Optimizacija oblika besposadne letjelice sa sjedinjenim trupom i krilom

Mostarčić, Branimir

FAKULTET STROJARSTVA I BRODOGRADNJE

SVEUČILIŠTE U ZAGREBU

DIPLOMSKI RAD

Student: Branimir Mostarčić

Matični broj: 0035226325

Zagreb, 2024

FAKULTET STROJARSTVA I BRODOGRADNJE

SVEUČILIŠTE U ZAGREBU

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Pero Prebeg

Student: Branimir Mostarčić

Matični broj: 0035226325

Zagreb, 2024

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

Zahvaljujem se: mentoru izv. prof. dr. sc. Peri Prebegu i prof. dr. sc. Milanu Vrdoljaku na savjetima koje su mi pružali tokom rada.

Branimir Mostarčić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija zrakoplovstva/zrakoplovnog inženjerstva

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

DIPLOMSKI ZADATAK

Student: **Branimir Mostarčić** JMBAG: 0035226325

Naslov rada na hrvatskom jeziku: **Optimizacija oblika besposadne letjelice sa sjedinjenim trupom i krilom**

Naslov rada na engleskom jeziku: **Shape optimization of a blended wing body unmanned aerial vehicle**

Opis zadatka:

Konfiguracija zrakoplova sa sjedinjenim trupom i krilom predstavlja konfiguraciju letjelice kod koje su krila i trup ujedinjeni u jednu aerodinamičku cjelinu, što poboljšava učinkovitost i smanjuje otpor. Ova konfiguracija može uključivati manje vertikalne stabilizatore kako bi se poboljšala stabilnost i upravljivost.

U radu je potrebno u programskom jeziku Python izraditi modul za optimizaciju geometrije letjelice sa sjedinjenim trupom i krilom, primjenom prethodno razvijenih matematičkih modela koji su implementirani u Python module.

Zadatak obuhvaća sljedeće:

- formulaciju optimizacijskog problema projektiranja oblika letjelice sa sjedinjenim trupom i krilom koji uključuje provjeru dinamičke uzdužne i bočne stabilnosti
- opis prethodno razvijenih matematičkih modela za definiranje geometrije letjelice sa stopljenim krilom i trupom, određivanje aerodinamičkih karakteristika, izračun inercijskih karakteristika, te provjeru dinamičke stabilnosti
- usporedbu rezultata matematičkih modela iz prethodne točke, koji su implementirani u Python module, s dostupnim rezultatima na nekoliko reprezentativnih modela
- izradu Python modula za optimizaciju oblika letjelice sa sjedinjenim trupom i krilom koji integrira prethodno razvijene module s postojećom implementacijom primjenjivog optimizacijskih algoritama
- provedbu optimizacije oblika besposadne letjelice za dostavu organa za presađivanje primjenom Python modula izrađenog u prethodnoj točci.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
26. rujna 2024.	28. studeni 2024.	5., 6. i 9. prosinca 2024.

Zadatak zadao:

Izv. prof. dr. sc. Pero Prebeg

Predsjednik Povjerenstva:

Prof. dr. sc. Milan Vrdoljak

SADRŽAJ

SADRŽAJ I
POPIS SLIKA III
POPIS TABLICA VI
POPIS OZNAKA VII
SAŽETAK IX
SUMMARY X
1 UVOD 1
2 RJEŠAVANJE OPTIMIZACIJSKIH PROBLEMA 4
2.1 Generalna metodologija 4
2.2 Opis korištenog algoritma 8
2.3 Automatsko deriviranje 10
3 GEOMETRIJA 13
3.1 NURBS krivulja 13
3.2 Elementarna ploha 14
3.3 Spajanje ploha 17
3.4 Diskretizacija geometrije 19
4 GEOMETRIJSKA IZVEDIVOST 22
4.1 Kolizija oplate i tla 22
4.2 Kolizija pogonskih komponenti i tla 23
4.3 Kolizija oplate i pogonskih komponenti 23
4.4 Međusobna kolizija između pogonskih komponenti 24
4.5 Kolizija trupa i unutrašnjih komponenti 24
4.6 Međusobna kolizija između unutarnjih komponenti 25
5 INERCIJSKE ZNAČAJKE 26
5.1 Oplata 26
5.2 Rebra 27
5.3 Ramenjače 28
5.4 „Puno“ krilo i spremnici goriva 29
5.5 Ukupna masa letjelice 31
5.6 Usporedba inercijskih značajki sa programom SolidWorks 32
6 AERODINAMIKA 34
6.1 Formulacija metode panela 34
6.2 Izračun sila i momenata 35
6.3 Provjera aerodinamičkog modela 39

6.3.1 Neviskozno optjecanje sfere ... 39
6.3.2 Usporedba sila i momenata sa programom AVL ... 40
7 DINAMIKA LETA ... 66
7.1 Linearizacija 6 DOF modela ... 69
7.2 Usporedba dinamičkog modela sa programom AVL ... 69
8 POSTAVLJANJE I RJEŠAVANJE OPTIMIZACIJSKOG PROBLEMA ... 70
8.1 Formulacija optimizacijskog problema ... 70
8.2 Rezultati optimizacije ... 72
ZAKLJUČAK ... 83
LITERATURA ... 84

POPIS SLIKA

Slika 1.1 Dijagramski prikaz funkcije za evaluaciju kvalitete unutar programa AVCOPro 2
Slika 1.2 Dijagramski prikaz optimizacijske petlje unutar programa AVCOPro 2

Slika 2.1 Jednostavni računani graf 11

Slika 3.1 Prikaz jedne instance geometrije u izometriji 18
Slika 3.2 Prikaz jedne instance geometrije s prednje strane 18
Slika 3.3 Prikaz jedne instance geometrije sa strane 18
Slika 3.4 Prikaz jedne instance geometrije u odozgo 188
Slika 3.5 Prikaz jedne diskretizirane instance geometrije u izometriji 20
Slika 3.6 Prikaz jedne diskretizirane instance geometrije s prednje strane 20
Slika 3.7 Prikaz jedne diskretizirane instance geometrije sa strane 20
Slika 3.8 Prikaz jedne diskretizirane instance geometrije odozgo 200

Slika 5.1 Testna geometrija u SolidWorks-u sa prikazanom konstrukcijom 32

Slika 6.1 Distribucija koeficijenta tlaka na sferi 39
Slika 6.2 Usporedba distribucija tlakova 39
Slika 6.3 Distribucija koeficijenta tlaka na konfiguracija za usporedbu rezultata sa programom AVL 400
Slika 6.4 Koeficijent sile u smjeru osi x, u ovisnosti o α 411
Slika 6.5 Koeficijent sile u smjeru osi x, u ovisnosti o β 411
Slika 6.6 Koeficijent sile u smjeru osi x, u ovisnosti o $pb/(2V)$ 422
Slika 6.7 Koeficijent sile u smjeru osi x, u ovisnosti o $qc/(2V)$ 422
Slika 6.8 Koeficijent sile u smjeru osi x, u ovisnosti o $rb/(2V)$ 433
Slika 6.9 Koeficijent sile u smjeru osi y, u ovisnosti o α 433
Slika 6.10 Koeficijent sile u smjeru osi y, u ovisnosti o β 444
Slika 6.11 Koeficijent sile u smjeru osi y, u ovisnosti o $pb/(2V)$ 444
Slika 6.12 Koeficijent sile u smjeru osi y, u ovisnosti o $qc/(2V)$ 455
Slika 6.13 Koeficijent sile u smjeru osi y, u ovisnosti o $rb/(2V)$ 455
Slika 6.14 Koeficijent sile u smjeru osi z, u ovisnosti o α 466
Slika 6.15 Koeficijent sile u smjeru osi z, u ovisnosti o β 466
Slika 6.16 Koeficijent sile u smjeru osi z, u ovisnosti o $pb/(2V)$ 477

Slika 6.17 Koeficijent sile u smjeru osi z, u ovisnosti o $qc/(2V)$ 477
Slika 6.18 Koeficijent sile u smjeru osi z, u ovisnosti o $rb/(2V)$ 488
Slika 6.19 Koeficijent momenta oko osi x, u ovisnosti o α 488
Slika 6.20 Koeficijent momenta oko osi x, u ovisnosti o β 49
Slika 6.21 Koeficijent momenta oko osi x, u ovisnosti o $pb/(2V)$ 49
Slika 6.22 Koeficijent momenta oko osi x, u ovisnosti o $qc/(2V)$ 500
Slika 6.23 Koeficijent momenta oko osi x, u ovisnosti o $rb/(2V)$ 500
Slika 6.24 Koeficijent momenta oko osi y, u ovisnosti o α 511
Slika 6.25 Koeficijent momenta oko osi y, u ovisnosti o β 511
Slika 6.26 Koeficijent momenta oko osi y, u ovisnosti o $pb/(2V)$ 522
Slika 6.27 Koeficijent momenta oko osi y, u ovisnosti o $qc/(2V)$ 522
Slika 6.28 Koeficijent momenta oko osi y, u ovisnosti o $rb/(2V)$ 533
Slika 6.29 Koeficijent momenta oko osi z, u ovisnosti o α 533
Slika 6.30 Koeficijent momenta oko osi z, u ovisnosti o β 544
Slika 6.31 Koeficijent momenta oko osi z, u ovisnosti o $pb/(2V)$ 544
Slika 6.32 Koeficijent momenta oko osi z, u ovisnosti o $qc/(2V)$ 555
Slika 6.33 Koeficijent momenta oko osi z, u ovisnosti o $rb/(2V)$ 555
Slika 6.34 Koeficijent sile u smjeru osi x, u ovisnosti otklona krilaca 566
Slika 6.35 Koeficijent sile u smjeru osi x, u ovisnosti otklona kormila visine 566
Slika 6.36 Koeficijent sile u smjeru osi x, u ovisnosti otklona kormila pravca 577
Slika 6.37 Koeficijent sile u smjeru osi y, u ovisnosti otklona krilaca 577
Slika 6.38 Koeficijent sile u smjeru osi y, u ovisnosti otklona kormila visine 588
Slika 6.39 Koeficijent sile u smjeru osi y, u ovisnosti otklona kormila pravca 588
Slika 6.40 Koeficijent sile u smjeru osi z, u ovisnosti otklona krilaca 59
Slika 6.41 Koeficijent sile u smjeru osi z, u ovisnosti otklona kormila visine 59
Slika 6.42 Koeficijent sile u smjeru osi z, u ovisnosti otklona kormila pravca 600
Slika 6.43 Koeficijent momenta oko osi x, u ovisnosti otklona krilaca 600
Slika 6.44 Koeficijent momenta oko osi x, u ovisnosti otklona kormila visine 611
Slika 6.45 Koeficijent momenta oko osi x, u ovisnosti otklona kormila pravca 611
Slika 6.46 Koeficijent momenta oko osi y, u ovisnosti otklona krilaca 622
Slika 6.47 Koeficijent momenta oko osi y, u ovisnosti otklona kormila visine 622
Slika 6.48 Koeficijent momenta oko osi y, u ovisnosti otklona kormila pravca 633
Slika 6.49 Koeficijent momenta oko osi z, u ovisnosti otklona krilaca 633
Slika 6.50 Koeficijent momenta oko osi z, u ovisnosti otklona kormila visine 644
Slika 6.51 Koeficijent momenta oko osi z, u ovisnosti otklona kormila pravca 644

Slika 7.1 Rezultat usporedbe vrijednosti korijena 69

Slika 8.1 Početna geometrija letjelice 720
Slika 8.2 Razvoj funkcije cilja 1..... 722
Slika 8.3 Razvoj funkcije cilja 2..... 73
Slika 8.4 Razvoj funkcije cilja 3..... 733
Slika 8.5 Razvoj funkcije ograničenja jednakosti 1 74
Slika 8.6 Razvoj funkcije ograničenja jednakosti 2 744
Slika 8.7 Razvoj funkcije ograničenja jednakosti 3 75
Slika 8.8 Razvoj funkcije ograničenja jednakosti 4 755
Slika 8.9 Razvoj funkcije ograničenja jednakosti 5 76
Slika 8.10 Razvoj funkcije ograničenja jednakosti 6 766
Slika 8.11 Razvoj funkcije ograničenja jednakosti 7 77
Slika 8.12 Razvoj funkcije ograničenja jednakosti 8 777
Slika 8.13 Razvoj funkcije ograničenja nejednakosti 1..... 78
Slika 8.14 Razvoj funkcije ograničenja nejednakosti 2..... 788
Slika 8.15 Razvoj funkcije ograničenja nejednakosti 3..... 79
Slika 8.16 Razvoj funkcije ograničenja nejednakosti 4..... 79
Slika 8.17 Razvoj funkcije ograničenja nejednakosti 5..... 80
Slika 8.18 Razvoj funkcije ograničenja nejednakosti 6..... 800
Slika 8.19 Razvoj funkcije ograničenja nejednakosti 7..... 81
Slika 8.20 Razvoj funkcije ograničenja nejednakosti 8..... 811

