenja

τ	-	Duljina vremenskog koraka
η	-	Stopa učenja
δ	-	Kroneckerov delta
$\|\cdot\|$	-	Općenita norma vektora
∇	-	Gradijent
min	-	Minimizacijski problem
max	-	Maksimizacijski problem
∂	-	Parcijalna derivacija
Tr()	-	Trag matrice

SAŽETAK

Diplomski rad obrađuje numerički algoritam za sintezu suboptimalnog zakona upravljanja linearnim dinamičkim sustavima koji će omogućiti zadržavanje utjecaja poremećaja ispod dozvoljene granice i asimptotski stabilizirati zatvoreni sustav. Algoritam je temeljen na metodi konjugiranog gradijenta za istovremenu minimizaciju i maksimizaciju funkcije troška pri čemu se gradijenti računaju unazadno primjenom pravila ulančanih derivacija. Metoda konjugiranog gradijenta je poboljšana SuperSAB algoritmom. Izvedeni algoritam je provjeren na dinamičkom sustavu jednostavne mase.

Ključne riječi: linearni vremenski invarijantni sustav, konjugirani gradijent, SuperSAB metoda, algoritam povratnog prostiranja kroz vrijeme, diferencijalne igre, igra nulte sume

SUMMARY

The master's theses addresses a numerical algorithm for the suboptimal control of linear dynamic systems that ensures that the disturbance influence remains below a permissible limit and asymptotically stabilizes the closed-loop system. The algorithm is based on conjugate gradient method for simultaneous minimization and maximization of the cost function, with gradients computed backwards using the chain rule. The conjugate gradient method is enhanced with the SuperSAB algorithm. The implemented algorithm was tested on a simple mass dynamic system.

Key words: linear time-invarijant system, conjugate gradient, SuperSAB method, backpropagation through time algorithm, differential games, zero-sum game

1. UVOD

Teorija igara se može primijeniti u raznim domenama života, u živim sustavima, u svakodnevnom životu, različitim domenama ljudskog društva, pa tako i inženjerskom. Može se primijeniti u svrhu predviđanja događaja, njegova objašnjenja ili planiranja. U mnogim slučajevima, strateški prostor donošenja odluka je prevelik što može učiniti problem gotovo nerješivim. Stoga se diferencijalne igre koriste kao procedure koje takve probleme čine rješivima uz modele sustava i donošenja odluka. U inženjerskim područjima, kombinacija optimizacije i diferencijalnih igara daje alate za upravljanje dinamičkim sustavima kako bi postigli što bolje performanse, za pronalazak najboljih rješenja za probleme sa više varijabli i ograničenja kao što je maksimizacija efikasnosti, stabilnost i željene performanse sustava, odabir strategija te omogućuju analizu i optimizaciju sustava sa suprotstavljenim ciljevima. Takve metode imaju svoju primjenu u robotici za kretanje robota u dinamičkom okruženju, energetici, konstrukciji za postizanje maksimalne čvrstoće i minimalne težine itd.

U ovom radu će se primijeniti algoritam povratnog prostiranja kroz vrijeme za sintezu suboptimalnog zakona upravljanja linearnim dinamičkim sustavima koji će omogućiti zadržavanje utjecaja poremećaja ispod dozvoljene granice i asimptotski stabilizirati zatvoreni sustav. Algoritam će se primijeniti u svrhu rješavanja minimaks suboptimalnog problema upravljanja sustava gdje se vektor poremećaja promatra kao varijabla koja maksimizira kriterij optimalnosti, dok se vektor upravljanja promatra kao varijabla koja minimizira kriterij optimalnosti. Za izračun minimuma i maksimuma funkcije troška koristi se metoda konjugiranog gradijenta.

U prvom dijelu rada provest će se teorijska razmatranja, sa stanovišta diferencijalnih igara, o metodama optimalnog upravljanja linearnim sustavima uz H_∞ kriterij optimalnosti. Tu će se opisati sustav u prostoru stanja te primjena diferencijalnih igara na problem H_∞ upravljanja. Predstavljeni su koncepti o stabilnosti i upravljivosti linearnih vremenski invarijantnih sustava zatim je opisana veza H_∞ uvjeta optimalnosti i $\mathcal{L}_2$ stabilnosti te njegova primjena u diferencijalnoj igri za H_∞ upravljanje.

Dugi dio nudi matematičku analizu algoritma povratnog prostiranja kroz vrijeme te njegove primjene na problem sinteze suboptimalnog zakona upravljanja po svim varijablama stanja linearnog sustava uz prisustvo poremećaja. Usporedit će se i različite metode optimizacije te utjecaj početnih parametara na efikasnost i uspješnost algoritma. Na kraju će se provesti analiza i usporedba rezultata izvedenog algoritma s onima koja se dobiju rješavanjem pripadajuće Hamilton-Jacobi-Isaacsova odnosno Riccatijeve jednadžbe.

2. H_∞ UPRAVLJANJE DINAMIČKIM LINEARNIM SUSTAVIMA

2.1. Linearni dinamički sustavi

2.1.1. Dinamički model linearnih sustava

Vremenski-invarijantni linearni kontinuirani dinamički sustavi su sustavi čija je veza između ulaza i izlaza linearna, a da matrice stanja nisu ovisne o vremenu. Može se prikazati slijedećim dinamičkim jednadžbama stanja i izlaza [1]:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t),\ x(t_0) = x_0, \tag{2.1.}$$

$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t) \tag{2.2.}$$

gdje su:

$x(t) \in \mathbb{R}^n$ - vektor stanja,
$u(t) \in \mathbb{R}^m$ - vektor upravljanja,
$y(t) \in \mathbb{R}^p$ - vektor izlaza,
$\boldsymbol{A} \in \mathbb{R}^{n\times n}$ - matrica koeficijenata sustava,
$\boldsymbol{B} \in \mathbb{R}^{n\times m}$ - matrica ulaza sustava,
$\boldsymbol{C} \in \mathbb{R}^{p\times n}$ - matrica izlaza sustava,
$\boldsymbol{D} \in \mathbb{R}^{p\times m}$ - matrica prijenosa sustava,

U slučaju kada je vektor upravljanja jednak nuli, $u(t) = 0$, govorimo o autonomnom linearnom sustavu te ga možemo prikazati slijedećim dinamičkim jednadžbama:

$$\dot{x}(t) = \boldsymbol{A}x(t),\ x(t_0) = x_0, \tag{2.3.}$$

$$y(t) = \boldsymbol{C}x(t) \tag{2.4.}$$

Stanje nehomogenog linearnog invarijantnog kontinuiranog sustava u bilo kojem trenutku možemo odrediti poznavanjem početnog stanja sustava $x(0)$ i prijelazne matrice sustava $e^{\boldsymbol{A}t}$ što se vidi iz slijedećih jednadžbi [1]:

$$x(t) = e^{\boldsymbol{A}t}x(0) + \int_0^t e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}u(\tau)\,d\tau \tag{2.5.}$$

$$y(t) = \boldsymbol{C}e^{\boldsymbol{A}t}x(0) + \int_0^t \boldsymbol{C}e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}u(\tau)\,d\tau + \boldsymbol{D}u(t) \tag{2.6.}$$

Matrica prijenosnih funkcija je omjer Laplace-ovih transformacija ulaza i izlaza, drugim riječima, to je Laplace-ova transformacija jediničnog impulsnog odziva sustava [2]. Važna je za sustav jer nam daje informacije o dinamičkom ponašanju sustava. Slijedi njezina definicija:

$$G(s) = \frac{Y(s)}{U(s)} \tag{2.7.}$$

Primjenom Laplaceove transformacije na jednadžbe sustava (2.1.) i (2.2.) dobivamo:

$$s\boldsymbol{X}(s) - x_0 = \boldsymbol{A}\boldsymbol{X}(s) + \boldsymbol{B}\boldsymbol{U}(s) \tag{2.8.}$$

$$\boldsymbol{Y}(s) = \boldsymbol{C}\boldsymbol{X}(s) + \boldsymbol{D}\boldsymbol{U}(s) \tag{2.9.}$$

$$\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D} \tag{2.10.}$$

Matricom prijenosnih funkcija $\boldsymbol{G}(s)$ može se definirati stabilnost sustava. Sustav je stabilan ako su polovi matrice prijenosnih funkcija $\boldsymbol{G}(s)$, odnosno svojstvene vrijednosti matrice $\boldsymbol{A}$, smješteni na lijevoj polovini kompleksna s-ravnine što znači da realni dijelovi korijena karakteristične jednadžbe $det[s\boldsymbol{I} - \boldsymbol{A}]$ moraju biti negativni $Re(s) < 0$.

2.1.2. Upravljivost

Sustav je upravljiv ako se može prevesti iz proizvoljnog početnog stanja u proizvoljno konačno stanje u konačnom vremenu. Govori nam postoji li uopće rješenje koje tražimo.

Linearni vremenski invarijantni sustav (2.1.) je potpuno upravljiv po stanjima u zatvorenom vremenskom intervalu ako postoji vektor upravljanja $u(t)$: $t_0 \leq t \leq t_f$ koji će prevesti sustav iz proizvoljnog početnog stanja $x(t_0)$ u proizvoljno konačno stanja $x(t_f)$ [2].

Pomoću rješenja diferencijalne jednadžbe (2.5.) možemo odrediti vektor stanja u nekom trenutku t_1:

$$x(t_1) = e^{\boldsymbol{A}t_1}x(0) + \int_0^{t_1} e^{\boldsymbol{A}(t_1-\tau)}\boldsymbol{B}u(\tau)\,d\tau \tag{2.11.}$$

Dobivena je integralna jednadžba s obzirom da je nepoznati vektor upravljanja ispod integrala. Izlučit ćemo ga te primijeniti Cayley-Hamiltonov teorem na matričnu funkciju $e^{A\tau}$ i razviti ju u polinom po matrici $\boldsymbol{A}$ [1].

$$e^{-\boldsymbol{A}t_1}x(t_1) - x(0) = \int_0^{t_1} e^{-\boldsymbol{A}\tau}\boldsymbol{B}u(\tau)\,d\tau \tag{2.12.}$$

$$e^{-\boldsymbol{A}\tau} = \sum_{j=0}^{n-1} \alpha_j(\tau)\boldsymbol{A}^j \tag{2.13.}$$

$$\boldsymbol{B}u(\tau) = \sum_{k=1}^{m} \boldsymbol{b}_k u_k(\tau) \tag{2.14.}$$

Jednadžbe (2.12). i (2.13.) uvrstimo u (2.14.):

$$e^{-\boldsymbol{A}t_1}x(t_1) - x(0) = \sum_{k=1}^{m}\sum_{j=0}^{n-1} \boldsymbol{A}^j\,\boldsymbol{b}_k \int_0^{t_1} \alpha_j(\tau)u_k(\tau)\,d\tau \tag{2.15.}$$

$$v_{jk} = \int_0^{t_1} \alpha_j(\tau)u_k(\tau)\,d\tau \tag{2.16.}$$

Uvrštavanjem (2.16.) u (2.17.) dobijemo:

$$e^{-\boldsymbol{A}t_1}x(t_1) - x(0) = \sum_{k=1}^{m}\sum_{j=0}^{n-1} v_{jk}\,\boldsymbol{A}^j\,\boldsymbol{b}_k \tag{2.17.}$$

S obzirom da su početno i konačno stanje proizvoljni, izraz možemo dekomponirati po vektorima $\boldsymbol{A}^j\mathbf{b}_k$, a da bi to bilo moguće od $n \times m$ vektora, n ih mora biti linearno nezavisno. Dekomponiranu matricu po vektorima $\boldsymbol{A}^j\mathbf{b}_k$ možemo raspisati prema jednadžbi (2.18.) te s obzirom da nam broj linearno neovisnih redaka ili stupaca matrice daje njezina rang, tažimo rang matrice $\boldsymbol{E}$ (2.19.) koji je uvjet potpune upravljivosti stanja linearnih kontinuiranih stanja sustava.

$$\boldsymbol{E} = [\boldsymbol{A}^{n-1}\boldsymbol{B}\ \ \boldsymbol{A}^{n-2}\boldsymbol{B}\ \ldots\ \boldsymbol{A}^{2}\boldsymbol{B}\ \ \boldsymbol{A}\boldsymbol{B}\ \ \boldsymbol{B}] \tag{2.18.}$$

$$rank(\boldsymbol{E}) = n \tag{2.19.}$$

2.2. H_∞ kriterij optimalnosti

H_∞ norma je matematički koncept korišten u teoriji upravljanja za mjerenje veze ulaza i izlaza sustava te performansi sustava. Ona kvantificira najgore pojačanje ili prigušenje ulaznog signala dok prolazi kroz sustav. Koristi se u metodologijama sinteze robusnog upravljanja kao što je optimalno H_∞ upravljanje kako bi se osigurala stabilnost i performansa sustava u prisutnosti nesigurnosti. Stoga se parametri sustava dizajniraju tako da se minimizira H_∞ norma uz zadovoljenje stabilnosti i performansi. Ograničavanjem najgoreg pojačanja, H_∞ norma daje sposobnost sustavu da osigura željeno ponašanje sustava u širokom rasponu radnih uvjeta.

