

Usporedba numeričkih metoda za rješavanje problema optimalnog upravljanja u programskim alatima MATLAB i CasADi

Belokleić, Korina

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:235:262540>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom](#).

Download date / Datum preuzimanja: **2024-08-26**

Repository / Repozitorij:

[Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Korina Belokleić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

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

Student:

Korina Belokleić

Zagreb, 2024.

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

Velika zahvala mentoru izv. prof. dr. sc. Vladimiru Miliću na pristupačnosti i susretljivosti te pomoći prilikom izrade ovog rada.

Zahvaljujem kolegicama i kolegama za pomoć te zajednički rad i učenje tijekom cijelog studija.

Zahvaljujem obitelji, prijateljima i dečku koji su me uvijek poticali i pružili mi sve kako bi cijeli studij bio lakši i bezbolniji.

Korina Belokleić



SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomске ispite

Povjerenstvo za diplomске 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: **Korina Belokleić** JMBAG: 0035220475

Naslov rada na hrvatskom jeziku: **Usporedba numeričkih metoda za rješavanje problema optimalnog upravljanja u programskim alatima MATLAB i CasADi**

Naslov rada na engleskom jeziku: **Comparison of numerical methods for solving optimal control problems in MATLAB and CasADi software tools**

Opis zadatka:

Mnogi problemi iz teorije upravljanja mogu se formulirati kao problemi optimalnog upravljanja koji se efikasno rješavaju raznim metodama numeričke optimizacije. Razvijeni su mnogi programski alati za njihovo rješavanje. Tema ovog diplomskog rada je primjena i usporedba različitih metoda za numeričko rješavanje problema optimalnog upravljanja koje su implementirane u matematičkim programskim alatima MATLAB i CasADi.

U diplomskom radu je potrebno:

1. Matematički formulirati problem optimalnog upravljanja u općem slučaju nelinearnog sustava uz Bolza funkciju cilja te ograničenja tipa jednakosti i nejednakosti. Provesti transformaciju problema beskonačno-dimenzionalnog kontinuiranog problema optimalnog upravljanja u konačno-dimenzionalni problem nelinearnog programiranja koji se može riješiti poznatim numeričkim metodama optimizacije – princip „prvo diskretizirati, onda optimizirati“.
2. Opisati metode za numeričko rješavanje problema optimalnog upravljanja koje su implementirane u matematičkim programskim alatima MATLAB (Optimization Toolbox) i CasADi.
3. Odabrati barem dva primjera sustava za koje će se primjenom odgovarajućih funkcija iz MATLAB-a i CasADi-ja provesti numerička sinteza optimalnog zakona upravljanja, i to za: linearni sustav barem drugog reda s analitičkim rješenjem i nelinearni mehanički sustav kao npr. robot s barem dva stupnja slobode gibanja.
4. Na odabranim primjerima sustava provesti niz numeričkih eksperimenata, odnosno usporedbi točnosti, broja iteracija i vremena izvršavanja za: a) različite metode diskretizacije problema (Eulerova metoda, Runge-Kuttova metoda, Adamsova metoda); b) različite metode minimizacije funkcije cilja koje su implementirane u korištenim programskim alatima (npr. kvazi Newtonove metode, metoda unutarnje točke, sekvencijalno kvadratično programiranje i sl.); c) različite metode računanja derivacija funkcije cilja (metoda konačnih diferencija i automatsko diferenciranje). Sve popratiti grafičkim prikazima i tablicama s odgovarajućim komentarima i opisima.

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

Zadatak zadan:

18. siječnja 2024.

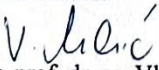
Datum predaje rada:

21. ožujka 2024.

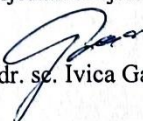
Predvideni datumi obrane:

25. – 29. ožujka 2024.

Zadatak zadao:


Izv. prof. dr. sc. Vladimir Milić

Predsjednik Povjerenstva:


Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA	III
POPIS TABLICA.....	VII
POPIS OZNAKA	VIII
SAŽETAK.....	IX
SUMMARY	X
1. UVOD.....	1
1.1. Cilj rada.....	2
1.2. Pregled literature i strukture.....	2
2. FORMULACIJA PROBLEMA I NUŽNI UVJETI OPTIMALNOSTI.....	3
2.1. Formulacija problema	3
2.2. Dinamički model nelinearnih sustava	4
2.3. Dinamička ograničenja	5
3. METODE ZA NUMERIČKO RJEŠAVANJE PROBLEMA OPTIMALNOG UPRAVLJANJA	7
3.1. Metode diskretizacije	7
3.1.1. Eulerova metoda	7
3.1.2. Runge – Kutta metoda 4. reda.....	8
3.1.3. Adams – Bashforth i Adams – Moulton metode	8
3.2. Metode za numeričku optimizaciju.....	9
3.2.1. Sekvencijalno kvadratično programiranje (sqp)	9
3.2.2. Metoda unutarnje točke (interior-point).....	10
4. MATEMATIČKI PROGRAMSKI ALATI ZA OPTIMIZACIJU	11
4.1. MATLAB.....	11
4.1.1. fminunc	12
4.1.2. fmincon	12
4.2. CasADi.....	13

5. PRIMJERI I USPOREDBE PRIMJENJENIH METODA	16
5.1. Prvi primjer – linearni sustav drugog reda.....	16
5.1.1. Analiza rješenja korištenjem MATLAB-ove funkcije fminunc	18
5.1.1.1. Eulerova metoda	18
5.1.1.2. Runge – Kutta metoda	24
5.1.1.3. Adamsova metoda.....	30
5.1.2. Analiza rješenja korištenjem MATLAB-ove funkcije fmincon	34
5.1.3. Analiza rješenja korištenjem CasADi-ja.....	36
5.2. Drugi primjer – mehanički sustav s dva stupnja slobode gibanja (jedan rotacijski, jedan translacijski)	38
5.2.1. Slučaj bez ograničenja korištenjem Eulerove metode	39
5.2.2. Slučaj uz ograničenja upravljačke varijable korištenjem Eulerove metode	42
5.2.3. Slučaj uz ograničenja na upravljačke varijable i kružne prepreke korištenjem Eulerove metode	44
5.2.4. Slučaj uz ograničenja na upravljačke varijable stanja i slijeđenje ravne linije korištenjem Eulerove metode.....	47
5.2.5. Slučaj uz ograničenja na upravljačke varijable stanja i slijeđenje kružne putanje korištenjem Eulerove metode.....	49
5.2.6. Slučaj uz ograničenja na upravljačke varijable i izbjegavanje kružne prepreke korištenjem CasADi-ja.....	51
6. ZAKLJUČAK.....	54
LITERATURA.....	55
PRILOZI.....	56

POPIS SLIKA

Slika 2.1. Varijable i ograničenja vremenski kontinuiranog problema optimalnog upravljanja [2]	4
Slika 5.1. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po Euleru.....	19
Slika 5.2. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po Euleru.....	19
Slika 5.3. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po Euleru	19
Slika 5.4. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po Euleru	19
Slika 5.5. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po Euleru.....	20
Slika 5.6. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po Euleru.....	20
Slika 5.7. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po Euleru	21
Slika 5.8. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po Euleru	21
Slika 5.9. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po Euleru.....	22
Slika 5.10. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po Euleru.....	22
Slika 5.11. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po Euleru	22
Slika 5.12. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po Euleru	22
Slika 5.13. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po RK4.....	24
Slika 5.14. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po RK4.....	24
Slika 5.15. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po RK4.....	25
Slika 5.16. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po RK4.....	25
Slika 5.17. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po RK4.....	26
Slika 5.18. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po RK4.....	26
Slika 5.19. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po RK4.....	26
Slika 5.20. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po RK4.....	26
Slika 5.21. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po RK4.....	27
Slika 5.22. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po RK4.....	27
Slika 5.23. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po RK4.....	27
Slika 5.24. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po RK4.....	27
Slika 5.25. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama	29
Slika 5.26. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama	29
Slika 5.27. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama	29

Slika 5.28. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama	29
Slika 5.29. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po Adamsu	31
Slika 5.30. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po Adamsu	31
Slika 5.31. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po Adamsu	31
Slika 5.32. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po Adamsu	31
Slika 5.33. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po Adamsu	32
Slika 5.34. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po Adamsu	32
Slika 5.35. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po Adamsu	32
Slika 5.36. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po Adamsu	32
Slika 5.37. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po Adamsu	33
Slika 5.38. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po Adamsu	33
Slika 5.39. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po Adamsu	33
Slika 5.40. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po Adamsu	33
Slika 5.41. Odstupanje x_1 prema analitičkom rješenju za $N=10$ korištenjem fmincon	35
Slika 5.42. Odstupanje x_2 prema analitičkom rješenju za $N=10$ korištenjem fmincon	35
Slika 5.43. Odstupanje u_1 prema analitičkom rješenju za $N=10$ korištenjem fmincon	35
Slika 5.44. Odstupanje u_2 prema analitičkom rješenju za $N=10$ korištenjem fmincon	35
Slika 5.45. Odstupanje x_1 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja ..	37
Slika 5.46. Odstupanje x_2 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja ..	37
Slika 5.47. Odstupanje u_1 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja ..	37
Slika 5.48. Odstupanje u_2 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja ..	37
Slika 5.49. Shema mehaničkog sustava s dva stupnja slobode gibanja [15]	38
Slika 5.50. Konvergencija funkcije cilja za različite N u slučaju bez ograničenja	40
Slika 5.51. First-order optimality measure za različite N u slučaju bez ograničenja	40
Slika 5.52. Konvergencija funkcije cilja u slučaju bez ograničenja za dva tipa algoritma	40
Slika 5.53. First-order optimality measure u slučaju bez ograničenja za dva tipa algoritma ..	40
Slika 5.54. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj bez ograničenja s interior-point algoritmom	41
Slika 5.55. Upravljačke varijable u_1 i u_2 tijekom vremena za slučaj bez ograničenja	42
Slika 5.56. Putanja robota u prostoru za slučaj bez ograničenja	42
Slika 5.57. Konvergencija funkcije cilja u slučaju ograničenja upravljačkih varijabli za dva tipa algoritma	42

Slika 5.58. First-order optimality measure u slučaju ograničenja upravljačkih varijabli za dva tipa algoritma.....	42
Slika 5.59. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli s interior-point algoritmom	43
Slika 5.60. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli.....	44
Slika 5.61. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli	44
Slika 5.62. Konvergencija funkcije cilja u slučaju ograničenja upravljačkih varijabli i kružne prepreke za dva tipa algoritma	44
Slika 5.63. First-order optimality measure u slučaju ograničenja upravljačkih varijabli i kružne prepreke za dva tipa algoritma	44
Slika 5.64. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i kružne prepreke sa sqp algoritmom	45
Slika 5.65. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i kružne prepreke – sqp	46
Slika 5.66. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i kružne prepreke – interior-point	46
Slika 5.67. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke	46
Slika 5.68. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke RK4 metodom za $N=50$	47
Slika 5.69. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke RK4 metodom za $N=100$	47
Slika 5.70. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije sa sqp algoritmom	48
Slika 5.71. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije	49
Slika 5.72. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije.....	49
Slika 5.73. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje sa sqp algoritmom.....	50
Slika 5.74. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje.....	50

Slika 5.75. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje	50
Slika 5.76. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja za $N=100$	51
Slika 5.77. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja za $N=1000$	51
Slika 5.78. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja	52
Slika 5.79. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja	52

POPIS TABLICA

Tablica 5.1. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=10$ i korištenjem fminunc funkcije.....	20
Tablica 5.2. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=100$ i korištenjem fminunc funkcije.....	21
Tablica 5.3. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=1000$ i korištenjem fminunc funkcije.....	22
Tablica 5.4. Analiza odstupanja rješenja od analitičkog za zadane parametre po Eulerovoj metodi.....	23
Tablica 5.5. Brojčane vrijednosti odstupanja po RK4 metodi za $N=10$ i korištenjem fminunc funkcije.....	25
Tablica 5.6. Brojčane vrijednosti odstupanja po RK4 metodi za $N=100$ i korištenjem fminunc funkcije.....	26
Tablica 5.7. Brojčane vrijednosti odstupanja po RK4 metodi za $N=1000$ i korištenjem fminunc funkcije.....	28
Tablica 5.8. Usporedba Eulerove i RK4 metode.....	29
Tablica 5.9. Brojčane vrijednosti odstupanja po Adamsovoj metodi za $N=1000$ i korištenjem fminunc funkcije.....	33
Tablica 5.10. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=10$ i korištenjem fmincon funkcije.....	35
Tablica 5.11. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=100$ i korištenjem fmincon funkcije.....	36
Tablica 5.12. Brojčane vrijednosti odstupanja za $N=10000$ i korištenjem CasADi-ja	37

POPIS OZNAKA

Oznaka	Jedinica	Opis
a	-	konstanta
b	-	konstanta
h	-	korak diskretizacije
H	-	Hamiltonijan (Hamiltonova funkcija)
I	kgm ²	moment tromosti
J	-	funkcija cilja
λ	-	Lagrangeov multiplikator
m	kg	masa
N	-	broj vremenskih intervala
t	s	vrijeme
t_0	s	početno vrijeme
T	s	konačno vrijeme
u	-	upravljačka varijabla
x	-	varijabla stanja
x_0	-	početno stanje

SAŽETAK

Optimalno upravljanje dinamičkim sustavima ključno je za postizanje visokih performansi i maksimalno iskorištavanje potencijala tehnologije. Ovaj rad pruža prvo teorijski, a onda i eksperimentalni pregled tri numeričke metode diskretizacije - Eulerove, Runge-Kuttove metode 4. reda i Adamsove metode 4. reda. Također i dvije metode za numeričku optimizaciju – sekvencijalno kvadratično programiranje (*sqp*) i metodu unutarnje točke (*interior-point*). Ističu se i relevantni alati poput MATLAB-a i CasADi-ja za implementaciju tih metoda te njihova primjena. Analiza učinkovitosti i točnosti ovih metoda kroz numeričke eksperimente pruža uvid u njihove prednosti, nedostatke i primjenjivost u stvarnim situacijama. Dva su primjera, linearni sustav drugog reda i mehanički sustav s dva stupnja slobode gibanja. Napomena je na tome da ne postoji univerzalna metoda koja bi odgovarala svakom problemu te je stoga bitno odabrati metodu sukladno specifičnim zahtjevima sustava. Kroz primjere riješenih problema, demonstrirana je primjena teorijskih koncepata i numeričkih metoda u praksi. Vide se odstupanja od analitičkih rješenja te međusobna usporedba metoda i korištenih algoritama iz programskih alata. Rad ukazuje na važnost optimalnog upravljanja i njegovu primjenu pogotovo kad se radi o složenijim problemima koji se žele što brže i točnije riješiti.

Ključne riječi: optimalno upravljanje, metode diskretizacije, metode numeričke optimizacije, MATLAB, CasADi

SUMMARY

Optimal control of dynamic systems is crucial for achieving high performance and maximizing the potential of technology. This study provides a theoretical and experimental overview of three numerical discretization methods - Euler's, Runge-Kutta's fourth-order method, and Adams' fourth-order method. Additionally, it covers two numerical optimization methods - sequential quadratic programming (SQP) and interior-point method. Relevant tools such as MATLAB and CasADi for implementing these methods and their application are highlighted. An analysis of the efficiency and accuracy of these methods through numerical experiments provides insight into their advantages, limitations, and applicability in real-life situations. Two examples are presented: a second-order linear system and a mechanical system with two degrees of freedom. It is noted that there is no universal method that would suit every problem, thus it is important to choose a method according to the specific requirements of the system. Through solved problem examples, the application of theoretical concepts and numerical methods in practice is demonstrated. Deviations from analytical solutions are observed, as well as a comparison between methods and algorithms used in software tools. This study emphasizes the importance of optimal control and its application, especially when dealing with more complex problems that require fast and accurate solutions.

Key words: optimal control, discretization methods, numerical optimization methods, MATLAB, CasADi

1. UVOD

U današnjem sve sofisticiranijem svijetu, gdje tehnološki napredak ide paralelno s brzim društvenim i ekonomskim promjenama, sustavi s kojima se svakodnevno susrećemo postaju izuzetno kompleksni. Od autonomnih uređaja do globalnih komunikacijskih mreža, ovi sustavi suočavaju se s dinamičnim izazovima, a upravljanje ovim dinamičnim sustavima postaje ključno kako bismo ostvarili željene performanse i iskoristili potencijal tehnologije za potrebe suvremenog društva. Kroz analizu njihove dinamike i identifikaciju ključnih varijabli, možemo razviti strategije optimalnog upravljanja koje će omogućiti sustavu postizanje visokih performansi u raznim situacijama. Optimalno upravljanje, stoga, postaje neizostavan element u postizanju ne samo efikasnog rada sustava, već i smanjenja troškova, povećanja učinkovitosti te optimizaciji. U suvremenoj tehnologiji, optimalno upravljanje može transformirati način na koji funkcioniraju industrijski procesi, energetske sustavi i mnogi drugi sektori, otvarajući put inovacijama i unapređenjima koja doprinose globalnom napretku. Zbog toga tema optimalnog upravljanja dinamičkim sustavima igra ključnu ulogu u suvremenoj tehnologiji, inženjerstvu i znanstvenim istraživanjima potičući na dublje razumijevanje kako optimizirati složene sustave za postizanje željenih ciljeva u dinamičnom okruženju.

Za optimizaciju bilo kojeg sustava potrebno je poznavanje modela sustava, kriterije i ograničenja. Ograničenja mogu biti različite vrste i mogu se izražavati na različite načine. Kada su u pitanju dinamički problemi, tipičan model optimizacije uključuje integralnu funkciju, skup ograničavajućih diferencijalnih jednadžbi i lokalna ograničenja koja se primjenjuju na upravljačke varijable. Cilj optimizacije je maksimizirati ili minimizirati tu funkciju, a zadatak je pronaći najbolju dinamiku u smislu najboljih upravljačkih funkcija i odgovarajuće optimalne trajektorije [1].

Optimalno upravljanje odnosi se na optimizaciju dinamičkih sustava. Dinamički sustavi obuhvaćaju procese koji se razvijaju tijekom vremena i koji se mogu opisati pomoću stanja x , što omogućuje predviđanje budućeg ponašanja sustava. Često se dinamičkim sustavima može upravljati odgovarajućim izborom ulaza, koji se nazivaju upravljačke varijable u . Obično se te varijable optimalno biraju kako bi se optimizirala određena funkcija cilja, uz poštivanje postavljenih ograničenja, a proces pronalaženja optimalnih upravljačkih varijabli često zahtijeva primjenu numeričkih metoda [2].

1.1. Cilj rada

Cilj ovog rada je pružiti pregled metoda diskretizacije u kontekstu optimalnog upravljanja dinamičkim sustavima. Prikazat će se rješavanje diferencijalnih jednadžbi koje proizlaze iz problema optimalnog upravljanja. Konkretno, istražiti će se tri ključne diskretizacijske metode: Eulerova metoda, Runge-Kuttova metoda 4. reda i Adamsova metoda 4. reda. Uz ove, još će se prikazati i dvije metode za numeričku optimizaciju: sekvencijalno kvadratično programiranje i metoda unutarnje točke. Osim toga, rad će pružiti pregled software alata koji se koriste za implementaciju tih metoda u kontekstu optimizacije i optimalnog upravljanja, s posebnim osvrtom na MATLAB i CasADi. Dodatno, rad će obuhvatiti numeričke eksperimente koji će biti provedeni kako bi se detaljno analizirala učinkovitost i točnost primijenjenih metoda. Cilj je pružiti razumijevanje praktičnih aspekata primijenjenih numeričkih tehnika, uključujući njihove prednosti, ograničenja i moguće primjene u stvarnim scenarijima optimalnog upravljanja. Kroz usporedbe rezultata dobivenih različitim metodama, rad će doprinijeti boljem razumijevanju optimalnog upravljanja u dinamičkim sustavima.

1.2. Pregled literature i strukture

Osnovni koncepti optimalnog upravljanja nelinearnim sustavima mogu se naći u [2] i [4]. U ovoj literaturi dani su izvodi nužnih uvjeta optimalnosti te su obrađene neke analitičke i numeričke metode za rješavanje problema optimalnog upravljanja i općenito optimizacije. Za eksperimentalnu primjenu i znanja o korištenim programskim paketima, najbolje je vidjeti [8] i [10] te također službenu dokumentaciju od MATLAB-a i CasADi-ja u kojoj se mogu naći dodatni primjeri koji mogu poslužiti u kreiranju novih rješenja zadataka.

U radu će prvo biti opisani teorijski izvodi i formule za dinamički sustav, uz njihova ograničenja. Slijedi, također, teorijski dio i opisi diskretizacijskih metode za numeričko rješavanje diferencijalnih jednadžbi i metoda za numeričku optimizaciju te programski alati koji su korišteni za optimizaciju. Na kraju su dva riješena primjera, jedan linearni sustav 2. reda te mehanički sustav s dva stupnja slobode gibanja.

2. FORMULACIJA PROBLEMA I NUŽNI UVJETI OPTIMALNOSTI

Povijest seže od pokušaja pronalaska analitičkih rješenja do razvoja numeričkih i računalnih metoda za njihovo modeliranje i analizu. Formulacija ovih modela često uključuje identifikaciju ključnih varijabli i parametara sustava te razumijevanje njihovih međusobnih interakcija. Znanstvenici su postepeno usavršavali i razrađivali modele kako bi došli do onih koji najbolje mogu opisati zadane sustave.

2.1. Formulacija problema

Formulirati problem optimalnog upravljanja u općem slučaju nelinearnog sustava, uz Bolza funkciju cilja i ograničenja tipa jednakosti i nejednakosti, možemo početi s općim oblikom problema optimalnog upravljanja. Neka je dana diferencijalna jednadžba opisana sa:

$$\dot{x} = f(x, u, t), \quad (2.1.1)$$

gdje je x vektor stanja sustava, u je vektor upravljačkih varijabli, a t je vrijeme. Također, neka je zadana Bolza funkcija cilja $J(x, u)$. Ograničenja tipa jednakosti i nejednakosti mogu biti oblika:

$$h(x, u, t) = 0, \quad (2.1.2)$$

$$g(x, u, t) \leq 0. \quad (2.1.3)$$

Cilj je pronaći optimalne vrijednosti upravljačkih varijabli $u(t)$ koje minimiziraju Bolza funkciju cilja uz zadovoljavanje ograničenja. Kako bismo ovaj problem pretvorili u konačnodimenzionalni problem nelinearnog programiranja (NLP), koristimo princip „prvo diskretizirati, onda optimizirati“. To znači da prvo diskretiziramo kontinuirani problem u diskretnim vremenskim koracima, što rezultira konačnim brojem varijabli. Onda možemo koristiti numeričke metode optimizacije za rješavanje dobivenog problema. Postupak diskretizacije obično uključuje zamjenu kontinuiranih varijabli s diskretnim vrijednostima u određenim vremenskim točkama. Na primjer, možemo diskretizirati varijable stanja $x(t)$ na N vremenskih koraka, što rezultira nizom diskretnih varijabli x_0, x_1, \dots, x_N . Također, varijable upravljanja $u(t)$ se isto diskretiziraju. Nakon diskretizacije, možemo formulirati konačnodimenzionalni problem nelinearnog programiranja koristeći te diskretizirane varijable. Problem se obično formulira kao minimizacija Bolza funkcije cilja uz zadovoljavanje diskretiziranih ograničenja jednakosti i nejednakosti. Ovaj konačnodimenzionalni problem

može se riješiti poznatim numeričkim metodama optimizacije, kao npr. metode konačnih razlika. Važno je odabrati odgovarajuću metodu ovisno o karakteristikama problema i zahtjevima performansi.

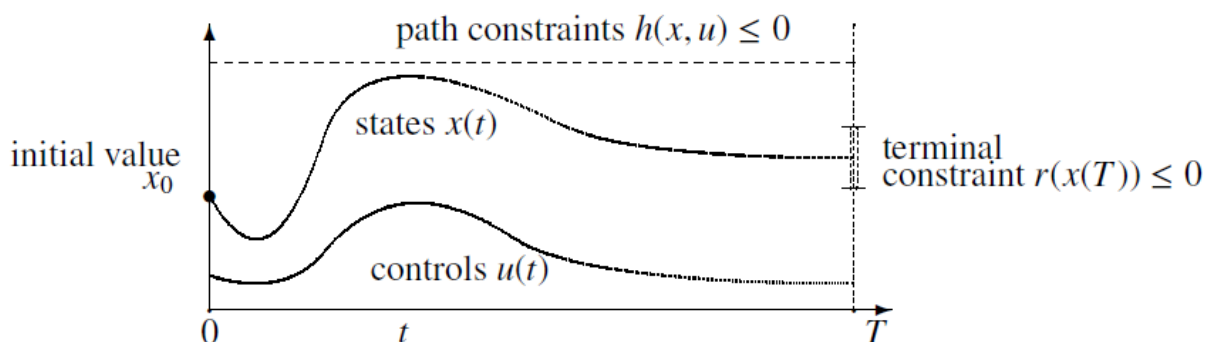
Formulacija problema kontinuiranog optimalnog upravljanja u ODE (*ordinary differential equation* – obična diferencijalna jednačina) postavkama može se izraziti kao:

$$\min_{x(\cdot), u(\cdot)} \int_0^T L(x(t), u(t)) dt + E(x(T)), \quad (2.1.1)$$

gdje imamo:

$$\begin{aligned} x(0) - x_0 &= 0, \text{ fiksna početna vrijednost,} \\ \dot{x}(t) - f(x(t), u(t)) &= 0, \quad t \in [0, T], \text{ ODE model,} \\ h(x(t), u(t)) &\leq 0, \quad t \in [0, T], \text{ ograničenja putanje,} \\ r(x(T)) &\leq 0, \text{ terminalna ograničenja.} \end{aligned}$$

Problem i njegove varijable prikazuje slika 2.1.