POPIS TABLICA

Tablica 2.1 Prednosti i nedostatci pojedinih metoda deriviranja..... 10

Tablica 5.1 Rezultati usporedbe inercijskih značajki za klasičnu konstrukciju 33
Tablica 5.2 Rezultati usporedbe inercijskih značajki za ispunjeno krilo..... 33

Tablica 8.1 Funkcije cilja 700
Tablica 8.2 Ograničenja jednakosti 71
Tablica 8.2 Ograničenja nejednakosti 711

POPIS OZNAKA

$\mathbf{A}_{\mu}$	Matrica utjecaja dipola
$\mathbf{A}_{\sigma}$	Matrica utjecaja izvora
$\mathcal{B}$	Oznaka za B-Spline krivulju
$\mathbf{CM}$	Centar mase letjelice
$\mathbf{C_p}$	Vektor koeficijenata tlakova na panelima
$c_{\mathrm{A}}(u)$	Distribucija duljine tetive
$\mathbf{F}_{\mathrm{A}}$	Vektor aerodinamičke sile
$\mathbf{F}_{\mathrm{G}}$	Vektor gravitacijske sile
$\mathbf{F}_{\mathrm{T}}$	Vektor pogonske sile
$\mathbf{f}(v;\, u)$	Profilna krivulja
$f(\mathbf{x})$	Funkcija cilja
$\mathbf{g}(u)$	Krivulja vodilja
$\mathbf{g}(\mathbf{x})$	Vektor ograničenja nejednakosti
$(g_{\mathrm{I}})_i(\mathbf{x})$	*i*-to „unutarnje“ ograničenje nejednakosti
$\mathbf{H}$	Eulerova kinematička matrica
$\mathbf{h}(\mathbf{x})$	Vektor ograničenja jednakosti
$\mathbf{I}$	Tenzor inercije letjelice
lb_i	Donja granica *i*-te varijable
$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$	Lagrangeova funkcija
$\hat{\mathcal{L}}(\mathbf{x}, \boldsymbol{\lambda})$	Proširena Lagrangeova funkcija
$\mathbf{M}_{\mathrm{A}}$	Vektor aerodinamičkog momenta
$\mathbf{M}_{\mathrm{T}}$	Vektor pogonskog momenta
m	Masa letjelice
$N_i^k(u)$	*i*-ta bazna funkcija *k*-tog reda
$\mathcal{N}$	Oznaka za NURBS krivulju

p	Kutna brzina oko *x*-osi k.s. letjelice
q	Kutna brzina oko *y*-osi k.s. letjelice
r	Kutna brzina oko *z*-osi k.s. letjelice
$\mathbf{s}(u,v)$	Elementarna ploha
ub_i	Gornja granica *i*-te varijable
$\mathbf{V}$	Vektor brzine letjelice u k.s. letjelice
v_∞	Brzina letjelice (nastrujavanja)
$\mathbf{x}$	Vektor projektnih varijabli
$\mathbf{x}^*$	Vektor optimalnih projektnih varijabli

$\alpha_{\mathrm{A}}(u)$	Distribucija uvijanja
α	Napadni kut
β	Kut klizanja
δ	Bezdimenzijski otklon kontrolne plohe
$\theta_{\mathrm{A}}(u)$	Distribucija dihedrala
$\boldsymbol{\lambda}$	Vektor Lagrangeovih multiplikatora
$\boldsymbol{\mu}$	Vektor jačina dipola na panelima
μ_{g}	Koeficijent penalizacije ograničenja nejednakosti
$\mu_{\mathrm{g_I}}$	Koeficijent penalizacije „unutarnjih“ ograničenja nejednakosti
μ_{h}	Koeficijent penalizacije ograničenja jednakosti
ξ	Intenzitet kolizije
$\boldsymbol{\sigma}$	Vektor jačina izvora na panelima
$\boldsymbol{\Phi}$	Vektor Eulerovih kuteva
$\varphi(\mathbf{x})$	Proširena funkcija cilja
$\boldsymbol{\omega}$	Vektor kutne brzine letjelice u k.s. letjelice

SAŽETAK

U ovom radu, predstavljena je izrada programa AVCOPro (*Aerial Vehicle Construction and Optimization Program*), modula izrađenog u programskom jeziku Python, koji je trenutno primjenjiv samo za optimizaciju projektiranja letjelica sa sjedinjenim trupom i krilom. Formuliran je optimizacijski problem koji uključuje projektiranje oblika takvih letjelica uz provjeru dinamičke uzdužne i bočne stabilnosti. Opisani su prethodno razvijeni matematički modeli za definiranje geometrije letjelica sa stopljenim krilom i trupom, izračun inercijskih značajki, određivanje aerodinamičkih karakteristika, te provjeru dinamičkih karakteristika letjelice. Ovi modeli, implementirani u Python module, uspoređeni su s dostupnim rezultatima na nekoliko reprezentativnih modela radi verifikacije njihove točnosti. Razvijen je Python modul koji integrira prethodno razvijene module s postojećim implementacijama primjenjivih optimizacijskih algoritama, koristeći biblioteku PyTorch za automatsko deriviranje putem računalnog grafa i propagacije unatrag (*backpropagation*). Ovo omogućava efikasno računanje gradijenata potrebnih za optimizaciju koristeći optimizacijski algoritam Adam. Razvijeni modul primijenjen je za optimizaciju oblika besposadne letjelice za dostavu organa za presađivanje. Na kraju su istaknuta ograničenja trenutne verzije programa i dan je pregled smjernica za budući razvoj.

Ključne riječi: AVCOPro, optimizacija, Adam, automatsko deriviranje, geometrija zrakoplova, izvedivost geometrije, inercijske značajke, aerodinamika, dinamika leta, performanse.

SUMMARY

This thesis presents the development of AVCOPro (Aerial Vehicle Construction and Optimization Program), a module created in the Python programming language, that is currently only applicable for optimizing the design of aircraft with blended fuselage and wing configurations. An optimization problem is formulated that includes designing the shape of such aircraft while verifying longitudinal and lateral dynamic stability. Previously developed mathematical models are described for defining the geometry of aircraft with blended wings and fuselage, calculating inertial properties, determining aerodynamic characteristics and verifying dynamic characteristics of the vehicle. These models, implemented in Python modules, are compared with available results on several representative models to verify their accuracy. A Python module has been developed that integrates the previously developed modules with existing implementations of applicable optimization algorithms, using the PyTorch library for automatic differentiation via backpropagation. This enables efficient calculation of gradients needed for optimization using the Adam optimizer. The developed module was applied to optimize the shape of an unmanned aircraft for the delivery of organs for transplantation. Finally, the limitations of the current version of the program are highlighted, and an overview of guidelines for future development is provided.

Key words: AVCOPro, optimization, Adam, automatic differentiation, aircraft geometry, geometric feasibility, inertia features, aerodynamics, flight dynamics, performances.

1 UVOD

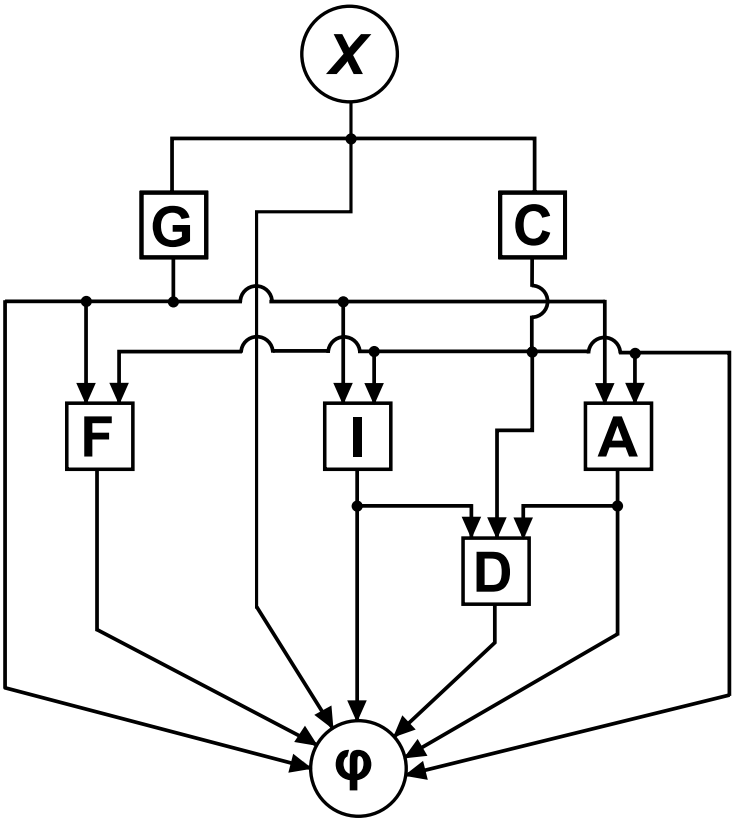
Ideja ovog rada je pokazati da je moguće proces projektirana letjelice automatizirati na način da se cijeli proces projektiranja razmatra kao funkcija, koja na temelju odluka (projektnih varijabli) vraća veličinu, koja je mjera kvalitete izabranih varijabli. Štoviše, moguće je ne samo zadovoljiti zahtijevana ograničenja (ukoliko su fizikalno izvediva) već i postići optimalne karakteristike, koje inače ne bi bilo lako postići na tradicionalni način, radi velikog broja varijabli, koje imaju utjecaj na više aspekata kvalitete letjelice. Srž ovog pristupa je matematičko modeliranje fizikalnih procesa, koji su relevantni pri evaluaciji kvalitete letjelice. Ti matematički modeli ne samo da moraju uhvatiti sve važne fenomene modelirane fizike, već moraju uz to biti i primjenjivi u kontekstu optimizacije a to znači da ti modeli moraju biti: brzi, stabilni te imati mogućnost efikasnog računanja gradijenata, čineći proces stvaranja takvih modela vrlo zahtjevnim.

Opisat će se predloženi program AVCOPro (*Aerial Vehicle Construction And Optimization Program*) za optimizaciju letjelica, koji je u trenutku pisanja ovog rada još uvijek u razvoju, te kao takav, ima određene limite (npr. trenutno se mogu modelirati samo leteća krila, BWB konfiguracije sa stabilizacijskim plohama i neke konfiguracije kod kojih nema izraženog trupa) te nedostatke (npr. nema modula za proračun čvrstoće konstrukcije i optimalnu kontrolu letjelice). Uz to, brzine leta mogu ići maksimalno do 0.3 Macha te je zbog toga korištenje trenutne verzije programa ograničeno na manje bespilotne letjelice, pa će se u skladu sa navedenim ograničenjima, na kraju riješiti jedan primjer sa konkretnim ciljevima i ograničenjima. Slika 1.1 prikazuje dijagram toka programa (funkcije za evaluaciju kvalitete) na najvišoj razini. Slova u dijagramu predstavljaju:

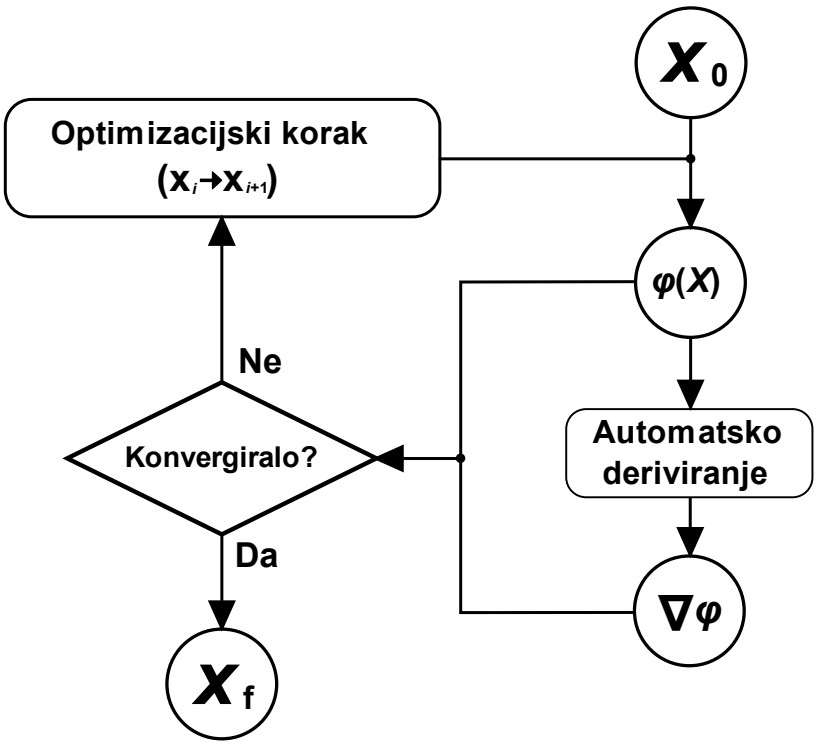
- X – projektne varijable,
- G (*Geometry*) – modul za generiranje geometrije,
- C (*Components*) – modul za modeliranje različitih komponenti (pogon, podvozje itd.),
- F (*Feasibility*) – modul za određivanje izvedivosti geometrije,
- I (*Inertia*) – modul za stvaranje inercijskog modela,
- A (*Aerodynamics*) – modul za stvaranje aerodinamičkog modela,
- D (*Dynamics*) – modul za stvaranje dinamičkog modela i
- φ – konačna vrijednost funkcije (sadrži ciljeve i ograničenja).