2.2.1. L_p stabilnost

Ako dinamički sustav prikažemo slijedećom jednadžbom [1]:

$$\dot{x} = \boldsymbol{f}(x) \tag{2.20.}$$

Kažemo da je ravnotežno stanje $x = 0$ stabilno ako za neki $\varepsilon > 0$ postoji $\delta > 0$ tako da iz $\|x(t_0)\| < \delta$, slijedi $\|x(t)\| < \varepsilon$ za sve $t \geq t_0$, inače je ravnotežno stanje nestabilno.
Ako ulazno-izlazno preslikavanje simbolički predstavimo operatorskim izrazom:

$$y = Hu \tag{2.21.}$$

gdje je

u: $[0, \infty\rangle \rightarrow \mathbb{R}^m$ - ulazni vektor
y: $[0, \infty\rangle \rightarrow \mathbb{R}^n$ - izlazni vektor
H: $\mathcal{L}_{pe}^m \rightarrow \mathcal{L}_{pe}^n$ - operator koji preslikava ulazni vektor u izlazni

$\mathcal{L}_p$ stabilnost nam govori da je sustav stabilan ako za sustav reprezentiran operatorom H: $\mathcal{L}_{pe}^m \rightarrow \mathcal{L}_{pe}^n$ postoje nenegativne konstante $\mathcal{L}_p$ pojačanje γ i bias β tako da vrijedi slijedeća relacija [1]:

$$\|(Hu)_\tau\|_{\mathcal{L}_p} \leq \gamma \|u_\tau\|_{\mathcal{L}_p} + \beta,\ \ u\epsilon\mathcal{L}_{pe}^m,\ \ \tau\epsilon[0, \infty\rangle \tag{2.22.}$$

U limesu $\tau \rightarrow \infty$ dobijemo:

$$\|y\|_{\mathcal{L}_p} = \|Hu\|_{\mathcal{L}_p} \leq \gamma \|u\|_{\mathcal{L}_p} + \beta \tag{2.23.}$$

Za $p = 2$ dobit ćemo izraz za $\mathcal{L}_2$ stabilnost:

$$\|y\|_{\mathcal{L}_2} \leq \gamma \|u\|_{\mathcal{L}_2} + \beta \tag{2.24.}$$

$\mathcal{L}_2$ stabilnost nam govori da ulazni signal konačne energije uzrokuje izlazni signal konačne energije. Ako imamo sustav u ravnotežnom stanju i sustav je globalno asimptotski stabilan, tada vanjski poremećaj konačne energije uzrokuje regulacijsku pogrešku konačne energije što znači da će sustav nakon izbacivanja iz ravnotežnog stanja s vremenom ponovno konvergirati ravnotežnom stanju.

Utvrđivanje $\mathcal{L}_2$ stabilnosti svodi se na računanje $\mathcal{L}_p$ pojačanja γ što je bitno iz dva razloga. Prvi, ako je ulazni signal poremećaj, tada na osnovu izraza za $\mathcal{L}_p$ pojačanja možemo utvrditi način podešavanja parametara sustava s ciljem minimizacije pojačanja, a time i minimizacije utjecaja poremećaja na izlazne varijable. Drugi, ako imamo sustav koji se sastoji od spregnutih podsustava za koje znamo odgovarajuća $\mathcal{L}_p$ pojačanja, tada stabilnost sustava možemo utvrditi primjenom small-grain teorema [1].

Za linearni multivarijabilni sustav s konstanim koeficijentima (2.1.) sa njegovom matricom prijenosnih funkcija, $\mathcal{L}_p$ pojačanje sustava je jednako:

$$\gamma = sup_{\omega \in \mathbb{R}} \|\boldsymbol{G}(j\omega)\|_2 \tag{2.25.}$$

gdje je:

$\|\boldsymbol{G}(j\omega)\|_2$ - inducirana $\mathcal{L}_2$ norma kompleksna matrice $\boldsymbol{G}(j\omega)$, što je zapravo njezina H_∞ norma

2.2.2. H_∞ upravljanje

H_∞ norma je mjera maksimalnog pojačanja sustava preko svih frekvencija. Za stabilan linearni, vremenski invarijantan sustav s prijenosnom funkcijom $\boldsymbol{G}(s)$, H_∞ norma je definirana kao:

$$\|\boldsymbol{G}(s)\|_{H_\infty} = sup_{\omega \in \mathbb{R}} \|\boldsymbol{G}(j\omega)\|_\infty, \quad \text{s= } j\omega \tag{2.26.}$$

gdje je:

$\boldsymbol{G}(s)$ - prijenosna funkcija

ω - frekvencija

$\|\boldsymbol{G}(j\omega)\|_\infty$ - najveća singularna vrijednost matrice $\boldsymbol{G}(j\omega)$

Ako je sustav disipativan u odnosu na $\mathcal{L}_2$*-gain supply rate* [1]:

$$\dot{V}(x) \leq \gamma^2 u^T(t)u(t) - y^T(t)y(t) \tag{2.27.}$$

sustav je $\mathcal{L}_2$ stabilan sa $\mathcal{L}_2$ pojačanjem manjim ili jednakim γ.

Za linearni sustav:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t), \tag{2.28.}$$

$$y(t) = \boldsymbol{C}x(t) \tag{2.29.}$$

Sa funkcijom energije $V(x) = x^T Px$ dobijemo uvijete za izračunavanje $\mathcal{L}_2$ pojačanja γ tako da funkciju energije $V(x)$ deriviramo, uvrstimo u prethodni izraz (2.27.) te prebacimo sve članove na lijevo stranu:

$$x^T\boldsymbol{A}^T\boldsymbol{P}x + x^T\boldsymbol{P}\boldsymbol{A}x + u^T\boldsymbol{B}^T\boldsymbol{P}x + x^T\boldsymbol{P}\boldsymbol{B}u - \gamma^2 u^T u + x^T\boldsymbol{C}^T\boldsymbol{C}x \leq 0 \tag{2.30.}$$

Njezin izraz u matričnoj formi glasi:

$$[x^T\ u^T]\begin{bmatrix}\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{P}\boldsymbol{B} \\ \boldsymbol{B}^T\boldsymbol{P} & -\gamma^2\boldsymbol{I}\end{bmatrix}\begin{bmatrix}x \\ u\end{bmatrix} \leq 0 \tag{2.31.}$$

Da bi gornja kvadratna forma bila pozitivno definitna, mora biti zadovoljena slijedeća nejednakost:

$$\begin{bmatrix}\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{P}\boldsymbol{B} \\ \boldsymbol{B}^T\boldsymbol{P} & -\gamma^2\boldsymbol{I}\end{bmatrix} < 0 \tag{2.32.}$$

Ako želimo na osnovu ove kvadratne forme dobiti kvadratnu formu samo po vektoru x, to ćemo učiniti maksimizacijom desne strane izraza po vektoru u te ćemo dobiti:

$$\boldsymbol{u} = \frac{1}{\gamma^2}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{x} \tag{2.33.}$$

Ako taj izraz uvrstimo u izraz (2.31.), dobivamo:

$$\boldsymbol{x}^T\left[\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^T\boldsymbol{C} + \frac{1}{\gamma^2}\boldsymbol{P}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}\right]x < 0 \tag{2.34.}$$

Što će biti zadovoljeno ako postoji takav $P \geq 0$ koji zadovoljava kvazi Riccati kvatratnu matričnu nejednadžbu:

$$\left[\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^T\boldsymbol{C} + \frac{1}{\gamma^2}\boldsymbol{P}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}\right] < 0 \tag{2.35.}$$

Zakon upravljanja (2.33.) garantira da je $\mathcal{L}_p$ pojačanje zatvorenog kruga manje ili jednako γ. Ovo se naziva i H_∞ upravljanje jer je $\mathcal{L}_p$ pojačanje linearnog sustava (2.1.) ekvivalentno H_∞ normi matrice prijenosnih funkcija.

2.3. Primjena diferencijalnih igara na problem H_∞ upravljanja

2.3.1. Diferencijalne igre i minimaks princip

Diferencijalne igre su pristup problemima u teoriji igara vezani za modeliranje i analizu sukoba u kontekstu dinamičkog sustava. To je grana teorija igara koja se bavi strateškim interakcijama igrača u dinamičkom okruženju gdje promjenu stanja sustava u vremenu opisuju diferencijalne jednadžbe. Svaki igrač svojim varijablama i strategijama želi postići što bolji ishod.

Ravnotežno stanje u diferencijalnoj igri predstavlja situaciju u kojoj nijedan igrač ne može poboljšati svoj rezultat mijenjanjem svoje strategije bez suglasnosti drugih igrača. Dinamičke trajektorije diferencijalnih opisuju razvoj igre tokom vremena, a definirane su kao diferencijalne jednadžbe koje opisuju ponašanje svakog igrača u kontekstu strategija drugih igrača. Njihova analiza omogućuje predviđanje konačnih ishoda igre.

Teorija igara s nultom sumom datira još iz ranih 1920-ih. Tada je na ovoj temi radio Borel, on je uveo pojam sukobljene situacije donošenja odluke koja uključuje više igrača tj. onih koji donose odluke. Također je uveo i koncepte čiste i miješane strategije. Ali nije razvio potpunu teoriju igara s nultom sumom, čak je pretpostavio da je min-max teorem, koji se koristi u ovom radu, netočan. Prvi koji je iznio dokaz min-max teorema i postavio temelje teorije igara kakvu

danas poznajemo je Von Neumann 1930-ih godina. Proučavanje diferencijalnih igara započeo je Isaac nakon razvoja Pontryaginovog principa maksimuma kojim se pronalazi najbolji mogući način za prevođenje dinamičkog sustava iz jednog stanja u drugo uz prisutnosti ograničenja za stanje ili upravljačke varijable. Time je postalo jasno da postoji veza između diferencijalnih igara i teorije optimalnog upravljanja. Zapravo, problemi diferencijalnih igara su zapravo generalizirani problemi optimalnog upravljanja u slučajevima s više igrača. Tijekom 1960-ih i 1970-ih godina prepoznata je uloga teorije igara u dizajnu minimaks upravljača gdje je cilj minimizirati ishod pod najgorim mogućim smetnjama ili varijacijama parametara. To vodi do zaključka kako upravljač mora imati dinamičku strukturu što zahtjeva postavljanje diferencijalnih igara. 1980-ih godina su istraživanja napredovala te naraslo razumijevanje složenih igara između koncepata rješenja i dinamičkih informacijskih obrazaca u igrama s nultom sumom. Tih godina se primjena metoda uglavnom koristila za probleme procjene i stohastičkog upravljanja s nestrukturiranom nesigurnošću te za probleme strukturiranih nesigurnosti. Tih godina u teoriji upravljanja dominirao je H_∞ pristup. Također je metodologija dizajna upravljača u najgorem slučaju, ali u frekvencijskoj domeni.

Minimaks pristup je vrsta algoritma povratnog pretraživanja koji se često koristi u teoriji igara i donošenju odluka kako bi pronašli optimalno kretanje igrača s pretpostavkom da i protivnik igra optimalno. Sastoji se od dva igrača, jedan je minimizator, a drugi maksimizator gdje maksimizator pokušava postići što volji ishod, a minimizator pokušava suprotno, postići što lošiji ishod tj. niži rezultat. Minimaks pristup se primjenjuje u raznim područjima od družtvenih igara za procjenjivanje svih mogućih poteza suparnika za odabiranje poteza koji minimizira maksimalni mogući gubitak, dizajna sustava koji moraju raditi pod različitim uvjetima i smetnjama za osiguravanje optimalne izvedbe u najgorem scenariju, dizajna kontrolera za automomna vozila kako bi se osiguralo sigurno ponašanje u prisutnosti nepredvidivih smetnji do pretraživačkih algoritama i planiranja projekata.

2.3.2. H_∞ upravljanje

U diferencijalnoj igri za H_∞ upravljanje imamo dva igrača, upravljačku varijablu u i poremećajnu varijablu d, te taj sustav možemo opisati diferencijalnom jednadžbom koja slijedi:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B_1}u(t) + \boldsymbol{B_2}d(t) \tag{2.36.}$$

Cilj H_∞ upravljanja sustavom (2.1.) je odrediti zakon upravljanja $u = \boldsymbol{K}x(t)$ i najgori slučaj poremećaja $d = \boldsymbol{W}x(t)$ takav da je $\gamma > 0$ minimiziran što minimizacija H_∞ norme prijenosne funkcije od poremećaja do izlaza sustava. Funkcija troška po kojoj će se definirati diferencijalna igra nulte sume:

$$J = \int_{t_0}^{t_f} (\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \|\boldsymbol{u}\|^2 - \gamma^2 \|\boldsymbol{d}\|^2)\, dt \tag{2.37.}$$

U zadanoj funkciji za matricu $\boldsymbol{Q}$ pretpostavlja se da simetrična i pozitivno semidefinitna. Diferencijalnom igrom nulte sume tražimo sedlenu točku u kojoj upravljačka varijabla minimizira funkciju troška J kako bi se utjecaj poremećaja zadržao ispod dozvoljene granice dok istovremeno varijabla poremećaja maksimizira istu funkciju troška.