Slika 2.1. Varijable i ograničenja vremenski kontinuiranog problema optimalnog upravljanja [2]

Integralni doprinos funkciji cilja $L(x, u)$ ponekad se naziva *Lagrangeov* izraz, a terminalna funkcija cilja $E(x(T))$ ponekad se naziva *Mayerov* izraz. Kombinacija oba naziva se *Bolza* funkcija cilja [2].

2.2. Dinamički model nelinearnih sustava

Netrivijalni dio svakog dijela upravljanja je modeliranje procesa. Cilj je dobiti najjednostavniji matematički opis koji adekvatno predviđa odgovor fizičkog sustava na sve predviđene ulaze. Ako to ograničimo na sustave opisane običnim diferencijalnim jednačinama (u obliku varijable stanja), onda su

$$x_1(t), x_2(t), \dots, x_n(t),$$

varijable stanja (ili skraćeno samo stanja) u trenutku t , a

$$u_1(t), u_2(t), \dots, u_m(t),$$

upravljačke varijable, tj. ulazi u vremenu t . Tada se sustav može opisati s n diferencijalnih jednadžbi prvog reda

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t), t) \\ \dot{x}_2(t) &= f_2(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t), t) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1(t), x_2(t), \dots, x_n(t), u_1(t), u_2(t), \dots, u_m(t), t) \end{aligned} \quad (2.2.1)$$

Sada se može definirati

$$x(t) \triangleq \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix},$$

kao vektor stanja te

$$u(t) \triangleq \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix},$$

kao vektor upravljanja. Tada se dolazi i do jednadžbe stanja u vektorskom obliku

$$\dot{x}(t) = f(x(t), u(t), t), \quad x(t_0) = x_0. \quad (2.2.2)$$

Pretpostavka je da su funkcije $f_1(\cdot), \dots, f_n(\cdot)$ kontinuirane, tako da sustav jednadžbi (2.1.2) ima jedinstveno rješenje. Ako nema upravljačkih varijabli $u(t)=0$, tada sustav (2.1.2) postaje

$$\dot{x}(t) = f(x(t), t), \quad x(t_0) = x_0, \quad (2.2.3)$$

nelinearni vremenski-varijabilni sustav [3].

2.3. Dinamička ograničenja

Problem optimalnog upravljanja može se tumačiti kao proširenje nelinearnog problema beskonačnog broj varijabli. Prvo se može promatrati jednostavan problem s jednom fazom i

bez ograničenja putanje. Točnije, pretpostavlja se odabiranje upravljačke varijable $u(t)$ koje treba minimizirati [4]

$$\min J = E(x(T), T) + \int_{t_0}^T L(x(t), u(t), t) dt, \quad (2.3.1)$$

uz jednadžbu stanja

$$\dot{x} = f(x(t), u(t), t), \quad (2.3.2)$$

i uvjete

$$x(t_0) = x_0, \quad t_0 \leq t \leq T, \quad (2.3.3)$$

gdje su početni uvjeti zadani u fiksnom početnom vremenu t_0 , a konačno vrijeme T je proizvoljno. Ova jednostavna verzija ima ograničenje tipa jednakosti. Ograničenje jednakosti (2.3.2.) može se promatrati kao „kontinuirano“ budući da mora biti zadovoljeno u cijelom intervalu $t_0 \leq t \leq T$ dok se jednakost (2.3.3) može promatrati kao „diskretna“ budući da je nametnuta u određenom vremenu T .

Da bi $u(t)$ bio optimalan, mora biti zadovoljeno:

$$\dot{x} = f(x, u, t), \quad (2.3.4)$$

$$\dot{\lambda} = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f}{\partial x}, \quad (2.3.5)$$

$$\frac{\partial L}{\partial u} + \lambda^T \frac{\partial f}{\partial u} = 0, \quad (2.3.6)$$

uz početne i rubne uvjete:

$$x(t_0) = x_0, \quad (2.3.7)$$

$$\lambda(T) = \frac{\partial E}{\partial x(T)}. \quad (2.3.8)$$

Ako se uz Lagrangeov mutiplikator λ definira i Hamiltonian

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t), \quad (2.3.9)$$

tada su uvjeti optimalnosti

$$\dot{x} = \frac{\partial H}{\partial \lambda}, \quad (2.3.10)$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x}, \quad (2.3.11)$$

$$\frac{\partial H}{\partial u} = 0. \quad (2.3.12)$$

Jednadžbe (2.3.10), (2.3.11) i (2.3.12) su tada kanonske jednadžbe optimalnog upravljanja.

3. METODE ZA NUMERIČKO RJEŠAVANJE PROBLEMA OPTIMALNOG UPRAVLJANJA

Numeričke metode za rješavanje problema optimalnog upravljanja ključne su tehnike u matematičkom modeliranju i inženjerskim problemima gdje se traži optimalni način upravljanja sustavom kako bi se postigao željeni cilj, uzimajući u obzir različita ograničenja i uvjete. Ovdje će se vidjeti pregled triju diskretizacijskih metoda koje se često koriste za rješavanje takvih problema i koje su korištene u ovom radu te dvije metode za numeričku optimizaciju.

3.1. Metode diskretizacije

3.1.1. Eulerova metoda

Ova metoda najjednostavnija je za rješavanje inicijalnog problema za običnu diferencijalnu jednačinu oblika

$$\dot{y} = f(x, y), \quad y(a) = y_0. \quad (3.1.1.1)$$

Metoda se zasniva na ideji da se \dot{y} u jednačini (3.1.1) zamijeni s podijeljenom razlikom

$$\dot{y}(x) = \frac{y(x+h) - y(x)}{h} + O(h), \quad (3.1.1.2)$$

pa rješenje diferencijalne jednačine zadovoljava

$$y(x+h) = y(x) + h\dot{y}(x) + O(h^2) = y(x) + hf(x, y(x)) + O(h^2), \quad (3.1.1.3)$$

gdje je h korak diskretizacije, a $O(h^2)$ označava drugi red greške.

Zanemarivanjem kvadratnog člana u razvoju (3.1.1.3) dobiva se aproksimacija

$$y(x+h) \approx y(x) + hf(x, y(x)). \quad (3.1.1.4)$$

Može se dokazati da je Eulerova integracijska metoda stabilna, tj. da je širenje lokalnih grešaka ograničeno konstantom koja je neovisna o veličini koraka h . Tako da aproksimacija postaje sve bolja kada se smanjuje veličina koraka h . Budući da je pogreška konzistentnosti u svakom koraku reda h^2 , a ukupni broj koraka reda $\Delta t/h$, akumulirana pogreška u zadnjem koraku je reda $h\Delta t$. Kako je to linearno, kaže se da je Eulerova metoda metoda prvog reda. Ako je broj koraka veći, metoda je preciznija, no zahtijeva veće vrijeme izračunavanja. U praksi nije toliko konkurentna jer druge metode daju točnije rezultate uz manje vrijeme računanja [2].

3.1.2. Runge – Kutta metoda 4. reda

Zajedno s Eulerovom metodom, ove dvije metode zovemo jednokoračne jer za aproksimaciju y_{i+1} koristimo samo vrijednosti y_i u prethodnoj točki x_i , tj. u jednom koraku dobijemo y_{i+1} iz y_i . Jedna je od najraširenijih metoda, a često se kraticom zove RK4. Jedan korak ove metode zahtijeva četiri procjene funkcije f .

$$k_1 = f(x_j, u_{const}), \quad (3.1.2.1)$$

$$k_2 = f\left(x_j + \frac{h}{2}k_1, u_{const}\right), \quad (3.1.2.2)$$

$$k_3 = f\left(x_j + \frac{h}{2}k_2, u_{const}\right), \quad (3.1.2.3)$$

$$k_4 = f(x_j + hk_3, u_{const}), \quad (3.1.2.4)$$

$$x_{j+1} = x_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \quad (3.1.2.5)$$

Jedan korak RK4 jednak je kao četiri koraka Eulerove metode. Može se pokazati da točnost konačne aproksimacije je reda $h^4\Delta t$. U praksi to znači da ova metoda obično zahtijeva puno manje procjena funkcija nego Eulerova metoda da bi se postigla ista razina točnosti [2].

3.1.3. Adams – Bashforth i Adams – Moulton metode

Kada se u Eulerovoj metodi računa aproksimacija u t_{k+1} koristi se aproksimacija samo iz prethodnog trenutka. Ipak, ima smisla uzimati u obzir i sve prije izračunate vrijednosti. Ovo su metode koje koriste više prethodnih aproksimacija i zovu se višekoračne metode. Adams – Bashdorth-ove metode su eksplicitne Adamsove metode, što znači da koraci računanja budućih vrijednosti ne zahtijevaju rješavanje sustava jednadžbi, već se mogu izračunati pomoću prethodno poznatih vrijednosti. Osnovna jednadžba za ovu metodu 4. reda je

$$y_{n+4} = y_{n+3} + \frac{h}{24}(55f_{n+3} - 59f_{n+2} + 37f_{n+1} - 9f_n), \quad (3.1.3.1)$$

Adams – Moulton-ove metode su implicitne, što znači da zahtijevaju rješavanje sustava jednadžbi kako bi se dobile buduće vrijednosti. Ova metoda koristi implicitni korak unaprijed u procjeni. Odabir između ovih metoda ovisi o konkretnim uvjetima problema, numeričkim svojstvima i preciznosti koja se traži. Adams – Moulton-ova metoda 4. reda koristi osnovnu jednadžbu

$$y_{n+4} = y_{n+3} + \frac{h}{24}(9f_{n+4} + 19f_{n+3} - 5f_{n+2} + f_{n+1}), \quad (3.1.3.2)$$

gdje su, za obje metode:

- y_{n+4} vrijednosti funkcije u budućem koraku $x_{n+4} = x_n + 4h$,
- y_{n+3} vrijednosti funkcije u budućem koraku $x_{n+3} = x_n + 3h$,
- $f_{n+i} = f(x_{n+i}, y_{n+i})$ je vrijednost izvedenice funkcije u točki x_{n+i} i y_{n+i} za $i=0, 1, 2, 3$ [5].

3.2. Metode za numeričku optimizaciju

3.2.1. Sekvencijalno kvadratično programiranje (sqp)

Ova metoda može rješavati probleme s proizvoljno nelinearnim funkcijama cilja i ograničenjima. Ipak, važno je imati na umu da nije uvijek u mogućnosti pronaći globalni optimum zbog različitih karakteristika funkcije i ograničenja. U svakoj iteraciji, cilj je aproksimirati nelinearni optimizacijski problem kvadratičnim programom i riješiti ga, na primjer, metodom aktivnih ograničenja. Ideja sekvencijalnog kvadratičnog programiranja (engl. *Sequential Quadratic programming - sqp*) jest modelirati nelinearni problem s trenutnom točkom x_k kao kvadratični program koji se minimizira kako bi se pronašla točka x_{k+1} za sljedeću iteraciju. Budući da kvadratično programiranje radi samo s kvadratnim funkcijama cilja i linearnim ograničenjima, nelinearnu funkciju cilja aproksimiramo kvadratnom funkcijom, dok ograničenja aproksimiramo linearnim funkcijama.

Ako je x_k trenutna točka našeg problema, onda se može napraviti Taylorov razvoj oko te točke

$$g(x_k + p) = \nabla g(x_k)^T p + g(x_k). \quad (3.2.1.1)$$

Jednako bi se mogla aproksimirati i funkcija cilja pomoću Taylorovog reda, no ta aproksimacija se pokazala jako lošom u određenim problemima. To se događa zbog toga što se informacija o nelinearnosti originalnih ograničenja izgubila. Zato se nameće da se umjesto aproksimacije originalne funkcije cilja aproksimira Lagrangeova funkcija koja se sastoji od funkcije cilja i ograničenja

$$L(x_k + p; \lambda_k) = \frac{1}{2} p^T \nabla^2 L(x_k; \lambda_k) p + \nabla f(x_k)^T p + f(x_k), \quad (3.2.1.2)$$

gdje je p korak, odnosno razlika između trenutne i nove točke [6].

3.2.2. Metoda unutarnje točke (interior-point)

Metoda unutarnje točke često se koristi u optimizacijskim postupcima koji uzimaju u obzir ograničenja nejednakosti. Prije pokretanja takvih postupaka, bitno je osigurati da početna točka leži unutar područja gdje su sva ograničenja nejednakosti zadovoljena. Zbog toga prije pokretanja takvih postupaka potrebno je provjeriti zadovoljava li zadana točka sva ograničenja nejednakosti te ako ne zadovoljava potrebno je odrediti novu početnu točku. Pretpostavimo da su nam ograničenja nejednakosti zadana u obliku

$$g_i(x) \geq 0. \quad (3.2.2.1)$$

To znači da će ograničenja imati negativnu vrijednost dokle god nisu zadovoljena. Također, što je točka x bliža granici ograničenja nejednakosti to vrijednost $g_i(x)$ postaje sve manja. Na temelju prethodnih pretpostavki, novu početnu točku možemo odrediti na način da definiramo pomoćni optimizacijski kod kojeg optimiramo funkcijom

$$G(x) = - \sum_{i=0}^{i < n} t_i g_i(x), \quad (3.2.2.2)$$

pri čemu je t definiran kao

$$t_i = \begin{cases} 0 & \text{ako } g_i(x) \geq 0 \\ 1 & \text{ako } g_i(x) < 0 \end{cases} \quad (3.2.2.3)$$

Ideja ovog pomoćnog optimizacijskog postupka je sumiranje svih ograničenja nejednakosti koja nisu zadovoljena u trenutno promatranoj točki x . Kako se približavamo granici ograničenja, vrijednost ove funkcije se smanjuje. Na taj način, postupak optimizacije koji se primjenjuje nastoji iterativno približiti točku granici dok točka ne zadovolji sva ograničenja. Koja ograničenja će biti uključena u sumu kontrolira se parametrom t_i , koji je definiran za svako pojedino ograničenje. Kada ograničenje nije zadovoljeno, njegov pripadni parametar t_i je jednak 1 te se na taj način vrijednost ograničenja dodaje u sumu. U suprotnom, kada je ograničenje zadovoljeno, parametar t_i postavlja se na 0, a time se to ograničenje isključuje iz sume. Kako se prethodno navelo, točke imaju negativne vrijednosti kada nisu zadovoljena ograničenja. Stoga, ispred sume dodajemo negaciju kako bismo definirali funkciju koja se minimizira. Minimalna vrijednost ove pomoćne funkcije je 0. Kada postignemo tu vrijednost, znamo da smo pronašli točku koja zadovoljava sva ograničenja [7].

4. MATEMATIČKI PROGRAMSKI ALATI ZA OPTIMIZACIJU

4.1. MATLAB

MATLAB je matematički programski paket namijenjen znanstvenim i inženjerskim numeričkim izračunima. Razvijen je kao interpreterski programski jezik visoke razine koji se prvotno temeljio na složenim matricama kao osnovnom tipu podataka, a ime "MATLAB" dobio je kao skraćenicu od "matrični laboratorij". Sposobnost povezivanja s programima napisanim u jezicima C ili Fortran čini ga pogodnim za složene projekte, a gotova rješenja za različita područja primjene neprestano proširuju njegove mogućnosti. S obzirom na svoj oblik, MATLAB je blizak načinu na koji inače zapisujemo matematičke formule, pa se jedan redak koda u MATLAB-u može usporediti sa stotinama redaka napisanim u općem programskom jeziku. MATLAB je, stoga, jezik visoke učinkovitosti u tehničkom računanju. On integrira računanje, vizualizaciju i programiranje, omogućujući izražavanje rješenja na konvencionalan matematički način.

Tipična upotreba MATLAB-a uključuje:

1. matematiku i računanje,
2. razvitak algoritama,
3. modeliranje, simulaciju i izgradnju prototipova,
4. analizu, obradu i vizualizaciju podataka,
5. znanstvenu i inženjersku grafiku,
6. razvitak gotovih rješenja (aplikacija) s GUI (*Graphical User Interface*) alatima [8].

Današnje mogućnosti MATLAB daleko nadmašuju originalni "matrični laboratorij". Osim osnovnog paketa, dostupni su brojni programski alati koji pokrivaju gotovo sva područja inženjerske djelatnosti, uključujući obradu signala, obradu slika, 2D i 3D grafičko oblikovanje, automatsko upravljanje, identifikaciju sustava, statističku analizu, neuronske mreže i mnoge druge.

Ovdje su korištene dvije osnovne funkcije:

- *fminunc* – koristi se za minimizaciju funkcije bez ograničenja,
- *fmincon* – koristi se za minimizaciju funkcije s ograničenjima.

4.1.1. *fminunc*

Ova funkcija za minimizaciju bez ograničenja daje dva izbora algoritma za njeno izvršavanje. Prvi, onaj koji je zadan „*quasi-newton*“, a također i korišten, te drugi „*trust-region*“ koji zahtijeva da se navede gradijent. Algoritam „*quasi-newton*“ daje tri metode za izračunavanje (bfgs, dfp, steepdesc). Ovdje je također korištena zadana BFGS (Broyden–Fletcher–Goldfarb–Shanno algoritam) za iterativno izračunavanje nelinearnog problema koja koristi postupak rada preko kubične funkcije te aproksimira Hessian matrice.

W. C. Davidon je prvi iznesao ideju quasi-Newtonovog algoritma koja je tada bila najkreativnija za razvoj nelinerane optimizacije. Zahtijeva se funkcija cilja kojom se mjeri promjena gradijenta te se konstruira model koji će dobro konvergirati. Budući da nisu potrebne druge derivacije, ova metoda učinkovitija je od Newtonove metode.

Trust region metoda definira područje (regiju) oko trenutne iteracije unutar koje vjeruje da je u modelu primijenjena funkcija cilja, a zatim odabire korak koji će biti aproksimiran za minimizaciju u definiranom području. Istovremeno se biraju i smjer i duljina koraka. Ako korak nije prihvatljiv, smanjuje se veličine regija i pronalaze se novi minimizatori [9].

4.1.2. *fmincon*

Ova funkcija za minimizaciju s ograničenjima, za razliku od funkcije *fminunc*, ima pet algoritama optimizacije:

- '*interior-point*' (default),
- '*trust-region-reflective*',
- '*sqp*',
- '*sqp-legacy*',
- '*active-set*'.

Preporuka je da se krene od onoga što je zadano, tj. '*interior-point*' jer rješava velike, rijetke probleme, ali i male, češće probleme. Algoritam zadovoljava granice u svim iteracijama i može se „oporaviti“ kad rezultat operacije nije validan numerički rezultat. Drugi algoritam koji je korišten u ovom radu je „*sqp*“ (*sequential quadratic programming* – sekvencijalno kvadratično programiranje) koji zadovoljava granice u svim iteracijama. Također se može „oporaviti“ kad rezultat operacije nije validan numerički rezultat, ali za razliku od „*interior-point*“ koji je algoritam velikih razmjera, „*sqp*“ to nije.

4.2. CasADi

CasADi (Computer Algebra System for Automatic Differentiation and Symbolic Computation in Optimization) je besplatni *open-source* alat za nelinearnu optimizaciju i algoritamsku diferencijaciju. Omogućava brzu i učinkovitu implementaciju različitih metoda za numeričko optimalno upravljanje, kako u kontekstu *offline* analize, tako i za nelinearnu prediktivnu kontrolu modela (engl. nonlinear model predictive control - NMPC). Opće je namjene te se može koristiti za modeliranje i rješavanje optimizacijskih problema s velikim stupnjem fleksibilnosti što je poželjno za probleme ograničene diferencijalnim jednadžbama¹ [10].

Započeo je kao alat za algoritamsko diferenciranje (AD) koristeći sintaksu sličnu sustavu računalne algebre (engl. computer-algebra system - CAS), što govori i njegov naziv. Iako je AD još uvijek ključna značajka, fokus se pomiče prema optimizaciji. U svom trenutnom obliku ima skup namjena za drastično smanjivanje napora potrebnog za implementaciju velikog skupa algoritama za numeričko optimalno upravljanje, bez „žrtvovanja“ učinkovitosti [10].

CasADi koristi sintaksu inspiriranu MATLAB-om, koristeći tip "sve je matrica", tj. skalari se tretiraju kao matrice dimenzija 1×1 , a vektori kao matrice dimenzija $n \times 1$. Za simbolički okvir poput CasADi-ja, rad s jednim tipom podataka olakšava učenje i održavanje alata.

CasADi je softver koji je posebno prilagođen za dinamičku optimizaciju i stekao je široku popularnost od svog izdanja 2012. godine. Sam po sebi CasADi nije rješavač za općenite probleme optimalnog upravljanja već pruža potrebne temeljne dijelove za formuliranje i rješavanje ovih općenitih problema. To je omogućilo inženjerima i istraživačima da formuliraju i riješe složenije probleme optimalnog upravljanja nego što je to bilo moguće koristeći standardni softver. Prednosti korištenja alata baziranog na CasADi-u, umjesto prilagođavanja koda za svaki problem, su mnoge. Na primjer, otklanjanje poteškoća u kodu baziranom na CasADi-u je teško jer je teško razlikovati greške u formulaciji problema od grešaka u metodi diskretizacije. Alat može pružiti metode diskretizacije koje su se pokazale učinkovitima na problemima s poznatim rješenjima što olakšava otklanjanje poteškoća. Također, može pružiti dobru separaciju između formulacije problema i metode diskretizacije što olakšava promjenu metode rješavača i postavki. Nedostatak alata je što je teško učiniti ga jednako fleksibilnim kao kôd koji je prilagođen za svaki problem. CasADi ima za cilj biti dovoljno fleksibilan da riješi

¹ dostupno na www.casadi.org, datum pristupanja 7.1.2024.

probleme optimalnog upravljanja iz stvarnog svijeta, dok je istovremeno dovoljno jednostavan za korištenje od osoba koje nisu stručnjaci u području numeričkog optimalnog upravljanja [11].

Algoritamsko diferenciranje (AD) znano i kao automatsko diferenciranje je način učinkovitog izračunavanja derivacija funkcija predstavljene algoritmom u CasADi-ju. AD koristi točne formule zajedno s vrijednostima umjesto nizova izraza numeričke diferencijacije. Može koristiti točne numeričke vrijednosti derivacija, gradijenta, Jacobiana i Taylorovih polinoma koje se ne mogu koristiti u drugim numeričkim metodama. Naziva se automatsko jer dobro radi kad ima zadane funkcije s velikim kodom. Ideja je proširiti ulaz kako bi se ukazalo na željenu derivaciju i omogućiti programskim operacijama da naprave numeričku derivaciju kao vrijednost funkcije.

Dok su osnovni principi i tehnike algoritamskog diferenciranja za nekoga prilično jednostavne, njihova primjena u većim problemima predstavlja dosta izazova. Za funkciju $y = f(x)$ gdje su x i y vektorske vrijednosti, AD način unaprijed pruža način za točno izračunavanje Jacobian-vektor produkta

$$\hat{y} := \frac{\partial f}{\partial x} \hat{x},$$

uz uračunati gubitak usporediv s vrednovanjem originalne funkcije $f(x)$. Ova definicija se po prirodi proširuje s matricnom vrijednosti funkcije $Y = F(X)$, s jednostavnim definiranjem $x := \text{vec}(X)$ i $y := \text{vec}(Y)$, gdje $\text{vec}(\cdot)$ označava slaganje stupaca okomito. To također generalizira daljnje slučajeve koji imaju veći broj vrijednosti matičnih ulaza i izlaza što je općenito slučaj za funkcije u CasADi-u.

S druge strane, AD način unatrag pruža način za točno izračunavanje Jacobian-transponiranog-vektor produkta

$$\bar{x} := \frac{\partial f^T}{\partial x} \bar{y},$$

također uz uračunati gubitak usporediv s vrednovanjem originalne funkcije $f(x)$. Za razliku od načina unaprijed, način unatrag obuhvaća veće opterećenje memorije. Ova definicija općenito se proširuje s funkcijom koja uzima više matičnih ulaza i izlaza.

Svaka implementacija AD-a funkcionira rastavljanjem računa u dijelove niza poznatih, ako je moguće eksplicitnih, lančanih pravila. Na primjer, AD pravilo načina unaprijed za množenje više matrica $Y = X_1 X_2$ dano je kao:

$$Y = \hat{X}_1 X_2 + X_1 \hat{X}_2,$$

dok je AD pravilo načina unazad dano kao:

$$\bar{X}_1 = \bar{Y}X_2^T,$$

$$\bar{X}_2 = X_1^T\bar{Y}.$$

Ova dva načina daju dva mehanizma za učinkovito i točno rješavanje usmjerenih derivacija. Rješavanje Jacobian matrice može bit velik i težak problem. Problemi većih derivacija mogu se tretirati kao posebni slučajevi ili korištenjem AD algoritama rekurzivno² [12, 13, 14].

² dostupno na www.casadi.org, datum pristupanja 7.1.2024.

5. PRIMJERI I USPOREDBE PRIMJENJENIH METODA

5.1. Prvi primjer – linearni sustav drugog reda

Zadan je linearni sustav drugog reda:

$$\begin{aligned}\dot{x}_1(t) &= u_1(t), & x_1(0) &= 0, \\ \dot{x}_2(t) &= x_1(t) + u_2(t), & x_2(0) &= 0.\end{aligned}$$

Odredi upravljačke varijable u_1 i u_2 koje minimiziraju funkciju cilja

$$J = a \cdot x_1(1) + b \cdot x_2(1) + \int_0^1 (u_1^2 + u_2^2 + x_1) dt,$$

gdje su a i b konstante, $x_1(1)$ i $x_2(1)$ vrijednosti stanja u konačnom vremenu $t_f=1$ s. Slijedi izračun analitičkog rješenja s kojim će se ono uspoređivati s tri različite metode.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ x_1 + u_2 \end{bmatrix},$$

gdje desni dio jednadžbe predstavlja f analogno poglavlju 2.2. Početni uvjeti slijede

$$x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

$$J = ax_1(T) + bx_2(T) + \int_0^{t_f} (u_1^2 + u_2^2 + x_1) dt, \quad T = 1s,$$

u funkciji cilja, prva dva člana predstavljaju E , a dio unutar integrala L , analogno poglavlju 2.3.