Radi velike opširnosti, neće se opisivati svi detalji programa, već će se opisati generalne ideje implementacije pojedinih modula. Isto tako, kako je modul „C“ svojevrsni *input* u program (ovisi koji se motori razmatraju, koje baterije itd.), on se neće posebno razmatrati.

Dalje na slici 1.2, vidi se optimizacijska petlja unutar programa. Važno je uočiti da je funkcija $\varphi(\mathbf{x})$ prikazana na slici 1.2, funkcija koja je prikazana na slici 1.1.



Slika 1.1 Dijagramski prikaz funkcije za evaluaciju kvalitete unutar programa AVCOPro



Slika 1.2 Dijagramski prikaz optimizacijske petlje unutar programa AVCOPro

Prije opisa sastavnih dijelova programa, potrebno je opisati kako se pristupa optimizacijskim problemima u općem slučaju.

2 RJEŠAVANJE OPTIMIZACIJSKIH PROBLEMA

2.1 Generalna metodologija

Općeniti optimizacijski problem sa ograničenjima, sastoji se od funkcije cilja koju želimo minimizirati (ili maksimizirati), te funkcija ograničenja jednakosti i nejednakosti. Taj problem se može zapisati na slijedeći način. Za funkciju $f(\mathbf{x})$, potrebno je pronaći točku $\mathbf{x}^*$, za koju vrijede sljedeće relacije:

$$f(\mathbf{x}^*) = \min_{\mathbf{x}}\big(f(\mathbf{x})\big), \tag{2.1}$$

$$\mathbf{h}(\mathbf{x}^*) = 0, \tag{2.2}$$

$$\mathbf{g}(\mathbf{x}^*) \geq 0, \tag{2.3}$$

$$lb_i \leq x_i \leq ub_i, \tag{2.4}$$

gdje je:
f – funkcija koju želimo minimizirati/maksimizirati,

$\mathbf{x}$ – nezavisne varijable funkcije f,

$\mathbf{x}^*$ – optimalna točka,

$\mathbf{h}$ – funkcije ograničenja jednakosti,

$\mathbf{g}$ – funkcije ograničenja nejednakosti,

lb_i – donja granica intervala, u kojem se i-ta varijabla može kretati i

ub_i – gornja granica intervala, u kojem se i-ta varijabla može kretati.

Valja ovdje naglasiti da funkcija f zapravo predstavlja linearnu kombinaciju više funkcija cilja, čime se može provoditi višeciljna optimizacija. Funkcija f u proširenom obliku glasi [1]:

$$f(\mathbf{x}) = \sum_{i=1}^{n_f} w_i f_i(\mathbf{x}), \tag{2.5}$$

gdje je $f_i(\mathbf{x})$ i-ta funkcija cilja a w_i težinski faktor, kojim se određuje utjecaj i-tog cilja.
Gornje relacije se mogu riješiti pomoću više algoritama, no na najvišem nivou, dijele se na algoritme koji koriste gradijente i algoritme koji ne koriste gradijente. Zadnja skupina ima mogućnost raditi sa diskretnim varijablama, te će sa većom vjerojatnošću pronaći globalni optimum funkcije, tj. rješenje će u tom slučaju biti manje osjetljivo na inicijalne vrijednosti

varijabli, no za probleme gdje ima puno varijabli i gdje je evaluacija funkcija relativno skupa, ti algoritmi jednostavno postaju neprihvatljivi, stoga se većinom za složene optimizacijske probleme koriste algoritmi, koji se baziraju na gradijentima funkcija. Bez obzira na nedostatke, ti algoritmi mogu biti atraktivan način da se pronađe „optimalna“ startna točka za algoritme koji se baziraju na gradijentima

Algoritmi koji se baziraju na gradijentima, mogu se u grubo podijeliti na algoritme koji koriste isključivo informacije prvog reda (gradijente) i algoritme koji koriste informacije drugog reda (gradijente i Hesseove matrice). Algoritmi koji koriste informacije drugog reda (to su algoritmi bazirani na Newtonovoj metodi) će u pravilu trebat manje iteracija od prve skupine (onih koji koriste informacije prvog reda), što je i logično, pošto barataju sa više informacija o geometriji funkcije, no nažalost, jedna iteracija je puno skuplja u odnosu na prvu skupinu, upravo zbog računanja drugih derivacija, što znači da je prednost tih algoritama ujedno i mana. Nedostatci algoritama koji rade sa informacijama drugog reda se mogu premostiti na način da se informacije drugog reda izračunaju na temelju informacija prvog reda (tzv. kvazi-Newtonove metode), no za sada se ti algoritmi neće razmatrati radi složenije integracije sa ostatkom programa, te će se zato u programu koristiti algoritam koji radi sa informacijama prvog reda. Konkretno, koristit će se algoritam Adam, koji bude opisan kasnije. Osim toga, algoritmi koji rade sa informacijama prvog reda, prirodno traže minimum (ili maksimum ako se tako formulira problem), pa nije potrebno provjeravati da li je pronađena stacionarna točka ujedno i minimum (ili maksimum). Nedostatak ovog tipa algoritma je da zahtijevaju namještanje tzv. hiperparametara, kako bi bili učinkoviti.

Najprije, potrebno je riješiti jedan problem, a to je kako riješiti sustav (2.1) – (2.3). Naime, odabran algoritam (Adam), „zna“ rješavati samo optimizacijske probleme bez ograničenja, što znači da je potrebno svesti sustav (2.1) – (2.3) na optimizacijski problem bez ograničenja.

Postoj više pristupa kako se to radi a za početak, objasni će se metoda penalizacije. U toj metodi, umjesto minimizacije funkcije cilja, minimizira se proširena funkcija cilja φ, koja ima sljedeći oblik [1]:

$$\varphi(\mathbf{x}) = f(\mathbf{x}) + \frac{\mu_{\mathrm{h}}}{2}\mathbf{h}(\mathbf{x})^{\boldsymbol{T}}\mathbf{h}(\mathbf{x}) + \frac{\mu_{\mathrm{g}}}{2}\min(\mathbf{g}(\mathbf{x}), \mathbf{0})^{\boldsymbol{T}}\min(\mathbf{g}(\mathbf{x}), \mathbf{0}), \qquad (2.6)$$

gdje veličine μ_{h} i μ_{g} predstavljaju koeficijente penalizacije za jednakosti odnosno nejednakosti, pri čemu vrijedi $\mu_{\mathrm{h}} > 0$ i $\mu_{\mathrm{g}} > 0$. Interpretacija dodanih penalizacijskih funkcija je intuitivna – penalizacijske funkcije su nenegativne, što znači da će se globalni minimum tih funkcija postići kada su jednake nuli tj. kad su sva ograničenja zadovoljena, pri čemu se stupanj penalizacije kontrolira penalizacijskim koeficijentima. Geometrijski, uvođenjem tih funkcija, originalna funkcija cilja počinje poprimati veće vrijednosti na mjestima domene gdje ograničenja nisi zadovoljena, dok na mjestima gdje su ograničenja zadovoljena, funkcija ostaje nepromijenjena. To stvara svojevrsne „kanjone“, koji postaju strmiji povećanjem penalizacijskih koeficijenata.

Time se može „natjerati“ optimizatora da stavi prioritet na zadovoljenje ograničenja, no nažalost, povećavanjem penalizacijskih koeficijenata, geometrija proširene funkcije cilja postaje vrlo strma, čime se može destabilizirati proces konvergencije, jer bilo kakva devijacija od ograničenja, uzrokuje naglo povećanje funkcije cilja i gradijenta. Zbog toga, ti koeficijenti se postepeno povećavaju sa iteracijama od neke početne vrijednosti.
Ponekad, postoje ograničenja nejednakosti, koja moraju uvijek biti zadovoljena, bez obzira na trenutno stanje varijabli. Zbog toga, u funkciju (2.5), dodaje se tzv. barijerna funkcija [1]:

$$\varphi(\mathbf{x}) = f(\mathbf{x}) + \frac{\mu_{\mathrm{h}}}{2}\mathbf{h}(\mathbf{x})^{\boldsymbol{T}}\mathbf{h}(\mathbf{x}) + \frac{\mu_{\mathrm{g}}}{2}\min(\mathbf{g}(\mathbf{x}), \mathbf{0})^{\boldsymbol{T}}\min(\mathbf{g}(\mathbf{x}), \mathbf{0}) + \sum_{\boldsymbol{i=1}}^{n_{g_{\mathrm{I}}}} \frac{\mu_{\mathrm{g_I}}}{(g_{\mathrm{I}})_i(\mathbf{x})}, \qquad (2.7)$$

gdje je $(g_{\mathrm{I}})_i(\mathbf{x})$ ograničenje nejednakosti, koje mora ostati unutar podobnog područja tokom optimizacije a $\mu_{\mathrm{g_I}}$ je penalizacijski koeficijent za barijernu funkciju te mora vrijediti: $\mu_{\mathrm{g_I}} > 0$.
Ako se pogleda dodana barijerna funkcija, vidi se da se vrijednost proširene funkcije cilja povećava kako se nejednakost približava rubu podobnog područja, ne bi li postalo beskonačno velika na samom rubu. Za razliku od prije uvedenih penalizacijskih funkcija, penalizacijski koeficijent uz barijernu funkciju mora biti čim manji, kako se ne bi nepotrebno smanjilo podobno područje. Barijerna funkcija se obično koristi za kontrolu matematičkih modela, koji su nedefinirani izvan određenog područja.
Kao što je već spomenuto, problem sa ovakvim pristupom je da pre velike vrijednosti penalizacijskih koeficijenata mogu destabilizirati konvergenciju. To se primarno odnosi na koeficijent uz ograničenja jednakosti μ_{h}, jer jednakosti drastično smanjuju podoban prostor (smanjuju ga za jednu dimenziju), što nije slučaj za ograničenja nejednakosti (nema redukcije dimenzije), pa se postavlja pitanje, da li je moguće izbjeći visoke vrijednosti koeficijenta μ_{h} a bez da se kompromitira zadovoljenje ograničenja jednakosti. Kako bi se odgovorilo na to pitanje, potrebno je detaljnije pogledati zahtjeve (2.1) i (2.2). Za početak, pretpostavimo da imamo samo ograničenja jednakosti. Ukoliko se (2.1) i (2.2) razviju u Taylorov red (zanemarujući članove višeg reda), dobiva se:

$$\nabla f(\mathbf{x}^*)\boldsymbol{\delta}^T \geq 0, \qquad (2.8)$$

$$\nabla \mathbf{h}(\mathbf{x}^*)\boldsymbol{\delta}^T = \mathbf{0}, \qquad (2.9)$$

gdje je $\boldsymbol{\delta}$ vektor, koji predstavlja podoban smjer tj. smjer u kojem se smijemo kretati, bez da se prekrše ograničenja. Relacija (2.8) nalaže da usmjerena derivacija funkcije cilja mora biti pozitivna u podobnom smjeru. S druge strane, iz relacije (2.9), vidi se da podoban smjer za slučaj jednog ograničenja, mora ležati u tangencijalnoj ravnini na nivo-plohu, koja predstavlja jedno ograničenje. To svojstvo mora biti zadovoljeno za sva ograničenja, pa će to biti moguće jedino ako svi gradijenti funkcija ograničenja (redci Jakobijana $\nabla \mathbf{h}(\mathbf{x}^*)$) leže u hiper ravnini, koja je okomita na podoban smjer. Kako predznak vektora $\boldsymbol{\delta}$ ne igra ulogu, slijedi da za relaciju