Optimalne strategije oba igrača mogu se dobiti rješavanjem Hamilton-Jacobi-Isaacs jednadžbom gdje je vrijednosnu funkciju pretpostavljamo kao $V(x) = x^T \boldsymbol{P} x$ i $\boldsymbol{P}$ pozitivno semidefinitno rješenje Riccatijeve jednadžbe.

$$min_u max_d \left\{ \frac{\partial V}{\partial x} (\boldsymbol{A}x + \boldsymbol{B_1}u + \boldsymbol{B_2}d) + x^T \boldsymbol{Q} x + u^T u - \gamma^2\, d^T d \right\} = 0 \tag{2.38.}$$

$$\frac{\partial V}{\partial x} = 2x^T \boldsymbol{P} \tag{2.39.}$$

Uvrstimo jednadžbu (2.39.) u HJI jednadžbu (2.38.) te dobijemo:

$$min_u max_d \{2x^T \boldsymbol{P}(\boldsymbol{A}x + \boldsymbol{B_1}u + \boldsymbol{B_2}d) + x^T \boldsymbol{Q} x + u^T u - \gamma^2\, d^T d\} = 0 \tag{2.40.}$$

Iz toga slijedi optimalni zakon za upravljanja koji ćemo dobiti maksimizacijom po u:

$$\frac{\partial}{\partial u} \{2x^T \boldsymbol{P}(\boldsymbol{A}x + \boldsymbol{B_1}u + \boldsymbol{B_2}d) + x^T \boldsymbol{Q} x + u^T u - \gamma^2\, d^T d\} = 0 \tag{2.41.}$$

$$\frac{\partial}{\partial u} \{2x^T \boldsymbol{P} \boldsymbol{B_1} u + u^T u\} = 0 \tag{2.42.}$$

$$2x^T \boldsymbol{P} \boldsymbol{B_1} + 2u = 0 \tag{2.43.}$$

$$u^* = -\boldsymbol{B_1}^T \boldsymbol{P} x \tag{2.44.}$$

Također, minimizacijom HJI jednadžbe po d, dobit ćemo optimalni zakon za poremećaj:

$$\frac{\partial}{\partial d}\{2x^T\boldsymbol{P}(\boldsymbol{A}x+\boldsymbol{B_1}u+\boldsymbol{B_2}d)+x^T\boldsymbol{Q}x+u^Tu-\gamma^2\,d^Td\}=0 \tag{2.45.}$$

$$\frac{\partial}{\partial d}\{2x^T\boldsymbol{P}\boldsymbol{B_2}d-\gamma^2\,d^T\gamma^2\}=0 \tag{2.46.}$$

$$2x^T\boldsymbol{P}\boldsymbol{B_2}-\boldsymbol{2}\gamma^2d=0 \tag{2.47.}$$

$$d^*=\frac{1}{\gamma^2}\boldsymbol{B_2}^T\boldsymbol{P}x \tag{2.48.}$$

Uvrštavanjem u^* (2.44.) i d^* (2.48.) u HIJ jednadžbu (2.40.), dobit će se Riccatijeva jednadžba (2.50.) gdje je $\boldsymbol{P}$ njezino rješenje.

$$\boldsymbol{P}\boldsymbol{A}+\boldsymbol{A}^T\boldsymbol{P}+\boldsymbol{Q}-\boldsymbol{P}\boldsymbol{B_1}\boldsymbol{B_1}^T\boldsymbol{P}+\frac{1}{\gamma^2}\boldsymbol{P}\boldsymbol{B_2}\boldsymbol{B_2}^T\boldsymbol{P}=\boldsymbol{0} \tag{2.49.}$$

Ova metoda optimalnog upravljanja sustavom koristit će se u ovom radu.

3. NUMERIČKI ALGORITAM ZA H_∞ UPRAVLJANJE LINEARNIM DINAMIČKIM SUSTAVIMA

Numerički algoritam za H_∞ upravljanje linearnim dinamičkim sustavima se temelji na algoritmu povratnog prostiranja kroz vrijeme. Primijenjen je tako da se gradijent propagira unazad kroz svaki korak te se tako optimiziraju težinski parametri. Kontinuirani dinamički sustav koji se optimizira, diskretiziran je Eulerovom metodom što nam omogućuje analizu njezinog ponašanja kroz vrijeme. Rješavanje postavljenog minimaks optimizacijskog problema temelji se na algoritmu konjugiranog gradijenta poboljšanog SuperSAB metodom koja prilagođava korake učenja tijekom optimizacije te time omogućuje bržu konvergenciju.

3.1. Formulacija problema

Zadani linearni vremenski invarijantni dinamički sustav bit će oblika:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B_1}u(t) + \boldsymbol{B_2}d(t), \qquad x(t_0) = x_0, \tag{3.1.}$$

$$z(t) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{3.2.}$$

gdje su:

$x(t) \in \mathbb{R}^n$ - vektor stanja,

$u(t) \in \mathbb{R}^{n_u}$ - vektor upravljanja,

$d(t) \in \mathbb{R}^{n_d}$ - vektor vanjskog poremećaja,

$z(t) \in \mathbb{R}^{n_z}$ - kaznena varijabla

$x_0 \in \mathbb{R}^n$ - vektor početnih stanja sustava

Pretpostavlja se da su sva stanja sustava (3.1.) i (3.2.) upravljiva, s obzirom na :

$$\boldsymbol{E} = [\boldsymbol{A}^{n-1}\boldsymbol{B_1} \;\; \boldsymbol{A}^{n-2}\boldsymbol{B_1} \ldots \boldsymbol{A}^2\boldsymbol{B_1} \;\; \boldsymbol{AB_1} \;\; \boldsymbol{B_1}] \tag{3.3.}$$

$$rank(\boldsymbol{E}) = n \tag{3.4.}$$

Cilj ovog rada je odrediti zakon upravljanja:

$$u = \boldsymbol{K}x(t) \tag{3.5.}$$

te „najgori mogući“ poremećaj sustava:

$$d = \boldsymbol{W}x(t) \tag{3.6.}$$

tako da je $\gamma > 0$ minimiziran.

Matrica $\boldsymbol{K}$ opisuje upravljače ulaze s obzirom za vektor stanja $x(t)$, a matrica $\boldsymbol{W}$ ulazne poremećaje sustava. Obje matrice će se izračunavati konjugiranim gradijentnim algoritmom te ćemo definirati u zajendičkoj matrici:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{W} \end{bmatrix} \tag{3.7.}$$

Za zadani sustav (3.1) i (3.2), definira se funkcija troška diferencijalnom igrom nulte sume za pronalazak zakona upravljanja i varijable poremećaja takvih da je:

$$J = min_u \, max_d \int_{t_0}^{t_f} \frac{1}{2}(x^T \boldsymbol{Q} x + \|u\|^2 - \gamma^2 \|d\|^2)\, dt \tag{3.8.}$$

3.2. Vremenska diskretizacija Eulerovom metodom

Prijmjena BPTT algoritma zahtjeva diskretizaciju dinamičkog sustava (3.1) koja će se u ovom radu riješiti Eulerovom metodom vremenske diskretizacije.

Ako uzmemo vremenski interval $[0, t_f\rangle$ nad kojim ćemo raditi diskretizaciju, podijelit ćemo ga na $N-1$ pod intervala jednakih duljina, vremenski raspon će se sastojati od $t_i = i\tau$ točaka za $i = 0, 1, 2 \ldots N-1$ gdje duljina vremenskog koraka jednaka $\tau = t_f/N$.

Eulerova aproksimacija dinamičke jednadžbe sustava (3.1.) je:

$$x(i+1) = x(i) + \tau \boldsymbol{f}(i), \qquad x(t_0) = x_0, \tag{3.9.}$$

gdje je:

$$\boldsymbol{f}(i) = \boldsymbol{A}\boldsymbol{x}(i) + \boldsymbol{B_1}\boldsymbol{u}(i) + \boldsymbol{B_2}\boldsymbol{d}(i) \tag{3.10.}$$

na intervalu $0, 1, 2 \ldots N-1$.

```
function x = my_euler(g, Z, h, Nint)

% ODE solution using Euler method

global n x0

x(1:n, 1) = x0;

for j = 1:Nint
    f = ODE_sys(x, g, Z, j);
    x(:, j+1) = x(:, j) + h*f;
end
```

Slika 3.1. Matlab kod za Eulerovu diskretizaciju

```
function dx = ODE_sys(x, g, Z, i)

% System ODE: x' = f(x, u, d)

global n uizl

x2 = x(2, i);

g(:,i) = Z*x(:,i);

u = g(1, i);
d = g(2, i);

uizl(:,i) = g(:,i);

dx = zeros(n, 1);

dx(1) = x2 + d;
dx(2) = u;
```

Slika 3.2. Matlab kod za izračun funkcije f(i)

Na slici 3.1. je prikazan Matlab kod koji ažurira sustav po Eulerovoj diskretizaciji za trenutno stanje. Funkcija *my_euler* kreće sa trenutnim stanjem sustava *x0*, upravljačkom varijablom i poremećajem *g*, matricam *K* i *W* koje su zapisane u matrici *Z*, vremenskim korakom *h* te broju iteracija *Nint*. Unutar nje se poziva funkcija *ODE_sys* koja je prikazana na slici 3.2. te izračunava jednadžbu (3.10.) *dx* kako bi u funkciji *my_euler* mogli ažuirarti trenutno stanje sustava *x* (3.9.) Eulerovom diskretizacijom.

Diskretizirane jednadžbe upravljačke varijable (3.11.) i poremećaja (3.12.) izražene su kao:

$$u_j(i) = \sum_{s=1}^{n} k_{js} x_s(i) \qquad j = 1, 2, \dots, n_u, \tag{3.11.}$$

$$d_r(i) = \sum_{s=1}^{n} w_{rs} x_s(i) \qquad r = 1, 2, \dots, n_d \tag{3.12.}$$

Diskretni oblik funkcije troška je:

$$J = \tau \sum_{i=0}^{N-1} F(i) \tag{3.13.}$$

gdje je:

$$F(i) = \frac{1}{2}(x(i)^T \boldsymbol{Q} x(i) + \|u(i)\|^2 - \gamma^2 \|d(i)\|^2) \tag{3.14.}$$

3.3. Metoda konjugiranog gradijenta

Metodu konjugiranog gradijenta koristimo za pronalazak sedlene točke diferencijalne igre nulte sume.

Metoda konjugiranog gradijenta počinje od početne pretpostavke x_0 te nadalje generira x_k za $k \geq 1$ pomoću ponavljanja [3]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3.15.}$$

gdje je vektor smjera pretraživanja:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \qquad d_0 = -g_0 \tag{3.16.}$$

gdje su:

α_k - parametar veličine koraka

β_k - parametar ažuriranja

$g_{k+1} = \nabla \mathrm{f}(x_k)^T$ - gradijent funkcije

Metoda je vrlo praktična za programiranje s obzirom da za računanje iteracije treba samo rješenja prethodne, a ne i svih ostalih iteracija što znači da treba pamtiti samo vektor zadnje iteracije, a ne matricu svih iteracija. Također, algoritam koristi prilagodbu stope konvergencije što daje bolju izvedbu nego ako je konstantna, prije će doći do rješenja.

3.3.1. Parametar β konjugirane gradijentne metode

U ovom radu koristili smo četiri poznate metode za izračunavanje parametra ažuriranja β: Hestenes-Stiefel, Fletcher-Reeves, Polak-Ribiere i Dai-Yuan te još dvije koje su hibrid tih četiriju metoda.

Svaka od tih metoda koristi $||g_{k+1}||^2$ ili $g_{k+1}^T y_k$ u brojniku i $||g_k||^2$ ili $d_k^T y_k$ u nazivniku gdje je $||\cdot||^2$ euklidska norma, a $y_k = g_{k+1} - g_k$. Njihovim kombinacijama dobijemo četiri različite metode [3].
Različiti β određuju različite metode konjugiranih gradijenata. Radovi u ovom području pokazuju da pravilan odabir ovog parametra vodi do boljih numerički izvedbi [5].

Hestenes-Stiefel metoda glasi:

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k} \tag{3.17.}$$

Fletcher-Reeves metoda glasi:

$$\beta_k^{FR} = \frac{||g_{k+1}||^2}{||g_k||^2} \tag{3.18.}$$

Polak-Ribiere metoda glasi:

$$\beta_k^{PR} = \frac{g_{k+1}^T y_k}{||g_k||^2} \tag{3.19.}$$

Dai-Yuan metoda glasi:

$$\beta_k^{DY} = \frac{||g_{k+1}||^2}{d_k^T y_k} \tag{3.20.}$$

Moguće je kategorizirati iznad prikazane formule u dvije kategorije. PR i HS su uključeni u prvu kategoriju. Ove formule smatraju se među najučinkovitijim dostupnim metodama

konjugiranih gradijenata za pronalaženje rješenja za funkcije velikih razmjera. To mogu postići zbog brojnika $g_{k+1}^T y_k$ u koji je integrirana funkcija automatskog ponovnog pokretanja koja sprječava da zapnu tijekom procesa izračunavanja. U drugu kategoriju su uključeni DY i DY. Unatoč činjenici da ovi algoritmi imaju izvrsne konvergencijske karakteristike, njihova numerička izvedba često je suboptimalna kao rezultat fenomena zagušenja.

Neki od najboljih izvedbi algoritma konjugiranog gradijenta su hibridnim metodama, koje dinamički prilagođavaju formulu za β_k kako se iteracija razvijaju. To može uključivati i promjenu načina na koji se β_k izračunava tijekom iteracija. Na primjer, moguće je koristiti različite metode računanja β_k u ranijim ili kasnijim iteracijama koje mogu bolje optimizirati funkciju za neki njezin specifičan oblik. Također, može spriječiti zagušenje tj. ne napredovanje algoritma. To omogućava algoritmu konjugiranog gradijenta da se prilagodi različitim karakteristikama funkcije troška što će poboljšati konvergenciju algoritma i smanjiti ukupno vrijeme potrebno za rješavanje problema optimizacije [3].