Hamiltonijan je tada

$$\begin{aligned}H &= F + \lambda^T f = u_1^2 + u_2^2 + x_1 + [\lambda_1 \quad \lambda_2] \begin{bmatrix} u_1 \\ x_1 + u_2 \end{bmatrix}, \\ H &= u_1^2 + u_2^2 + x_1 + \lambda_1 u_1 + \lambda_2 (x_1 + u_2).\end{aligned}$$

Slijede kanonske jednadžbe optimalnog upravljanja:

$$-\frac{\partial H}{\partial x_1} = -(1 - \lambda_2) = -1 - \lambda_2 = \dot{\lambda}_1,$$

$$-\frac{\partial H}{\partial x_2} = 0 = \dot{\lambda}_2,$$

$$\frac{\partial H}{\partial u_1} = 2u_1 + \lambda_1 = 0 \rightarrow u_1 = -\frac{1}{2}\lambda_1,$$

$$\frac{\partial H}{\partial u_2} = 2u_2 + \lambda_2 = 0 \rightarrow u_2 = -\frac{1}{2}\lambda_2.$$

Uvrštavanjem u_1 i u_2 u polazne jednadžbe za \dot{x}_1 i \dot{x}_2 :

$$\dot{x}_1 = -\frac{1}{2}\lambda_1,$$

$$\dot{x}_2 = x_1 - \frac{1}{2}\lambda_2.$$

Rješavajući prve dvije kanonske jednadžbe optimalnog upravljanja

$$\dot{\lambda}_1 = -1 - \lambda_2,$$

$$\dot{\lambda}_2 = 0,$$

uz rubne uvjete

$$\lambda_1(T) = \frac{\partial E}{\partial x_1(T)} = a,$$

$$\lambda_2(T) = \frac{\partial E}{\partial x_2(T)} = b.$$

Dolazi se do dijela rješavanja druge kanonske jednadžbe direktnim integriranjem s rubnim uvjetom $\lambda_2(1) = b$ iz čega slijedi

$$\lambda_2 = b.$$

Dalje imamo

$$\dot{\lambda}_1 = -1 - b,$$

$$\lambda_1(t) = \int (-1 - b) dt = -t - bt + C,$$

uz rubni uvjet

$$\lambda_1(1) = a \rightarrow -1 - b + C = a \rightarrow C = a + b + 1,$$

$$\lambda_1(t) = -(1 + b)t + a + b + 1.$$

λ_1 i λ_2 uvrštavamo u izraze za u_1 i u_2 te \dot{x}_1 i \dot{x}_2 pa rješavamo

$$u_1 = -\frac{1}{2}\lambda_1 = (1 + b)\frac{t}{2} - \frac{1}{2}(1 + a + b),$$

$$u_2 = -\frac{1}{2}\lambda_2 = -\frac{b}{2}.$$

$$\dot{x}_1 = -\frac{1}{2}\lambda_1 = \frac{1}{2}(1 + b)t - \frac{1}{2}(a + b + 1),$$

se integrira uz početni uvjet $x_1(0)=0$

$$x_1 = (1 + b)\frac{t^2}{4} - (1 + a + b)\frac{t}{2} + C,$$

$$0 = (1 + b)\frac{0^2}{4} - (1 + a + b)\frac{0}{2} + C \rightarrow C = 0,$$

$$\dot{x}_2 = (1 + b)\frac{t^2}{4} - (1 + a + b)\frac{t}{2} - \frac{1}{2}b,$$

zamjenom prva dva člana za x_1 te u trećem članu b za λ_2 i integriranjem uz početni uvjet $x_2(0)=0$

$$x_2 = (1 + b) \frac{t^3}{12} - (1 + a + b) \frac{t^2}{4} - \frac{b}{2}t + C,$$

$$0 = (1 + b) \frac{0^3}{12} - (1 + a + b) \frac{0^2}{4} - \frac{b}{2}0 + C \rightarrow C = 0,$$

$$x_2 = (1 + b) \frac{t^3}{12} - (1 + a + b) \frac{t^2}{4} - \frac{b}{2}t,$$

tako su se dobila analitička rješenja ovog zadatka.

Slijedi provedba rješenja koja se provodi primjenom MATLAB-a s korištenjem funkcije *fminunc*. Korištene metode bit će Eulerova, Runge-Kutta metoda 4. reda i Adamsova metoda 4. reda. Usporedit će se rješenja za različite vrijednosti konstanti a i b . Također će se usporediti točnost rješenja s obzirom na analitičko i brzine izvršavanja za tri različite metode diskretizacije.

Za ovdje najjednostavniju metodu, Eulerovu, primijenit će se i rješavanje korištenjem funkcije *fmincon* te primjermom CasADi-a.

Kako bi se riješio problem s računanjem integrala, uvodi se nova varijabla stanja. Tako se dobiva dodatna diferencijalna jednadžba

$$\dot{x}_3(t) = u_1^2 + u_2^2 + x_1.$$

U svakoj od slijedećih metoda korištena je ova dodatna diferencijalna jednadžba.

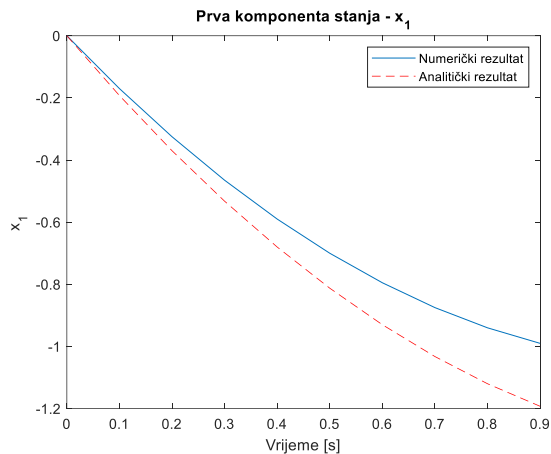
5.1.1. Analiza rješenja korištenjem MATLAB-ove funkcije *fminunc*

Primjenom MATLAB-ove funkcije *fminunc* napravljena su rješenja za sve tri gore navedene metode. U svakom od sljedećih poglavlja bit će prikazani rezultati za parametre koji su dobiveni i oni koji predstavljaju analitičko rješenje.

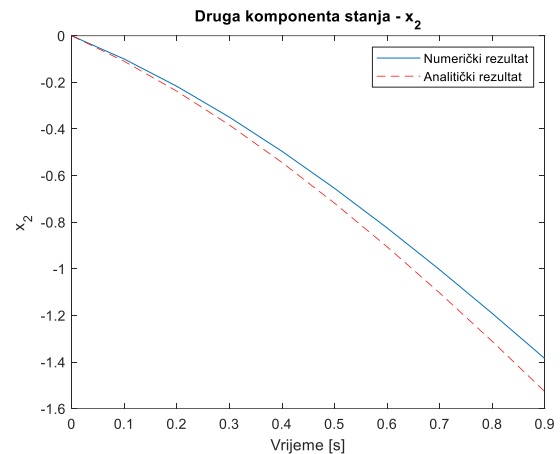
5.1.1.1. Eulerova metoda

Počelo se s odabirom konstanti a i b koje su ovdje uzete kao 1 i 2 te je nakon izračun išao prvo s najmanjim brojem vremenskih intervala N , odnosno s početkom od $N=10$ (10^1) te se povećavao za jedan red veličine.

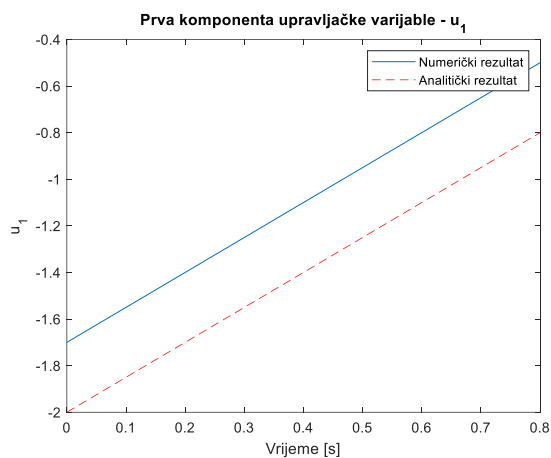
Dobiveni su grafovi prikazani su na slikama od 5.1 do 5.4. Vide se i dvije linije od kojih plava predstavlja dobiveno rješenje s obzirom na zadane uvjete, a crvena, isprekidana analitičko rješenje.



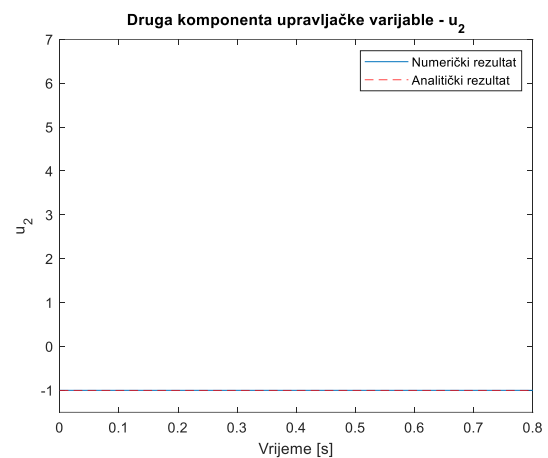
Slika 5.1. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po Euleru



Slika 5.2. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po Euleru



Slika 5.3. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po Euleru



Slika 5.4. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po Euleru

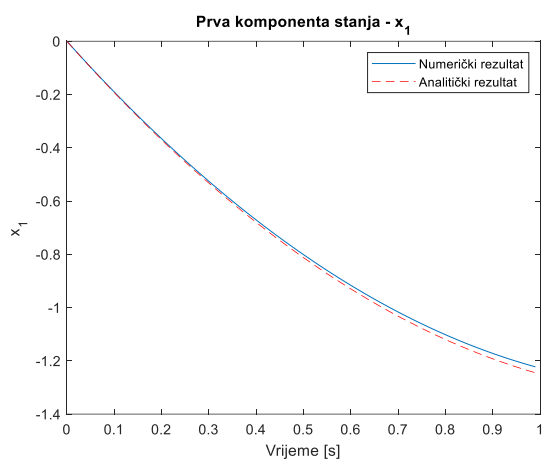
Za svako od navedenih rješenje jasno se okom vidi velika greška, osim kod parametra u_2 . To se događa zbog malog broja vremenskih intervala, no rješenje je približno te iz razloga malog broja N vrijeme izvršavanja je jako malo i iznosi 0,69 s.

Kako ne bi greške bile prikazane samo grafički, brojevima se u tablici 5.1 može vidjeti svako pojedino odstupanje.

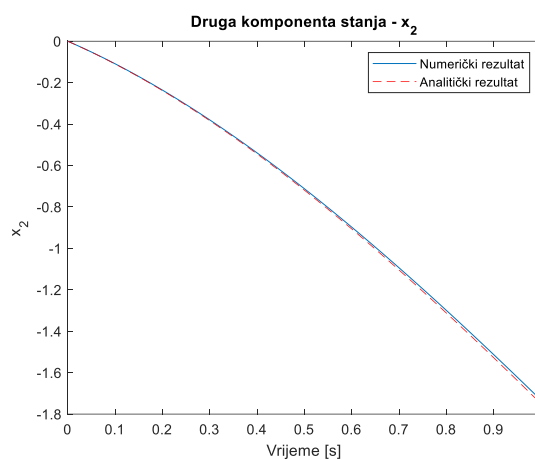
Tablica 5.1. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=10$ i korištenjem fminunc funkcije

Eulerova metoda; fminunc; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10$	0,69 s	Odstupanje za x_1 : 0.18595
		Odstupanje za x_2 : 0.13148
		Odstupanje za u_1 : 0.27273
		Odstupanje za u_2 : 1.5832e-07

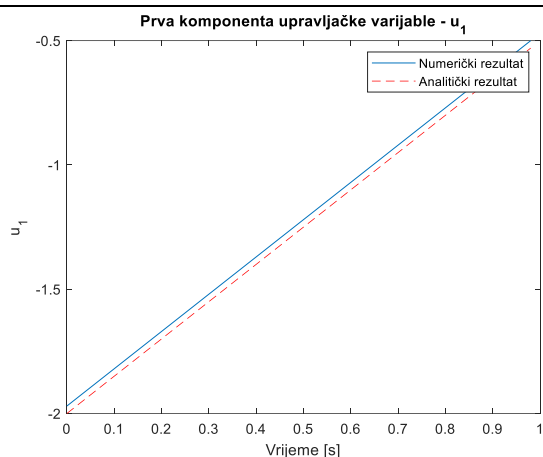
Za poboljšavanje rezultata i približavanje k analitičkom rješenju povećat će se broj vremenskih intervala za jedan red veličine na $N=100$ (10^2).



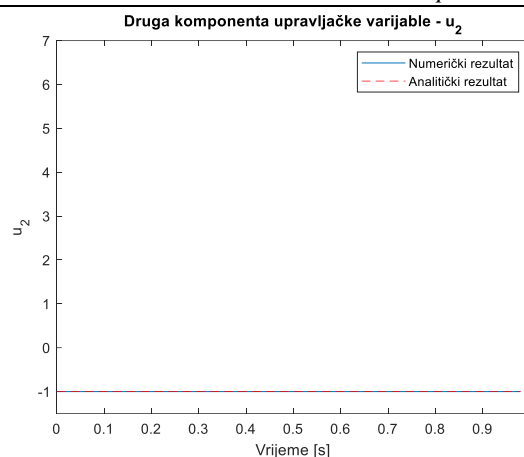
Slika 5.5. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po Euleru



Slika 5.6. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po Euleru



Slika 5.7. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po Euleru



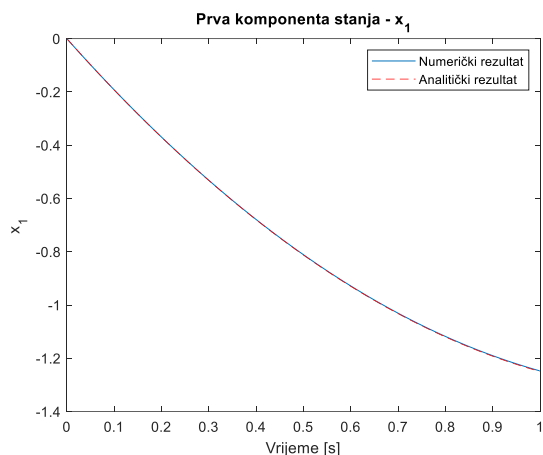
Slika 5.8. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po Euleru

Već samim pogledom na grafove na slikama od 5.5 do 5.8. vidi se znatno poboljšanje rezultata i njihovo približavanje željenom cilju. Ovdje vrijeme izvršavanja se malo povećalo, no neznatno te se proračun razvio nakon 1,069 s, a dobivena rješenja su bolja za jedan red veličine što prikazuje tablica 5.2.

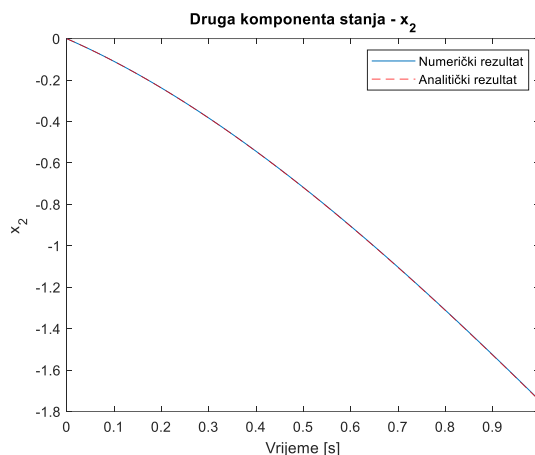
Tablica 5.2. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=100$ i korištenjem fminunc funkcije

Eulerova metoda; fminunc; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10^2$	1,069 s	Odstupanje za x_1 : 0.022058
		Odstupanje za x_2 : 0.016989
		Odstupanje za u_1 : 0.029708
		Odstupanje za u_2 : 2.603e-06

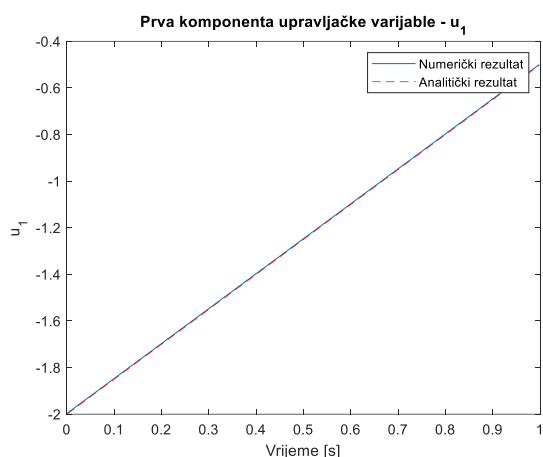
Ponovno se radi povećanje broja vremenskih intervala, ovaj puta na $N=1000$ (10^3). Vrijeme trajanja znatno se povećalo u odnosu na prva dva primjera Eulerove metode te iznosi 35,133 s. Na slikama od Slika 5.9 do Slika 5.12 prikazana su grafička rješenja.



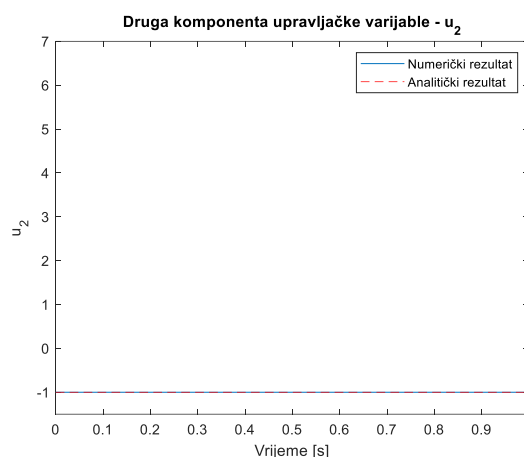
Slika 5.9. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po Euleru



Slika 5.10. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po Euleru



Slika 5.11. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po Euleru



Slika 5.12. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po Euleru

Brojčano se ona mogu vidjeti u tablici 5.3:

Tablica 5.3. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=1000$ i korištenjem fminunc funkcije

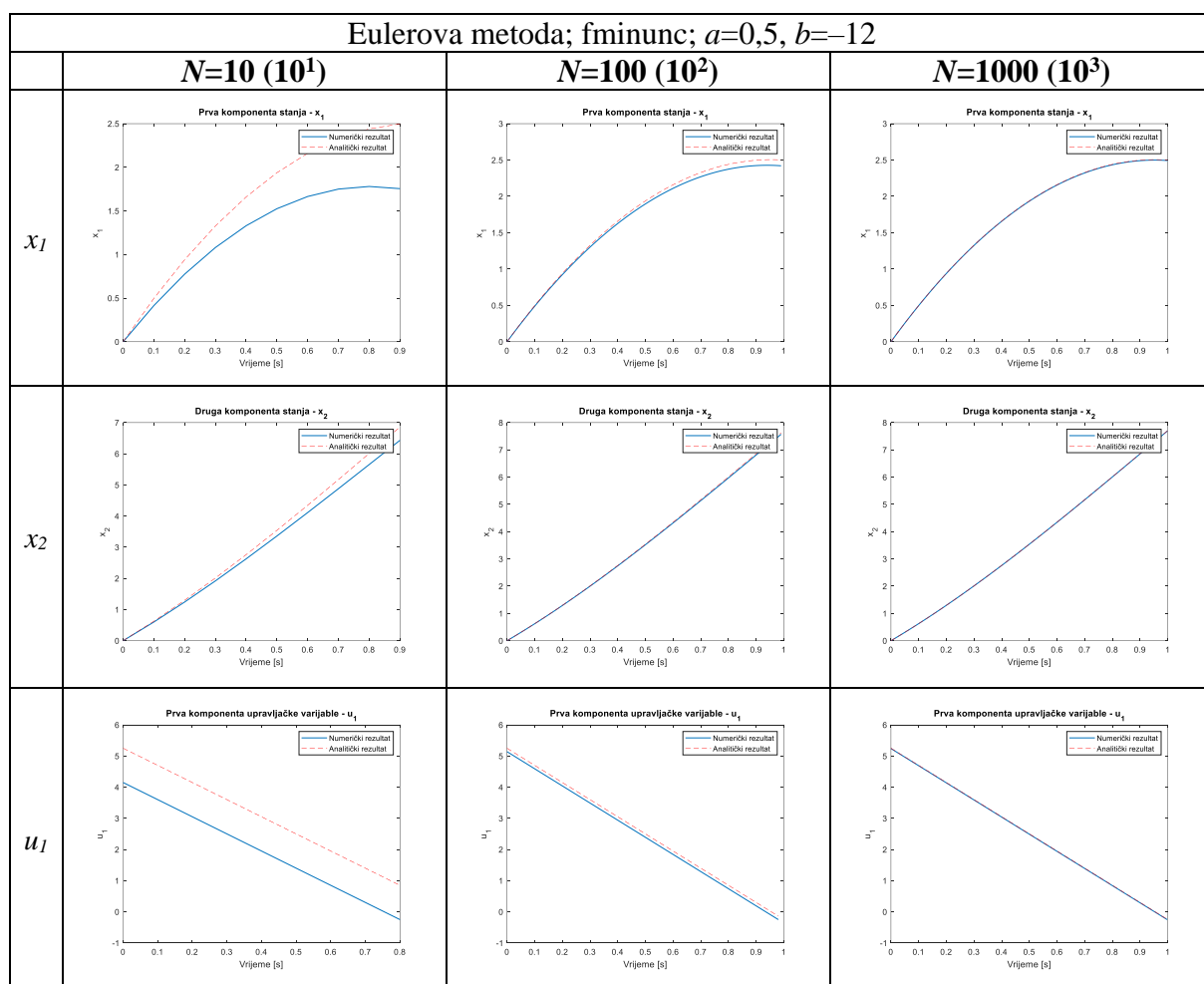
Eulerova metoda; fminunc; $a=1, b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10^3$	35,133 s	Odstupanje za x_1 : 0.0022306
		Odstupanje za x_2 : 0.0017169
		Odstupanje za u_1 : 0.0030743
		Odstupanje za u_2 : 2.8691e-05

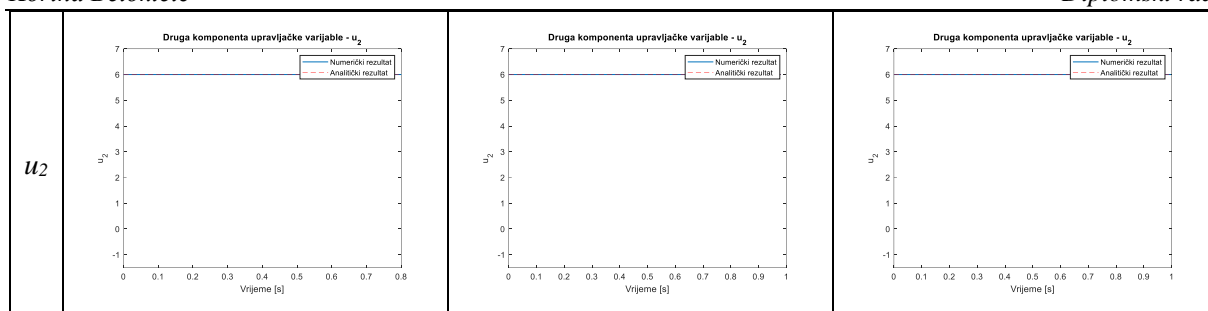
Greška u odnosu na analitičko rješenje smanjila se za još jedan stupanj veličine povećanjem broja vremenskih intervala. Grafički rješenja izgledaju praktički identična, no sve pretpostavke i napravljeni primjeri ukazuju da će ona biti još bolje ako se poveća N . Ipak, njegovim

povećanjem za još jedan stupanj veličine vrijeme izvršavanja postaje predugo. Takvo dugo vrijeme nije obećavajuće te se kao optimalno može uzeti treći primjer. Budući da je ovaj zadatak jednostavan, ovo vrijeme je prosječno što se tiče rješavanja ako nemamo druge izračune. Kada bi se na ovaj problem naslonili drugi, teži i veći problemi, vjerojatno je da bi se rješenja iz drugog pokušaja uzela kao relevantna s obzirom na puno kraće vrijeme izvršavanja.

Za istu metodu primijenit će se drugi parametri konstanti a i b , ovdje će one biti $a=0,5$, $b=-12$. Rješenja će se raditi za iste N -ove kao i u gornjim primjerima, a rezultati su prikazani u tablici 5.4.

Tablica 5.4. Analiza odstupanja rješenja od analitičkog za zadane parametre po Eulerovoj metodi





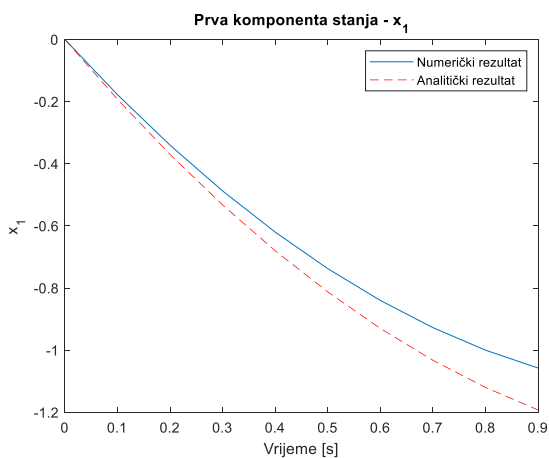
Vremena izvršavanja su redom 0,709 s, 2,619 s, 224,555 s. Rješenja za odstupanja se u većini slučajeva poboljšavaju za jedan red veličine s povećanjem N . Rješenja zadnjeg slučaja su najutjecajnija, no vrijeme izvršavanja je oko 100 puta veće.

Može se vidjeti kako inicijalno postavljanje konstanti a i b dosta utječe na vrijeme izvršavanja zadatka.