(2.8) da vrijedi $\nabla f(\mathbf{x}^*)\boldsymbol{\delta}^T = 0$, što znači da i gradijent funkcije cilja leži u istoj hiper ravnini. To znači da je moguće gradijent funkcije cilja u optimalnoj točki, prikazati linearnom kombinacijom gradijenata ograničenja jednakosti na slijedeći način:

$$\nabla f(\mathbf{x}^*) = -\boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*), \tag{2.10}$$

gdje je $\boldsymbol{\lambda}$ vektor Lagrangeovih multiplikatora. Relacija (2.10) vrijedi za bilo koje nivo-plohe $\mathbf{h}(\mathbf{x}^*)$, no nas zanimaju samo ona rješenja za koja vrijedi:

$$\mathbf{h}(\mathbf{x}^*) = \mathbf{0}. \tag{2.11}$$

Jednadžbe (2.10) i (2.11), mogu se kompaktno zapisati pomoću Lagrangeove funkcije:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}). \tag{2.12}$$

Deriviranjem funkcije (2.12) po projektnim varijablama $\mathbf{x}$ i Lagrangeovim multiplikatorima $\boldsymbol{\lambda}$, te izjednačavanjem tih derivacija sa nulom, dobivaju se uvjeti (2.10) i (2.11):

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}) = \nabla f(\mathbf{x}^*) + \boldsymbol{\lambda}^T \nabla \mathbf{h}(\mathbf{x}^*) = \mathbf{0}, \tag{2.13}$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}) = \mathbf{h}(\mathbf{x}^*) = \mathbf{0}. \tag{2.14}$$

Dakle, stacionarna točka Lagrangeove funkcije, predstavlja optimalnu točku funkcije cilja, koja zadovoljava sva ograničenja jednakosti. Ključna riječ je *stacionarna* točka, ne minimum ili maksimum. Kako nas zanima minimzacija funkcije cilja, tada se traži da za $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$, $\nabla f(\mathbf{x}^*) = 0$ bude minimum, pa će onda i (2.13) biti minimum. S druge strane, (2.14) mora biti maksimum. Razlog tomu jest činjenica da želimo da Lagrangeova funkcija bude čim veća za zadani skup ograničenja koja nisu zadovoljena. Na taj način se pri minimizaciji Lagrangeove funkcije po projektnim varijablama, maksimalno penaliziraju ograničenja, koja nisu zadovoljena. Time se može riješiti optimizacijski problem sa ograničenjima jednakosti, bez potrebe da se uvode penalizacijski koeficijenti, koji mogu destabilizirati proces konvergencije. Za maksimalno ubrzanje konvergencije, u programu AVCOPro, koristi se proširena Lagrangeova funkcija, koja je kombinacija metode penalizacije i Lagrangeove metode. Proširena Lagrangeova funkcija ima oblik kao (2.12), samo što se umjesto čiste funkcije cilja, ubacuje funkcija $\varphi(\mathbf{x})$ (vidi (2.7)) [1]:

$$\hat{\mathcal{L}}(\mathbf{x}, \boldsymbol{\lambda}) = \varphi(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x}). \tag{2.15}$$

Korištenjem proširene Lagrangeove funkcije, mogu se izbjeći visoki penalizacijski koeficijenti uz funkcije ograničenja jednakosti, čime se stabilizira konvergencija.

Konačna stvar koja je iznimno važna za uspješnu konvergenciju je izotropnost projektnog prostora. Pod „izotropnost“, misli se na to da se sve varijable mijenjaju u rasponu koji je istog reda veličine za sve varijable. Npr., moguće je imati u optimizacijskom zadatku varijable koje se mijenjaju u rasponu reda veličine 10^{-3} mjernih jedinica i varijable koje se mijenjaju u

rasponu reda veličine 10^3 mjernih jedinica. Očito je da će se brojčano puno više mijenjati zadnja skupina varijabli, iako možda prva skupina ima značajniji utjecaj na funkciju cilja i/ili ograničenja. Isto vrijedi i za vrijednosti funkcija cilja i ograničenja jer bi npr. neki ciljevi mogli „zasjeniti“ druge, iako su možda po prioritetu međusobno ravnopravni.

Kako bi se postigla izotropnost varijabli, koristi se sjedeća transformacija za *i*-tu varijablu:

$$\tilde{x}_i = \frac{x_i - lb_i}{ub_i - lb_i}, \tag{2.16}$$

čime se dobiva izvrsno svojstvo: $\tilde{x}_i \in [0, 1]$. Osim bolje konvergencije, poboljšava se i floating-point aritmetika.
Za funkcije cilja i ograničenja, koriste se sljedeće relacije:

$$\tilde{f}(\tilde{\mathbf{x}}) = \sum_{i=1}^{n_f} w_i \frac{f_i(\tilde{\mathbf{x}})}{|f_i(\tilde{\mathbf{x}})| + 1}, \tag{2.17}$$

$$\tilde{h}_j(\tilde{\mathbf{x}}) = \frac{h_j(\tilde{\mathbf{x}})}{|h_j(\tilde{\mathbf{x}})| + 1}, \tag{2.18}$$

$$\tilde{g}_k(\tilde{\mathbf{x}}) = \frac{g_k(\tilde{\mathbf{x}})}{|g_k(\tilde{\mathbf{x}})| + 1}. \tag{2.19}$$

Za funkcije (2.17) - (2.19), vrijedi: $\tilde{f}(\tilde{\mathbf{x}}), \tilde{h}_j(\tilde{\mathbf{x}}), \tilde{g}_k(\tilde{\mathbf{x}}) \in [-1, 1]$ za sve $\tilde{\mathbf{x}}$.
Sljedeće, potrebno je iskoristiti neki algoritam za minimizaciju funkcija, kako bi riješili optimizacijski zadatak.

2.2 Opis korištenog algoritma

Optimizacijski algoritam Adam (*Adaptive Moment Estimation*), originalno je bio razvijen za potrebe strojnog učenja, no kako ima svojstva koja su pogodna za rješavanje šireg skupa problema te pošto ga je lako integrirati sa programom, ovaj algoritam je izabran. Optimizacijski algoritmi koji rade sa gradijentima, svode se na odabir optimalnog koraka tj. optimalne aktualizacije varijabli. Ukoliko imamo varijable $\mathbf{x}_i$ funkcije cilja *f* u *i*-toj iteraciji, tada će varijable u sljedećoj iteraciji po algoritmu Adam biti [2]:

$$\mathbf{m}_{i+1} = \beta_1 \mathbf{m}_i + (1 - \beta_1) \nabla f(\mathbf{x}_i), \tag{2.20}$$

$$\mathbf{v}_{i+1} = \beta_2 \mathbf{v}_i + (1 - \beta_2) \nabla f(\mathbf{x}_i)^2, \tag{2.21}$$

$$\widehat{\mathbf{m}}_{i+1} = \frac{\mathbf{m}_{i+1}}{1-\beta_1^{i+1}}, \tag{2.22}$$

$$\hat{\mathbf{v}}_{i+1} = \frac{\mathbf{v}_{i+1}}{1-\beta_2^{i+1}}, \tag{2.23}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \frac{\widehat{\mathbf{m}}_{i+1}}{\sqrt{\hat{\mathbf{v}}_{i+1}} + \varepsilon}, \tag{2.24}$$

gdje je:
α – parametar za kontrolu veličine koraka (*learning rate*),

β_1 – parametar za kontrolu utjecaja gradijenata iz prijašnjih iteracija,

β_2 – parametar za kontrolu adaptacije koraka i

ε – broj (obično 10^{-8}), koji je postavljen radi izbjegavanja potencijalnog dijeljenja sa nulom.

Sve operacije u jednadžbama od (2.20) – (2.24) su operacije *element-po-element*.

Prvi korigirani moment $\widehat{\mathbf{m}}$, predstavlja prosjek svih gradijenata, s time da „stariji“ gradijenti imaju sve manji utjecaj. Time gradijent dobiva svojevrsnu „inerciju“, ukoliko se gradijent ne mijenja puno od iteracije do iteracije pa se time napredak ubrzava što daje ovom algoritmu mogućnost da „preskoči“ lokalne minimume. S druge strane, ukoliko gradijent naleti na naglo povišenje funkcije cilja (što je često kod penalizacijskih metoda), ponašanje gradijenta će se prigušiti, pošto će uprosječenje trenutnog gradijenta (koji je naglo promijenio smjer i magnitudu) sa prošlima dati gradijent koji je „pitomiji“, čime se stabilizira konvergencija. Korijenom drugog korigiranog momenta $\hat{\mathbf{v}}$, normaliziraju se gradijenti, te se na taj način manji gradijenti pojačavaju, a veći prigušuju čime se postiže ravnomjerni napredak, ukoliko je funkcija cilja u jednom smjeru strma a u drugom ravna.

U kontekstu rješavanog problema (optimizacijski problem sa ograničenjima), pravilo aktualiziranja projektnih varijabli će biti:

$$\mathbf{m}_{i+1} = \beta_1 \mathbf{m}_i + (1-\beta_1)\, \nabla_{\mathbf{x}} \hat{\mathcal{L}}(\mathbf{x}_i, \boldsymbol{\lambda}_i), \tag{2.25}$$

$$\mathbf{v}_{i+1} = \beta_2 \mathbf{v}_i + (1-\beta_2) \nabla_{\mathbf{x}} \hat{\mathcal{L}}(\mathbf{x}_i, \boldsymbol{\lambda}_i)^2, \tag{2.26}$$

$$\widehat{\mathbf{m}}_{i+1} = \frac{\mathbf{m}_{i+1}}{1-\beta_1^{i+1}}, \tag{2.27}$$

$$\hat{\mathbf{v}}_{i+1} = \frac{\mathbf{v}_{i+1}}{1-\beta_2^{i+1}}, \tag{2.28}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \frac{\hat{\mathbf{m}}_{i+1}}{\sqrt{\hat{\mathbf{v}}_{i+1}} + \varepsilon}, \tag{2.29}$$

$$\boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}_i + \mu_{\mathrm{h}} \mathbf{h}(\mathbf{x}_i). \tag{2.30}$$

Parametri: α, β_1, β_2 (tzv. hiperparametri), obično imaju vrijednosti: $\alpha = 0.001$, $\beta_1 = 0.9$ i $\beta_2 = 0.999$.

2.3 Automatsko deriviranje

Kako bi se primijenio gore navedeni algoritam, potrebno je izračunati tražene gradijente. Gradijente je moguće računati na tri načina:

- analitički/simbolički,
- metodom konačnih razlika i
- automatskim deriviranjem.

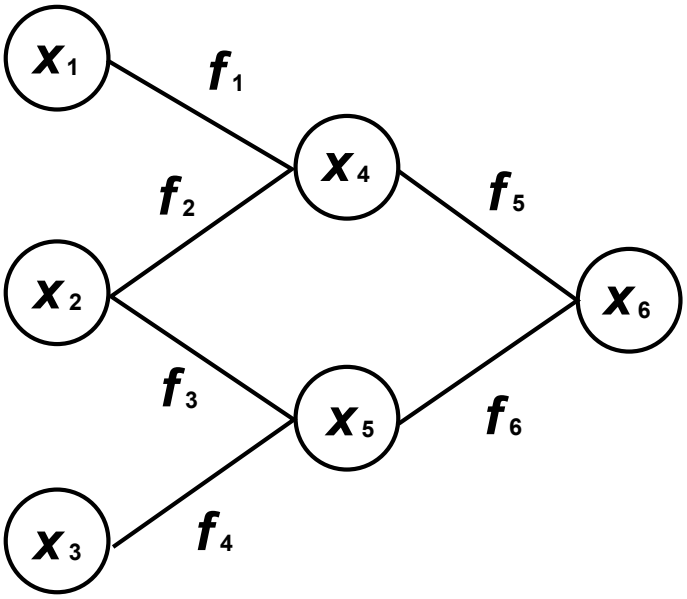
Prednosti i nedostatci pojedine metode, mogu se vidjeti u tablici 2.1.

Tablica 2.1 Prednosti i nedostatci pojedinih metoda deriviranja

Metoda	Prednost/i	Nedostatak/ci
Analitičko/simboličko deriviranje	• Apsolutna točnost	• Nije primjenjivo na složene funkcije/programe (loše skaliranje)
Metoda konačnih razlika	• Jednostavnost • Primjenjivo na proizvoljno složene funkcije/programe	• Podložno pogreškama uslijed zaokruživanja • Sporo računanje gradijenata velikih dimenzija
Automatsko deriviranje	• Točnost do strojne preciznosti • Primjenjivo na proizvoljno složene funkcije/programe • Brzo računanje gradijenata velikih dimenzija	• Složenija primjena • Može biti memorijski zahtjevno (ovisno o korištenoj varijanti)

Prema tablici 2.1, vidi se da automatsko deriviranje praktički kombinira točnost analitičkog/simboličkog deriviranja i fleksibilnost metode konačnih razlika. Uz te prednosti, nudi mogućnost da se brzo izračuna gradijent funkcije sa puno varijabli, te je zato ova metoda vrlo atraktivna u strojnom učenju, gdje se broj varijabli za najbolje modele mjeri u milijardama, pa je stoga automatsko deriviranje uključeno u program AVCOPro.