U ovom radu se koriste dvije hibridne metode: Touati-Ahmed-Storey metoda i Dai-Yuhan hibridna metoda. Touati-Ahmed-Storey metoda koja je hibrid Polak-Ribiere i Fletcher-Reeves metoda glasi:

$$\beta_k = \begin{cases} \beta_k^{PR}, & 0 \leq \beta_k^{PR} \leq \beta_k^{FR} \\ \beta_k^{FR}, & inače \end{cases} \tag{3.21.}$$

Dai-Yuhan hibridna metoda koja je hibrid Dai-Yuan i Hestenes-Stiefel metoda glasi:

$$\beta_k = \max\{0\,, \min(\beta_k^{HS}, \beta_k^{DY}) \tag{3.22.}$$

3.3.2. SuperSAB metoda

Standardna metoda za izračunavanje μ je metoda najbržeg spusta što je računalno zahtjevan način jer može zahtijevati mnogo evaluacija funkcije troška u jednoj iteraciji gradijentnog algoritma. Također, ako funkcija nije odgovarajuće skalirana, može pokazati loše konvergencijske karakteristike. Za izbjegavanje tih problema koristi se SuperSAB metoda [4] koja zahtjeva samo informacije o smjerovima gradijenta u dvije uzastopne iteracije gradijentnog algoritma. Smjer gradijenta se određuje prema predznacima dviju uzastopnih iteracija gradijenta pa se stopa učenja prilagođava tako da ako gradijenti ima jednak predznak,

stopa učenja se povećava što ubrzava konvergenciju, a ako imaju različit predznak znači da je prešao lokalni minimum te se stopa učenja smanjuje. Jednadžbe koje ju opisuju su slijedeće:

$$x_i(k+1) = x_i(k) + \Delta x_i(k) \tag{3.23.}$$

$$\Delta x_i(k) = -\eta_i(k)\frac{\partial f}{\partial x_i(k)} + \alpha \Delta x_i(k-1) \tag{3.24.}$$

$$\eta_i(k) = \begin{cases} d^+ \cdot \eta_1(k-1), & ako\ \dfrac{\partial f}{\partial x_i(k)} \cdot \dfrac{\partial f}{\partial x_i(k-1)} > 0 \\ d^- \cdot \eta_1(k-1), & ako\ \dfrac{\partial f}{\partial x_i(k)} \cdot \dfrac{\partial f}{\partial x_i(k-1)} < 0 \\ \eta_1(k-1), & ako\ \dfrac{\partial f}{\partial x_i(k)} \cdot \dfrac{\partial f}{\partial x_i(k-1)} = 0 \end{cases} \tag{3.25.}$$

Ovaj algoritam lokalne adaptivnosti pruža brzu konvergenciju jer se optimizirana varijabla x_i ažurira samo po lokalnim vrijednostima gradijenta funkcija $\frac{\partial f}{\partial x_i(k)}$ i $\frac{\partial f}{\partial x_i(k-1)}$. Međutim, tu se javlja i problem diskontinuiteta upravljačke varijable. Kako bi to izbjegli izmijenit će se postupak tako da omogućava globalnu adaptaciju jedne stope učenja u svakoj iteraciji k kako bi došli do optimalnog kontinuiranog upravljačkog rješenja.

3.3.3. Računanje gradijenta za zadani problem

Minimaks optimizacijski pristup se temelji na usponu ili spust konjugiranog gradijenta za matrice $\boldsymbol{K} = \left[k_{js}\right]$ i $\boldsymbol{W} = [w_{rs}]$.

Kako bi jednostavnije primijenili algoritam konjugiranog gradijenta, matrice $\boldsymbol{K}$ i $\boldsymbol{W}$ će se zajednički izračunavati u matrici $\boldsymbol{Z}$ tako da vrijedi:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{W} \end{bmatrix} \tag{3.26.}$$

Algoritam konjugiranog gradijenta će se izvesti u slijedećem obliku:

$$\boldsymbol{Z} = \boldsymbol{Z}^{(j-1)} + \eta^{(j-1)}\boldsymbol{S}^{(j-1)} \tag{3.27.}$$

$$\boldsymbol{S}^{(j)} = \lambda \nabla J\left(\boldsymbol{Z}^{(j-1)}\right) + \beta^{(j-1)}\boldsymbol{S}^{(j-1)} \tag{3.28.}$$

gdje je:

$\boldsymbol{S}$ - smjer pretraživanja

Za prvi redak matrice $\boldsymbol{Z}$, čiji su elementi zapravo elementi matrice $\boldsymbol{K}$, $\lambda = -1$, dok je za drugi redak čiji su elementi elementi matrice $\boldsymbol{W}$, $\lambda = 1$.

Kao što je već spomenuto, kako bi zaobišli problem diskontinuiteta upravljačke varijable koji ova metoda donosi, modificiran je SuperSAB algoritam za računanje stope učenja:

$$\eta_i(j) = \begin{cases} d^+ \cdot \eta_1(j-1), & ako \;\; Tr\left\{\left[\frac{\partial J}{\partial vec(Z^{(j)})}\right]^T \frac{\partial J}{\partial vec(Z^{(j-1)})}\right\} \geq 0 \\ d_1^- \cdot \eta_1(j-1), & ako \;\; Tr\left\{\left[\frac{\partial J}{\partial vec(Z^{(j)})}\right]^T \frac{\partial J}{\partial vec(Z^{(j-1)})}\right\} < 0 \\ d_2^- \eta_1(j-1), & ako \;\; J\left(Z^{(j)}\right) \geq J\left(Z^{(j-1)}\right) \end{cases} \tag{3.29.}$$

gdje su $0 < d_2^- < d_1^- < 1 < d^+$ dilatacijski parametri.

```
1  function eta = eta_update(eta_old, Jz, Jz_prev, J, J_prev)
2
3
4
5  if (trace(Jz.'*Jz_prev) >= 0)
6      eta = 1.2 * eta_old;
7      'Ispunjen prvi uvjet'
8  elseif (J >= J_prev)
9      eta = 0.4 * eta_old;
10     'Ispunjen treci uvjeet'
11 elseif (trace(Jz.'*Jz_prev) < 0)
12     eta = 0.95 * eta_old;
13     'Ispunjen drugi uvjet'
14 end
```

Slika 3.3. Matlab kod SuperSAB algoritma

Na slici 3.3. je Matlab kod implementacije modificiranog SuperSAB algoritma za zadani problem. Uvjeti određivanja veličine stope učenja se određuju s obzirom na trag umnoška prošlog i trenutnog koraka gradijenta funkcije troška i prema trenutnom iznosu funkcije troška i one iz prethodnog koraka.

Za izračun stope učenja pomoću SuperSAB metode, potreban nam je gradijent funkcije troška po elementima matrice $\boldsymbol{Z}$.

Gradijent funkcije troška (3.8) po elementima matrice $\boldsymbol{Z}$ u j-toj iteraciji gradijentnog algoritma i i-tom intervalu uzorkovanja dan je jednadžbom (3.30.):

$$\frac{\partial J}{\partial z_{pq}} = \tau \sum_{i=0}^{N-1} \left(\sum_{r=1}^{n} \frac{\partial F(i)}{\partial x_r(i)} \frac{\partial x_r(i)}{\partial z_{pq}} + \sum_{r=1}^{n_u+n_d} \frac{\partial F(i)}{\partial g_r(i)} \frac{\partial g_r(i)}{\partial z_{pq}} \right) \tag{3.30.}$$

gdje je $p = 1, 2, \ldots n_u$ i $q = 1, 2, \ldots . n_d$.

Kao što smo matrice $\boldsymbol{K}$ i $\mathbf{W}$ stavili u zajedničku matricu $\boldsymbol{Z}$ za jednostavniju implementaciju gradijentnog algoritma, analogno ćemo i upravljaču varijablu u i poremećaj d staviti u zajedničku matricu $\mathbf{g_r}(\mathrm{i})$:

$$[\boldsymbol{g_r}(i)] = \boldsymbol{g}(i) = \begin{bmatrix} u(i) \\ d(i) \end{bmatrix} \tag{3.31.}$$

Izraz (3.31.) se može zapisati u matričnom obliku:

$$\frac{\partial J}{\partial z_{pq}} = \tau \sum_{i=0}^{N-1} \left(\frac{\partial F(i)}{\partial x(i)} \cdot \frac{\partial x(i)}{\partial z_{pq}} + \frac{\partial F(i)}{\partial g(i)} \cdot \frac{\partial g(i)}{\partial z_{pq}} \right) \tag{3.32.}$$

Pomoću jednadžbi (3.13.) i (3.14.) možemo odrediti elemente:

$$\frac{\partial F(i)}{\partial x(i)} = \frac{\partial}{\partial x(i)} \left[\frac{1}{2} (x(i)^T \boldsymbol{Q} x(i) + \|u(i)\|^2 - \gamma^2 \|d(i)\|^2) \right] \tag{3.33.}$$

$$\frac{\partial F(i)}{\partial x(i)} = \boldsymbol{Q} x(i) \tag{3.34.}$$

$$\frac{\partial F(i)}{\partial g(i)} = \frac{\partial}{\partial g(i)} \left[\frac{1}{2} (x(i)^T \boldsymbol{Q} x(i) + \|u(i)\|^2 - \gamma^2 \|d(i)\|^2) \right] \tag{3.35.}$$

$$\frac{\partial F(i)}{\partial g(i)} = \begin{bmatrix} I_{n_u} & 0 \\ 0 & -\gamma^2 I_{n_d} \end{bmatrix} g(i) \tag{3.36.}$$

Parcijalna derivacija $\frac{\partial x(\mathrm{i})}{\partial z_{pq}}$ se može izračunati prema slijedećoj jednadžbi:

$$\frac{\partial x(i)}{\partial z_{pq}} = \sum_{j=1}^{n} \frac{\partial f_r(i-1)}{\partial x_j(i-1)} \frac{\partial x_j(i-1)}{\partial z_{pq}} + \sum_{j=1}^{n_u+n_d} \frac{\partial f_r(i-1)}{\partial g_m(i-1)} \frac{\partial g_m(i-1)}{\partial z_{pq}} \tag{3.37.}$$

gdje je $r = 1, 2, \ldots n$, $p = 1, 2, \ldots n_u + n_d$ i $q = 1, 2, \ldots n.$

Također, možemo jednadžbu (3.37.) zapisati u matričnom obliku:

$$\frac{\partial x(i)}{\partial z_{pq}} = \frac{\partial f(i-1)}{\partial x(i-1)} \cdot \frac{\partial x(i-1)}{\partial z_{pq}} + \frac{\partial f(i-1)}{\partial g(i-1)} \cdot \frac{\partial g(i-1)}{\partial z_{pq}} \tag{3.38.}$$

Uvrstimo li jednadžbe (3.9.) i (3.10.) u $\frac{\partial f(i-1)}{\partial x(i-1)}$ i $\frac{\partial f(i-1)}{\partial g(i-1)}$, dobit ćemo izraze:

$$\frac{\partial f(i-1)}{\partial x(i-1)} = \frac{\partial}{\partial x(i-1)}[x(i-1) + \tau(\boldsymbol{A}x(i-1) + \boldsymbol{B}_1 u(i-1) + \boldsymbol{B}_2 d(i-1))] \tag{3.39.}$$

$$\frac{\partial f(i-1)}{\partial x(i-1)} = I_n + \tau\boldsymbol{A} \tag{3.40.}$$

$$\frac{\partial f(i-1)}{\partial g(i-1)} = \frac{\partial}{\partial g(i-1)}[x(i-1) + \tau(\boldsymbol{A}x(i-1) + \boldsymbol{B}_1 u(i-1) + \boldsymbol{B}_2 d(i-1))] \tag{3.41.}$$

$$\frac{\partial f(i-1)}{\partial g(i-1)} = \tau[\boldsymbol{B}_1 \quad \boldsymbol{B}_2] \tag{3.42.}$$

Nadalje, parcijalna derivacija $\frac{\partial g_m(\mathrm{i}-1)}{\partial z_{pq}}$ se može izračunati kao slijedeća jednadžba:

$$\frac{\partial g_m(i)}{\partial z_{pq}} = \sum_{j=1}^{n}\left(\frac{\partial z_{mj}}{\partial z_{pq}} x_q(i) + z_{mj}\frac{\partial x_j(i)}{\partial z_{pq}}\right) \tag{3.43.}$$

gdje je $m = 1, 2, \ldots n_u + n_d$, $r = 1, 2, \ldots n,$, $p = 1, 2, \ldots n_u + n_d$ *i* $q = 1, 2, \ldots n$, a $\frac{\partial z_{\mathrm{mj}}}{\partial z_{\mathrm{pq}}} = \delta_{mp}$ što je Kroneckerov delta.

Tada se jednadžba (3.43.) može zapisati u slijedećem matričnom obliku:

$$\frac{\partial g(i)}{\partial z_{pq}} = \begin{bmatrix} \delta_{1p} & \cdots & 0 & 0 \\ \vdots & \delta_{2p} & \vdots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_{mp} \end{bmatrix} \begin{bmatrix} x_q(i) \\ x_q(i) \\ \vdots \end{bmatrix} + \boldsymbol{Z}\frac{\partial x(i)}{\partial z_{pq}} \tag{3.44.}$$

Slijedeći korak je odrediti početne uvjete za gore navede rekurzivne izraze.

Početni uvjeti vektora stanja sustava su neovisni o matrici $\boldsymbol{Z}$, iz toga slijedi:

$$\frac{\partial x_r(0)}{\partial z_{pq}} = 0 \tag{3.45.}$$

Drugi početni uvjet glasi:

$$\frac{\partial g_m(0)}{\partial z_{pq}} = \delta_{mp} x_q(0) \tag{3.46.}$$

Na slici 3.4. je prikazana implementacija gradijenta u Matlabu:

```
function J_z = gradient_Jz(x, g, Z, h)

% Calculation of cost function gradient w.r.t. matrix weights

global nu n N A B Q R1 R2 gama

J_z = zeros(nu, n);

for p = 1:nu
    for q = 1:n


        Xz = zeros(n, 1); % dx/dz
        Gz = zeros(nu, 1); % dg/dz

        zz = 0;

        for i = 2:N

            fx = eye(n) + h*A; % df/dx
            fu = h*B; % df/du
            Xz = fx*Xz + fu*Gz;

            Gz = diag([delta(1, p), delta(2, p)])*[x(q,i);x(q,i)] + Z*Xz;

            Fx = Q*x(:,i); % dF/dx
            Fu = blkdiag(R1, -gama^2*R2)*g(:,i); % dF/du
            ss = dot(Fx, Xz) + dot(Fu, Gz);
            zz = zz + ss;
        end

        J_z(p, q) = zz*h;

    end
end
```

Slika 3.4. Matlab kod za računanje gradijenta

3.4. Sumiranje algoritma

Prema slijedećim koracima je sumiran algoritam objašnjen u ovom poglavlju:

Tablica 3.1. Suma algoritma po koracima

1. korak	Zadavanje parametara za vremensku diskretizaciju sustava: vrijeme izvedbe t_f, broj vremenskih podintervala N, duljina vremenskog koraka τ. Postavljanje inicijalnih vrijednosti matrice pojačanja $\boldsymbol{Z}_0$. Odabir metode za izračun parametra ažuriranja β te njegove maksimalne vrijednosti β_{max}. Postavljanje početnog vektora stanja x_0.
2. korak	Eulerova aproksimacija jednadžbe stanja i povećanje iteracija.