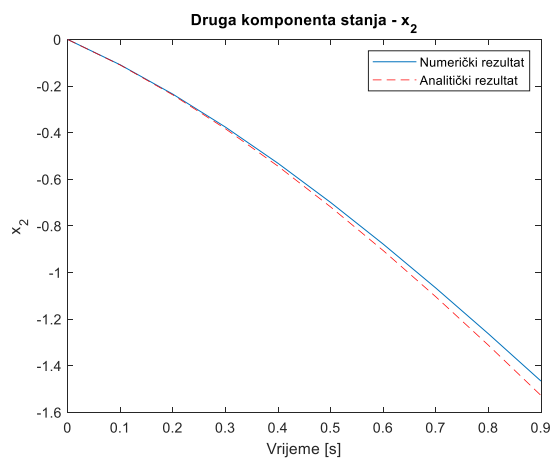
5.1.1.2. Runge – Kutta metoda

Počelo se s odabirom konstanti a i b koje su ovdje uzete kao 1 i 2 te je nakon izračun išao prvo s najmanjim brojem vremenskih intervala N , odnosno s početkom od $N=10$ (10^1) te se povećavao za jedan red veličine.

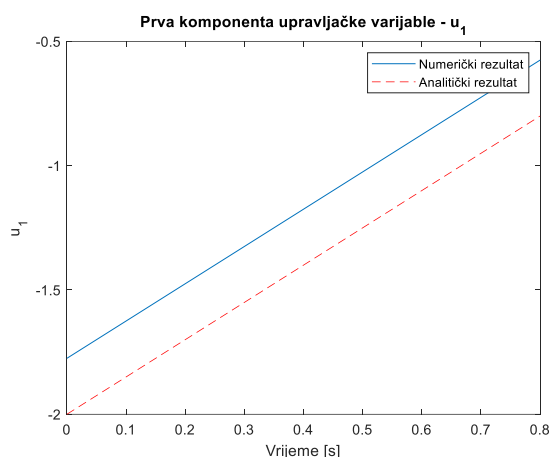
Dobiveni su grafovi prikazani na slikama od 5.13 do 5.16. Vide se i dvije linije od kojih plava predstavlja dobiveno rješenje s obzirom na zadane uvjete, a crvena, isprekidana analitičko rješenje.



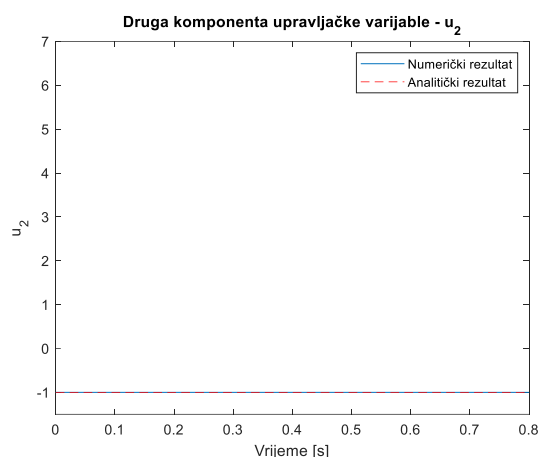
Slika 5.13. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po RK4



Slika 5.14. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po RK4



Slika 5.15. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po RK4



Slika 5.16. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po RK4

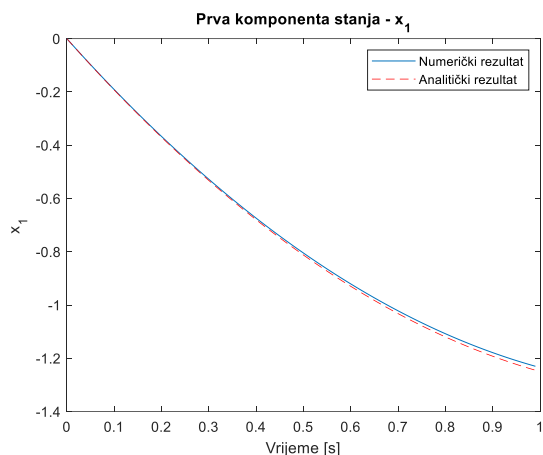
Za svako od navedenih rješenje jasno se okom vidi velika greška, osim kod parametra u_2 . To se događa zbog malog broja vremenskih intervala, no rješenje je približno te iz razloga malog broja N , vrijeme izvršavanja je jako malo te iznosi 0,592 s te nešto manje nego kod Eulerove metode za istu vrijednost N , no ipak neosjetno.

Kako ne bi greške bile prikazane samo grafički, brojevima se u tablici 5.5 može vidjeti svako pojedino odstupanje.

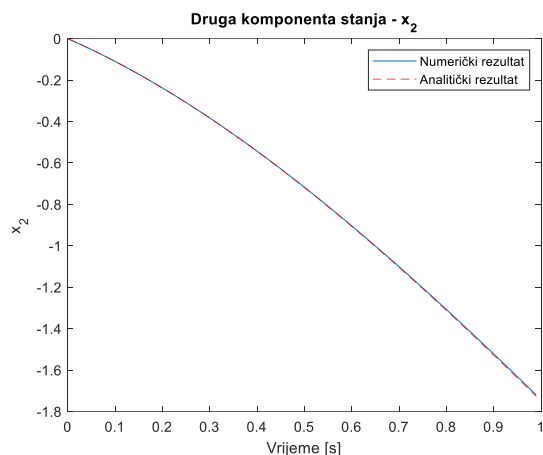
Tablica 5.5. Brojčane vrijednosti odstupanja po RK4 metodi za $N=10$ i korištenjem fminunc funkcije

Runge – Kutta metoda; fminunc; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10$	0,592 s	Odstupanje za x_1 : 0.12397
		Odstupanje za x_2 : 0.057288
		Odstupanje za u_1 : 0.20455
		Odstupanje za u_2 : 1.5832e-07

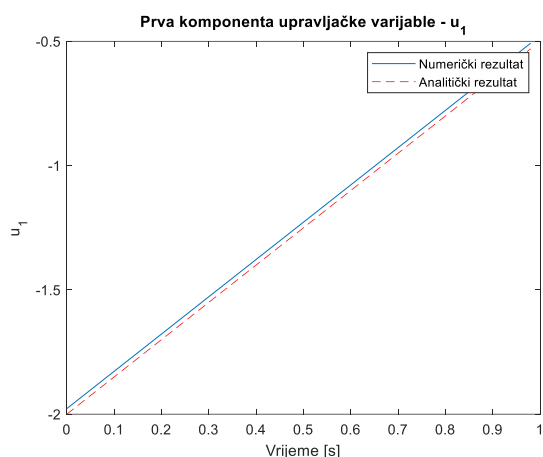
Za poboljšavanje rezultata i približavanje k analitičkom rješenju povećat će se broj vremenskih intervala za jedan red veličine na $N=100$ (10^2).



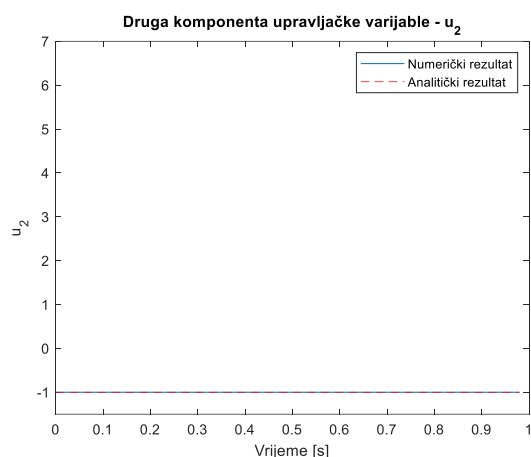
Slika 5.17. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po RK4



Slika 5.18. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po RK4



Slika 5.19. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po RK4



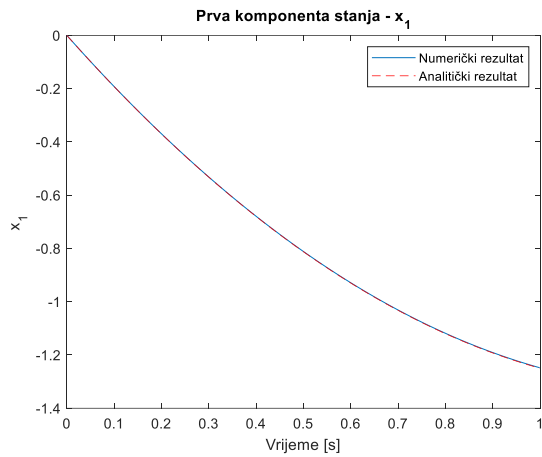
Slika 5.20. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po RK4

Već samim pogledom na grafove na slikama od 5.17 do 5.20 vidi se znatno poboljšanje rezultata i njihovo približavanje željenom cilju. Ovdje vrijeme izvršavanja se malo povećalo, no neznatno te se proračun razvio nakon 1,707 s, a dobivena rješenja su bolja za jedan red veličine što se može vidjeti u tablici 5.6.

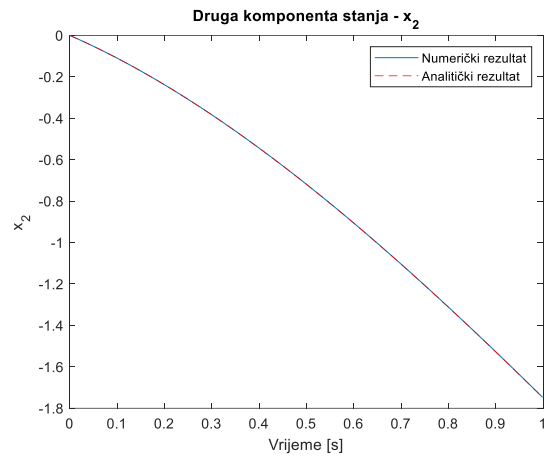
Tablica 5.6. Brojčane vrijednosti odstupanja po RK4 metodi za $N=100$ i korištenjem fminunc funkcije

Runge – Kutta metoda; fminunc; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10^2$	1,707 s	Odstupanje za x_1 : 0.014703
		Odstupanje za x_2 : 0.0072902
		Odstupanje za u_1 : 0.022282
		Odstupanje za u_2 : 1.1971e-06

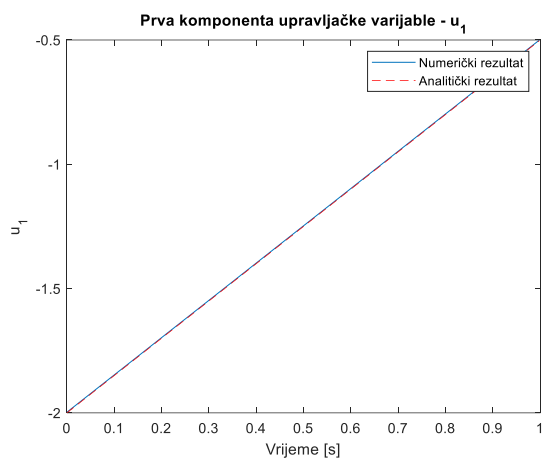
Ponovno se radi povećanje broja vremenskih intervala, ovaj puta na $N=1000$ (10^3). Vrijeme trajanja znatno se povećalo u odnosu na prva dva primjera Eulerove metode te iznosi 114,92 s. Na slikama od 5.21 do 5.24 prikazana su grafička rješenja.



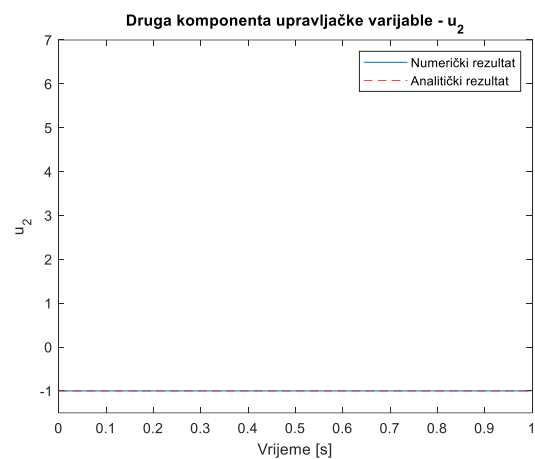
Slika 5.21. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po RK4



Slika 5.22. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po RK4



Slika 5.23. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po RK4



Slika 5.24. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po RK4

Brojčano ona mogu vidjeti u tablici 5.7:

Tablica 5.7. Brojčane vrijednosti odstupanja po RK4 metodi za $N=1000$ i korištenjem fminunc funkcije

Runge – Kutta metoda; fminunc; $a=1, b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10^3$	114,92 s	Odstupanje za x_1 : 0.0014863
		Odstupanje za x_2 : 0.0007319
		Odstupanje za u_1 : 0.0023051
		Odstupanje za u_2 : 2.8691e-05

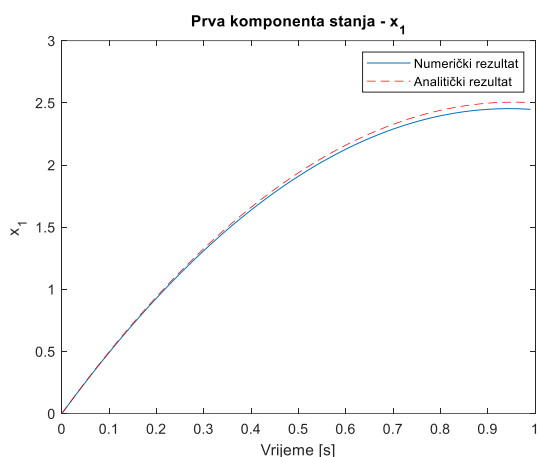
Greška u odnosu na analitičko rješenje smanjila se za još jedan stupanj veličine povećanjem broja vremenskih intervala. Grafički rješenja izgledaju praktički identična, no sve pretpostavke i napravljeni primjeri ukazuju da će ona biti još bolje ako se poveća N . Ipak, njegovim povećanjem za još jedan stupanj veličine vrijeme izvršavanja postaje predugo, isto kao i kod Eulerove metode. Takvo dugo vrijeme nije obećavajuće te se kao optimalno može uzeti treći primjer. Budući da je ovaj zadatak jednostavan, ovo vrijeme je prosječno što se tiče rješavanja ako nemamo druge izračune. Kada bi se na ovaj problem naslonili drugi, teži i veći problemi, vjerojatno je da bi se rješenja iz drugog pokušaja uzela kao relevantna s obzirom na puno kraće vrijeme izvršavanja.

Ovdje će se još riješiti primjer s drugim vrijednostima konstanti a i b .

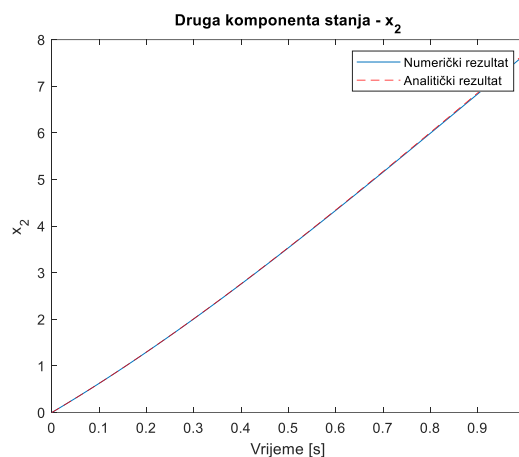
Za slučaj kad je $N=100$ i vrijednosti konstanti su $a=0,5$, $b=-12$ prikazani su grafovi na slikama od 5.25 do 5.28. Isti slučaj s istim parametrima je izračunat gore prema Eulerovoj metodi, a njihovu usporedbu prikazuje tablica 5.8.

Tablica 5.8. Usporedba Eulerove i RK4 metode

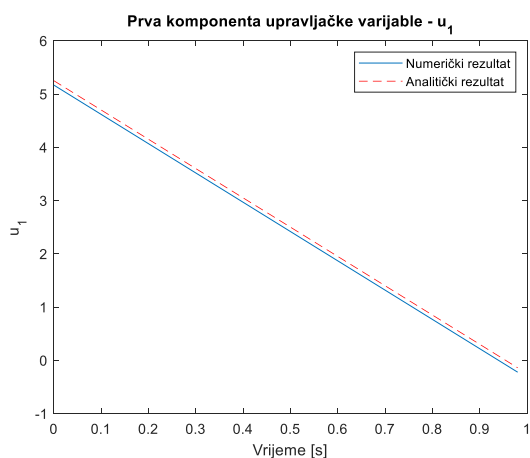
	Eulerova metoda	Runge – Kutta metoda
Vrijeme izvršavanja:	2,619 s	5,663 s
Odstupanje od analitičkog za x_1:	0.080864	0,053885
Odstupanje od analitičkog za x_2:	0.052063	0,026723
Odstupanje od analitičkog za u_1:	0,10897	0,081748
Odstupanje od analitičkog za u_2:	7,8533e-06	1,2496e-05



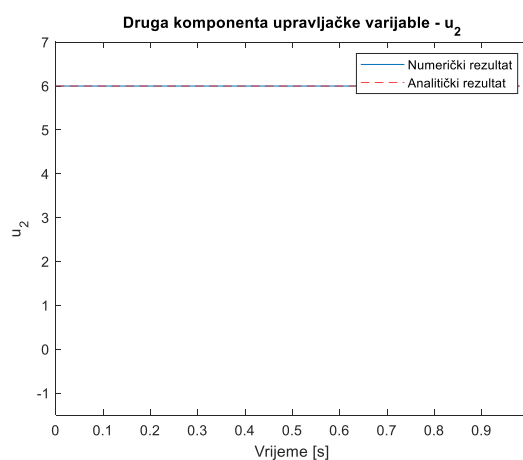
Slika 5.25. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama



Slika 5.26. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama



Slika 5.27. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama



Slika 5.28. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po RK4 s novim konstantama

Kako bi se vidjele točne razlike između ove dvije metode poslužit će tablica 5.8. Osim grafičkih rješenja koja se mogu vidjeti za svaku od metoda, ovdje je usporedba u vremenu izvršavanja i točnosti podataka.

Ovdje se postavlja pitanje optimalnosti. Koja metoda je optimalna za rješenje ovog sustava. To većinom ovisi o specifičnim zahtjevima i uvjetima rada i zadatka pa je važno izabrati metodu koja najbolje zadovoljava prioritete. Je li to brzo izvršavanje i vremenska učinkovitost uz manju preciznosti ili je važnije točnost rezultata unatoč većem vremenu izvršavanja.

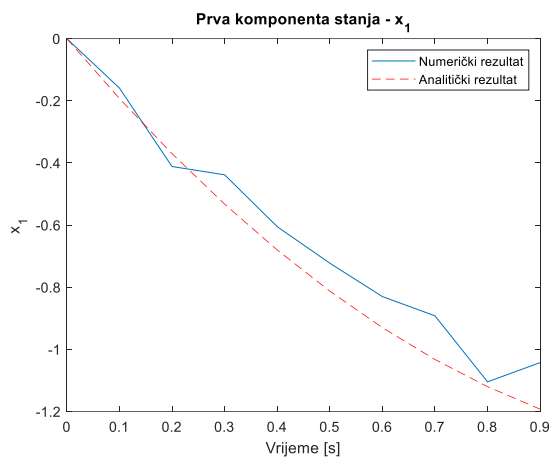
U ovom konkretnom primjeru, vrijeme izvršavanja po RK4 dvostruko je veće od Eulerove metode, no ti brojevi nisu veliki pa i dulje vrijeme izvršavanja (koje nije predugo, no u usporedbi dvostruko veće) dat će bolje rezultate i moći će se dobiti precizniji i točniji rezultati.

5.1.1.3. Adamsova metoda

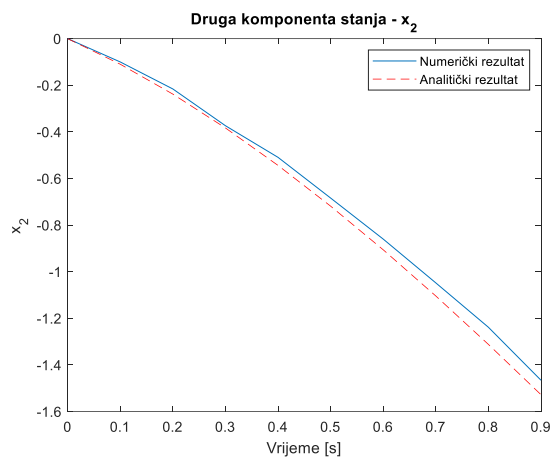
Ostala je još jedna metoda, Adamsova. Također se počelo s odabirom konstanti a i b koje su ovdje uzete kao 1 i 2 te je nakon izračun išao prvo s najmanjim brojem vremenskih intervala N , odnosno s početkom od $N=10$ (10^1) te se povećavao za jedan red veličine.

Kod Adams – Bashforthove metode 4.reda javlja se velika akumulacija pogreške na kraju intervala koja se dodatno multiplicira i na početku intervala u sljedećoj iteraciji optimizacijskog problema. Kako bi se ublažio ovaj problem, u ovom primjeru smanjio se red metode te su rješenja dobivena Adams – Bashforth-ovom metodom 3. reda.

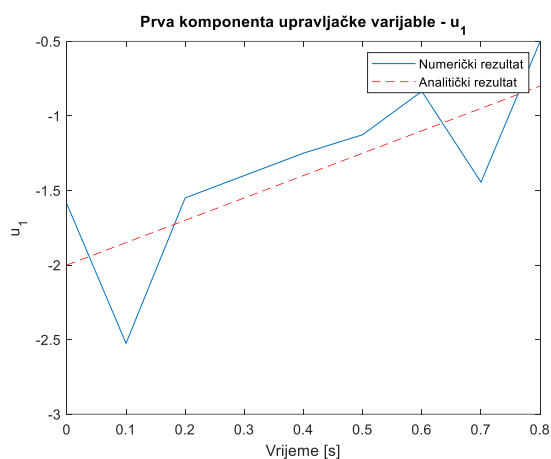
Dobiveni su grafovi prikazani su za svako povećanje N . Redom su prikazani x_1 , x_2 , u_1 i u_2 . Vide se i dvije linije od kojih plava predstavlja dobiveno rješenje s obzirom na zadane uvjete, a crvena analitičko rješenje.



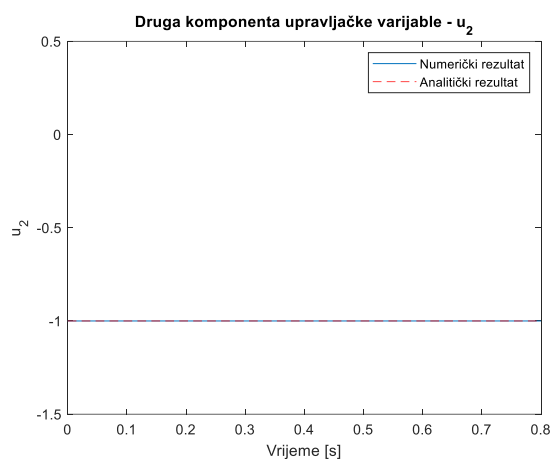
Slika 5.29. Odstupanje x_1 prema analitičkom rješenju za $N=10$ po Adamsu



Slika 5.30. Odstupanje x_2 prema analitičkom rješenju za $N=10$ po Adamsu



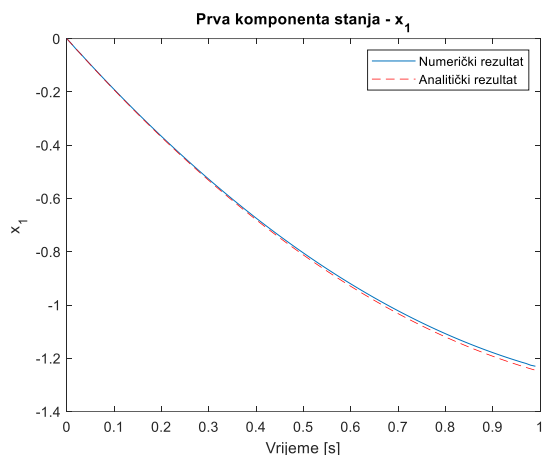
Slika 5.31. Odstupanje u_1 prema analitičkom rješenju za $N=10$ po Adamsu



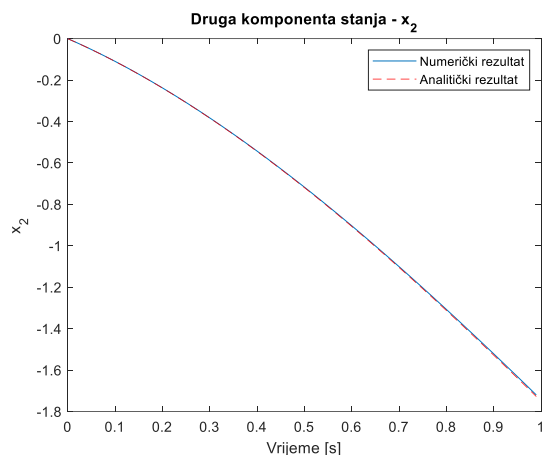
Slika 5.32. Odstupanje u_2 prema analitičkom rješenju za $N=10$ po Adamsu

Rješenja su prikazana na slikama od 5.29 do 5.32, a za svako od navedenih rješenja jasno se okom vidi velika greška, osim kod parametra u_2 . To se događa zbog malog broja vremenskih intervala, no rješenje je približno te iz razloga malog broja N , vrijeme izvršavanja je malo te iznosi 0,776 s.

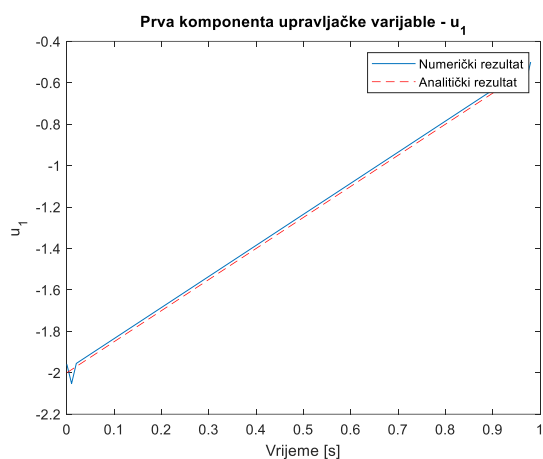
Povećanjem broja intervala pokušat će se maknuti ta pogreška. Za znatno bolja rješenja povećat će se broj vremenskih intervala za jedan red veličine na $N=100$ (10^2).