Kako bi se objasnio princip rada automatskog deriviranja, promotrimo na slici 2.1 jednostavan program/računalni graf tj. funkciju sa tri ulazne varijable (x_1, x_2 i x_3) i jednom izlaznom varijablom (x_6).



Slika 2.1 Jednostavni računani graf

Kao što se vidi sa slike 2.1, računalni graf se sastoji od varijabli (čvorova) i funkcija (spojnica). Matematički, računalni graf sa slike možemo zapisati na slijedeći način:

$$x_6 = x_6(x_4(x_1, x_2), x_5(x_2, x_3)) \tag{2.31}$$

Gradijent varijable x_6 će glasiti:

$$\begin{bmatrix} \frac{\partial x_6}{\partial x_1} \\ \frac{\partial x_6}{\partial x_2} \\ \frac{\partial x_6}{\partial x_3} \end{bmatrix}^T = \begin{bmatrix} \frac{\partial x_6}{\partial x_4}\frac{\partial x_4}{\partial x_1} \\ \frac{\partial x_6}{\partial x_4}\frac{\partial x_4}{\partial x_2} + \frac{\partial x_6}{\partial x_5}\frac{\partial x_5}{\partial x_2} \\ \frac{\partial x_6}{\partial x_5}\frac{\partial x_5}{\partial x_3} \end{bmatrix}^T \tag{2.32}$$

Bitno je uočiti iz (2.32), da se komponente gradijenta sastoje od jednostavnih dijelova, konkretno, jedna komponenta je sastavljena od niza derivacija susjednih varijabli, što znači da možemo izračunati cijeli gradijent neke potencijalno složene funkcije na način da gledamo elementarne dijelove, od kojih je ta funkcija načinjena. Postoje dva moda računanja gradijenata pomoću automatskog deriviranja. Prvi mod se zove *propagacija unaprijed*, te je efikasniji za slučajeve gdje funkcija ima više izlaznih nego ulaznih argumenata, dok se drugi mod zove *propagacija unatrag*, te je efikasniji kada funkcija ima više ulaznih od izlaznih argumenata. Kako je drugi slučaj puno češći, opisat će se drugi mod. Kako bi *propagacija unatrag* funkcionirala, potrebno je sastaviti računalni graf. Računalni graf sadržava informaciju o tome kako su varijable povezane, kako su nastale i koje su im vrijednosti. Stoga je za računanje gradijenata potrebno prvo proći kroz funkciju, te zabilježiti njenu strukturu (*forward pass*). Kada smo došli do kraja funkcije, kreće propagacija unatrag, gdje se računaju derivacije elementarnih funkcija, te se množe i zbrajaju, ovisno o strukturi računalnog grafa kroz koji se prolazi. Kada se kaže *elementarne* funkcije, ne misli se samo na klasične elementarne funkcije iz matematike već i na složenije rutine, kao npr. rješavanje linearnog sustava.
Zbog potrebe za stvaranjem računalnog grafa, *propagacija unatrag* je memorijski skuplja, no kako je već rečeno, ujedno i češća varijanta u praksi.

3 GEOMETRIJA

Kao što je već navedeno u uvodu, jedine geometrije koje se mogu trenutno modelirati su geometrije uzgonskih poha (krila, stabilizacijskih ploha, te trupova koji su u obliku uzgonskih tijela). Sve te geometrije se mogu modelirati kao plohe jednostavnom superpozicijom dvije krivulje. Te plohe se onda međusobno spajaju G1 kontinuitetom, da bi se dobila ukupna geometrija letjelice. Kako bi se dao konkretan oblik plohe, potrebno je najprije promotriti fundamentalni dio modeliranih ploha – NURBS krivulju.

3.1 NURBS krivulja

Razlog zbog kojeg su izabrane NURBS krivulje kao fundamentalni dio jest činjenica da su NURBS krivulje izrazito fleksibilne. Sa malo stupnjeva slobode, mogu poprimiti gotovo bilo koji oblik pri čemu je utjecaj bilo kojeg stupnja slobode krivulje lokaliziran, što je definitivno poželjno kada je u pitanju optimizacija oblika. NURBS krivulja $\mathcal{N}$, definirana je na slijedeći način:

$$\mathcal{N} = \mathcal{N}(u; \mathbf{K}, \mathbf{w}, n, k) = \frac{\mathcal{B}(u; \mathbf{wK}, n, k)}{\mathcal{B}(u; \mathbf{w}, n, k)}, \tag{3.1}$$

gdje je:
u – slobodni parametar ($u \in [0, 1]$),

K – vektor kontrolnih točaka,

w – vektor težinskih faktora,

n – broj kontrolnih točaka, umanjen za jedan,

k – red krivulje a

$\mathcal{B}$ predstavlja B-Spline krivulju, koja je za neki skup čvorova **P**, definirana kao:

$$\mathcal{B}(u; \mathbf{P}, n, k) = \sum_{i=0}^{n} P_i N_i^k(u), \tag{3.2}$$

gdje $N_i^k(u)$ predstavlja *i*-tu baznu funkciju *k*-tog reda, koja je definirana preko Cox-de Boorove rekurzivne formule [3]:

$$N_i^k(u) = \begin{cases} \begin{cases} 1 \text{ za } (u \in [u_i, u_{i+1}\rangle \text{ ako } u \in [0,1\rangle) \vee (u \in [u_i, u_{i+1}] \text{ ako } u = 1) \\ 0 \text{ za } (u \notin [u_i, u_{i+1}\rangle \text{ ako } u \in [0,1\rangle) \vee (u \notin [u_i, u_{i+1}] \text{ ako } u = 1) \end{cases} \text{za } k = 1 \\ \dfrac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \dfrac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u) \text{ za } k \geq 2 \end{cases} . \quad (3.3)$$

Od sada pa nadaje, koristit će se notacija $\mathcal{N}(u; \mathbf{K}, \mathbf{w}, n, k)$, za NURBS krivulju i pri tome treba imati na umu da **K** i **w** predstavljaju stupnjeve slobode krivulje. Isto tako, kada se kaže „krivulja“, ne misli se nužno na geometrijsku krivulju u prostoru, već se može misliti i na različite distribucije veličina.

3.2 Elementarna ploha

Kao što je već rečeno na početku ovog poglavlja, plohe u vidu se mogu modelirati superpozicijom dvije krivulje; krivulje vodilje **g** i profilne krivulje **f**:

$$\mathbf{s}(u, v) = \mathbf{g}(u) + \mathbf{f}(v;\, u), \quad (3.4)$$

gdje su *u* i *v* slobodni parametri, koji se kreću u rasponu od 0 do 1.
Krivulja vodilja $\mathbf{g}(u)$ je jednostavno NURBS krivulja u 3D prostoru:

$$\mathbf{g}(u) = \mathcal{N}\left(u; \mathbf{P}_\mathbf{g}, \boldsymbol{w}_\mathrm{g}, k_\mathrm{g}\right), \quad (3.5)$$

a parametri $\mathbf{P}_\mathbf{g}, \boldsymbol{w}_\mathrm{g}$ će biti projektne varijable, kojima se kontrolira oblik krivulje vodilje.
Krivulja **f** je krivulja, koja je dobivena nizom transformacija krivulja jediničnih aeroprofila i zato je prisutan još dodatni parametar *u*, kojim se kontrolira transformacija, ovisno o tome gdje se „kvači“ krivulja aeroprofila na krivulju vodilju. Aeroprofili sa tetivom jedinične duljine, kao profilne krivulje, nalaze se u ne transformiranom stanju u *x*-*z* ravnini, sa napadnim rubom u ishodištu i izlaznim bridom na koordinati (1, 0, 0). Te krivulje imaju oblik:

$$\tilde{\mathbf{f}}(v; u) = [v \quad 0 \quad \tilde{z}_\mathrm{A}(v; u)]^T, \quad (3.6)$$

$$\tilde{z}_\mathrm{A}(v; u) = \tilde{z}_{\mathrm{A}1}(v) \cos\left(\frac{\pi}{2}u\right)^2 + \tilde{z}_{\mathrm{A}2}(v) \sin\left(\frac{\pi}{2}u\right)^2, \quad (3.7)$$

gdje su $\tilde{z}_{\mathrm{A}1}(v)$ i $\tilde{z}_{\mathrm{A}2}(v)$ funkcije, koje predstavljaju gornju ili donju krivulju početnog (na početku krivulje vodilje) odnosno krajnjeg (na kraju krivulje vodilje) aeroprofila te imaju oblik (*v* se može zamijeniti i sa koordinatom *x* u ovom slučaju):

$$\tilde{z}_{\mathrm{A}i}(v) = \mathcal{N}(v; \mathbf{P}_{\mathrm{f}i}, \boldsymbol{w}_{\mathrm{f}i}, k_{\mathrm{f}i}). \tag{3.8}$$

Parametri $\mathbf{P}_{\mathrm{f}i}$ i $\boldsymbol{w}_{\mathrm{f}i}$, mogu se ostaviti kao projektne varijable ili se mogu namjestiti tako da funkcija $\tilde{z}_{\mathrm{A}i}(v)$ aproksimira već postojeći aeroprofil, pomoću metode najmanjih kvadrata. Zadnja metoda je trenutno jedini siguran način da se dobi funkcija (3.8), zato što trenutna verzija programa nema ugrađenu analizu 3D graničnog sloja, koja je neophodna kada je u pitanju optimizacija aeroprofila.
Kako je poželjno modelirati kontrolne plohe, potrebno je uvesti relacije koje transformiraju krivulju aeroprofila u neutralnom obliku, u oblik aeroprofila sa otklonjenom (zarotiranom) plohom. Ukoliko označimo sa c_{cs} postotak duljine tetive kontrolne plohe (dakle udaljenost od izlaznog brida do točke rotacije kontrolne plohe jediničnog aeroprofila) te ako se kontrolna ploha otkloni za δ_{cs}, tada će funkcija (3.7) poprimiti slijedeći oblik:

$$\tilde{z}_{\mathrm{CA}}(v;\, u) = \begin{cases} \tilde{z}_{\mathrm{A}}(v; u) \;\text{ za } v \in [0, 1 - c_{\mathrm{cs}}\rangle \\ \beta \mathbf{R}_y(\delta_{\mathrm{cs}})\big(\tilde{z}_{\mathrm{A}}(v;\, u) - \tilde{z}_{\mathrm{A}}(1 - c_{\mathrm{cs}}; u)\big) + \tilde{z}_{\mathrm{A}}(1 - c_{\mathrm{cs}}; u) \text{ za } v \in [1 - c_{\mathrm{cs}}, 1]' \end{cases} \tag{3.9}$$

gdje je β predstavlja faktor skaliranja segmenta krivulje te je definiran kao omjer duljine segmenta krivulje nakon i prije otklona. Razlikujemo faktor skaliranja za gornju (β_{u}) i donju (β_{d}) krivulju:

$$\beta_{\mathrm{u}} = \frac{l_{\mathrm{u}}^{\delta}}{l_{\mathrm{u}}}, \qquad \beta_{\mathrm{d}} = \frac{l_{\mathrm{d}}^{\delta}}{l_{\mathrm{d}}}. \tag{3.10}$$

Ako se ti segmenti aproksimiraju pravcima, tada će primjenom kosinusovog poučka duljene prije i poslije otklona biti:

$$l_{\mathrm{u}} = \sqrt{\frac{b^2}{4} + \delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2 - b\sqrt{\delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2}\cos(\gamma)}, \tag{3.11}$$

$$l_{\mathrm{u}}^{\delta} = \sqrt{\frac{b^2}{4} + \delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2 - b\sqrt{\delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2}\cos(\gamma + \delta_{\mathrm{cs}})}, \tag{3.11}$$

$$l_{\mathrm{d}} = \sqrt{\frac{b^2}{4} + \delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2 - b\sqrt{\delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2}\cos(\pi - \gamma)}, \tag{3.12}$$

$$l_{\mathrm{d}}^{\delta} = \sqrt{\frac{b^2}{4} + \delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2 - b\sqrt{\delta_{\mathrm{cs}}^2 + (h_{\mathrm{cs}})^2}\cos(\pi - \gamma - \delta_{\mathrm{cs}})}, \tag{3.13}$$

gdje je:
b – debljina aeroprofila na mjestu rotacije kontrolne plohe,

h_{cs} – z koordinata srednje linije aeroprofila na mjestu rotacije kontrolne plohe i

γ – kut između z osi i pravca koji spaja centar rotacije sa izlaznim bridom.