3. korak	Računanje gradijenta i funkcije troška.
4. korak	Računanje novih pojačanja koristeći konjugiranu metodu, odabranu metodu za računanje parametra ažuriranja β i SuperSAB algoritam.
5. korak	Provjera uvjeta za zaustavljanje i vraćanje na 2. korak.

4. SIMULACIJSKI PRIMJER

4.1. Dinamički model linearnog sustava

Potrebno je izabrati model sustava za provedbu usporedbe rezultata dobivenog algoritmom i onoga koji se dobije rješavanjem pripadajuće Hamilton-Jacobi-Isaacsove (Riccatijeve) jednadžbe. Primjer sustava je linearni sustav jednostavne mase koji je opisan sljedećom jednadžbom [2]:

$$m\ddot{x}(t) = u(t) + d(t) \tag{4.1.}$$

gdje je:

- $x(t)$ - pozicija mase m u nekom trenutku
- $\dot{x}(t)$ - brzina mase m
- $\ddot{x}(t)$ - ubrzanje mase m
- $u(t)$ - upravljački ulaz koji utječe na masu m
- $d(t)$ - vanjski utjecaj na masu m

Diferencijalnu jednadžbu drugog reda je potrebno prebaciti u prostor stanja. To ćemo napraviti tako da izlučimo ubrzanje mase $\ddot{x}(t)$ s lijeve strane:

$$\ddot{x} = \frac{1}{m}u + \frac{1}{m}d \tag{4.2.}$$

Definiramo varijable stanja:

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} \end{aligned} \tag{4.3.}$$

Uvrštavanjem jednadžbi (4.3.) u (4.2.) iz diferencijalnih jednadžbi drugog reda, dobijemo dvije diferencijalne jednadžbe prvog reda koje glase:

$$\begin{aligned} \dot{x}_1 &= \dot{x} = x_2 \\ \dot{x}_2 &= \ddot{x} = \frac{1}{m}u + \frac{1}{m}d \end{aligned} \tag{4.4.}$$

Model sustava prema (4.4.) može se kraće zapisati kao:

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{m}u + \frac{1}{m}d \end{bmatrix} \tag{4.5.}$$

Sustav želimo zapisati u obliku:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} d \\ y &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \end{aligned} \tag{4.6.}$$

Uz iznos mase $m = 1kg$, matrice sustava su:

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \boldsymbol{B}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \boldsymbol{B}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \boldsymbol{C}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{D}_{12} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \tag{4.7.}$$

Za prethodno opisani sustav potrebno je odrediti upravljačku varijablu u i poremećajnu varijablu d tako da je:

$$J = min_u \, max_d \int_{t_0}^{t_f} \frac{1}{2} (x^T \boldsymbol{Q} x + \|u\|^2 - \gamma^2 \|d\|^2) \, dt \tag{4.8.}$$

pri čemu su $\boldsymbol{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ i $\gamma = 2$.

4.2. Rezultati simulacije

Za rješavanje danog problema koristi će se algoritam iz 3. poglavlja, a njegova će izvedba biti u programskom jeziku Matlab.

Ovdje su dane numeričke vrijednosti početnih parametara potrebnih za pokretanje. Parametri za vremensku diskretizaciju sustava su vrijeme izvedbe $t_f = 10\,s$ i broj vremenskih podintervala $N = 5000$ na koji će se podijeliti vrijeme izvedbe. Na temelju toga se izračuvana duljina vremenskog koraka $\tau = 0.002s$.

Parametri relevantni za konjugirani gradijentni algoritam su početna stopa konvergencije gradijentnog algoritma $\eta_0 = 0.0001$, početna stopa konjugacije $\beta_0 = 0.0008$ i parametar dopuštene greške $\varepsilon = 0.001$ na kojem se temelji kriterij zaustavljanja algoritma.

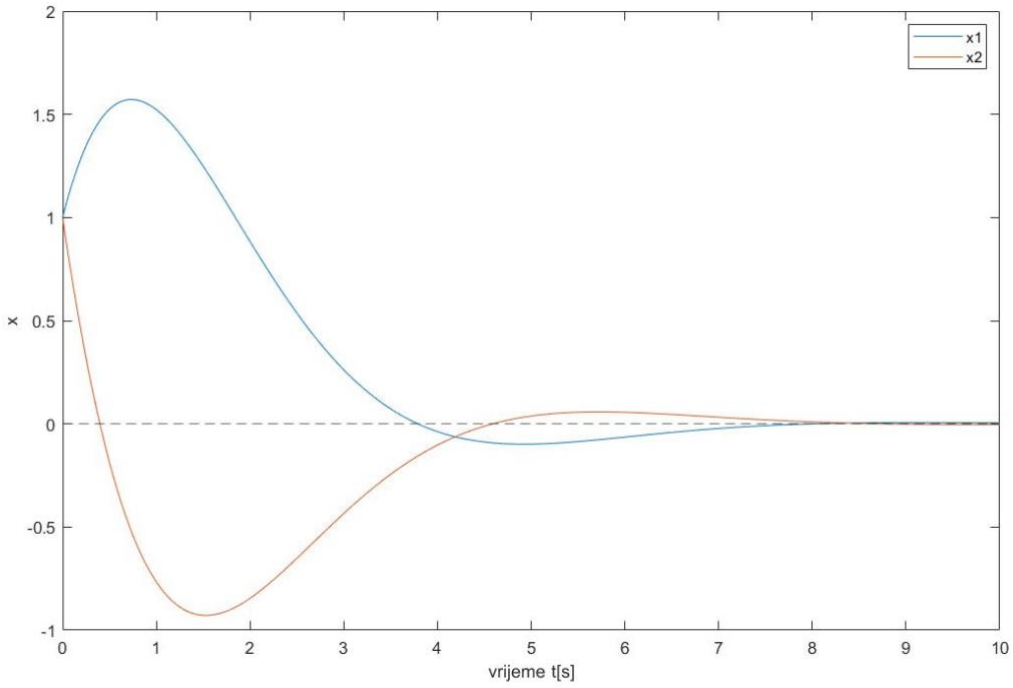
Također su dane i numeričke vrijednosti dilatacijskih parametara SuperSAB metode: pozitivni dilatacijski parametar za povećanje veličine koraka $d^+ = 1.2$, negativni dilatacijski parametar za smanjenje veličine koraka $d_1^- = 0.95$ i negativni dilatacijski parametar za daljnje prilagođavanje veličine koraka $d_2^- = 0.4$. Korištena je Dai-Yuan metoda za izračun stope konjugacije β.

Početne vrijednosti matrica $\boldsymbol{K}$ i $\boldsymbol{W}$ postavljane su na vrijednosti $\boldsymbol{K} = [-1 \; -1]$ i $\boldsymbol{W} = [1 \quad 1]$. U algoritmu su matrice pojačanja $\boldsymbol{K}$ i $\boldsymbol{W}$ zapisane u zajedničku matricu $\boldsymbol{Z}$ pa je njezina početna vrijednost:

$$\boldsymbol{Z}_0 = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \tag{4.9.}$$

Algoritam je konvergirao na 106. iteraciji nakon što je zadovoljio uvjet $\left\| \nabla J\left(z^{(j)}\right) \right\|_\infty \leq \varepsilon$ za $\varepsilon = 0.001$. Nakon konvergencije algoritma nova vrijednosti matrice $\boldsymbol{Z}$, ujedno i matrica pojačanja $\boldsymbol{K}$ i $\boldsymbol{W}$, iznosi:

$$\boldsymbol{Z} = \begin{bmatrix} -1.3320 & -1.7639 \\ 0.4414 & 0.3338 \end{bmatrix} \tag{4.10.}$$

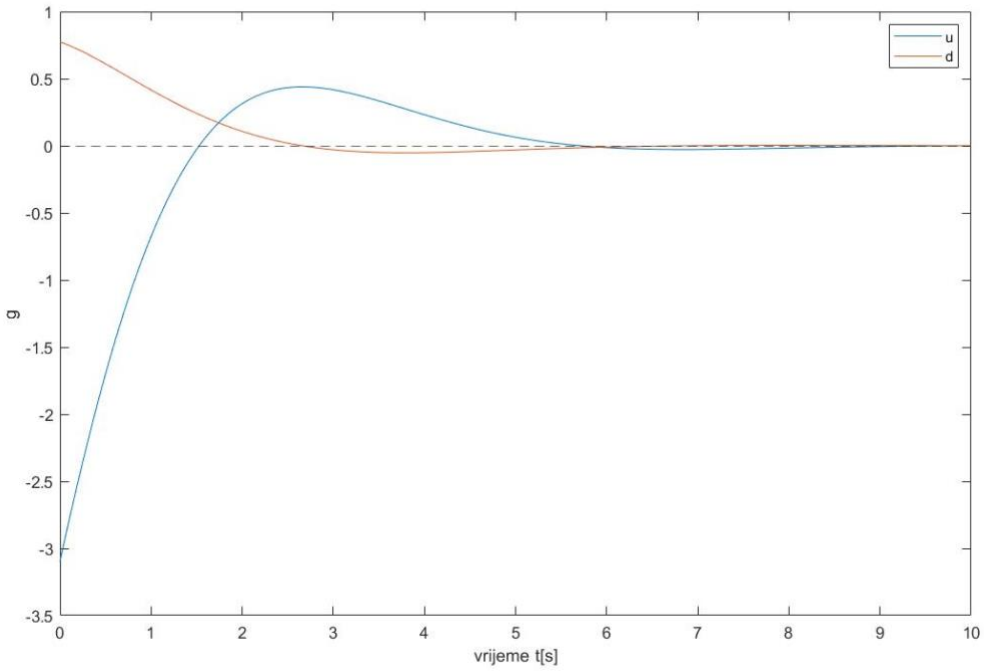


Slika 4.1. Varijable stanja u ovisnosti o vremenu

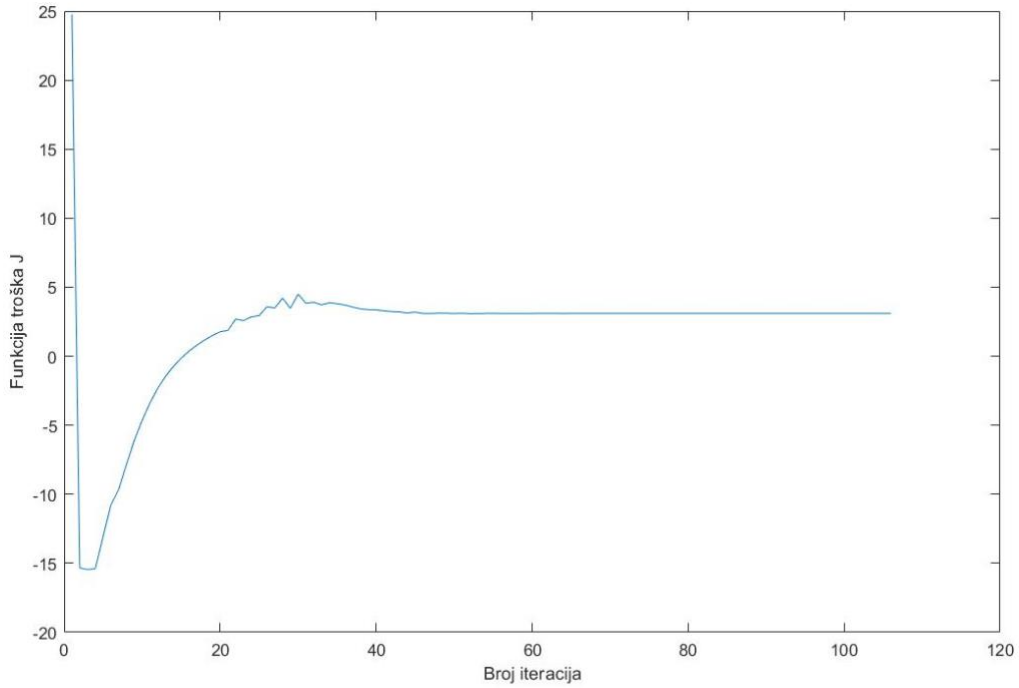
Na slici 4.1. prikazana je ovisnost varijable stanja o vremenu. Graf nam pokazuje kakav je odgovor sustava na utjecaj upravljačke varijable i poremećaja tokom vremena ako vektor početnih uvjeta varijable stanja iznosi $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Na slici 4.2. prikazana je ovisnost upravljačke varijable i varijable poremećaja o vremenu. Graf nam pokazuje naglo djelovanje poremećaja te pokušaj upravljačke varijable da se suprotstavi i neutralizira to djelovanje.

Na slici 4.3. je prikazana funkcija troška u ovisnosti o broju iteracija algoritma. Nakon pojave naglog pada zbog pogreške u početnim uvjetima te blagog osciliranja, funkcija troška se stabilizira što indicira da ju je algoritam uspješno minimizirao te je postigla svoje (sub)optimalno rješenje. Funkcija troška je konvergirala u $J = 3.1025$ i nakon toga ostaje konstantna.