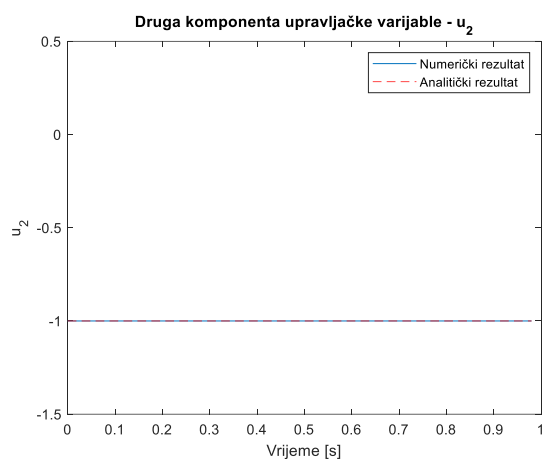
Slika 5.33. Odstupanje x_1 prema analitičkom rješenju za $N=100$ po Adamsu



Slika 5.34. Odstupanje x_2 prema analitičkom rješenju za $N=100$ po Adamsu



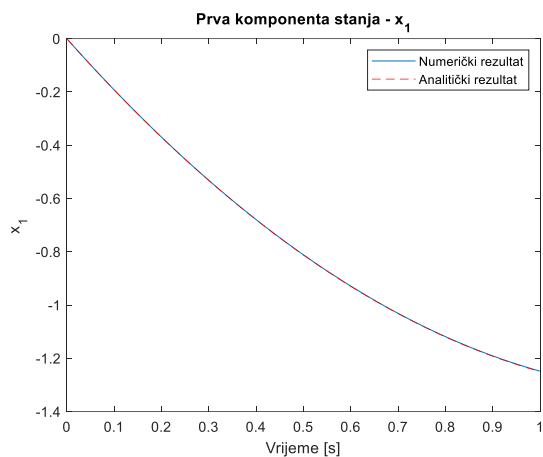
Slika 5.35. Odstupanje u_1 prema analitičkom rješenju za $N=100$ po Adamsu



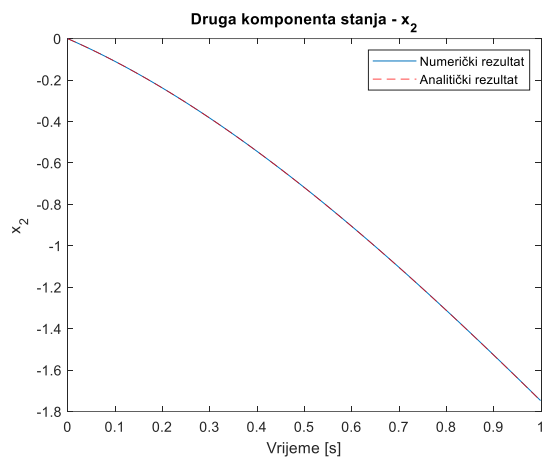
Slika 5.36. Odstupanje u_2 prema analitičkom rješenju za $N=100$ po Adamsu

Već samim pogledom na grafove na slikama od 5.33 do 5.36 vidi se poboljšanje rezultata i njihovo približavanje željenom cilju. Ovdje vrijeme izvršavanja se malo povećalo, no neznatno te se proračun razvio nakon 1,94 s, a dobivena rješenja su bolja za jedan red veličine.

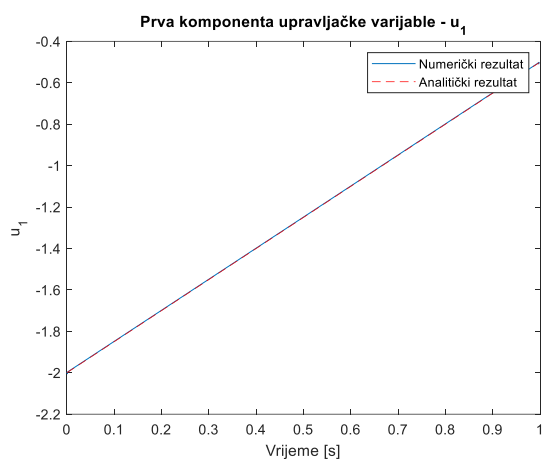
Ponovno se povećava broj vremenskih intervala, $N=1000$ (10^3). A rezultati su prikazani grafovima na slikama od 5.37 do 5.40 kako bi se vidjela odstupanja od analitičkog rješenja.



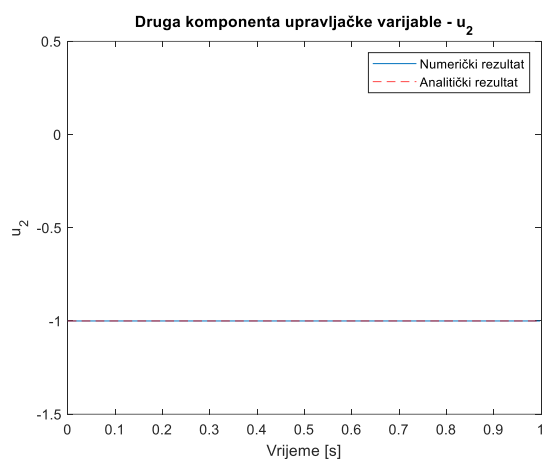
Slika 5.37. Odstupanje x_1 prema analitičkom rješenju za $N=1000$ po Adamsu



Slika 5.38. Odstupanje x_2 prema analitičkom rješenju za $N=1000$ po Adamsu



Slika 5.39. Odstupanje u_1 prema analitičkom rješenju za $N=1000$ po Adamsu



Slika 5.40. Odstupanje u_2 prema analitičkom rješenju za $N=1000$ po Adamsu

Brojčani rezultati prikazani su samo za slučaj računanja s najvećim N -om i prikazani su u tablici 5.9. Kao i kod prethodne dvije metode, povećanjem broja vremenskih intervala za jedan red veličine, vrijeme izvršavanja predugo je za dobivanje rezultata te se iz tog razloga ne uzima kao relevantno u procesu optimizacije.

Tablica 5.9. Brojčane vrijednosti odstupanja po Adamsovoj metodi za $N=1000$ i korištenjem `fminunc` funkcije

Adamsova metoda; <code>fminunc</code> ; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10^3$	114,135 s	Odstupanje za x_1 : 0.0014593
		Odstupanje za x_2 : 0.00073135
		Odstupanje za u_1 : 0.0048408
		Odstupanje za u_2 : 0.00082115

Iako se prvotno predviđalo kako će ova metoda dati najbolje rezultate, iz priloženog se vidi da u ovom primjeru to nije bio slučaj. Optimalna metoda se pokazala Runge – Kutteova metoda 4. reda gdje su rezultati najbliži analitičkom rješenju.

MATLAB-ova funkcija *fminunc* ima dva algoritma za optimizaciju. *Quasi-newton*, što je zadana opcija i ovdje korištena. Još postoji i drugi algoritam, *trust-region*. Korištena *quasi-newton* metoda koristi se kada je *Newtonovu* metodu teško koristiti ili je računanje *Hessiana* komplicirano. Ova metoda aproksimira *Hessian* korištenjem gradijenta. „Prave“ *Newtonove* metode računale bi *Hessian* matricu direktno te se spuštaju i lociraju minimum nakon određenog broja ponavljanja.

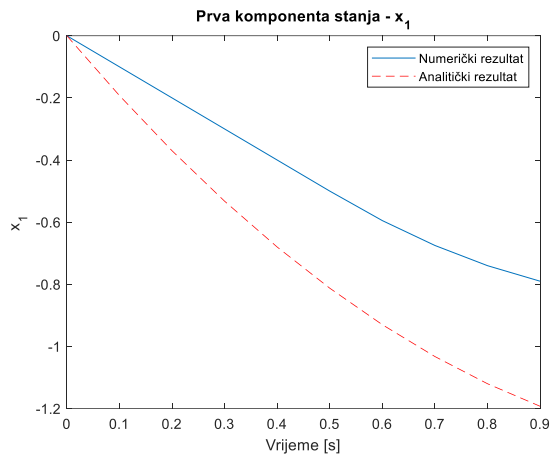
Kod *trust-region* metode se promatra problem neograničene minimizacije gdje funkcija uzima vektorske argumente i vraća skalare. Ideja je aproksimirati funkciju f s jednostavnijom funkcijom q koja utječe na ponašanje f u okruženju oko neke točke. To okruženje predstavlja *trust-region*. Ovo je još jedan način na koji bi se mogao riješiti ovaj problem i vidjeti koje razlike su u odnosu na zadanu metodu i hoće li to moći poboljšati rezultate.

5.1.2. Analiza rješenja korištenjem MATLAB-ove funkcije *fmincon*

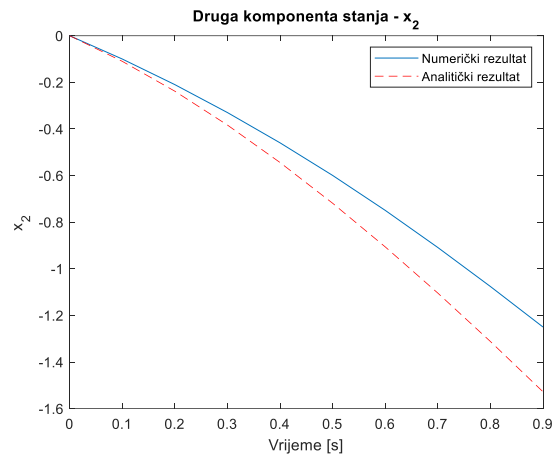
Gornji zadatak riješen je i korištenjem *fmincon* MATLAB-ovom funkcijom koja ima ograničenja. Preporuka je da se krene od onoga što je zadano u funkciji (to je također korišteno i u ovom zadatku), a to je '*interior-point*' algoritam.

Analiza se radi prema Eulerovoj metodi. Počelo se s odabirom konstanti a i b koje su ovdje uzete kao 1 i 2 te je nakon izračun išao prvo s najmanjim brojem vremenskih intervala N , odnosno s početkom od $N=10$ (10^1) te se povećavao za jedan red veličine. Ograničenja su ovdje postavljena kao granice (*lower bounds* – donje granice, lb i *upper bounds* – gornje granice, ub) za varijable upravljanja u na interval od -1 do 1. Postoje još i ograničenja linearnih jednadžbi i nejednadžbi, no one su postavljene kao prazna matrica '[]'. Dobiveni su grafovi prikazani na slikama od 5.41 do 5.44. Vide se i dvije linije od kojih plava predstavlja dobiveno rješenje s obzirom na zadane uvjete, a crvena analitičko rješenje.

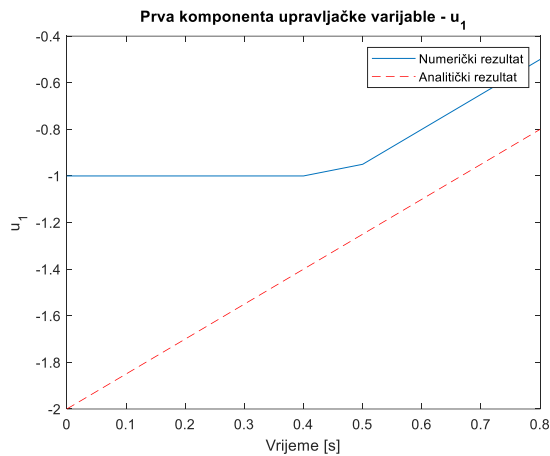
Tablica 5.10 prikazuje i brojčana odstupanja.



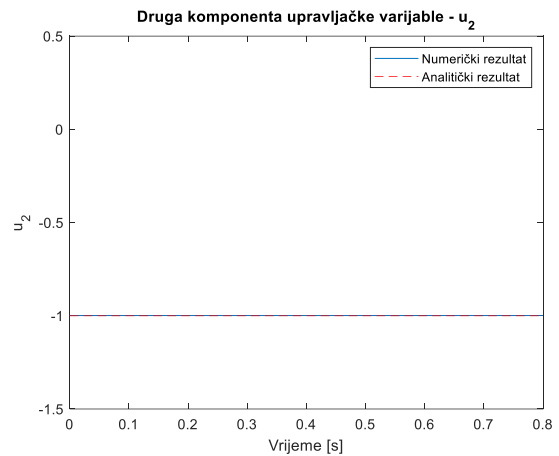
Slika 5.41. Odstupanje x_1 prema analitičkom rješenju za $N=10$ korištenjem fmincon



Slika 5.42. Odstupanje x_2 prema analitičkom rješenju za $N=10$ korištenjem fmincon



Slika 5.43. Odstupanje u_1 prema analitičkom rješenju za $N=10$ korištenjem fmincon



Slika 5.44. Odstupanje u_2 prema analitičkom rješenju za $N=10$ korištenjem fmincon

Tablica 5.10. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=10$ i korištenjem fmincon funkcije

Eulerova metoda; fmincon; $a=1, b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10$	1,66 s	Odstupanje za x_1 : 0.3967
		Odstupanje za x_2 : 0.27694
		Odstupanje za u_1 : 1
		Odstupanje za u_2 : 0.0013271

S obzirom na sve gore primjere, rezultati su očekivani. Pokušat će se povećati broj vremenskih intervala kako bi rezultati bili bolji.

Nakon 3,059 s, izvršio se zadatak gdje je $N=100$ (10^2). Rezultati se brojčano vide u tablici 5.11.

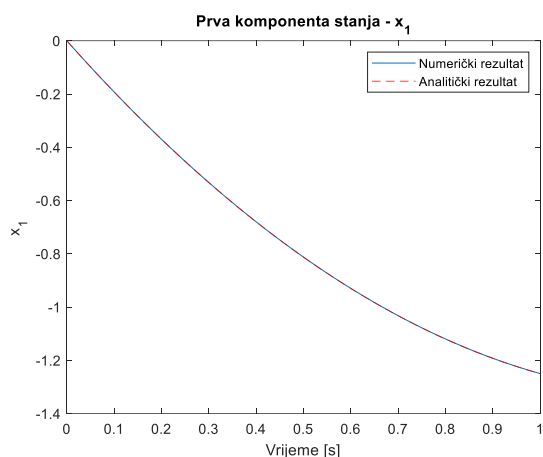
Tablica 5.11. Brojčane vrijednosti odstupanja po Eulerovoj metodi za $N=100$ i korištenjem `fmincon` funkcije

Eulerova metoda; <code>fmincon</code> ; $a=1$, $b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=100$	3,059 s	Odstupanje za x_1 : 0.4501
		Odstupanje za x_2 : 0.51432
		Odstupanje za u_1 : 1.0505
		Odstupanje za u_2 : 0.20935

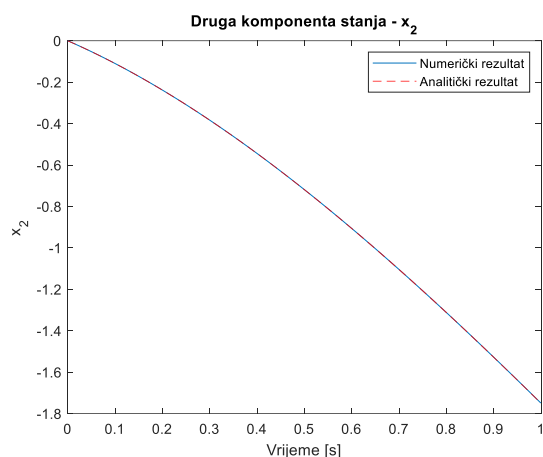
Ova rješenja su lošija nego kod manjeg broja N . Pretpostavka je da je to zbog postavke ograničenja koje ova funkcija zahtijeva. Promjena tih parametara utjecala bi i na promjenu rješenja. Bitno je znati kako te parametre postaviti i ima li zadano u zadatku ono što se treba ostvariti te na taj način prilagoditi postavke.

5.1.3. Analiza rješenja korištenjem `CasADi`-ja

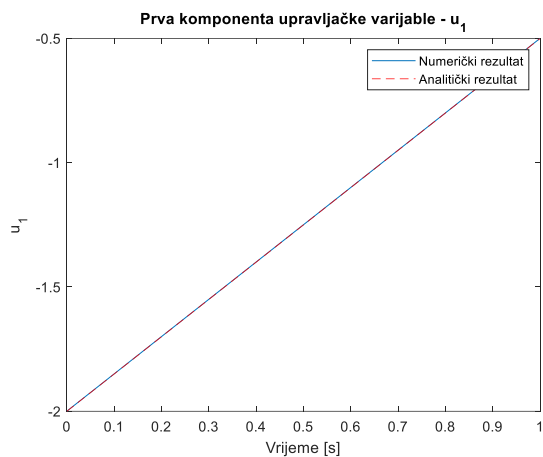
Budući da se u ovom dijelu rješava zadatak pomoću `CasADi`-ja, njegovo brzo izvršavanje (pogotovo u usporedbi s gornjim primjerima) daje nam mogućnost rješavanja s većim brojem N , tj. većim brojem vremenskih intervala. Svakim povećanjem N i ovdje se rezultati poboljšavaju. Tako kada se koristi $N=1000$ (10^3) vrijeme trajanja izvršavanja je 5,944 s što je puno manje u odnosu na primjere s istim brojem vremenskih intervala, ali bez korištenja `CasADi`-ja. Za bolje rezultate još će se povećati N na 10000 (10^4), a tada je vrijeme izvršavanja 59,964 s. Rješenja su izvođena odabirom konstanti a i b koje su ovdje uzete kao 1 i 2.



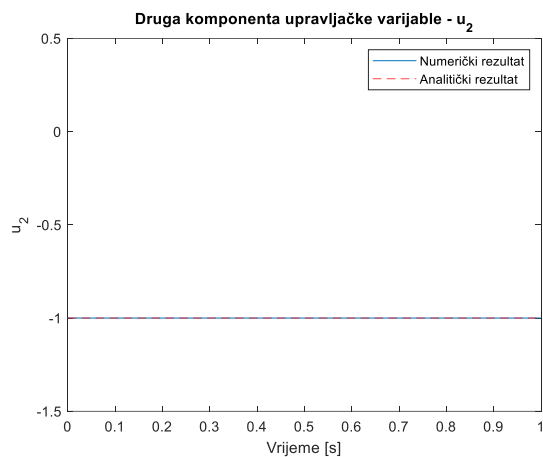
Slika 5.45. Odstupanje x_1 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja



Slika 5.46. Odstupanje x_2 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja



Slika 5.47. Odstupanje u_1 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja



Slika 5.48. Odstupanje u_2 prema analitičkom rješenju za $N=10000$ korištenjem CasADi-ja

Grafička rješenja mogu se vidjeti na slikama od 5.45 do 5.48 gdje se ona praktički preklapaju s analitičkim rješenjima. Da bi se vidjela točna odstupanja u odnosu na analitičko rješenje, poslužit će tablica 5.12.

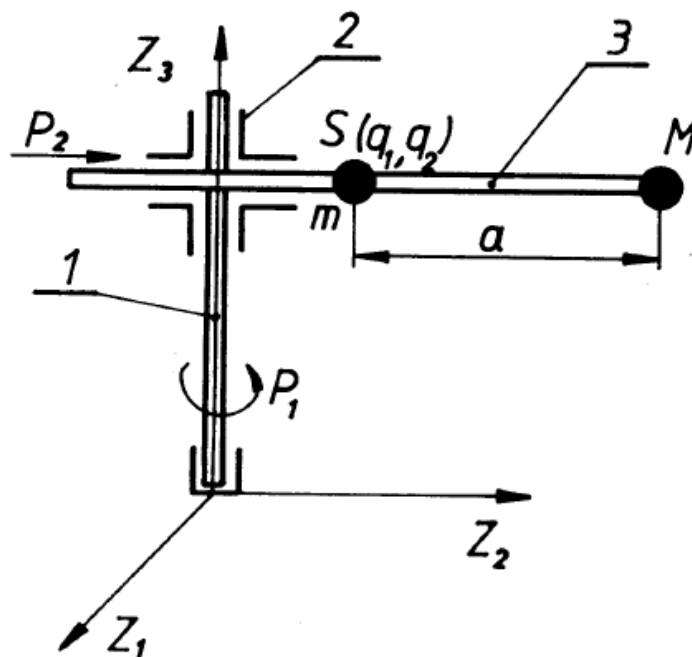
Tablica 5.12. Brojčane vrijednosti odstupanja za $N=10000$ i korištenjem CasADi-ja

CasADi; $a=1, b=2$		
Broj koraka	Vrijeme izvršavanja	Greška u odnosu na analitičko rješenje
$N=10000$	59,964 s	Odstupanje za x_1 : 0.00038099
		Odstupanje za x_2 : 0.00025889
		Odstupanje za u_1 : 0.00046336
		Odstupanje za u_2 : 7.9565e-06

Kad bi se povećao N za još jedan red veličine, vrijeme izvršavanja bi bilo dugo i nepraktično za rješavanje ovog jednostavnog primjera. Ovdje su greške najmanje pa s tog stajališta se može reći kako je CasADi izvršio „svoj posao“ i riješio zadatak s više iteracija nego prijašnji primjeri. Svakako za puno brže, a dovoljno točno rješenje moglo se uzeti i ono s jednim redom veličine manjim N .

5.2. Drugi primjer – mehanički sustav s dva stupnja slobode gibanja (jedan rotacijski, jedan translacijski)

Zadan je mehanički sustav s dva stupnja slobode gibanja, čiju shemu prikazuje slika 5.49, gdje je potrebno riješiti problem optimalne transformacije iz nekih početnih stanja u neka konačna stanja uz ograničenja na upravljačke varijable. Također, razmotriti još dodatna ograničenja tipa jednakosti, nejednakosti i slijeđenja određenih putanja.



Slika 5.49. Shema mehaničkog sustava s dva stupnja slobode gibanja [15]

Dinamički (Euler – Lagrangeov) model uzet je iz knjige [15].

Inicijalno su postavljeni parametri:

$$m_1 = 3 \text{ kg},$$

$$m_2 = 1 \text{ kg},$$

$$I_1 + I_2 = 0,1 \text{ kgm}^2,$$

$$a = 0,25 \text{ m}.$$

Ograničenja upravljačkih varijabli, za svaku minimalna i maksimalna vrijednosti:

$$U_{1max} = 9 \text{ Nm},$$

$$U_{1min} = -9 \text{ Nm},$$

$$U_{2max} = 7 \text{ N},$$

$$U_{2min} = -7 \text{ N}.$$

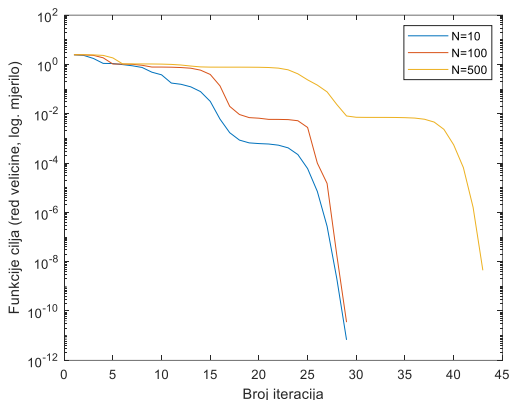
Više je različitih slučajeva koji će se razmotriti i u svima su korišteni gore navedene vrijednosti određenih parametara i varijabli.

5.2.1. Slučaj bez ograničenja korištenjem Eulerove metode

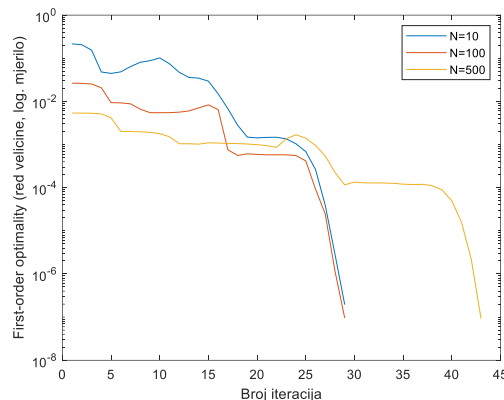
Prvi slučaj poslužit će da se vidi koji je optimalni broj vremenskih intervala za koje će se kasnije izvršavati svaki od slučajeva. Ovdje će se temeljna usporedba raditi s obzirom na algoritam koji se koristi i koje je njegovo vrijeme izvršavanja.

Počinja se po onom što je u MATLAB-u definirano kao zadano, odnosno algoritam „*interior-point*“. Kreće se s brojem vremenskih intervala $N=10$ za čije izvršavanje je bilo potrebno 2,515 s, dok je za $N=100$ bilo potrebno 11,528 s, a s $N=500$ 412,706 s. Vidi se slika 5.50 koja prikazuje konvergenciju funkcije cilja kroz iteracije. Uz to gledat ćemo graf koji prikazuje slika 5.51 koji nam daje prikaz *first-order optimality measure*, tj. mjeru koja se koristi za procjenu brzine optimizacijskog algoritma prema optimalnom rješenju, odnosno koliko je neka točka blizu optimalne te se gleda postizanje lokalnog minimuma. Kad vrijednost *first-order optimality measure* postane dovoljno mala, algoritam se zaustavlja jer smatra da je pronašao prihvatljivo rješenje. Također, onaj algoritam koji brže konvergira prema nuli smatra se efikasniji. Na grafu se vidi kako se vrijednost ove mjere mijenja kroz iteracije. Idealno bi bilo da se graf kreće prema nuli kako algoritam napreduje kroz iteracije jer bi to značilo da se algoritam sve više približava lokalnom minimumu.

Bolji rezultati se vide s većim brojem N jer je sustav bolje aproksimiran kroz vrijeme što dovodi do brže konvergencije jer su preciznije informacije o sustavu tijekom svake iteracije. Ipak, obzirom na sve navedene uvjete i vrijeme izvršavanja, ovdje se kao optimalno uzima $N=100$ s kojim će se ići dalje kroz slučajeve i promjene algoritama. U drugim slučajevima postojat će i ograničenja koja će dodatno odužiti vrijeme izvršavanja pa bi za $N=500$ ono bilo predugo.