b, h_{cs} i γ, definirani su slijedećim jednadžbama:

$$b = \tilde{z}_{\mathrm{Au}}(1 - c_{\mathrm{cs}}; u) - \tilde{z}_{\mathrm{Ad}}(1 - c_{\mathrm{cs}}; u), \tag{3.14}$$

$$h_{\mathrm{cs}} = \frac{\tilde{z}_{\mathrm{Au}}(1 - c_{\mathrm{cs}}; u) + \tilde{z}_{\mathrm{Ad}}(1 - c_{\mathrm{cs}}; u)}{2}, \tag{3.15}$$

$$\gamma = \frac{\pi}{2} - \arctan\left(\frac{h_{\mathrm{cs}}}{c_{\mathrm{cs}}}\right), \tag{3.16}$$

gdje se naravno indeks „u“ odnosi na gornji a „d“ na donji aeroprofil.
Konačno, možemo krivulju $\tilde{\mathbf{f}}(v; u)$ transformirati u krivulju $\mathbf{f}(v;\ u)$ na slijedeći način:

$$\mathbf{f}(v;\ u) = c_{\mathrm{A}}(u)\mathbf{R}_{\mathrm{x}}\big(\theta_{\mathrm{A}}(u)\big)\mathbf{R}_{y}\big(\alpha_{\mathrm{A}}(u)\big)\big(\tilde{\mathbf{f}}(v; u) - [1 - c_{\mathrm{cs}} \quad 0 \quad h_{\mathrm{cs}}]^{T}\big), \tag{3.17}$$

$$c_{\mathrm{A}}(u) = \mathcal{N}\big(u; \mathbf{p}_{c_{\mathrm{A}}}, \boldsymbol{w}_{c_{\mathrm{A}}}, k_{c_{\mathrm{A}}}\big), \tag{3.18}$$

$$\theta_{\mathrm{A}}(u) = \mathcal{N}\big(u; \mathbf{p}_{\theta_{\mathrm{A}}}, \boldsymbol{w}_{\theta_{\mathrm{A}}}, k_{\theta_{\mathrm{A}}}\big), \tag{3.19}$$

$$\alpha_{\mathrm{A}}(u) = \mathcal{N}\big(u; \mathbf{p}_{\alpha_{\mathrm{A}}}, \boldsymbol{w}_{\alpha_{\mathrm{A}}}, k_{\alpha_{\mathrm{A}}}\big), \tag{3.20}$$

gdje je:
$c_{\mathrm{A}}(u)$ – distribucija duljine tetive (skaliranje aeroprofila),

$\alpha_{\mathrm{A}}(u)$ – distribucija uvijanja (rotacija aeroprofila oko y-osi) i

$\theta_{\mathrm{A}}(u)$ – distribucija dihedrala (rotacija aeroprofila oko x-osi).

Važno je naglasiti da distribucija dihedrala zapravo odgovara distribuciji kuta između y-osi i projiciranog vektora tangente krivulje vodilje tj.:

$$\theta_{\mathrm{A}}(u) = \arctan\left(\frac{\frac{dz_{\mathrm{g}}}{du}}{\frac{dy_{\mathrm{g}}}{du}}\right). \tag{3.21}$$

Elementarnu plohu je moguće prilagođavati na različite načine, ne bi li se dobile specifične plohe za određene dijelove letjelice. Npr. trup se modelira kao elementarna ploha koja ima kontrolne točke u *x-y* ravnini, bez uvijanja i dihedrala. S druge strane, za segment gdje se nalazi kontrolna ploha, krivulja vodilja mora biti pravac, kako bi se osigurala mogućnost instalacije rotirajuće kontrolne plohe a posljedično tome, distribucija duljine tetive na tom segmentu treba biti linearna, dok je distribucija uvijanja proizvoljna.

3.3 Spajanje ploha

Kao što je već spomenuto, ukupna geometrija letjelice se dobiva spajanjem segmenata plohe. Taj spoj se ostvaruje G1 kontinuitetom, ne bi li se sačuvao jedan stupanj slobode, koji bi inače bio izgubljen kada bi se plohe spajale C1 kontinuitetom. Pošto se segmenti spajaju duž raspona, uvjet za spajanje plohe $\mathbf{s}_2$ na plohu $\mathbf{s}_1$ glasi:

$$\frac{\partial \mathbf{s}_2}{\partial u} = \chi \frac{\partial \mathbf{s}_1}{\partial u}, \tag{3.22}$$

gdje je χ realni broj, koji predstavlja konstantu skaliranja.
Ovdje se neće izvoditi uvjeti spajanja, pošto je izvod dosta složen, nego će se samo navesti da ostvarivanjem uvjeta (3.22), svaki sastavni dio plohe gubi stupnjeve slobode. Pa tako će kod krivulje vodilje prva točka biti unaprijed određena, dok će druga trebat ležat na određenom pravcu. Kod distribucija geometrijskih veličina, „izgube“ se prve dvije kontrolne točke.

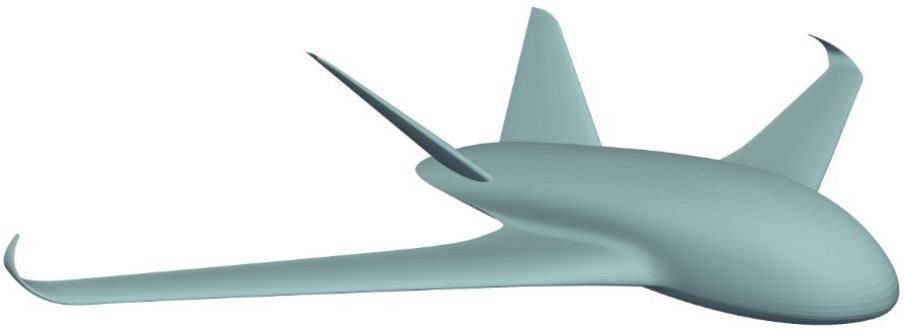
Osim spajanja segmenata, moguće je i „prikvačiti“ jednu plohu na drugu. To se čini na način da se definira tzv. *korijenska* ploha, koja se spaja na *roditeljsku* plohu pomoću presječne krivulje (krivulje koja se dobije kao presjek između *roditeljske* plohe i plohe *rezne* plohe). Ako označimo *roditeljsku* plohu sa $\mathbf{s}_\mathrm{p}$ a *reznu* sa $\mathbf{s}_\mathrm{c}$, tada će presječna krivulja $\mathbf{f}_\cap$ bit:

$$\mathbf{f}_\cap(v_\mathrm{c}) = \mathbf{s}_\mathrm{p}\big(u_\mathrm{p}, v_\mathrm{p}\,\big) \cap \mathbf{s}_\mathrm{c}(u_\mathrm{c}, v_\mathrm{c}\,), \tag{3.23}$$

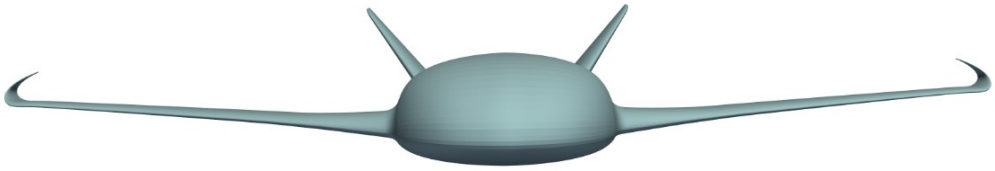
gdje je v_c slobodni parametar, pa se može označiti i sa *v*. Važno je naglasiti da je ploha $\mathbf{s}_\mathrm{c}$ ograničena na način da joj je krivulja vodilja pravac (to je bitno radi daljnje konstrukcije korijenske plohe). Kako je ploha $\mathbf{s}_\mathrm{c}$ istog oblika kao i (3.4), tada se iz te plohe može izdvojiti krivulja $\mathbf{s}_\mathrm{c}(1, v)$. Linearnom interpolacijom krivulja $\mathbf{f}_\cap(v)$ i $\mathbf{s}_\mathrm{c}(1, v)$, dobiva se *korjenska* ploha

$$\mathbf{s}_\mathrm{r}(u, v) = \mathbf{f}_\cap(v)(1 - u) + \mathbf{s}_\mathrm{c}(1, v)u. \tag{3.24}$$

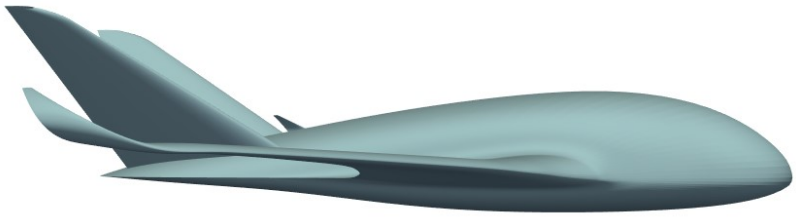
Primjer geometrije (ujedno i geometrije koja se ubacuje u optimizacijsku petlju kao početna geometrija), koja se može postići gore prikazanim modeliranjem, vidi se na slikama 3.1. – 3.4.



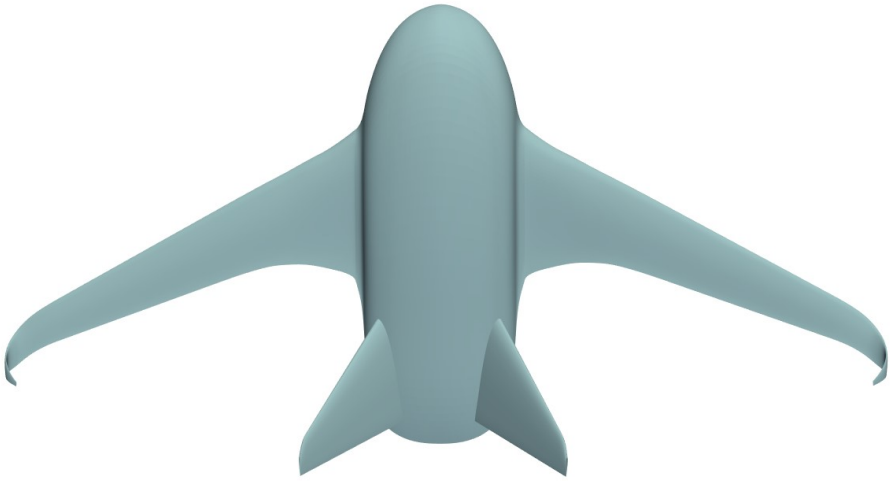
Slika 3.1 Prikaz jedne instance geometrije u izometriji



Slika 3.2 Prikaz jedne instance geometrije s prednje strane



Slika 3.3 Prikaz jedne instance geometrije sa strane



Slika 3.4 Prikaz jedne instance geometrije u odozgo

Kao što je već spomenuto, elementarna ploha je glavni dio sa kojim se stvara geometrija. No geometrija se može podijeliti na veće dijelove – klastere. Klasteri predstavljaju skup segmenata (elementarnih ploha), te čine neki konkretni dio letjelice. Tako npr. gore prikazana geometrija se sastoji od tri klastera:

- trup,
- krila i
- stabilizatori.

To je bitno radi dodjeljivanja svojstava svakom klasteru (npr. konstrukcijski elementi krila će se zasigurno razlikovati od konstrukcije vertikalnih stabilizatora).

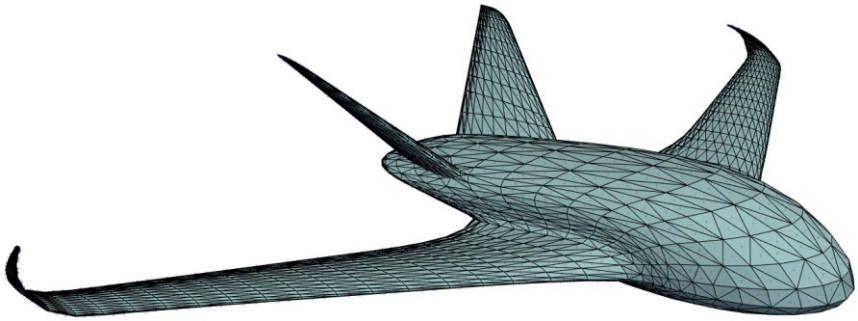
3.4 Diskretizacija geometrije

Nakon definirane plohe, potrebno je stvorit mrežu plohe, koja se dalje koristi u drugim modulima. Izabrana je trokutasta mreža, pošto je trokutima uvijek moguće prekriti bilo kakvu plohu. Kako je proces stvaranja mreže dosta složen, neće se detaljno opisivati, već će se navesti glavne ideje.