Slika 4.2. Upravljačka varijabla *u* i varijabla poremećaja *d* u ovisnosti o vremenu



Slika 4.3. Funkcija troška u ovisnosti s brojem iteracija

Osim Dai-Yuan metode za izračun stope konjugacije uzete su u obzir i ostale metode iz poglavlja 3.3.1. Međutim, s obzirom na jednostavnost obrađenog modela sustava, rezultati su gotovo jednaki. Rezultati različitih optimizacijskih metoda dani su u tablici 4.1. Funkcija troška je pomoću svih metoda konzistentno konvergirala na vrijednosti 3,1025. Neznatna razlika u metodi Polak-Ribiere je zanemariva. Iako su metode, osim hibridne metode Dai-Yuan - Hestenes-Stiefel, uspješne u minimizaciji funkcije troška, značajno se razlikuju u vremenu izvedbe što je važan faktor za odabir određene metode. Dai-Yuan metoda je najučinkovitija, pruža (sub)optimalnu vrijednost funkcije troška u najkraćem vremenu. Konvergirala je u 106. iteraciji i ostala stabilna. Osim hibridne metode Dai-Yuan - Hestenes-Stiefel, najlošije rezultate je dala metoda Hestenes-Stiefel sa vremenom izvedbe dvostruko dužim od Dai-Yuan metode, a konvergirala je u 233. iteraciji algoritma. Hibridna metoda Fletcher-Reeves - Polak-Ribiere pokazuje obećavajuće smanjenje vremena izvedbe u usporedbi sa samom metodom Fletcher-Reeves. Konvergirala je u 133. iteraciji što najbolji rezultat nakon Dai-Yuan metode. Mogla bi biti poboljšana njezina izvedba uz dodatna podešavanja parametara i provjere stabilnosti.

Tablica 4.1. Usporedba različitih optimizacijskih metoda

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$t_f = 10$ $N = 5000$ $\tau = 0.002$ $\varepsilon = 0.001$ $\beta_{max} = 0.7$	3.1025	38.6430
Fletcher-Reeves		3.1025	58.1514
Polak-Ribiere		3.1027	49.3014
Hestenes-Stiefel		3.1025	92.3342
Fletcher-Reeves - Polak-Ribiere		3.1025	47.7793
Dai-Yuan - Hestenes-Stiefel		divergencija	N/A

4.3. Prilagođavanje ulaznih parametara

Za uspješnost konjugirano gradijentnog algoritma, veliki utjecaj ima postavljanje njegovih parametara koji mogu bitno utjecati na stabilnost algoritma, ali i brzinu. Nekoliko je takvih parametara, a to su: duljina vremenskog koraka τ koji ovisi o vremenu izvedbe t_f i broju vremenskih pod intervala N, zatim stopa konvergencije η_0, kriterij zaustavljanja ε, maksimalna stopa konjugacije β_{max} te dilatacijski parametri SuperSAB metode d^+, d_1^- i d_2^-.

Prilikom računanja skalarnih vrijednosti β parametra, došlo je do velikog variranja njezinog iznosa što može biti rezultat nekoliko razloga. Ako funkcija troška ima područja s velikim nagibima, to će rezultirati drastičnim promjenama smjera gradijenta između iteracija što će dovesti i do nagle promjene β. Također, veliki utjecaj može imati i odabir početnih vrijednosti. Ako su početne vrijednosti daleko od optimalnog rješenja, to će također dovesti velike varijacije β-i. Te nagle promjene β mogu destabilizirati algoritam jer umjesto da postepeno konvergira prema optimumu, ono krene skakati između različitih smjerova što može i usporiti proces konvergencije. Zato će se u algoritmu ograničiti vrijednosti parametra β kako bi osigurali stabilnost algoritma i postepenu konvergenciju prema rješenju što je prikazano na slici 4.4.

```
5 -   if (metoda == 1)
6         % Dai-Yuan
7 -       brojnik1_DY = norm(Jz(1,:));
8 -       nazivnik1_DY = S(1,:)*(Jz(1,:)-Jz_prev(1,:)).';
9
10 -      brojnik2_DY = norm(Jz(2,:));
11 -      nazivnik2_DY = S(2,:)*(Jz(2,:)-Jz_prev(2,:)).';
12
13 -      if (abs(brojnik1_DY / nazivnik1_DY) < beta_max)
14 -          beta1 = brojnik1_DY / nazivnik1_DY;
15 -      else
16 -          beta1 = beta_max;
17 -      end
18
19 -      if (abs(brojnik2_DY / nazivnik2_DY) < beta_max)
20 -          beta2 = brojnik2_DY / nazivnik2_DY;
21 -      else
22 -          beta2 = beta_max;
23 -      end
```

Slika 4.4. Dai-Yuan metoda izračuna parametra β

Stabilnost sustava i vrijeme izvedbe jako ovisi o izboru parametra β_{max} što je prikazano u tablici 4.2. uz ostale parametre koji ostaju konstantni: $t_f = 10$, $N = 5000$, $\tau = 0.002$ i $\varepsilon = 0.001$. Najbolja kombinacija minimalne vrijednosti troška i najkraćeg vremena izvedbe je za vrijednost parametra $\beta_{max} = 0.7$ gdje je vrijednost funkcije troška konvergirala na 3.1025 u 39,4606s. Jako bliski rezultati postižu se i za vrijednosti $\beta_{max} = 0.6$ gdje je metoda konvergirala za 40,2541s. Vrijednosti $\beta_{max} > 0.8$ znatno povećavaju vrijeme izvedbe bez dodatnog poboljšanja funkcije troška, dok za vrijednosti $\beta_{max} < 0.05$ algoritam divergira.

Tablica 4.2. Utjecaj parametra β_{max}

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$\beta_{max} = 0.005$	divergencija	N/A
Dai-Yuan	$\beta_{max} = 0.001$	divergencija	N/A
Dai-Yuan	$\beta_{max} = 0.05$	3.1025	42.0105
Dai-Yuan	$\beta_{max} = 0.4$	3.1025	48.4141
Dai-Yuan	$\beta_{max} = 0.5$	3.1025	41,1855
Dai-Yuan	$\beta_{max} = 0.55$	3.1025	51,2256
Dai-Yuan	$\beta_{max} = 0.6$	3.1024	40,2541
Dai-Yuan	$\beta_{max} = 0.65$	3.1025	57,1141
Dai-Yuan	$\beta_{max} = 0.7$	3.1025	39,4606
Dai-Yuan	$\beta_{max} = 0.8$	3.1025	71,9649
Dai-Yuan	$\beta_{max} = 0.9$	3.1025	762,2038

U konjugiranoj gradijentnoj metodi stopa konvergencije η_0 i kriterij zaustavljanja ε imaju značajan utjecaj na izvedbu algoritma. Utječu na brzinu konvergencije, točnost konačnog rješenja i ukupno vrijeme izvedbe. Međutim, treba odrediti i njihovu optimalnu vrijednost jer mogu donijeti i dodatne probleme u algoritam. Visoka stopa konvergencije može početno brzo smanjiti funkciju troška, međutim u toj brzini joj se može dogoditi da preskoči optimalne točke te počne nepotrebno oscilirati oko rješenja. U suprotnom, niska stopa konvergencije osigurava stabilnije i preciznije korake prema (sub)optimalnom rješenju, ali manji koraci mogu povećati ukupno vrijeme izvođenja. Analogno tome, strogi kriterij zaustavljanja ε će povećati točnost dobivenog rješenja, ali i znatno produžiti vrijeme izvedbe algoritma. Dok slabiji kriterij zaustavljanja ε riskira preciznost i točnost rješenja. Te parametre treba prilagoditi prema zahtjevima i očekivanjima greške koje želimo dobiti algoritmom.

Tablica 4.3. Utjecaj kriterija zaustavljanja ε

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$\varepsilon = 0.0001$	3.1025	144.6262
Dai-Yuan	$\varepsilon = 0.0005$	3.1025	112.1161
Dai-Yuan	$\varepsilon = 0.001$	3.1025	38.0450
Dai-Yuan	$\varepsilon = 0.005$	3.1026	34.6714
Dai-Yuan	$\varepsilon = 0.01$	3.1027	30.2038
Dai-Yuan	$\varepsilon = 0.05$	3.1033	22.2430
Dai-Yuan	$\varepsilon = 0.1$	3.1033	21.9820

U tablici 4.3. je prikazana ovisnost funkcije troška i vremena izvedbe o kriteriju zaustavljanja uz ostale parametre koji ostaju konstantni: $t_f = 10$, $N = 5000$, $\eta_0 = 0{,}0001$, $\tau = 0.002$ i $\beta_{max} = 0.7$. Najbolja kombinacija minimalne vrijednosti funkcije troška i najkraćeg vremena izvedbe je za vrijednost $\varepsilon = 0.001$. Iako se vrijeme izvedbe smanjuje slabijim kriterijem

zaustavljanja smanjuje, njezina točnost pada, dok strožim kriterijem zaustavljanja ne dolazi do poboljšanja rezultata funkcije troška, a vrijeme izvedbe se drastično povećava.

Još će se provjeriti utjecaj početne stope konvergencije što je prikazano u tablici 4.4. uz ostale parametre koji ostaju konstantni: $t_f = 10$, $N = 5000$, $\tau = 0.002$, $\varepsilon = 0.001$ i $\beta_{max} = 0.7$. Promjene kriterija zaustavljanja, iako djeluju na izvedbu algoritma, uglavnom su sposobne doći do (sub)optimalnog rješenja. Međutim, i male promjene početne stope konvergencije η_0 drastično djeluju vrijeme izvedbe algoritma kao što je pretpostavljeno. Stoga su granice uspješnosti algoritma s obzirom na $\eta_0 = [0{,}0001\ \ 0{,}0005]$ unutar kojih imamo minimalnu vrijednost funkcije troška. Za vrijednosti $\eta_0 > 0{,}0005$, algoritam divergira tj. ne može pronaći optimalno rješenje, a za vrijednosti početne stope konvergencije $\eta_0 < 0{,}0001$, zbog malog koraka mu se drastično povećava vrijeme izvedbe tj. vrijeme potrebno za pronalazak (sub)optimalnog rješenja.

Tablica 4.4. Utjecaj početne stope konvergencije η_0

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$\eta_0 = 0.00005$	3.1025	254.3802
Dai-Yuan	$\eta_0 = 0.0001$	3.1025	40.7885
Dai-Yuan	$\eta_0 = 0.0005$	3.1025	67.0519
Dai-Yuan	$\eta_0 = 0.001$	divergencija	N/A

Uz odabir optimalnih parametara $\eta_0 = 0{,}0001$, $\varepsilon = 0.001$ i $\beta_{max} = 0.7$, razmatrat će se i utjecaj duljine vremenskog koraka τ Eulerove diskretizacije što je prikazano u tablici 4.5. Uspješnost izvedbe algoritma uvelike ovisi o duljini vremenskog koraka te je interval uspješnosti s obzirom na $\tau = [0{,}00143\ \ 0{,}002]$. Izvan tih granice, algoritam divergira i ne može pronaći optimalno rješenje problema. Utjecaj duljine vremenskog koraka na minimazciju funkcije troška i vrijeme izvedbe djeluje suprotno. Povećanjem koraka, vrijeme izvedbe se smanjuje, ali funkcija troška ima veću grešku, dok se smanjenjem koraka, događa obratno. Stoga odabir optimalne duljine vremenskog koraka ovisi o zahtjevima. Ako je cilj minimizirati

vrijeme izvedbe najbolje rješenje daje $\tau = 0{,}0025$, a ako je cilj minimizirati funkciju troška, najbolju izvedbu daje $\tau = 0{,}00143$. S obzirom da težimo optimalnom djelovanju s obzirom na oba zahtjeva, kao zadovoljavajuće rješenje uzimamo $\tau = 0{,}002$.

Tablica 4.5. Utjecaj duljine vremenskog koraka τ

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$\tau = 0{,}0001$	divergencija	N/A
Dai-Yuan	$\tau = 0{,}001$	divergencija	N/A
Dai-Yuan	$\tau = 0{,}00143$	3,1010	59,5948
Dai-Yuan	$\tau = 0{,}002$	3,1025	39,9651
Dai-Yuan	$\tau = 0{,}0025$	3,1038	32,6397
Dai-Yuan	$\tau = 0.0033$	divergencija	N/A
Dai-Yuan	$\tau = 0.005$	divergencija	N/A

Neke okvirne vrijednosti dilatacijskih parametara SuperSAB metode su $0.85 \leq d^{+} \leq 0.95$ i $1.05 \leq d^{-} \leq 1.2$, a njihova preporučena veza je $d^{+} \approx 1/d_1^{-}$. Njihov utjecaj algoritam je prikazan u tablici 4.6. Dilatacijski parametri utječu uglavnom na vrijeme izvedbe. Imaju mali, gotovo zanemariv utjecaj na funkciju troška kao što je vidljivo za izbor parametara $d^{+} = 1.2, d_1^{-} = 0.85, d_2^{-} = 0.4$ s obzirom da su to već parametri na rubovima preporučenih intervala. Stoga je optimalno rješenje za parametre $d^{+} = 1.2, d_1^{-} = 0.95, d_2^{-} = 0.4$ jer rezultira povoljnom funkcijom troška uz minimalno vrijeme izvedbe.