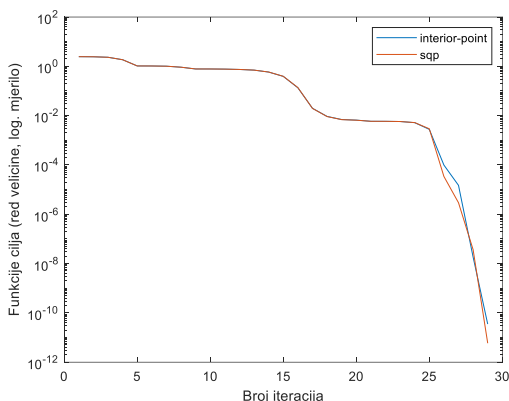


Slika 5.50. Konvergencija funkcije cilja za različite N u slučaju bez ograničenja

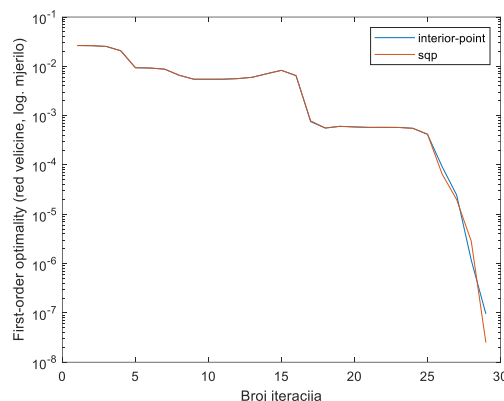


Slika 5.51. First-order optimality measure za različite N u slučaju bez ograničenja

S jednakim brojem N , vidjet će se usporedba dva korištena algoritma, *inetrrior-point* i *sqp*.

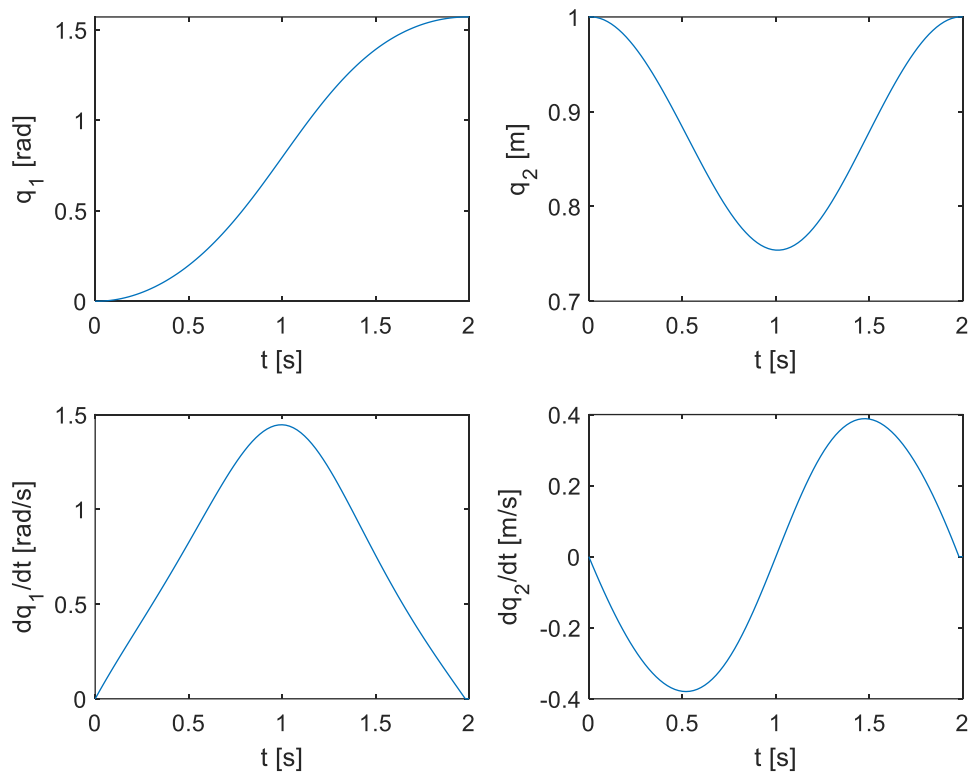


Slika 5.52. Konvergencija funkcije cilja u slučaju bez ograničenja za dva tipa algoritma



Slika 5.53. First-order optimality measure u slučaju bez ograničenja za dva tipa algoritma

Slika 5.52 i slika 5.53 prikazuju kako su ova dva algoritma za ovaj primjer poprilično slični, da im se grafovi preklapaju u većem dijelu i da je teško odrediti koji je bolji. Ostaje još pregled vremena izvršavanja da se na taj način izabere optimalan algoritam. Oko sekundu brže izvršavanje ima *inetrrior-point* u odnosu na *sqp* pa se to uzima kao optimalno. Slika 5.54 prikazuje četiri grafa korištenjem *inetrrior-point* algoritma gdje se (redom s lijeva na desno) vidi rotacijski položaj (q_1) kako se mijenja tijekom vremena te promjena translacijskog položaja (q_2) tijekom vremena. Slijedi još prikaz brzine rotacije robota kroz vrijeme $\frac{dq_1}{dt}$, tj. kutna promjena i brzinu translacije kroz vrijeme $\frac{dq_2}{dt}$.



Slika 5.54. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj bez ograničenja s interior-point algoritmom

Za prikaz upravljačkih varijabli tijekom vremena služi slika 5.55, a sama putanja robota od početne do krajnje točke slika 5.56. Početna stanja ovdje, a i u svim sljedećim slučajevima su:

$$x_1(0) = 0 \text{ rad} = \dot{q}_1,$$

$$x_2(0) = 1 \text{ m} = \dot{q}_2,$$

$$x_3(0) = 0 \frac{\text{rad}}{\text{s}} = \ddot{q}_1,$$

$$x_4(0) = 0 \frac{\text{m}}{\text{s}} = \ddot{q}_2,$$

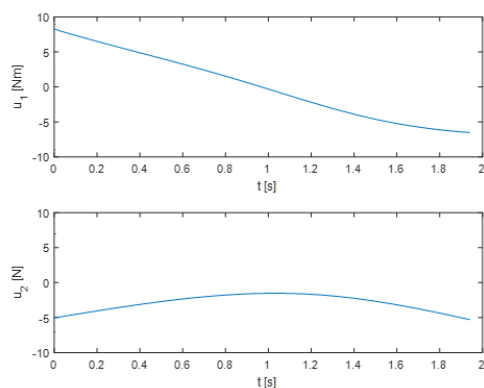
dok su konačna stanja:

$$x_1(T) = \frac{\pi}{2} \text{ rad},$$

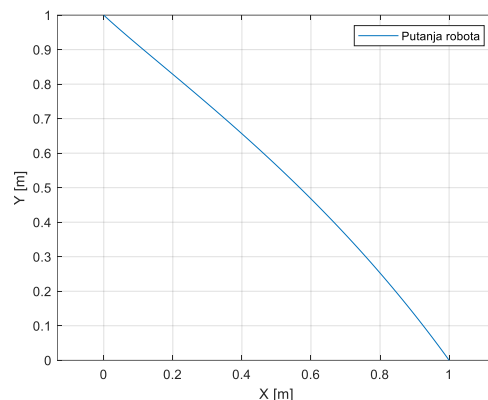
$$x_2(T) = 1 \text{ m},$$

$$x_3(T) = 0 \frac{\text{rad}}{\text{s}},$$

$$x_4(T) = 0 \frac{\text{m}}{\text{s}}.$$



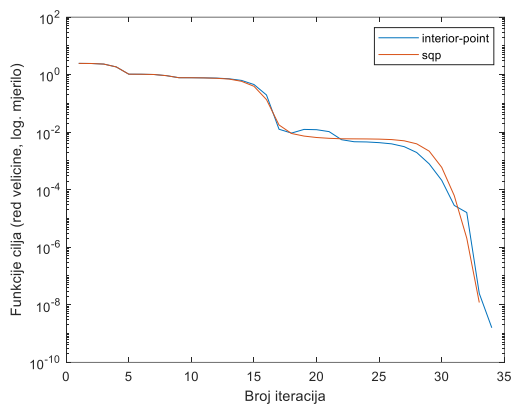
Slika 5.55. Upravljačke varijable u_1 i u_2 tijekom vremena za slučaj bez ograničenja



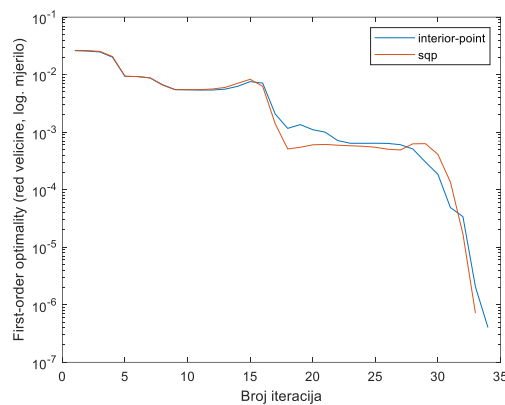
Slika 5.56. Putanja robota u prostoru za slučaj bez ograničenja

5.2.2. Slučaj uz ograničenja upravljačke varijable korištenjem Eulerove metode

Ovaj slučaj je vrlo sličan prethodnome, no uvedena su ograničenja na obje upravljačke varijable na vrijednosti koje su navedene u poglavlju 5.2.



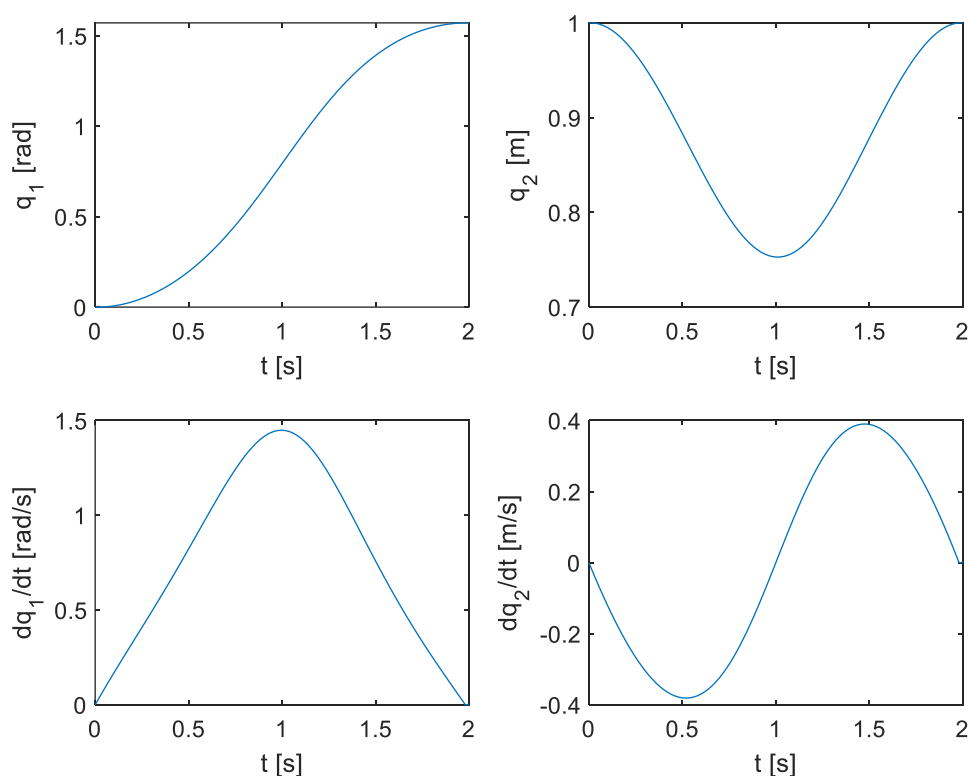
Slika 5.57. Konvergencija funkcije cilja u slučaju ograničenja upravljačkih varijabli za dva tipa algoritma



Slika 5.58. First-order optimality measure u slučaju ograničenja upravljačkih varijabli za dva tipa algoritma

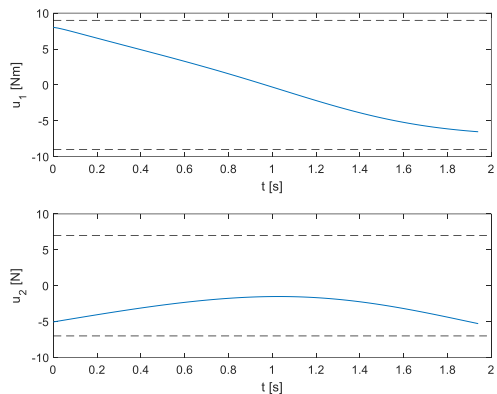
Slika 5.57 i slika 5.58 prikazuju kako su ova dva algoritma za ovaj primjer poprilično slični, da im se grafovi preklapaju u dijelovima i da je teško odrediti koji je bolji. Ostaje još pregled vremena izvršavanja da se na taj način izabere optimalan algoritam. I ovdje, oko

sekundu brže izvršavanje ima *inetrrior-point* u odnosu na *sqp* pa se to uzima kao optimalno. Slika 5.59 prikazuje četiri grafa korištenjem *inetrrior-point* algoritma gdje se (redom s lijeva na desno) vidi rotacijski položaj (q_1) kako se mijenja tijekom vremena te promjena translacijskog položaja (q_2) tijekom vremena. Slijedi još prikaz brzine rotacije robota kroz vrijeme $\frac{dq_1}{dt}$, tj. kutna promjena i brzinu translacije kroz vrijeme $\frac{dq_2}{dt}$.

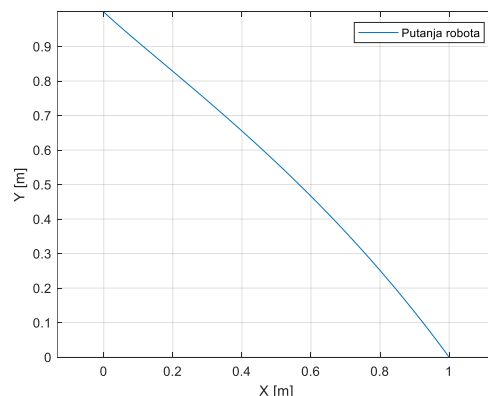


Slika 5.59. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli s interior-point algoritmom

Za prikaz upravljačkih varijabli tijekom vremena služi slika 5.60 gdje se vidi kako tijekom vremena ne izlaze iz postavljenih granica i u zadanom su intervalu. Sama putanja robota od početne do krajnje točke vidi se na slici 5.61.



Slika 5.60. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli



Slika 5.61. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli

5.2.3. Slučaj uz ograničenja na upravljačke varijable i kružne prepreke korištenjem Eulerove metode

U ovom slučaju imamo ograničenje tipa nejednakosti gdje je potrebno zadovoljiti izbjegavanje kružne prepreke

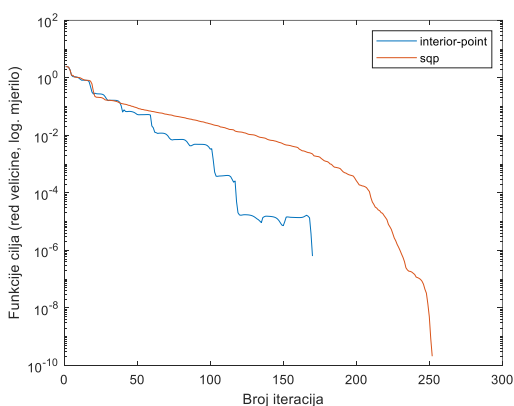
$$(x - x_0)^2 - (y - y_0)^2 \geq R^2,$$

gdje su

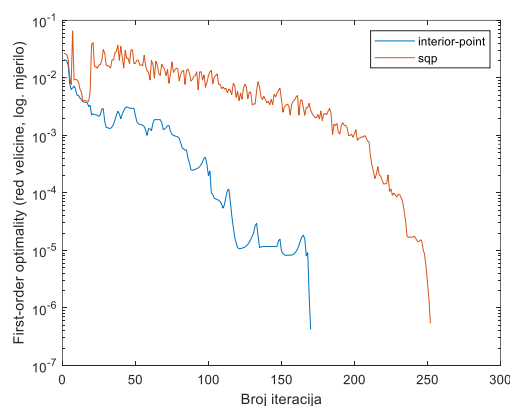
$$x = x_2 \cos(x_1),$$

$$y = x_2 \sin(x_1),$$

koordinate vrha robota u vanjskom koordinatnom sustavu, a x_0 i y_0 koordinate centra kružne prepreke polumjera R .

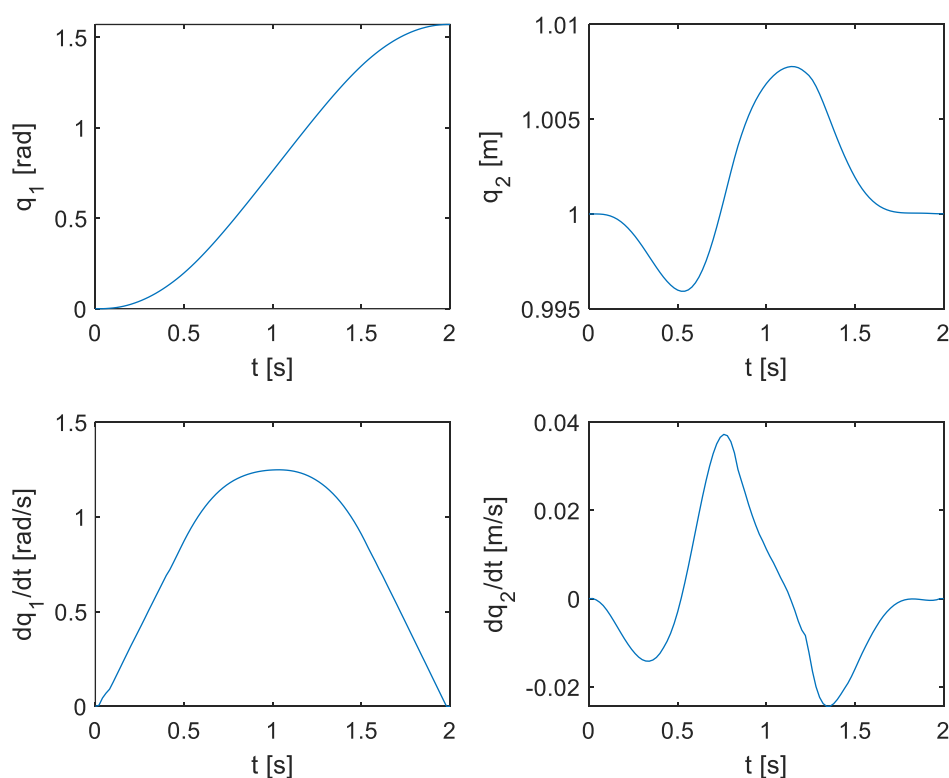


Slika 5.62. Konvergencija funkcije cilja u slučaju ograničenja upravljačkih varijabli i kružne prepreke za dva tipa algoritma



Slika 5.63. First-order optimality measure u slučaju ograničenja upravljačkih varijabli i kružne prepreke za dva tipa algoritma

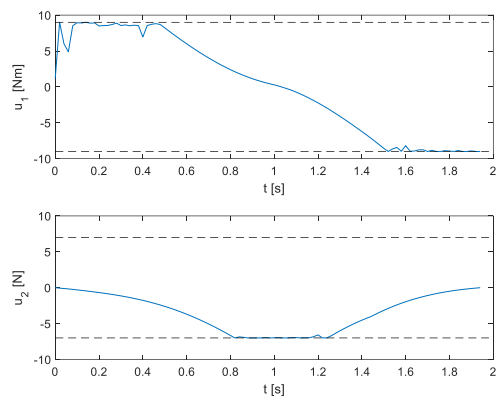
Kad se pogleda konvergencija funkcije cilja koju daje slika 5.62, uočavaju se znatno manje oscilacije, što bi značilo da je tu konvergencija stabilnija i da je *sqp* algoritam poželjniji. Slika 5.63 prikazuje kako se s oba algoritma povećavanjem broja iteracija približavamo nuli, tj. lokalnom minimumu. Obje također imaju i oscilacije, odnosno nagle skokove i padove. Vrijeme izvršavanja s *interior-point* metodom je 133,068 s dok za isti problem *sqp* algoritam izvrši za 190,342 s. To ukazuje i broj iteracija koji je veći kod *sqp* algoritma. Iako mu je vrijeme izvršavanja duže u ovom slučaju će se *sqp* uzeti kako bi se prikazali ostali parametri sustava.



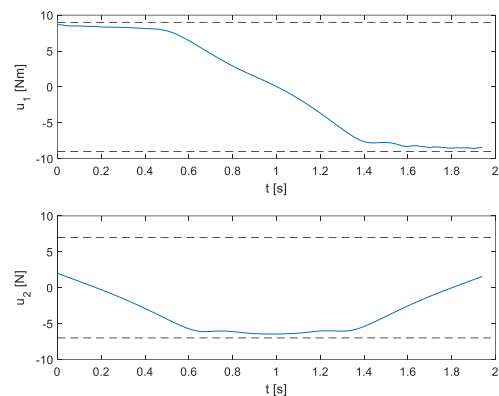
Slika 5.64. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i kružne prepreke sa *sqp* algoritmom

Slika 5.64 prikazuje četiri grafa korištenjem *sqp* algoritma gdje se (redom s lijeva na desno) vidi rotacijski položaj (q_1) kako se mijenja tijekom vremena te promjena translacijskog položaja (q_2) tijekom vremena. Slijedi još prikaz brzine rotacije robota kroz vrijeme $\frac{dq_1}{dt}$, tj. kutna promjena i brzinu translacije kroz vrijeme $\frac{dq_2}{dt}$.

Slika 5.65 i slika 5.66 prikazuju upravljačke varijable (sa zadanim postavljenim granicama) tijekom vremena. Ovom usporedbom može se vidjeti, posebno za upravljačku varijablu u_1 , kako *interior-point* algoritam ne traži rješenja izravno na granici dopuštenih ograničenja, nego se pretražuju unutarnje točke prostora.

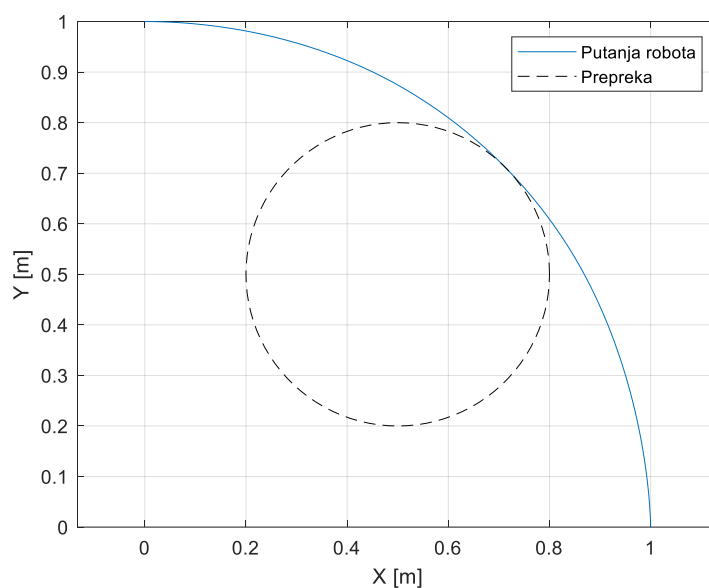


Slika 5.65. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i kružne prepreke – sqp



Slika 5.66. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i kružne prepreke – interior-point

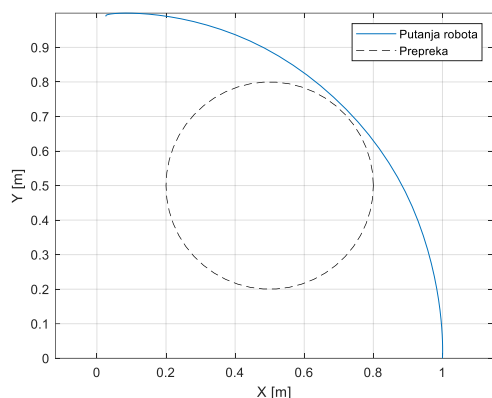
Putanja robota, gdje se vidi izbjegavanje kružne prepreke, može se vidjeti na slici 5.67.



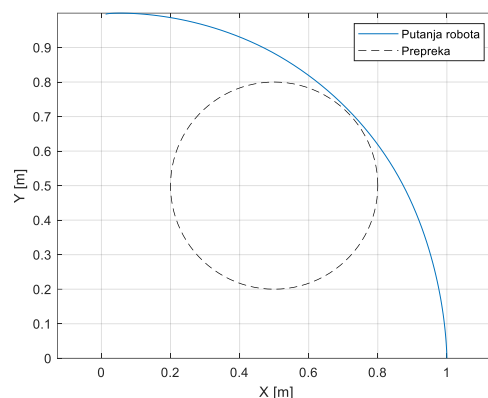
Slika 5.67. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke

Svi slučajevi riješeni su i Runge-Kuttovom metodom 4. reda (RK4). U ovom će se primjeru vidjeti i ta rješenja i usporedbe RK4 s Eulerovom metodom. Rješenje će se raditi za dva različita broja vremenskih intervala kako bi se vidjele razlike.

Za $N=50$ i sqp algoritam vrijeme izvršavanja je 38,786 s, dok za isti algoritam uz $N=100$ ono iznosi 185,369 s.



Slika 5.68. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke RK4 metodom za $N=50$



Slika 5.69. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i kružne prepreke RK4 metodom za $N=100$

Slika 5.68 i slika 5.69 prikazuju usporedbu u veličini broja vremenskih intervala gdje se jasno može vidjeti kako veći broj N znači veći broj iteracija, a samim time i veće vrijeme izvršavanja, no putanja je bliža prepreci što bi značilo da radi kraći, tj. brži put.

Sliku 5.69 također možemo usporediti sa slikom 5.67. Iako neočekivano (s obzirom na primjer iz poglavlja 5.1), ovdje je RK4 metoda lošija od Eulerove. Na slici 5.67 možemo vidjeti kako se Eulerovom metodom prepreka i putanja robota diraju što dovodi do najboljeg mogućeg puta u izbjegavanju zadane kružne prepreke.