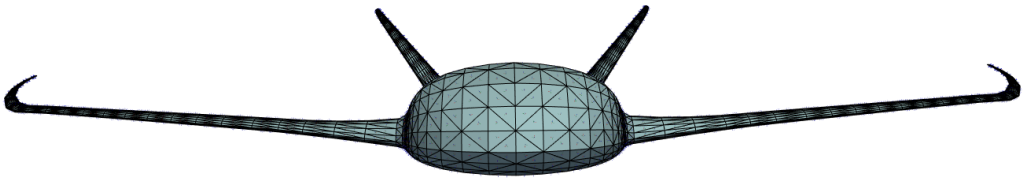
U kontekstu stvaranja plošne mreže, razlikujemo dvije situacije. Prva situacija je da se mreža stvara na krilu a druga da se stvara na trupu. Razlog razlikovanju te dvije situacije jest da se na trupu mogu nalaziti uzgonske plohe (kao na prethodnim slikama), što znači da se mora koristiti dodatni mehanizam, pomoću kojeg će se stvarati mreža oko uzgonske plohe, koja se veže na trup.

Ukoliko se stvara mreža na krilu, dovoljno je stvoriti strukturirani oblak točaka, čijim se spajanjem (indeksiranjem) dobiva mreža.

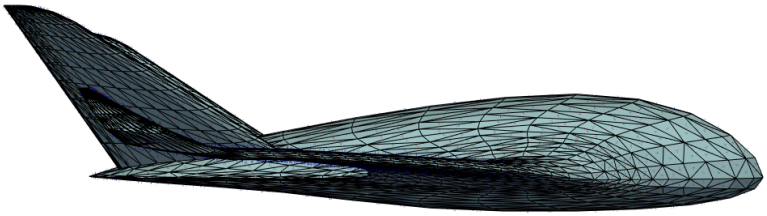
U slučaju da se stvara mreža na trupu sa uzgonskom plohom, onda se stvaranje mreže odvija u četiri koraka. Prvo se izračuna oblak točaka (kao kod krila), no ovaj put u parametarskom prostoru plohe (*u*-*v* prostor). Nakon toga, detektiraju se sve točke unutar poligona, koji predstavlja presječnu krivulju uzgonske plohe i trupa, te se miču iz oblaka točaka. Sljedeće, stvara se mreža u parametarskom prostoru pomoću Delaunayeve triangulacije i na posljetku, potrebno je maknuti sve trokute koju su generirani unutar krivulje tj. trokute, čiji vrhovi leže na točkama presječne krivulje. To se čini detekcijom težišta trokuta unutar poligona presječne krivuje, nakon čega se konačno dobiva gotova mreža. Na slikama 3.5 – 3.8, vidi se diskretizirana geometrija sa prethodnih slika.



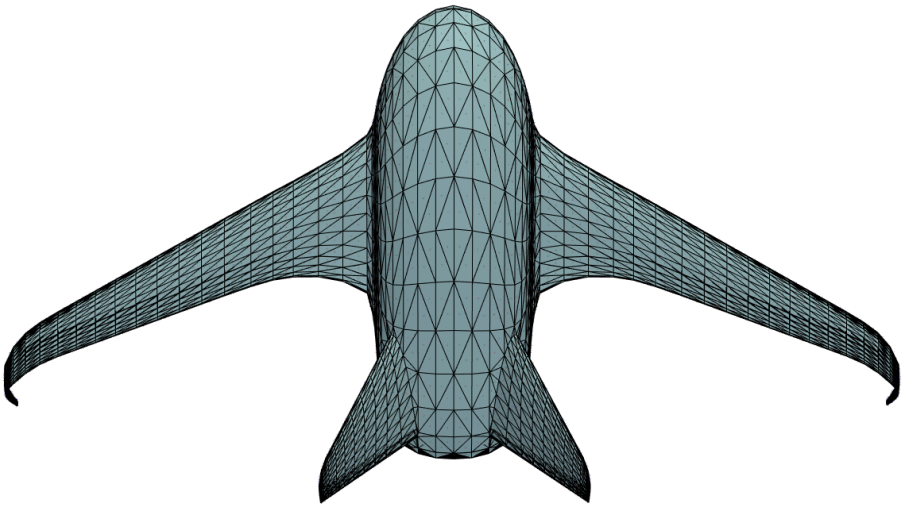
Slika 3.5 Prikaz jedne diskretizirane instance geometrije u izometriji



Slika 3.6 Prikaz jedne diskretizirane instance geometrije s prednje strane



Slika 3.7 Prikaz jedne diskretizirane instance geometrije sa strane



Slika 3.8 Prikaz jedne diskretizirane instance geometrije odozgo

Diskretizacija plohe je rađena na način da mreža postaje sve finija kako se približava vrhovima krila, čime se pospješuje konvergencija aerodinamičke analize.

Važno je još naglasiti na kraju ovog poglavlja, da je koordinatni sustav u kojem se definira geometrija uvijek orijentiran na način da je: *x*-os usmjerena duž osi trupa unatrag, *y*-os u smjeru desnog krila a *z*-os je onda prema pravilu desne ruke usmjerena prema gore. Taj koordinatni sustav će se zvati *konstrukcijski* koordinatni sustav.

4 GEOMETRIJSKA IZVEDIVOST

Prilikom optimizacije, potrebno je osigurati da dobivena geometrija bude fizikalna tj. ne smije biti nikakvih fizičkih kolizija. Moguće su slijedeće kolizije:

- kolizija oplate i tla,
- kolizija pogonskih komponenti i tla,
- kolizija oplate i pogonskih komponenti,
- međusobna kolizija između pogonskih komponenti,
- kolizija trupa i unutarnjih komponenti i
- međusobna kolizija između unutarnjih komponenti.

Svaka kolizija je predstavljena negativnim brojem (intenzitet kolizije), koji raste kako kolizija postaje izraženija. Taj se broj koristi kao ograničenje nejednakosti, koji se implicitno ubacuje u proširenu funkciju cilja. Pri analizi kolizija, važna je pretpostavka da su dimenzije svih pogonskih jedinica iste te da se mogu modelirati kao cilindri radijusa r i visine h. Uz te pretpostavke, pretpostavlja se da se ti cilindri mogu rotirati isključivo oko y-osi te da te rotacije nisu velike. Također, unutrašnje komponente, modeliraju se ako kutije dimenzija $l_x \times l_y \times l_z$.

4.1 Kolizija oplate i tla

Kako bi se odredila kolizija između oplate i tla, prvo se definira detekcijska ravnina na način da se odredi referentna točka (obično je to koordinata prednjeg ili stražnjeg podvozja) te vektor normale. Vektor normale može biti određen pomoću kontaktnih točaka podvozja ili nagiba letjelice (npr. pri polijetanju). Kako u ovom trenutku imamo dostupnu mrežu tj. vrhove mreže, ono što će se gledati jest da li su vrhovi točaka ispod te detekcijske ravnine. Ukoliko označimo referentnu točku detekcijske ravnine sa $\mathbf{P}_0$, vektor normale sa $\mathbf{n}$ i ako vrhove mreže stavimo u matricu $\mathbf{V}$ (svaki stupac predstavlja koordinate jedne točke), tada će se intenzitet kolizije ξ odrediti na slijedeći način:

$$\xi = \sum \min\left(\mathbf{n}^T(\mathbf{V} - \mathbf{P}_0), 0\right). \tag{4.1}$$

4.2 Kolizija pogonskih komponenti i tla

Ovaj se problem svodi na problem iz pod poglavlja 4.1. Izazov je ovdje pronaći najnižu točku pojedine pogonske jedinica. Imajući na umu činjenicu da se cilindri koji predstavljaju pogonske jedinice mogu rotirati oko *y*-osi, u općem slučaju, najniža točka $\mathbf{P}_\mathrm{b}$ će biti:

$$\mathbf{P}_\mathrm{b} = \mathbf{R}_y(\alpha_\mathrm{T})\left(\mathbf{P}_0 + \begin{bmatrix} \mathrm{sign}(\alpha_\mathrm{T} + \alpha_0)\dfrac{h}{2} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix}\right), \tag{4.2}$$

gdje je:
$\mathbf{P}_0$ – ishodište pogonske jedinice,

α_T – zakret pogonske jedinice oko *y*-osi i

α_0 – zakret ravnine (letjelice) oko *y*-osi.

Sada sve što se mora učiniti za određivanje intenziteta kolizije je primijeniti izraze prikazane u pod poglavlju 4.1, na točke $\mathbf{P}_\mathrm{b}$.

4.3 Kolizija oplate i pogonskih komponenti

U ovom slučaju, pogonske komponente (propeleri ili mlazni motori) su predstavljene cilindrima odgovarajućih dimenzija (polumjer *r* i visina *h*), pa se problem svodi na traženje vrhova mreže, koji se nalaze unutar tih cilindara. Pravac $\mathbf{p}$, koji definira os cilindra, definiran je kako slijedi:

$$\mathbf{p} = \mathbf{P}_0 + t\mathbf{d}, \tag{4.3}$$

gdje je $\mathbf{P}_0$ točka u sredini cilindra a $\mathbf{d}$ je jedinični vektor smjera pravca (cilindra).
Kako bi se detektiralo neku točku unutar cilindra, potrebno je prvo odrediti najmanju udaljenost između točke i osi cilindra. Ukoliko je udaljenost manja od polumjera, tada je točka kandidat za daljnju provjeru a u suprotnom se odbacuje. Najmanje udaljenosti između vrhova $\mathbf{V}$ i pravca $\mathbf{p}$, dane su slijedećom relacijom:

$$\mathbf{L_{min}} = \|\mathbf{V} - (\mathbf{P}_0 + \mathbf{t}_\mathrm{min} \odot \mathbf{d})\|, \tag{4.4}$$

$$\mathbf{t}_\mathrm{min} = \mathbf{d}^T(\mathbf{V} - \mathbf{P}_0), \tag{4.5}$$

gdje operator $\odot$ predstavlja operaciju množenja *element-po-element*. Dalje, označimo vrhove $\mathbf{V}$ za koje vrijedi $\mathbf{L_{min}} < r$ sa $\mathbf{V}_\mathrm{r}$. U tom slučaju, vrhovi će biti unutar cilindra ako ispunjavaju slijedeći uvjet:

$$\mathbf{d}^T\left(\mathbf{V}_\mathrm{r}-\left(\mathbf{P}_0+\frac{h}{2}\mathbf{d}\right)\right)\leq 0\cap-\mathbf{d}^T\left(\mathbf{V}_\mathrm{r}-\left(\mathbf{P}_0-\frac{h}{2}\mathbf{d}\right)\right)\leq 0. \tag{4.6}$$

Ako se broj vrhova unutar cilindra označi sa n_c, tada će intenzitet kolizije ξ biti:

$$\xi=-n_\mathrm{c}h+\sum\min(\mathbf{L_{min}}-r,0)\,. \tag{4.7}$$

4.4 Međusobna kolizija između pogonskih komponenti

Za detekciju kolizije između pogonskih jedinica, jednostavno se provjerava da li je su središta pogonskih jedinica (cilindara), projiciranih na *y*-*z* ravninu, na udaljenosti koja je manja od 2*r*. Ovdje je pretpostavljeno da zakret pogonske jedinice oko *y*-osi nije velik, što je i više nego razumna pretpostavka. Ako imamo *n* pogonskih jedinica sa projiciranim ishodištima $\mathbf{P}_{yz}$, tada će intenzitet kolizije ξ biti:

$$\xi=\sum_{i=1,\ i\neq j}^{n}\sum_{j=1,\ j\neq i}^{n}\min\left(\left\|\left(\mathbf{P}_{yz}\right)_i-\left(\mathbf{P}_{yz}\right)_j\right\|-2r,0\right). \tag{4.8}$$

4.5 Kolizija trupa i unutrašnjih komponenti

Za ovaj problem, moglo bi se iskoristiti praćenje zraka (*ray tracing*), no umjesto toga, iskoristit će se geometrija trupa. Kao što je već spomenuto pred kraj trećeg poglavlja, trup se modelira kao ploha bez uvijanja i dihedrala., što znači da su svi aeroprofili u ravninama, koje su paralelne sa ravninom simetrije (*x*-*z* ravninom), pa se može jednostavno provjeriti da li je vrh kutije iznad donje i ispod gornje plohe. Da bi se to provjerilo, potrebno je prvo pronaći parametre plohe *u* i *v*, koji daju *x* i *y* koordinate na plohama, koje se poklapaju sa *x* i *y* koordinatama vrha kutije. Uzimajući u obzir prethodno navedeno, intenzitet kolizije ξ između trupa i unutarnjih komponenti dobiva se preko slijedećih relacija:

$$\xi=\sum\min\left(\min\left(z_\mathrm{v}-z_{\mathrm{s_d}}(u_\mathrm{cd},v_\mathrm{cd}),\ z_{\mathrm{s_u}}(u_\mathrm{cu},v_\mathrm{cu})-z_\mathrm{v}\right),0\right), \tag{4.9}$$

gdje je:

z_v – *z* koordinata vrha komponente,

$z_{\mathrm{s_d}}$ – *z* koordinata donje plohe,

z_{s_u} – *z* koordinata gornje plohe a

u_c i v_c su parametri, za koje se *x* i *y* koordinate na plohama poklapaju sa *x* i *y* koordinatama vrhova, te ih se dobiva iz sljedećeg sustava(isto je za gornju i donju plohu):

$$\begin{aligned} x_{\mathrm{s}}(u_{\mathrm{c}}, v_{\mathrm{c}}) - x_{\mathrm{v}} &= 0 \\ y_{\mathrm{s}}(u_{\mathrm{c}}, v_{\mathrm{c}}) - y_{\mathrm{v}} &= 0. \end{aligned} \tag{4.10}$$

Ukoliko nije moguće pronaći parametre u_{c} i v_{c} (vrh se uopće ne nalazi iznad trupa), to će prouzrokovati rezidual sustava (4.10), koji je različit od nule, pa se ta veličina može koristiti kao intenzitet kolizije.