Tablica 4.6. Utjecaj dilatacijskih parametara

Metoda	Konfiguracija	Funkcija troška J	Vrijeme izvedbe, [s]
Dai-Yuan	$d^+ = 1.2$ $d_1^- = 0.95$ $d_2^- = 0.4$	3.1025	39,9651
Dai-Yuan	$d^+ = 1.05$ $d_1^- = 0.95$ $d_2^- = 0.4$	3.1025	63.6481
Dai-Yuan	$d^+ = 1.2$ $d_1^- = 0.85$ $d_2^- = 0.4$	3.1024	52.7978
Dai-Yuan	$d^+ = 1.2$ $d_1^- = 0.95$ $d_2^- = 0.2$	3,1025	50.4269
Dai-Yuan	$d^+ = 1.05$ $d_1^- = 0.85$ $d_2^- = 0.4$	3.1034	48.7828
Dai-Yuan	$d^+ = 1.05$ $d_1^- = 0.85$ $d_2^- = 0.2$	3.1034	49.9107

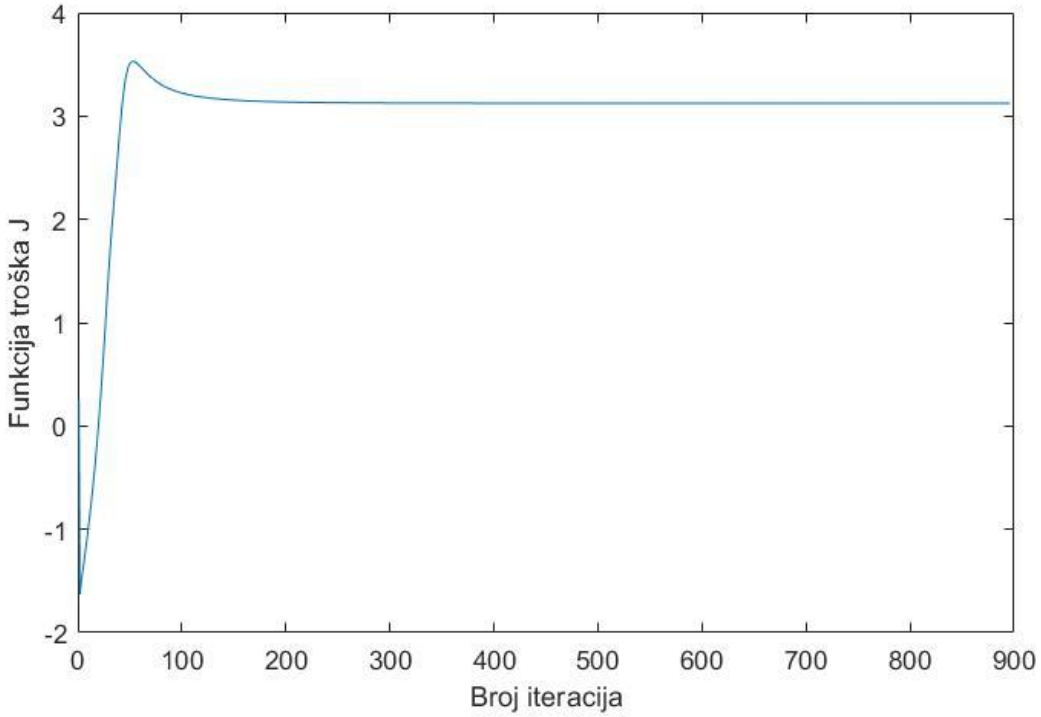
Odabir početnih vrijednosti matrice pojačanja je kritičan korak u dizajnu algoritma optimizacije. Prvi pokušaj biranja nul matrice kao početne matrice pojačanja nije bio dobar izbor iz nekoliko razloga. Kada su svi elementi matrice jednaki nuli, početni parametri će biti jednaki za sve parametre tj. algoritam će napraviti isti korak za sve parametre. S obzirom da rješavamo problem optimizacije linearnog sustava, rane iteracije nekad neće pružiti korisne informacije za smjerove pretrage što će onda usporiti konvergenciju. Daljnjim eksperimentiranjem, pojačanja su postavljena u blizini konačnog rješenja $\boldsymbol{Z}_0 = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$ što ubrzava konvergenciju, smanjuje rizik da algoritam zaglavi u nekom od lokalnih minimuma i

povećava šanse da pronađe globalni minimum. Zbog te blizine konačnom rješenju i gradijenti su manji što vodi stabilnijim i manjim koracima tijekom optimizacije. Daljnjim približavanjem konačnom rješenju dolazi se do veće greške funkcije troška i dužem vremenu izvedbe. S toga se ostalo pri izboru matrice pojačanja (4.9.).

4.4. Usporedba

4.4.1. Usporedba s rješenjem standardnog gradijentnog algoritma

Ako za isti problem primijenimo standardni gradijentni algoritam čiji se koraci određuju direktno na osnovi gradijenta funkcije troška, a ne za određivanje smjera pretrage, za iste početne parametre: kriterij zaustavljanja $\varepsilon = 0.001$, početna stopa konvergencije $\eta_0 = 0{,}0001$, vrijeme izvedbe $t_f = 10\ s$, broj vremenskih pod intervala $N = 5000$ i duljina vremenskog koraka $\tau = 0.002s$ te jednaku početnu matricu pojačanja (4.9.), ne postiže konvergenciju funkcije troška, ali niti ne divergira, nego oscilira oko konačnog rješenja bez da ga postigne. Tek nakon prilagodbe početnih parametara tako da je matrica pojačanja $Z_0 = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$, kriterij zaustavljanja $\varepsilon = 0.01$, broj vremenskih pod intervala $N = 1000$, duljina vremenskog koraka $\tau = 0.01s$ i početna stopa konvergencije $\eta_0 = 0{,}001$, standardni gradijentni algoritam je postigao konvergenciju nakon 896. iteracija u vrijednost funkcije troška $J = 3.1238$ što je dosta lošiji rezultat u usporedbi sa konjugiranom gradijentnom metodom. Konvergencija funkcije troška u ovisnosti o broju iteracija je prikazana na slici 8. To nam pokazuje kako SuperSAB metoda omogućuje puno bržu konvergenciju bez velikog utjecaja na stabilnost. Matrica pojačanja postiže vrijednosti $\boldsymbol{Z} = \begin{bmatrix} -1.3270 & -1.7650 \\ 0.4419 & 0.3363 \end{bmatrix}$,



Slika 4.5. Funkcija troška standardna gradijentne metode u ovisnosti o broju iteracija

4.4.2. Usporedba s rješenjem Riccatijeve jednadžbe

Prema [2] za sustav (4.7.) rješenje Riccatijeve jednadžbe (2.49.) daje matricu:

$$\boldsymbol{P} = \begin{bmatrix} 1.7638 & 1.3333 \\ 1.3333 & 1.7638 \end{bmatrix} \tag{4.11.}$$

Matrice pojačanja upravljačke varijable i poremećaja iz rješenja Riccatijeve jednadžbe glase:

$$\boldsymbol{K} = -\boldsymbol{B_1}^T \boldsymbol{P} \tag{4.12.}$$

$$\boldsymbol{W} = \frac{1}{\gamma^2} \boldsymbol{B_2}^T \boldsymbol{P} \tag{4.13.}$$

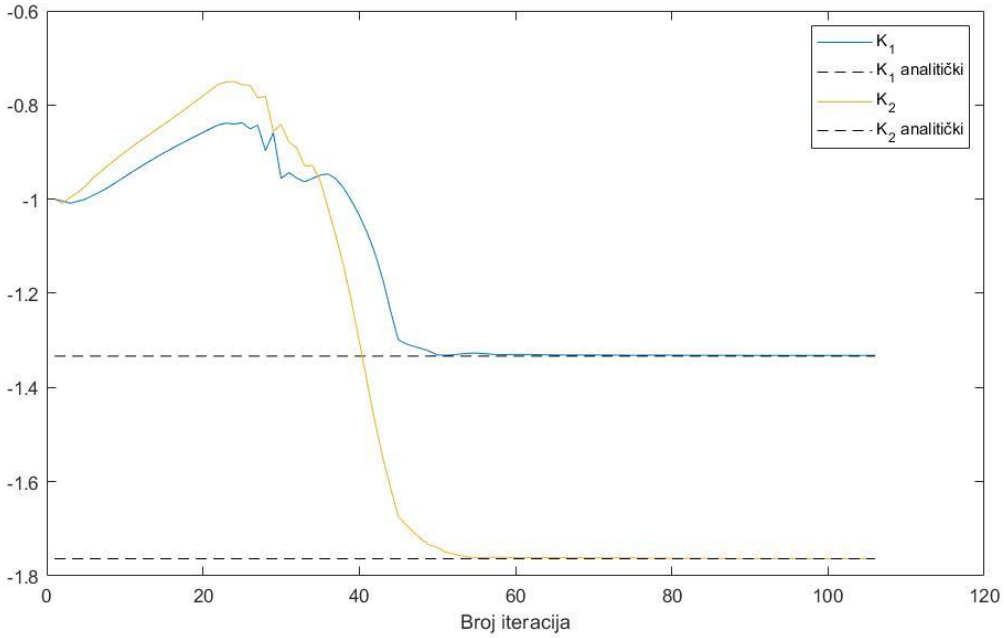
Ako se matrica $\boldsymbol{P}$ (4.11.) uvrsti u (4.12.) i (4.13.) dobit će se njihove vrijednosti:

$$\boldsymbol{K} = [-1.3333 \quad -1.7638] \tag{4.14.}$$

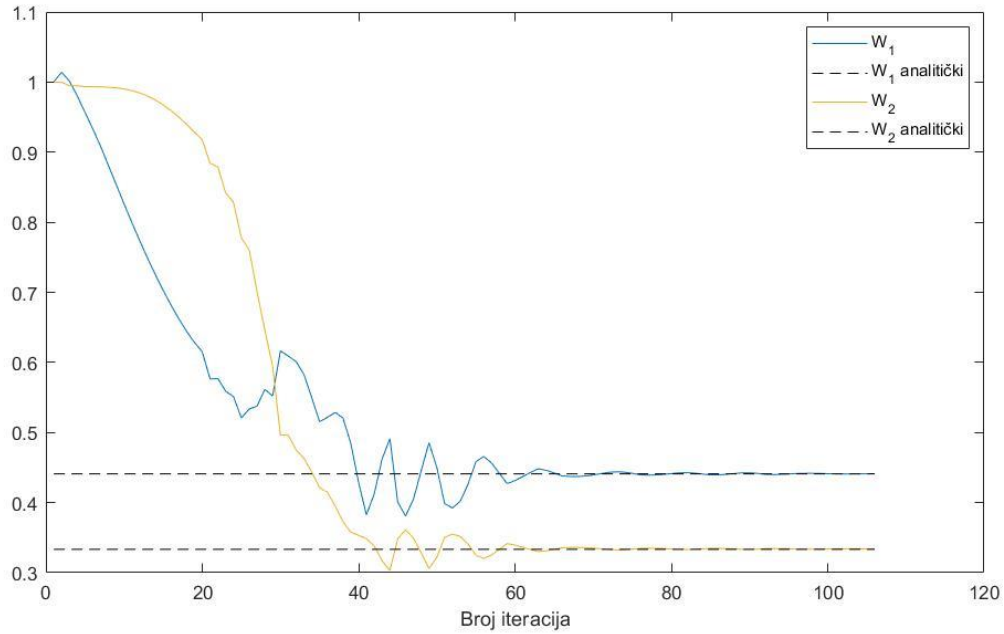
$$\boldsymbol{W} = [0.4410 \quad 0.3333] \tag{4.15.}$$

Rezultati dobiveni rješavanjem Riccatijeve jednadžbe (4.14.) i (4.15.), vrlo su bliski rezultatima dobivenim konjugiranom gradijentnom metodom $\boldsymbol{Z} = \begin{bmatrix} -1.3270 & -1.7650 \\ 0.4419 & 0.3363 \end{bmatrix}$ prikazani na

slici 4.6. i 4.7. Prema tome, naš konjugirani gradijentni algoritam dobro optimizira zadani sustav. Međutim, to je bilo očekivano s obzirom na jednostavnost sustava koji se obrađuje.



Slika 4.6. Ovisnost matrice pojačanja K o broju iteracija



Slika 4.7. Ovisnost matrice pojačanja W o broju iteracija

5. ZAKLJUČAK

U prvom dijelu rada postavljena je teorijska podloga za linearne vremenski invarijantne sustave te važnost upravljivosti za taj sustav. Nakon toga je objašnjena veza $\mathcal{L}_p$ pojačanja i H_∞ norme te kako je ta veza temelj H_∞ upravljanja kojim se određuje zakon upravljanja minimizacijom funkcije troška.

U drugom dijelu rada razjašnjen je numerički algoritam za H_∞ upravljanje linearnim dinamičkim sustavom. Korištena je struktura algoritma propagiranja unatrag kroz vrijeme koja uz diskretizaciju dinamičkog sustava Eulerovom metodom koristi konjugiranu gradijentnu metodu poboljšanu SuperSAB algoritmom te nekoliko metoda za izračun parametra ažuriranja poput Hestenes-Stiefel, Fletcher-Reeves, Polak-Ribiere i Dai-Yuan te njihove hibride. Algoritam je izveden u Matlab-u, a njegova analiza je provedena na dinamičkom sustavu jednostavne mase. Dobiveni rezultati su uspoređeni sa izvedbom standardnog gradijentnog algoritma te s rješenjem pripadajuće Hamilton-Jacobi-Isaacsova (odnosno Riccatijeve) jednadžbe. Nakon podešavanja ulaznih parametara, postignuta je konvergencija funkcije troška te su dobiveni zadovoljavajući rezultati. Najveći utjecaj na efikasnost i sposobnost algoritma da daje rješenje je imao izbor početne matrice pojačanja $\boldsymbol{Z}$. Loše postavljanje početnih vrijednosti te matrice može u potpunosti onesposobiti algoritam za davanje rješenja i uz prilagođavanje ostalih ulaznih parametara. U radu se fokusiralo na analizu i poboljšanje početnih parametara za Dai-Yuan metodu računanja parametra ažuriranja. Obećavajuće rezultate je dala i hibridna metoda Fletcher-Reeves - Polak-Ribiere te bi se daljnjim prilagođavanjem njezinih početnih parametara mogla postići bolja efikasnost.

LITERATURA

[1] Kasać, J., Opća teorija sustava, skripta, Fakultet strojarstva i brodogradnje, Zagreb, 2007.

[2] Sinha, A., Linear Systems: Optimal and Robust Control, CRC Press, Boca Raton FL, 2007.

[3] Hager, W.W., Zhang, H., A survey of nonlinear conjugate gradient methods, Pacific Journal of Optimization, Volume 2, Number 1, pp. 35-58, 2006.

[4] Tollenaere T., SuperSAB: fast adaptive backpropagation with good scaling properties, Neural Networks Volume 3, Number 5, pp. 561–573, 1990.

[5] Wais, O. M. T., Abbo, K. K., Saleh, I.A., Modified Hestenes - Stiefel Conjugate Gradient (MHS-CG) Method for Solving Unconstrained Optimization, Journal of Multidisciplinary Modeling and Optimization, Volume 4, Number 2, pp. 32-42, 2021.