5.2.4. Slučaj uz ograničenja na upravljačke varijable stanja i slijeđenje ravne linije korištenjem Eulerove metode

U ovom slučaju je ograničenje tipa jednakosti gdje se ono zasniva na slijeđenju ravne linije da bude zadovoljeno

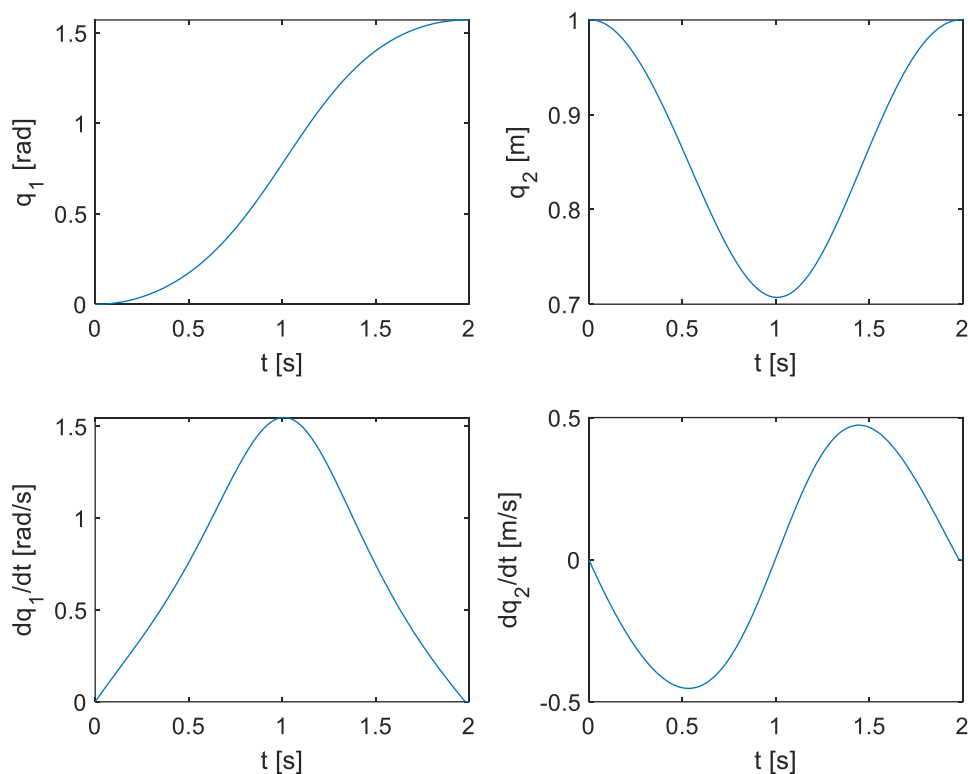
$$x + y - 1 = 0,$$

gdje su

$$x = x_2 \cos(x_1),$$

$$y = x_2 \sin(x_1).$$

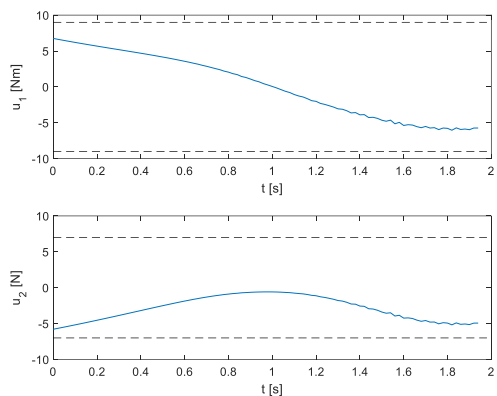
Ovdje se koristio *sqp* algoritam koji je dvostruko brže izvršio zadatak, a s obzirom na sličnost rezultata, ovaj se može uzeti kao optimalni.



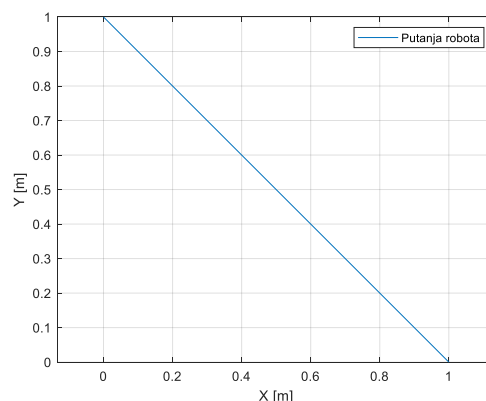
Slika 5.70. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije sa sqp algoritmom

Slika 5.70 prikazuje četiri grafa korištenjem *sqp* algoritma gdje se (redom s lijeva na desno) vidi rotacijski položaj (q_1) kako se mijenja tijekom vremena te promjena translacijskog položaja (q_2) tijekom vremena. Slijedi još prikaz brzine rotacije robota kroz vrijeme $\frac{dq_1}{dt}$, tj. kutna promjena koja je najveća u sredini procesa, a pred kraj pada simetrično kao i rast, i brzinu translacije kroz vrijeme $\frac{dq_2}{dt}$.

Za prikaz upravljačkih varijabli tijekom vremena služi slika 5.71 gdje se vidi kako tijekom vremena ne izlaze iz postavljenih granica i u zadanom su intervalu.



Slika 5.71. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije



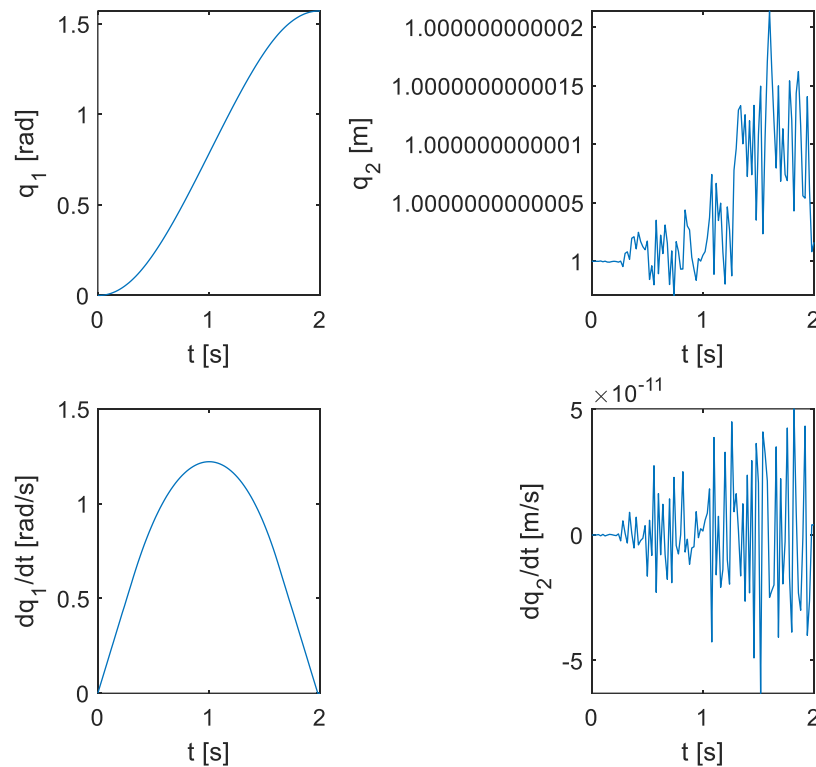
Slika 5.72. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i slijeđenje ravne linije

Sama putanja robota od početne do krajnje točke vidi se na slici 5.72 gdje se jasno vidi zadovoljen uvjet slijeđenja ravne linije.

5.2.5. Slučaj uz ograničenja na upravljačke varijable stanja i slijeđenje kružne putanje korištenjem Eulerove metode

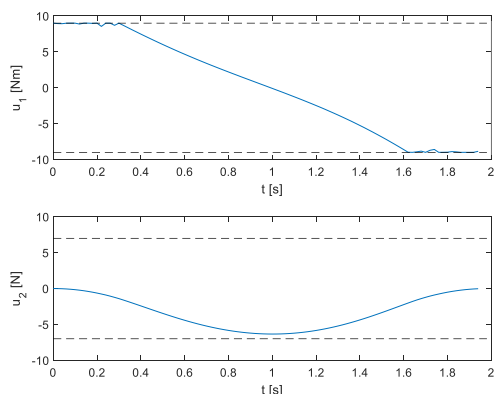
U ovom slučaju dobit će se rješenje uz slijeđenje kružne putanje koja spaja točke $(1, 0)$ i $(0, 1)$. Algoritam *sqp* dvostruko brže je riješio zadatak čiji rezultati slijede.

Slika 5.73 prikazuje četiri grafa korištenjem *sqp* algoritma gdje se (redom s lijeva na desno) vidi rotacijski položaj (q_1) kako se mijenja tijekom vremena. Promjena translacijskog položaja (q_2) tijekom vremena ima značajne oscilacije, iako se to nije očekivalo s obzirom na prethodne primjere. To može biti posljedica složenih svojstava ili nestabilnosti iako su zadovoljena sva zadana ograničenja. Slijedi još prikaz brzine rotacije robota kroz vrijeme $\frac{dq_1}{dt}$, tj. kutna promjena i brzinu translacije kroz vrijeme $\frac{dq_2}{dt}$. Ona je cijelo vrijeme oko nule što je potencijalno posljedica nedostatka preciznosti. Svakako je važno da se provjere parametri modela kako bi se osigurala dovoljna preciznost.

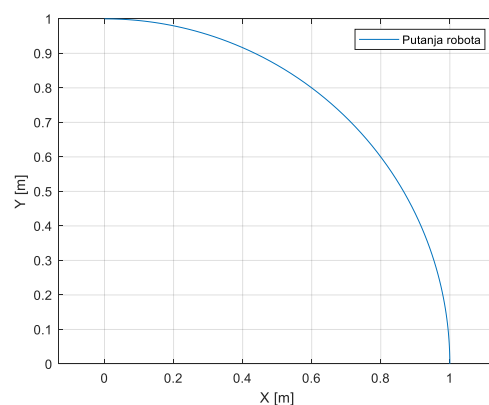


Slika 5.73. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje sa sqp algoritmom

Za prikaz upravljačkih varijabli tijekom vremena služi slika 5.74 gdje se vidi kako tijekom vremena ne izlaze iz postavljenih granica i u zadanom su intervalu. Samu putanja robota od početne do krajnje točke prikazuje slika 5.75 gdje se jasno vidi kretanje između točki koje je slijeđeno kružnom putanjom.



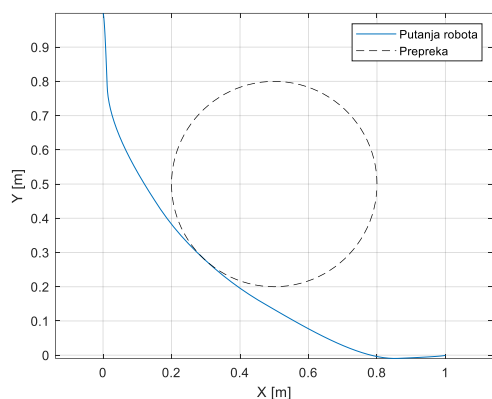
Slika 5.74. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje



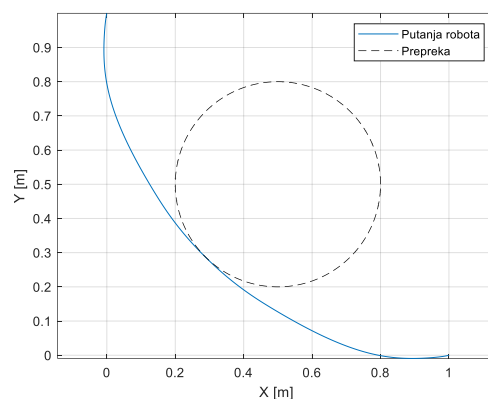
Slika 5.75. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i slijeđenje kružne putanje

5.2.6. Slučaj uz ograničenja na upravljačke varijable i izbjegavanje kružne prepreke korištenjem CasADi-ja

Budući da se u ovom dijelu rješava zadatak pomoću CasADi-ja, njegovo brzo izvršavanje (pogotovo u usporedbi s gornjim primjerima) daje nam mogućnost rješavanja s većim brojem N , tj. većim brojem vremenskih intervala. Svakim povećanjem N i ovdje se rezultati poboljšavaju. Tako kada se koristi $N=100$ vrijeme trajanja izvršavanja je 3,364 s što je puno manje u odnosu na primjere s istim brojem vremenskih intervala, ali bez korištenja CasADi-ja. Za pokušaj dobivanja boljih rezultate još će se povećati N na 1000, a tada je vrijeme izvršavanja 11,391 s.

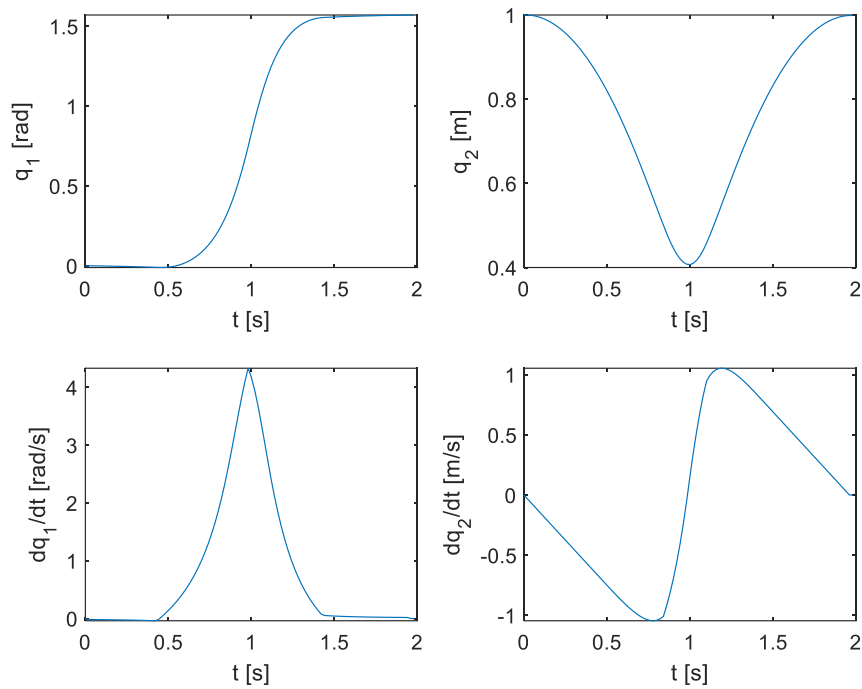


Slika 5.76. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja za $N=100$

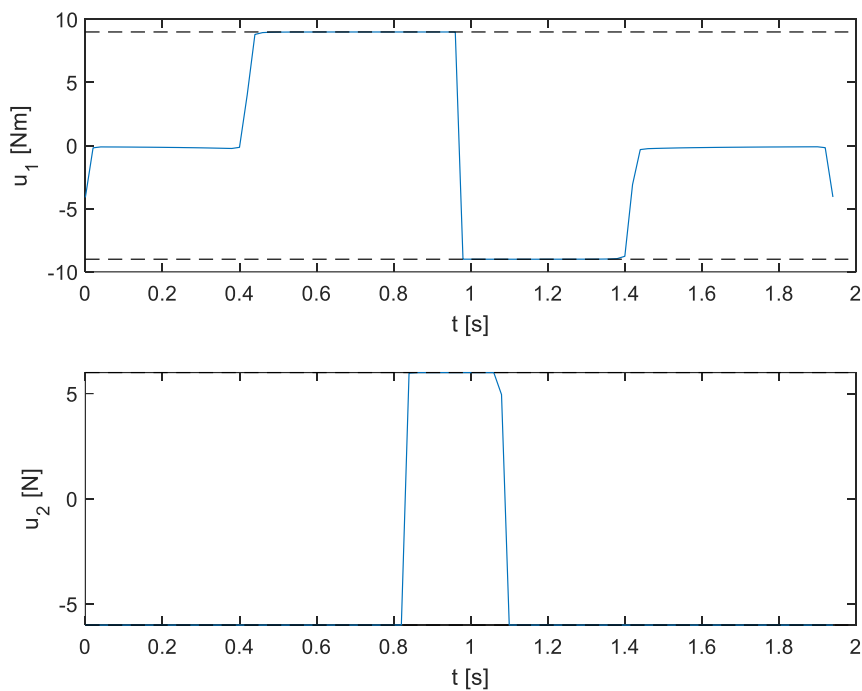


Slika 5.77. Putanja robota u prostoru za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja za $N=1000$

Pogledom na sliku 5.76 i sliku 5.77 rezultati su vrlo slični, prepreka je izbjegnuta uz njeno dodirivanje, no uz CasADi to je s druge strane prepreke za razliku od korištenja Eulerove ili RK4 metode. Numerički rezultati sigurno su bolji korištenjem većeg broja N , no u ovom slučaju to nije niti potrebno pa će ostala rješenje biti prikazana za $N=100$ kako bi bila najbolja usporedba s prijašnjim primjerima gdje se koristio isti taj broj vremenskih intervala.



Slika 5.78. Prikaz q_1 , q_2 , dq_1/dt i dq_2/dt za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja



Slika 5.79. Upravljačke varijable u_1 i u_2 (s granicama) tijekom vremena za slučaj ograničenja upravljačkih varijabli i izbjegavanje kružne prepreke korištenjem CasADi-ja

Slika 5.78 i slika 5.79 također su različite od onih koje smo imali prije jer CasADi u svojoj trenutnoj formi pruža skupove koji značajno smanjuju napor potreban za implementaciju

velikog broja algoritama za numeričko optimalno upravljanje i pruža metode diskretizacije koje su se pokazale učinkovitima na problemima s poznatim rješenjima, što olakšava otklanjanje poteškoća.

Ovdje je *solver* (rješavač) postavljen na „ipopt“ (*Interior Point OPTimizer*) što ukazuje na to koja se metoda koristila. Ovdje se implementiraju unutrašnje točke što se često koristi za nelinearne optimizacijske probleme s ograničenjima kakav je i ovaj primjer. Ovo je metoda koja se često koristi za različite vrste problema optimizacije i može biti vrlo efikasna za širok spektar problema.

6. ZAKLJUČAK

U današnjem sve složenijem svijetu i uz mnoge probleme iz teorije upravljanja, optimalno upravljanje dinamičkim sustavima pokazalo se ključnim za postizanje visokih performansi i iskorištavanje potencijala tehnologije u različitim sektorima. Kroz analizu dinamike sustava i identifikaciju ključnih varijabli, moguće je razviti strategije optimalnog upravljanja koje omogućuju sustavima da se prilagode dinamičnim izazovima, optimiziraju performanse i minimiziraju troškove. U ovom radu istražene su tri ključne numeričke metode diskretizacije u kontekstu optimalnog upravljanja dinamičkim sustavima: Eulerova metoda, Runge-Kutteova metoda 4. reda i Adamsova metoda 4. reda te metode numeričke optimizacije. Osim toga, prikazani su i relevantni programski alati, poput MATLAB-a i CasADi-ja, koji se koriste za implementaciju ovih metoda u optimizaciji.

Numerički eksperimenti provedeni su kako bi se analizirala učinkovitost i točnost primijenjenih metoda, pružajući dublje razumijevanje njihovih praktičnih aspekata, prednosti i ograničenja. Kroz usporedbe rezultata dobivenih različitim metodama, rad je doprinio boljem razumijevanju optimalnog upravljanja dinamičkim sustavima. Iako su istražene metode pokazale određene prednosti i nedostatke u različitim scenarijima, važno je istaknuti da ne postoji univerzalna metoda koja bi bila optimalna za sve situacije. Zato je važno pažljivo odabrati metodu ovisno o konkretnim zahtjevima sustava i ciljevima optimizacije. Kroz primjere riješenih problema, uključujući linearni sustav 2. reda i mehanički sustav s dva stupnja slobode gibanja, demonstrirana je primjena teorijskih koncepta i numeričkih metoda u praksi. Ovi primjeri pružaju uvid u praktičnu primjenu optimalnog upravljanja u različitim domenama, naglašavajući važnost kontinuiranog istraživanja i razvoja u području optimizacije dinamičkih sustava.

LITERATURA

- [1] Locatelli, A.: *Optimal Control: An Introduction*, Volume 55, Issue 3, Birkhauser Verlag AG, Basel, Switzerland, 2002.
- [2] Diehl, M., Gros, S.: *Numerical Optimal Control*, Lecture notes, University of Freiburg, 2022.
- [3] Kasać, J., *Opća teorija sustava*, Fakultet strojarstva i brodogradnje, Sveučilište u Zagrebu, Zagreb, 2007.
- [4] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM Press, Philadelphia, 2001.
- [5] Grbeš, A., *Numeričke metode za rješavanje običnih diferencijalnih jednadžbi*, Diplomski rad, Prirodoslovno matematički fakultet, Sveučilište u Zagrebu, Zagreb, 2021.
- [6] Đurasević, M., *Kvadratično i sekvencijalno kvadratično programiranje*, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2015.
- [7] Đurasević, M., Jakobović, D., *Postupci numeričke optimizacije*, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2023.
- [8] Essert M., Žilić T.: *MATLAB – Matrični laboratorij*, Zagreb, 2004.
- [9] Nocedal, J., Wright S. J., *Numerical Optimization*, 2nd edition, Springer, New York, 2006.
- [10] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., Diehl, M.: *CasADi: a software framework for nonlinear optimization and optimal control*, *Math. Prog. Comp.* 11, 1–36, 2019.
- [11] Leek, V., *An Optimal Control Toolbox for MATLAB Based on CasADi*, Master of Science Thesis, Linköping University, Linköping, 2016.
- [12] Griewank, A., Walther, A., *Evaluating Derivatives - Principles and Techniques of Algorithmic Differentiation*, 2nd edition, SIAM, Philadelphia, 2008.
- [13] Giles, M. B., *Collected matrix derivative results for forward and reverse mode algorithmic differentiation*. In: Bischof, C.H., Bücker, H.M., Hovland, P., Naumann, U., Utke, J. (eds.) *Advances in Automatic Differentiation*, pp. 35–44. Springer, Berlin, 2008.
- [14] Neidinger, R. D., *Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming*, Volume 52, No. 3, pp. 545-563, SIAM, Philadelphia, 2009.
- [15] Novaković, B., *Metode vođenja tehničkih sistema*, ŠK, Zagreb, 1990.: 104 – 105

PRILOZI

- I. MATLAB kod – linearni sustav drugog reda – *fminunc* (Eulerova, RK4 i Adamsova metoda)
- II. MATLAB kod – linearni sustav drugog reda – *fmincon*
- III. MATLAB kod – linearni sustav drugog reda – uz CasADi
- IV. MATLAB kod – mehanički sustav s dva stupnja slobode gibanja
- V. MATLAB kod – mehanički sustav s dva stupnja slobode gibanja – uz CasADi

I. MATLAB kod – linearni sustav drugog reda – *fminunc* (Eulerova, RK4 i Adamsova metoda)

```

close all
clear all
clc

global t0 T N tau t_vec x0 n

n = 3; % red sustava
nu = 2; % broj upravljackih varijabli

t0 = 0; % pocetno vrijeme
T = 1; % konacno vrijeme
N = 1000; % broj vremenskih intervala
tau = (T-t0)/N; % korak integracije
t_vec = t0:tau:(T-tau); % vremenski interval

tu = t0:tau:(T-2*tau);

x0 = [0 0 0]; % pocetna stanja
u0 = zeros(2,N-1); % pocetna upravljacka varijable

u1 = control(u0);
xout = my_euler(u1, tau, N-1);

% Analiticko rjesenje problema optimalnog upravljanja
a = 0.5; b = -12;
u1a = (1+b)*(tu/2) - (1+a+b)/2; % opt upr
u2a = -b/2*ones(size(tu)); % opt upr
x1a = (1+b)*t_vec.^2/4 - (1+a+b)*t_vec/2;
x2a = (1+b)*t_vec.^3/12 - (1+a+b)*t_vec.^2/4 - b/2*t_vec;

figure(1)
plot(t_vec, xout(1,:), t_vec, x1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta stanja - x_1');
xlabel('Vrijeme [s]');
ylabel('x_1');

figure(2)
plot(t_vec, xout(2,:), t_vec, x2a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta stanja - x_2');
xlabel('Vrijeme [s]');
ylabel('x_2');

figure(3)
plot(tu, u1(1,:), tu, u1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta upravljačke varijable - u_1');
xlabel('Vrijeme [s]');
ylabel('u_1');

figure(4)
plot(tu, u1(2,:), tu, u2a, '--r')
ylim([-1.5 7])
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta upravljačke varijable - u_2');
xlabel('Vrijeme [s]');
ylabel('u_2');

% Usporedba numeričkih i analitičkih rezultata
disp('Usporedba numeričkih i analitičkih rezultata:');
disp('-----');

% Odstupanje za x1
diff_x1 = norm(xout(1,:) - x1a, inf);

```



```

disp(['Odstupanje za x1: ', num2str(diff_x1)]);

% Odstupanje za x2
diff_x2 = norm(xout(2,:) - x2a, inf);
disp(['Odstupanje za x2: ', num2str(diff_x2)]);

% Odstupanje za u1
diff_u1 = norm(u1(1,:) - u1a, inf);
disp(['Odstupanje za u1: ', num2str(diff_u1)]);

% Odstupanje za u2
diff_u2 = norm(u1(2,:) - u2a, inf);
disp(['Odstupanje za u2: ', num2str(diff_u2)]);

function u = control(u0)

options=optimset('Display','iter');

u = fminunc('cost_function', u0, options);

function J = cost_function(x)

global N tau

a = 0.5; b = -12;

xout = my_euler(x, tau, N-1);

J = a*xout(1,N) + b*xout(2,N) + xout(3,N);

function x = my_euler(u, h, n_i)

% Euler method for ODEs solution

global x0 n

x(1:n, 1) = x0;

%% Euler %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j = 1:n_i
    f = ODE_sys(x, u, j);
    x(:,j+1) = x(:,j) + h*f;
end

function dx = ODE_sys(x, u, i)

% Diferencijalne jednadzbe sustava
% dx_1(t) = u_1(t)
% dx_2(t) = x_1(t) + u_2(t)

% x_1(t), x_2(t) su varijable stanja
% u_1(t) i u_2(t) su upravljacke varijable

% x3 = \int{u_1^2 + u_2^2 + x_1} integralni clan funkcije cilja, odnosno
% dx_3(t) = u_1^2 + u_2^2 + x_1

dx = zeros(3, 1);

x1 = x(1,i);
cont1 = u(1,i); % optimalno upravljanje
cont2 = u(2,i); % optimalno upravljanje

dx(1) = cont1;
dx(2) = x1 + cont2;
dx(3) = x1 + cont1^2 + cont2^2;

```

```

function x = my_rk4(u, h, n_i)

global x0 n

x(:, 1) = x0;

for j = 1:n_i
    k1 = ODE_sys(x(:, j), u(:, j), j);
    k2 = ODE_sys(x(:, j) + h / 2 * k1, u(:, j), j);
    k3 = ODE_sys(x(:, j) + h / 2 * k2, u(:, j), j);
    k4 = ODE_sys(x(:, j) + h * k3, u(:, j), j);

    x(:, j + 1) = x(:, j) + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);
end

function x = my_adams4(u, h, n_i)

global x0 n

x(1:n, 1) = x0;

%% Adams-Bashforth 4th order %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for j = 1:n_i
%     f = ODE_sys(x, u, j);
%     if j <= 4
%         % Koristi Eulerov korak za prva 3 koraka
%         x(:,j+1) = x(:,j) + h*f;
%     else
%         % Adams-Bashforth korak za ostale korake
%         x(:,j+1) = x(:,j) + (h/24)*(55*f - 59*ODE_sys(x, u, j-1) + 37*ODE_sys(x,
u, j-2) - 9*ODE_sys(x, u, j-3));
%     end
% end

%Ovi tezijski koeficijenti (9, 19, -5, 1) su dobiveni iz matematičkih razvoja
interpolacijskog
%polinoma koji povezuje prethodne četiri brzine promjene. Formula omogućava
aproksimaciju
%sljedećeg stanja sustava koristeći Adams-Bashforth metodu 4. reda.

%% Adams-Bashforth 2nd order %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f1 = ODE_sys(x, u, 1);
% x(:, 2) = x(:, 1) + h*f1;
%
% for j = 2:n_i
%     f2 = ODE_sys(x, u, j);
%     x(:, j+1) = x(:, j) + (h/2)*(-f1 + 3*f2);
%     f1 = f2;
% end

%% Adams-Bashforth 3rd order %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f1 = ODE_sys(x, u, 1);
x(:, 2) = x(:, 1) + h*f1;

f2 = ODE_sys(x, u, 2);
x(:, 3) = x(:, 2) + h*f2;

for j = 3:n_i
    f3 = ODE_sys(x, u, j);
    x(:, j+1) = x(:, j) + (h/12)*(5*f1 - 16*f2 + 23*f3);
    f1 = f2;
    f2 = f3;
end

```

```
%% Adams 4th order %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% f1 = ODE_sys(x, u, 1);
% x(:, 2) = x(:, 1) + h*f1;
%
% f2 = ODE_sys(x, u, 2);
% x(:, 3) = x(:, 2) + h*f2;
%
% f3 = ODE_sys(x, u, 3);
% x(:, 4) = x(:, 3) + h*f3;
%
% for j = 4:n_i
%     f4 = ODE_sys(x, u, j);
%     x(:, j+1) = x(:, j) + (h/24)*(f1 - 5*f2 + 19*f3 + 9*f4); % Adams-Moulton
%     %x(:, j+1) = x(:, j) + (h/24)*(-9*f1 + 37*f2 - 59*f3 + 55*f4); % Adams-
    Bashforth
%     f1 = f2;
%     f2 = f3;
%     f3 = f4;
% end
```

II. MATLAB kod – linearni sustav drugog reda – *fmincon*

```

close all
clear all
clc

global t0 T N tau t_vec x0 n

n = 3; % red sustava
nu = 2; % broj upravljackih varijabli

t0 = 0; % pocetno vrijeme
T = 1; % konacno vrijeme
N = 10; % broj vremenskih intervala
tau = (T-t0)/N; % korak integracije
t_vec = t0:tau:(T-tau); % vremenski interval

tu = t0:tau:(T-2*tau);

x0 = [0 0 0]; % pocetna stanja
u0 = zeros(2,N-1); % pocetna upravljacka varijable

u = control_fmincon(u0);
xout = my_euler(u, tau, N-1);

% Analiticko rjesenje problema optimalnog upravljanja
a = 1; b = 2;
u1a = (1+b)*(tu/2) - (1+a+b)/2; % opt upr
u2a = -b/2*ones(size(tu)); % opt upr
x1a = (1+b)*t_vec.^2/4 - (1+a+b)*t_vec/2;
x2a = (1+b)*t_vec.^3/12 - (1+a+b)*t_vec.^2/4 - b/2*t_vec;

figure(1)
plot(t_vec, xout(1,:), t_vec, x1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta stanja - x_1');
xlabel('Vrijeme [s]');
ylabel('x_1');

figure(2)
plot(t_vec, xout(2,:), t_vec, x2a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta stanja - x_2');
xlabel('Vrijeme [s]');
ylabel('x_2');

figure(3)
plot(tu, u(1,:), tu, u1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta upravljačke varijable - u_1');
xlabel('Vrijeme [s]');
ylabel('u_1');

figure(4)
plot(tu, u(2,:), tu, u2a, '--r')
ylim([-1.5 0.5])
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta upravljačke varijable - u_2');
xlabel('Vrijeme [s]');
ylabel('u_2');

function u = control_fmincon(u0)

options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'interior-
point');

```

```
A = [];  
b = [];  
Aeq = [];  
beq = [];  
lb = -ones(size(u0)); %donje granice  
ub = ones(size(u0)); %gornje granice  
  
u = fmincon(@cost_function_fmincon, u0, A, b, Aeq, beq, lb, ub, @constraints,  
options);  
  
%funkcija constraints.m koja za definiranje ograničenja  
function [c, ceq] = constraints(u)  
    %primjrr nejednakosnih ograničenja:  
    c = [];  
    ceq = [];  
end  
end  
  
function J = cost_function_fmincon(u)  
  
global N tau  
  
a = 1; b = 2;  
  
xout = my_euler(u, tau, N-1);  
  
J = a*xout(1,N) + b*xout(2,N) + xout(3,N);
```

III. MATLAB kod – linearni sustav drugog reda – uz CasADi

```

%%addpath('C:\Users\Windows\Desktop\casadi-3.6.4-windows64-matlab2018b');

close all;
clear all;
clc;

import casadi.*

global t0 T N tau t_vec x0 n nu

n = 3; % red sustava
nu = 2; % broj upravljačkih varijabli

t0 = 0; % početno vrijeme
T = 1; % konačno vrijeme
N = 11; % broj vremenskih intervala
tau = (T - t0) / N; % korak integracije
t_vec = t0:tau:(T-tau); % vremenski interval

tu = t0:tau:(T-2*tau);

x0 = [0 0 0]; % početna stanja
u0 = zeros(2, N - 1); % početna upravljačke varijable

[u_opt, x_opt] = optimize();

% Analitičko rješenje problema optimalnog upravljanja
a = 1; b = 2;
u1a = (1 + b) * (tu / 2) - (1 + a + b) / 2; % opt upr
u2a = -b / 2 * ones(size(tu)); % opt upr
x1a = (1 + b) * t_vec.^2 / 4 - (1 + a + b) * t_vec / 2;
x2a = (1 + b) * t_vec.^3 / 12 - (1 + a + b) * t_vec.^2 / 4 - b / 2 * t_vec;

figure(1)
plot(t_vec, x_opt(1, :), t_vec, x1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta stanja - x_1');
xlabel('Vrijeme [s]');
ylabel('x_1');

figure(2)
plot(t_vec, x_opt(2, :), t_vec, x2a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta stanja - x_2');
xlabel('Vrijeme [s]');
ylabel('x_2');

figure(3)
plot(tu, u_opt(1, :), tu, u1a, '--r')
legend('Numerički rezultat', 'Analitički rezultat');
title('Prva komponenta upravljačke varijable - u_1');
xlabel('Vrijeme [s]');
ylabel('u_1');

figure(4)
plot(tu, u_opt(2, :), tu, u2a, '--r')
ylim([-1.5 0.5])
legend('Numerički rezultat', 'Analitički rezultat');
title('Druga komponenta upravljačke varijable - u_2');
xlabel('Vrijeme [s]');
ylabel('u_2');

% Usporedba numeričkih i analitičkih rezultata
disp('Usporedba numeričkih i analitičkih rezultata:');
disp('-----');

```

```

% Odstupanje za x1
diff_x1 = norm(x_opt(1,:) - x1a, inf);
disp(['Odstupanje za x1: ', num2str(diff_x1)]);

% Odstupanje za x2
diff_x2 = norm(x_opt(2,:) - x2a, inf);
disp(['Odstupanje za x2: ', num2str(diff_x2)]);

% Odstupanje za u1
diff_u1 = norm(u_opt(1,:) - u1a, inf);
disp(['Odstupanje za u1: ', num2str(diff_u1)]);

% Odstupanje za u2
diff_u2 = norm(u_opt(2,:) - u2a, inf);
disp(['Odstupanje za u2: ', num2str(diff_u2)]);

function [u_opt, x_opt] = optimize()
    import casadi.*

    % Definiraj globalne simbole
    global n nu N tau umax1 umax2 x0 a b

    n = 3; % red sustava
    nu = 2; % broj upravljačkih varijabli

    %umax1 = 1; % Postavi ovo na odgovarajuću vrijednost
    %umax2 = 1; % Postavi ovo na odgovarajuću vrijednost
    a = 1;
    b = 2;

    % Definiraj simbole unutar optimize funkcije kao globalne
    global x u

    x = MX.sym('x', n); %stanje
    u = MX.sym('u', nu); %upravljanje

    %definiranje funkcije sustava
    dx1 = u(1);
    dx2 = x(1) + u(2);
    dx3 = x(1) + u(1)^2 + u(2)^2;
    ode = vertcat(dx1, dx2, dx3);

    %casadi funkcije sustava
    ode_func = Function('ode_func', {x, u}, {ode});

    %Casadi funkcija cilja
    cost = a * x(1) + b * x(2) + x(3);
    cost_func = Function('cost_func', {x, u}, {cost});

    %inicijalizacija optim problema
    opti = casadi.Opti();

    %definiranje varijable
    U = opti.variable(nu, N-1);

    % Postavi početne uvjete
    X = opti.variable(n, N);
    X(:, 1) = x0;

    %postavljanje ograničenja sustava
    for k = 1:N-1
        opti.subject_to(X(:, k + 1) == ode_func(X(:, k), U(:, k)) * tau + X(:, k));
    end

    %Postavi ograničenja na upravljanje??
    %opti.subject_to(U(1, :) <= umax1);
    %opti.subject_to(U(2, :) <= umax2);

```

```
% Postavi početne guess vrijednosti
opti.set_initial(U, zeros(nu, N - 1));

%funk cilja
J = cost_func(X(:, N-1), U(:, N-1));
opti.minimize(J);

opti.solver('ipopt'); %postavljanje solvera

sol = opti.solve(); %rj opt slovera

%Dohvati optimalna rješenja
u_opt = sol.value(U);
x_opt = sol.value(X);
end
```


IV. MATLAB kod – mehanički sustav s dva stupnja slobode gibanja

```

%close all
clear all
clc

global x0 n a m1 m2 I12 U1min U1max U2min U2max X0 Y0 R N tau Jopt j

global output_fmin_data

% parametri robota
m1 = 3;
m2 = 1;
I12 = 0.1;
a = 0.25;

% ogranichenja upr. varijable
U1min = -9; U1max = 9;
U2min = -7; U2max = 7;

% prepreka
X0 = 0.5;
Y0 = 0.5;
R = 0.3;

n = 4; % red sustava
nu = 2; % broj upravljackih varijabli

t0 = 0; % pocetno vrijeme
T = 2; % konacno vrijeme
N = 100; % broj vremenskih intervala
tau = (T-t0)/N; % korak integracije
t_vec = t0:tau:T; % vremenski interval

tu = t0:tau:T-tau;

x0 = [0 1 0 0]; % pocetna stanja

u0 = zeros(nu, N); % pocetne upravljacke varijable

j = 1;
uopt = control(u0);

xout = my_euler(uopt, tau, N);

figure(1)
subplot(221)
plot(t_vec, xout(1, :))
xlabel('t [s]'), ylabel('q_1 [rad]')

subplot(222)
plot(t_vec, xout(2, :))
xlabel('t [s]'), ylabel('q_2 [m]')

subplot(223)
plot(t_vec, xout(3, :))
xlabel('t [s]'), ylabel('dq_1/dt [rad/s]')

subplot(224)
plot(t_vec(1:end), xout(4, :))
xlabel('t [s]'), ylabel('dq_2/dt [m/s]')

figure(2)
subplot(211)
plot(tu(1:end-2), uopt(1, 1:end-2), tu, U1min*ones(size(tu)), '--k', tu,
U1max*ones(size(tu)), '--k')

```

```

xlabel('t [s]'), ylabel('u_1 [Nm]')

subplot(212)
plot(tu(1:end-2), uopt(2, 1:end-2), tu, U2min*ones(size(tu)), '--k', tu,
U2max*ones(size(tu)), '--k')
xlabel('t [s]'), ylabel('u_2 [N]')

% Cartesian coordinate
x2 = xout(2,:);
x1 = xout(1,:);
X = x2.*cos(x1);
Y = x2.*sin(x1);

% prepreka
omega = 2*pi/T;
Xc = X0 + R*cos(omega*t_vec);
Yc = Y0 + R*sin(omega*t_vec);

figure(3)
plot(X, Y)
xlabel('X [m]')
ylabel('Y [m]')
legend('Putanja robota')
grid on
axis equal

% konvergencija funkcije cilja
Nfunc_eval = N*nu+1;
Niter = int32(length(Jopt)/Nfunc_eval); % broj iteracija treba uzeti int32 zbog
numerike dijeljenja

% uzima se svaki N*nu+1 element iz onoga sto fmincon vraca u feval jer
% u jednoj iteraciji fmincon izracuna nu opt. var. (u1 i u2) i iz toga izracuna
vrijednost funkcije cilja (feval)
% a dinamika sustava se rjesava N puta pa iz toga slijedi N*nu+1 racunanja,
% a na kraju tog racunanja se dobiva nova funkcija cilja u sljedecoj
% iteraciji, zbog toga se uzima:
JJopt = Jopt(1:(N*nu+1):end-1);

figure(4)
semilogy(1:Niter, JJopt)
xlabel('Broj iteracija')
ylabel('Funkcije cilja (red velicine, log. mjerilo)')
legend('interior-point', 'sqp')
hold on

% konvergencija inf. norme gradijenta (First-order optimality)
for k=1:Niter
    gradinf(k) = output_fmin_data(k).optimValues;
end

figure(5)
semilogy(1:Niter, gradinf)
xlabel('Broj iteracija')
ylabel('First-order optimality (red velicine, log. mjerilo)')
legend('interior-point', 'sqp')
hold on

function uizl = control(u)

global N U1min U1max U2min U2max

u1min = U1min*ones(1,N);
u1max = U1max*ones(1, N);

u2min = U2min*ones(1,N);

```

```

u2max = U2max*ones(1, N);

Umin = [ulmin;u2min];
Umax = [ulmax;u2max];

% % 1. slucaj: bez ogranicenja
%options = optimset('OutputFcn', @output_fmin, 'Display', 'iter', 'Algorithm',
'interior-point', 'Tolx', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxFunEvals',
100000);
%uizl = fmincon('cost_function', u, [], [], [], [], [], [], [], options);

% % 2. slucaj: ogranicenja upr. varijable
%options = optimset('OutputFcn', @output_fmin, 'Display', 'iter', 'Algorithm',
'interior-point', 'Tolx', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxFunEvals',
100000);
%uizl = fmincon('cost_function', u, [], [], [], [], Umin, Umax, [], options);

% 3. slucaj: ogranicenja upr. varijable i kruzna prepreka (ogranicenje tipa
nejednakosti)
%options = optimset('OutputFcn', @output_fmin, 'Display', 'iter', 'Algorithm',
'interior-point', 'Tolx', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxFunEvals',
100000);
%uizl = fmincon('cost_function', u, [], [], [], [], Umin, Umax, 'constr_func',
options);

% % 4. slucaj: ogranicenja upr. varijable i slijedenje ravne linije (ogranicenje
tipa jednakosti)
%options = optimset('OutputFcn', @output_fmin, 'Display', 'iter', 'Algorithm',
'sqp', 'Tolx', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxFunEvals', 100000);
%uizl = fmincon('cost_function', u, [], [], [], [], Umin, Umax, 'constr_func',
options);

% % 5. slucaj: ogranicenja upr. varijable i slijedenje kruzne putanje (ogranicenje
tipa jednakosti)
options = optimset('OutputFcn', @output_fmin, 'Display', 'iter', 'Algorithm',
'sqp', 'Tolx', 1e-6, 'TolFun', 1e-6, 'TolCon', 1e-6, 'MaxFunEvals', 100000);
uizl = fmincon('cost_function', u, [], [], [], [], Umin, Umax, 'constr_func',
options);

function J = cost_function(x)

global N tau Jopt j

xout = my_euler(x, tau, N);

x1df = pi/2; x2df = 1; x3df = 0; x4df = 0; % zeljena konacna stanja

x1f = xout(1, N);
x2f = xout(2, N);
x3f = xout(3, N);
x4f = xout(4, N);

J = (x1df - x1f).^2 + (x2df - x2f).^2 + (x3df - x3f).^2 + (x4df - x4f).^2;

Jopt(j) = J;
j = j + 1;

function x = my_euler(u, h, n_i)

% Euler method for ODEs solution

global x0 n

x(1:n, 1) = x0;

%% Euler %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

for j = 1:n_i
    f = ODE_RTrobot(x, u, j);
    x(:, j+1) = x(:, j) + h*f;
end
function dx = ODE_RTrobot(x, u, i)

% System ODE: x' = f(x) + g(x)*u = f(x, u)

global I12 m1 m2 a

% x1=q1 x2=q2 x3=dq1 x4=dq2
x1 = x(1, i); x2 = x(2, i);
x3 = x(3, i); x4 = x(4, i);
u1 = u(1, i); u2 = u(2, i);

M11 = I12 + (m1+m2)*x2^2 + 2*m2*a*x2 + m2*a^2;
M22 = m1+m2;

C1 = 2*((m1+m2)*x2+m2*a)*x3*x4;
C2 = -(m1*x2+m2*(a+x2))*x3^2;

M = [M11 0;0 M22];
C = [C1;C2];

dq_vec = [x3;x4];
u_vec = [u1;u2];
ddq_vec = M\u_vec - M\C;

dx = [dq_vec;ddq_vec];

function [ineq_const, eq_const] = constr_func(x)

global N tau X0 Y0 R

xout = my_euler(x, tau, N);

x1 = xout(1,:);
x2 = xout(2,:);

X = x2.*cos(x1);
Y = x2.*sin(x1);

% % 3. slucaj: izbjegavanje krugne prepreke (ogranicenje tipa nejednakosti)
%ineq_const = R^2 - ((X-X0).^2+(Y-Y0).^2);
%eq_const = [];

% % 4. slucaj: slijedenje ravne linije (ogranicenje tipa jednakosti)
%ineq_const = [];
%eq_const = X + Y - 1;

% % 5. slucaj: slijedenje krugne putanje koja spaja tocke (1, 0) (0, 1)
(ogranicenje tipa jednakosti)
ineq_const = [];
eq_const = X.^2 + Y.^2 - 1;

function stop = output_fmin(x, optimValues, state)

stop = false;

global output_fmin_data

switch state
case 'init'

```

```
    hold on
    case 'iter'
        ni = length(output_fmin_data)+1;
        output_fmin_data(ni).x = x;
        output_fmin_data(ni).optimValues = optimValues.firstorderopt;
    case 'done'
        hold off
    otherwise
end
end
```

V. MATLAB kod – mehanički sustav s dva stupnja slobode gibanja – uz CasADi

```

close all;
clear all;
clc;

import casadi.*

global t0 T N tau t_vec x0 n nu
global U1min U1max U2min U2max
global X0 Y0 R

n = 4; % red sustava
nu = 2; % broj upravljackih varijabli

t0 = 0; % pocetno vrijeme
T = 2; % konacno vrijeme
N = 100; % broj vremenskih intervala
tau = (T - t0) / N; % korak integracije
t_vec = t0:tau:(T-tau); % vremenski interval

tu = t0:tau:(T-2*tau);

x0 = [0 1 0 0]; % pocetna stanja
u0 = zeros(2, N - 1); % pocetna upravljacke varijable

% ogranicenja upr. varijable
U1min = -9; U1max = 9;
U2min = -6; U2max = 6;

% prepreka
X0 = 0.5;
Y0 = 0.5;
R = 0.3;

[u_opt, x_opt] = optimize();

figure(1)
subplot(221)
plot(t_vec, x_opt(1, :))
xlabel('t [s]'), ylabel('q_1 [rad]')

subplot(222)
plot(t_vec, x_opt(2, :))
xlabel('t [s]'), ylabel('q_2 [m]')

subplot(223)
plot(t_vec, x_opt(3, :))
xlabel('t [s]'), ylabel('dq_1/dt [rad/s]')

subplot(224)
plot(t_vec(1:end), x_opt(4, :))
xlabel('t [s]'), ylabel('dq_2/dt [m/s]')

figure(2)
subplot(211)
plot(t_vec(1:end-2), u_opt(1, 1:end-1), t_vec, U1min*ones(size(t_vec)), '--k',
t_vec, U1max*ones(size(t_vec)), '--k')
xlabel('t [s]'), ylabel('u_1 [Nm]')

subplot(212)
plot(t_vec(1:end-2), u_opt(2, 1:end-1), t_vec, U2min*ones(size(t_vec)), '--k',
t_vec, U2max*ones(size(t_vec)), '--k')
xlabel('t [s]'), ylabel('u_2 [N]')

% Cartesian coordinate

```

```

x2 = x_opt(2,:);
x1 = x_opt(1,:);
Xr = x2.*cos(x1);
Yr = x2.*sin(x1);

% prepreka
omega = 2*pi/T;
Xc = X0 + R*cos(omega*t_vec);
Yc = Y0 + R*sin(omega*t_vec);

figure(3)
plot(Xr, Yr, Xc, Yc, 'k--')
xlabel('X [m]')
ylabel('Y [m]')
legend('Putanja robota', 'Prepreka')
grid on
axis equal

function [u_opt, x_opt] = optimize()
    import casadi.*

    % Definiraj globalne simbole
    global n nu N tau x0
    global Ulmin Ulmax U2min U2max
    global X0 Y0 R

    n = 4; % red sustava
    nu = 2; % broj upravljackih varijabli

    % parametri robota
    m1 = 3;
    m2 = 1;
    I12 = 0.1;
    a = 0.25;

    % Definiraj simbole unutar optimize funkcije kao globalne
    global x u

    x = MX.sym('x', n); %stanje
    u = MX.sym('u', nu); %upravlj

    %definiranje funkcije sustava
    dx1 = x(3);
    dx2 = x(4);
    dx3 = -(2*a*m2*x(3)*x(4) - u(1) + 2*m1*x(2)*x(3)*x(4) +
2*m2*x(2)*x(3)*x(4))/(I12 + a^2*m2 + m1*x(2)^2 + m2*x(2)^2 + 2*a*m2*x(2));
    dx4 = (u(2) + a*m2*x(3)^2 + m1*x(2)*x(3)^2 + m2*x(2)*x(3)^2)/(m1 + m2);
    ode = vertcat(dx1, dx2, dx3, dx4);

    %casadi funkcije sustava
    ode_func = Function('ode_func', {x, u}, {ode});

    %Casadi funkcija cilja
    x1df = pi/2; x2df = 1; x3df = 0; x4df = 0; % zeljena konacna stanja
    cost = (x1df - x(1)).^2 + (x2df - x(2)).^2 + (x3df - x(3)).^2 + (x4df -
x(4)).^2;
    cost_func = Function('cost_func', {x, u}, {cost});

    %inicijalizacija optim problema
    opti = casadi.Opti();

    %dfiniranje varijable
    U = opti.variable(nu, N-1);

    % Postavi pocetne uvjete
    X = opti.variable(n, N);
    X(:, 1) = x0;

```

```

%postavljanje ograničenja sustava
for k = 1:N-1
    opti.subject_to(X(:, k + 1) == ode_func(X(:, k), U(:, k)) * tau + X(:, k));
end

% ograničenja na upravljanje
opti.subject_to(U(1, :) <= U1max);
opti.subject_to(U(1, :) >= U1min);
opti.subject_to(U(2, :) <= U2max);
opti.subject_to(U(2, :) >= U2min);

% izbjegavanje kružne prepreke (ograničenje tipa nejednakosti)
Xr = X(2, :).*cos(X(1, :));
Yr = X(2, :).*sin(X(1, :));
opti.subject_to(R^2 - ((Xr-X0).^2+(Yr-Y0).^2) <= 0);

% Postavi početne guess vrijednosti
opti.set_initial(U, zeros(nu, N - 1));

%funk cilja
J = cost_func(X(:, N-1), U(:, N-1));
opti.minimize(J);

opti.solver('ipopt'); %postavljanje solvera

sol = opti.solve(); %rj opt solvera

%Dohvati optimalna rjesenja
u_opt = sol.value(U);
x_opt = sol.value(X);
end

close all
clear all
clc

syms I12 m1 m2 a
syms x1 x2 x3 x4 u1 u2

M11 = I12 + (m1+m2)*x2^2 + 2*m2*a*x2 + m2*a^2;
M22 = m1+m2;

C1 = 2*((m1+m2)*x2+m2*a)*x3*x4;
C2 = -(m1*x2+m2*(a+x2))*x3^2;

M = [M11 0;0 M22];
C = [C1;C2];

dq_vec = [x3;x4];
u_vec = [u1;u2];
ddq_vec = M\u_vec - M\C;

dx = [dq_vec;ddq_vec];

simplify(dx)

```