PRILOZI

I. Matlab kod

MATLAB KOD

main_results

```
close all
clear all
clc

global A B Q R1 R2 nu n...
    x0 t0 T tau N t_vec...
    gama...
    qmin epsilon...
    d1 d2 k1 k2...
    cost_func_out kiter...
    metoda beta_max
%------------------------------------------------------------------------



% Example: Sinha Linear Systems: Optimal and Robust Control pp. 317 Example
5.10.1
% System matrices
A = [0 1;0 0];
B1 = [0;1]; n_u = 1;
B2 = [1;0]; n_d = 1;

B = [B1 B2];

% Cost function
gama = 2;
Q = [1 0;0 0];
R1 = eye(n_u);
R2 = eye(n_d);
%------------------------------------------------------------------------


% Dimensions of system vectors
n = 2; % state vector
nu = 2; % input vector ie. nu = n_u + n_d (n_u is number of control inputs,
n_d is number of disturbance input)
%------------------------------------------------------------------------



% Time discretization
t0 = 0; % initial time
T = 10; % final time
N = 1000; % number of time intervals
tau = (T-t0)/N; % time step length
t_vec = t0:tau:T; % time vector
%------------------------------------------------------------------------


% Initial conditions
x0 = [1 1]'; % initial state vector
Z0 = -1*ones(nu,n); % initial control+disturbance matrix
%------------------------------------------------------------------------

% Gradient algorithm
qmin = 1e-4; % convergence rate for GA
```

```
epsilon = 1e-3; % stopping criteria

% Odabir metode za računanje beta:
%   1 = Dai-Yuan
%   2 = Fletcher-Reeves
%   3 = Polak-Ribiere
%   4 = Hestenes-Stiefel
%   5 = hibrid - Fletcher-Reeves - Polak-Ribiere
%   6 = hibrid - Dai-Yuan - Hestenes-Stiefel
metoda = 5;
beta_max = 0.1;
%--------------------------------------------------------------------------


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%% MIN-MAX OPTIMAL CONTROL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic
[Xopt, Gopt] = control(Z0, 1); % optimal control vector calculation
time_exe = toc

% analytic solution
p12 = gama^2/(gama^2-1);
p22 = gama*sqrt(2*gama^2-1)/(gama^2-1);
p11 = p12*p22/(1+p12/gama^2);

P = [p11 p12;p12 p22];

Kc = -[0;1]'*P
[k1(end) k2(end)] % obtained

Kd = [1;0]'*P/gama^2
[d1(end) d2(end)] % obtained


plotter
```

control

```
function [xizl, g] = control(Z_init, n_show)

% Optimal control vector calculation using conjugate gradient algorithm

global nu tau N qmin cost_func_out n ...
    d1 d2 k1 k2 epsilon kiter


Z = Z_init;
g = zeros(nu, N+1);

kiter = 0;
while 1

    kiter = kiter + 1;

    k1(kiter) = Z(1,1); k2(kiter) = Z(1,2);
    d1(kiter) = Z(2,1); d2(kiter) = Z(2,2);

    x = my_euler(g, Z, tau, N+1);
```

```
    Jz = gradient_Jz(x, g, Z, tau);
    fnc2_u = cost_function(x, g);
    J = fnc2_u;

    lambda = 1;

    % eta = qmin;

    if (kiter == 1)
        S(1,:) = -lambda*Jz(1,:);
        S(2,:) = lambda*Jz(2,:);
    elseif (kiter > 1)
        S(1,:) = -lambda*Jz(1, :).' + beta1*S(1,:).';
        S(2,:) = lambda*Jz(2, :).' + beta2*S(2,:).';
    end

    if (kiter == 1)
        eta = qmin;
    elseif (kiter > 1)
        eta = eta_update(eta, Jz, Jz_prev, J ,J_prev);
    end

    if (kiter == 1)
        beta1 = 0.0008;
        beta2 = 0.0008;
    elseif (kiter > 1)
        [beta1, beta2] = beta_update(Jz_prev, Jz, S);
    end

    Z(1,:) = Z(1,:) + eta*S(1,:); % matrix K

    Z(2,:) = Z(2,:) + eta*S(2,:); % matrix W

    Z_konacno = Z

    Jz_prev = Jz;
    J_prev = J;
    xizl = x;
    g = Z*x;

    if (mod(kiter, n_show) == 0)
        number_of_iteration = kiter
        cost_func = fnc2_u
        cost_func_out(kiter) = fnc2_u;
    end

    if norm(Jz, inf) < epsilon
        break
    end
end
```

my_euler

```
function x = my_euler(g, Z, h, Nint)

% ODE solution using Euler method

global n x0
```

```
x(1:n, 1) = x0;

for j = 1:Nint
    f = ODE_sys(x, g, Z, j);
    x(:, j+1) = x(:, j) + h*f;
end
```

ODE_sys

```
function dx = ODE_sys(x, g, Z, i)

% System ODE: x' = f(x, u, d)

global n uizl

x2 = x(2, i);

g(:,i) = Z*x(:,i);

u = g(1, i);
d = g(2, i);

uizl(:,i) = g(:,i);

dx = zeros(n, 1);

dx(1) = x2 + d;
dx(2) = u;
```

gradient_Jz

```
function J_z = gradient_Jz(x, g, Z, h)

% Calculation of cost function gradient w.r.t. matrix weights

global nu n N A B Q R1 R2 gama

J_z = zeros(nu, n);

for p = 1:nu
    for q = 1:n


        Xz = zeros(n, 1); % dx/dz
        Gz = zeros(nu, 1); % dg/dz

        zz = 0;

        for i = 2:N

            fx = eye(n) + h*A; % df/dx
            fu = h*B; % df/du
            Xz = fx*Xz + fu*Gz;

            Gz = diag([delta(1, p), delta(2, p)])*[x(q,i);x(q,i)] + Z*Xz;

            Fx = Q*x(:,i); % dF/dx
            Fu = blkdiag(R1, -gama^2*R2)*g(:,i); % dF/du
```

```
            ss = dot(Fx, Xz) + dot(Fu, Gz);
            zz = zz + ss;
        end

        J_z(p, q) = zz*h;

    end
end
```

delta

```
function delta_out = delta(i, j)

% Kronecker delta

if (i == j)
    delta_out = 1;
else
    delta_out = 0;
end
```

cost_function

```
function J = cost_function(x, g)

% Cost function calculation

global nu N tau gama

s1 = 0;
s2 = 0;

for i=1:N+1

    s1 = s1 + x(1,i)*x(1,i);

    for j=1:nu
        if(j==1)
            pen = 1;
        end
        if (j==2)
            pen=-gama^2;
        end
        s2 = s2 + pen*g(j,i)*g(j,i);
    end
end

J = 0.5*tau*(s1+s2);
```

beta_update

```
function [beta1, beta2] = beta_update(Jz_prev, Jz, S)

global metoda beta_max

if (metoda == 1)
    % Dai-Yuan
```

```
    brojnik1_DY = norm(Jz(1,:));
    nazivnik1_DY = S(1,:)*(Jz(1,:)-Jz_prev(1,:)).';

    brojnik2_DY = norm(Jz(2,:));
    nazivnik2_DY = S(2,:)*(Jz(2,:)-Jz_prev(2,:)).';

    if (abs(brojnik1_DY / nazivnik1_DY) < beta_max)
        beta1 = brojnik1_DY / nazivnik1_DY;
    else
        beta1 = beta_max;
    end

    if (abs(brojnik2_DY / nazivnik2_DY) < beta_max)
        beta2 = brojnik2_DY / nazivnik2_DY;
    else
        beta2 = beta_max;
    end

elseif (metoda == 2)
    % Fletcher-Reeves
    brojnik_FR = norm(Jz);
    nazivnik_FR = norm(Jz_prev);

    if (abs(brojnik_FR / nazivnik_FR) < beta_max)
        beta1 = brojnik_FR / nazivnik_FR;
        beta2 = brojnik_FR / nazivnik_FR;
    else
        beta1 = beta_max;
        beta2 = beta_max;
    end

elseif (metoda == 3)
    % Polak-Ribiere
    brojnik1_PR = Jz(1,:)*(Jz(1,:)-Jz_prev(1,:)).';
    nazivnik1_PR = norm(Jz_prev(1,:));

    brojnik2_PR = Jz(2,:)*(Jz(2,:)-Jz_prev(2,:)).';
    nazivnik2_PR = norm(Jz_prev(2,:));

    if (abs(brojnik1_PR / nazivnik1_PR) < beta_max)
        beta1 = brojnik1_PR / nazivnik1_PR;
    else
        beta1 = beta_max;
    end

    if (abs(brojnik2_PR / nazivnik2_PR) < beta_max)
        beta2 = brojnik2_PR / nazivnik2_PR;
    else
        beta2 = beta_max;
    end

elseif (metoda == 4)
    % Hestenes-Stiefel
    brojnik1_HS = Jz(1,:)*(Jz(1,:)-Jz_prev(1,:)).';
    nazivnik1_HS = S(1,:)*(Jz(1,:)-Jz_prev(1,:)).';

    brojnik2_HS = Jz(2,:)*(Jz(2,:)-Jz_prev(2,:)).';
    nazivnik2_HS = S(2,:)*(Jz(2,:)-Jz_prev(2,:)).';

    if (abs(brojnik1_HS / nazivnik1_HS) < beta_max)
        beta1 = brojnik1_HS / nazivnik1_HS;
```

```
    else
        beta1 = beta_max;
    end

    if (abs(brojnik2_HS / nazivnik2_HS) < beta_max)
        beta2 = brojnik2_HS / nazivnik2_HS;
    else
        beta2 = beta_max;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% HIBRID %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif (metoda == 5)

    % Fletcher-Reeves
    brojnik_FR = norm(Jz);
    nazivnik_FR = norm(Jz_prev);

    if (abs(brojnik_FR / nazivnik_FR) < beta_max)
        beta1_FR = brojnik_FR / nazivnik_FR;
        beta2_FR = brojnik_FR / nazivnik_FR;
    else
        beta1_FR = beta_max;
        beta2_FR = beta_max;
    end

    % Polak-Ribiere
    brojnik1_PR = Jz(1,:)*(Jz(1,:)-Jz_prev(1,:)).';
    nazivnik1_PR = norm(Jz_prev(1,:));

    brojnik2_PR = Jz(2,:)*(Jz(2,:)-Jz_prev(2,:)).';
    nazivnik2_PR = norm(Jz_prev(2,:));

    if (abs(brojnik1_PR / nazivnik1_PR) < beta_max)
        beta1_PR = brojnik1_PR / nazivnik1_PR;
    else
        beta1_PR = beta_max;
    end

    if (abs(brojnik2_PR / nazivnik2_PR) < beta_max)
        beta2_PR = brojnik2_PR / nazivnik2_PR;
    else
        beta2_PR = beta_max;
    end

    if (0 <= beta1_PR <= beta1_FR)
        beta1 = beta1_PR;
    else
        beta1 = beta1_FR;
    end

    if (0 <= beta2_PR <= beta2_FR)
        beta2 = beta2_PR;
    else
        beta2 = beta2_FR;
    end
```

```
elseif (metoda == 6)

    % Dai-Yuan
    brojnik1_DY = norm(Jz(1,:));
    nazivnik1_DY = S(1,:)*(Jz(1,:)-Jz_prev(1,:)).';

    brojnik2_DY = norm(Jz(2,:));
    nazivnik2_DY = S(2,:)*(Jz(2,:)-Jz_prev(2,:)).';

    if (abs(brojnik1_DY / nazivnik1_DY) < beta_max)
        beta1_DY = brojnik1_DY / nazivnik1_DY;
    else
        beta1_DY = beta_max;
    end

    if (abs(brojnik2_DY / nazivnik2_DY) < beta_max)
        beta2_DY = brojnik2_DY / nazivnik2_DY;
    else
        beta2_DY = beta_max;
    end

    % Hestenes-Stiefel
    brojnik1_HS = Jz(1,:)*(Jz(1,:)-Jz_prev(1,:)).';
    nazivnik1_HS = S(1,:)*(Jz(1,:)-Jz_prev(1,:)).';

    brojnik2_HS = Jz(2,:)*(Jz(2,:)-Jz_prev(2,:)).';
    nazivnik2_HS = S(2,:)*(Jz(2,:)-Jz_prev(2,:)).';

    if (abs(brojnik1_HS / nazivnik1_HS) < beta_max)
        beta1_HS = brojnik1_HS / nazivnik1_HS;
    else
        beta1_HS = beta_max;
    end

    if (abs(brojnik2_HS / nazivnik2_HS) < beta_max)
        beta2_HS = brojnik2_HS / nazivnik2_HS;
    else
        beta2_HS = beta_max;
    end

    beta1 = max(0, min(beta1_HS,beta1_DY));
    beta2 = max(0, min(beta2_HS,beta2_DY));
end
```

eta_update

```
function eta = eta_update(eta_old, Jz, Jz_prev, J, J_prev)


if (trace(Jz.'*Jz_prev) >= 0)
    eta = 1.2 * eta_old;
    'Ispunjen prvi uvjet'
elseif (J >= J_prev)
    eta = 0.4 * eta_old;
    'Ispunjen treci uvjeet'
elseif (trace(Jz.'*Jz_prev) < 0)
    eta = 0.95 * eta_old;
    'Ispunjen drugi uvjet'
end
```

```
end
```

plotter

```
% close all
% clear all
% clc

MM = 1:kiter;

figure(1)
plot(MM, cost_func_out)

figure(2)
plot(t_vec, Xopt(:,1:N+1))

figure(3)
plot(t_vec, Gopt(:,1:N+1))

figure(4)
subplot(121)
plot(MM, k1, MM, Kc(1)*ones(1,kiter),'--k', MM, k2, MM,
Kc(2)*ones(1,kiter),'--k')

subplot(122)
plot(MM, d1, MM, Kd(1)*ones(1,kiter),'--k', MM, d2, MM,
Kd(2)*ones(1,kiter),'--k')
```