4.6 Međusobna kolizija između unutarnjih komponenti

Kako se komponente modeliraju kao kutije, koje su poravnate sa osima koordinatnog sustava, detekcija kolizije dvije kutije je vrlo jednostavna. Za intenzitet kolizije ξ, izabrao se volumen preklapanja dvije kutije, pa će za *n* komponenti intenzitet kolizije biti:

$$\xi = \sum_{i=1,\ i\neq j}^{n} \sum_{j=1,\ j\neq i}^{n} V_{ij}, \tag{4.11}$$

$$V_{ij} = \prod_{k=1}^{3} \max\left(\min\left(\left(x_{k,\max}\right)_i, \left(x_{k,\max}\right)_j\right) - \max\left(\left(x_{k,\min}\right)_i, \left(x_{k,\min}\right)_j\right), 0\right), \tag{4.12}$$

gdje je $x_{k,\max}$ gornja granica kutije u smjeru *k*-te koordinate a $x_{k,\min}$ donja granica kutije u smjeru *k*-te koordinate.

5 INERCIJSKE ZNAČAJKE

Klasična konstrukcija aviona, sastoji se od tri glavne komponente:

- oplate,
- rebara i
- ramenjače.

Nazivi gornjih komponenti mogu varirati, ovisno o tome gdje se nalaze (da li su na trupu ili krilu), no njihova uloga se ne mijenja značajno. Ponekad (to primarno vrijedi za male bespilotne letjelice), krila su ispunjena nekom masom kao npr. stiropor. Isto tako, u krila se redovito smješta gorivo radi oslobođenja prostora u trupu i rasterećenja konstrukcije krila u letu. I na kraju, svaki avion ima neke pomoćne sustave, pogonske sustave (osim jedrilica) i naravno, koristan teret. Sve te komponente se moraju uzeti u obzir kako bi se ispravno izračunale inercijske značajke aviona, koje su neophodne za ispravno određivanje stabilnosti i dinamičkih značajki letjelice.

5.1 Oplata

Kako bi se dobila masa oplate, najjednostavnije je iskoristiti već stvorenu plošnu mrežu. To se čini tako da se izračuna površina svakog trokuta, te se pomnoži sa debljinom i gustoćom oplate. Na taj se način postupak izračuna inercijskih značajki plohe, može provesti vrlo brzo. Zbrajanjem svih trokuta, dobiva se ukupna masa oplate. Masa oplate m_{skin} iskazana je sljedećom jednadžbom:

$$m_{\mathrm{skin}} = \sum_{i=1}^{n_{\mathrm{C}}} (\rho_{\mathrm{skin}})_i (t_{\mathrm{skin}})_i \sum_{j=1}^{n_{\mathrm{P}i}} A_{ij}, \tag{5.1}$$

gdje je:
n_{C} – broj klastera,

$n_{\mathrm{P}i}$ – broj panela na *i*-tom klasteru,

$(\rho_{\mathrm{skin}})_i$ – gustoća oplate na *i*-tom klasteru,

$(t_{\mathrm{skin}})_i$– debljina oplate na *i*-tom klasteru i

A_{ij} – površina *j*-tog panela na *i*-tom klasteru.

Kao i masa, centar mase oplate $\mathbf{CM}_{\mathrm{skin}}$ također se određuje iz geometrije mreže na sljedeći način:

$$\mathbf{CM}_{\text{skin}} = \sum_{i=1}^{n_{\text{C}}} \frac{(\rho_{\text{skin}})_i (t_{\text{skin}})_i}{m_{\text{skin}}} \sum_{j=1}^{n_{\text{P}i}} \mathbf{CM}_{\text{P}ij} A_{ij}, \tag{5.2}$$

gdje je $\mathbf{CM}_{\text{P}ij}$ centar mase individualnog panela.

Tenzor inercije oplate $\mathbf{I}_{\text{skin}}$, računa se na analogan načina kao i prethodne dvije veličine – preko mreže:

$$\mathbf{I}_{\text{skin}} = \sum_{i=1}^{n_{\text{C}}} (\rho_{\text{skin}})_i (t_{\text{skin}})_i \sum_{j=1}^{n_{\text{P}i}} \begin{bmatrix} y_{\text{CM}_{\text{P}ij}}^2 + z_{\text{CM}_{\text{P}ij}}^2 & 0 & -x_{\text{CM}_{\text{P}ij}} z_{\text{CM}_{\text{P}ij}} \\ 0 & x_{\text{CM}_{\text{P}ij}}^2 + z_{\text{CM}_{\text{P}ij}}^2 & 0 \\ -x_{\text{CM}_{\text{P}ij}} z_{\text{CM}_{\text{P}ij}} & 0 & x_{\text{CM}_{\text{P}ij}}^2 + y_{\text{CM}_{\text{P}ij}}^2 \end{bmatrix} A_{ij}, \tag{5.3}$$

gdje su: $x_{\text{CM}_{\text{P}i}}$, $y_{\text{CM}_{\text{P}i}}$ i $z_{\text{CM}_{\text{P}i}}$, koordinate centra mase *i* – tog panela. Bitno je uočiti da je pri izračunu tenzora inercije, zanemarena geometrija individualnog panela, te se svaki panel tretira kao jedna točka. Uspostavilo se da taj pristup daje točne rezultate uz zanemarive pogreške. Komponente I_{xy} i I_{zy} nisu uključene, pošto će finalni tenzor inercije imati nule na tim mjestima, zbog simetrije oko *x-z* ravnine. Također bitno za naglasiti jest da se tenzor inercije računa oko ishodišta. To je napravljeno da se ne treba voditi računa o globalnom centru mase, koji u ovom trenutku još nije poznat. Ukupni tenzor inercije cijele letjelice (sa komponentama) će naravno bit transformiran u onaj, koji je definiran oko centra mase. Na isti način će se računati tenzori inercije ostalih komponenti

5.2 Rebra

Masa rebara se dobiva na analogan način kao i masa oplate, samo što se sada koristi površina rebara a ne površina panela. Jednadžba za izračun mase rebara m_{ribs}, glasi:,

$$m_{\text{ribs}} = \sum_{i=1}^{n_{\text{C}}} (\rho_{\text{ribs}})_i (t_{\text{ribs}})_i \sum_{j=1}^{(n_{\text{ribs}})_i} A_{\text{R}ij}, \tag{5.4}$$

gdje je:

$(n_{\text{ribs}})_i$ – broj rebara na *i*-tom klasteru,

$(\rho_{\text{ribs}})_i$ – gustoća rebara na *i*-tom klasteru,

$(t_{\text{ribs}})_i$ – debljina rebara na *i*-tom klasteru i

$A_{\text{R}i_i}$ – površina jednog rebra *j*-tog rebra na *i*-tom klasteru.

Za izračun centra mase svih rebara, pretpostavit će se da se centar mase individualnog rebra nalazi na sredini između točke prednjeg i izlaznog brida. Ta pretpostavka je provjerena, te se

došlo do zaključka da nema značajnih odstupanja u odnosu na korištenje pravog centra mase individualnog rebra. To je još dodatno legitimno radi činjenice da se rebra u realnim situacijama izrađuju kao štapna konstrukcija, pa tražiti centar mase „punog“ rebra postaje manje realno. Izračun centra mase svih rebara $\mathbf{CM}_{\mathrm{ribs}}$ glasi:

$$\mathbf{CM}_{\mathrm{ribs}} = \sum_{i=1}^{n_{\mathrm{c}}} \frac{(\rho_{\mathrm{ribs}})_i (t_{\mathrm{ribs}})_i}{m_{\mathrm{ribs}}} \sum_{j=1}^{(n_{\mathrm{ribs}})_i} \left(\frac{\mathbf{r}_{\mathrm{LE}ij} + \mathbf{r}_{\mathrm{TE}ij}}{2} \right) A_{\mathrm{R}ij}, \tag{5.5}$$

gdje je $\mathbf{r}_{\mathrm{LE}ij}$ položaj točke na prednjem bridu aeroprofila a $\mathbf{r}_{\mathrm{TE}ij}$ točka na izlaznom bridu aeroprofila.

Za izračun tenzora inercije rebara, uvodi se pretpostavka da se rebra mogu tretirati kao štapovi, čija duljina odgovara duljini tetive aeroprofila. Na taj način su se izbjegle transformacije (koje bi se u protivnom trebale provoditi), bez da se značajno naruši točnost dobivenih rezultata, čime se dobiva brz i točan način izračuna tenzora inercije rebara. Za proizvoljno orijentiran štap u prostoru, definiran točkama $\mathbf{r}_1$ i $\mathbf{r}_2$, mogu se izvesti komponente tenzora inercije štapa (podijeljene sa masom štapa), koje glase (komponente I_{xy} i I_{zy} nisu navedene):

$$(\tilde{I}_{\mathrm{rod}})_{xx}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{3}(y_1^2 + y_1 y_2 + y_2^2 + z_1^2 + z_1 z_2 + z_1^2), \tag{5.6}$$

$$(\tilde{I}_{\mathrm{rod}})_{yy}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{3}(x_1^2 + x_1 x_2 + x_2^2 + z_1^2 + z_1 z_2 + z_1^2), \tag{5.7}$$

$$(\tilde{I}_{\mathrm{rod}})_{zz}(\mathbf{r}_1, \mathbf{r}_2) = \frac{1}{3}(x_1^2 + x_1 x_2 + x_2^2 + y_1^2 + y_1 y_2 + y_2^2), \tag{5.8}$$

$$(\tilde{I}_{\mathrm{rod}})_{xz}(\mathbf{r}_1, \mathbf{r}_2) = \frac{-1}{6}(2x_1 z_1 + x_1 z_2 + x_2 z_1 + 2x_2 z_2), \tag{5.9}$$

gdje su: x_1, y_1 i z_1, koordinate prvog vrha a: x_2, y_2 i z_2, koordinate drugog vrha. Sada se konačno može napisati konačan izraz za tenzor inercije rebara $\mathbf{I}_{\mathrm{ribs}}$:

$$\mathbf{I}_{\mathrm{ribs}} = \sum_{i=1}^{n_{\mathrm{c}}} (\rho_{\mathrm{ribs}})_i (t_{\mathrm{ribs}})_i \sum_{j=1}^{(n_{\mathrm{ribs}})_i} \begin{bmatrix} (\tilde{I}_{\mathrm{rod}})_{xx}(\mathbf{r}_{\mathrm{LE}ij}, \mathbf{r}_{\mathrm{TE}ij}) & 0 & (\tilde{I}_{\mathrm{rod}})_{xz}(\mathbf{r}_{\mathrm{LE}ij}, \mathbf{r}_{\mathrm{TE}ij}) \\ 0 & (\tilde{I}_{\mathrm{rod}})_{yy}(\mathbf{r}_{\mathrm{LE}ij}, \mathbf{r}_{\mathrm{TE}ij}) & 0 \\ (\tilde{I}_{\mathrm{rod}})_{xz}(\mathbf{r}_{\mathrm{LE}ij}, \mathbf{r}_{\mathrm{TE}ij}) & 0 & (\tilde{I}_{\mathrm{rod}})_{zz}(\mathbf{r}_{\mathrm{LE}ij}, \mathbf{r}_{\mathrm{TE}ij}) \end{bmatrix} A_{\mathrm{R}ij}. \tag{5.10}$$

5.3 Ramenjače

Kako bi se izračunala masa ramenjača, pretpostavit će se da se ramenjače mogu tretirati kao krivulje, što je analogno načinu na koji su se tretirala rebra. Uz tu pretpostavku, masa ramenjača m_{ribs} će biti:

$$m_{\text{spars}} = \sum_{i=1}^{n_c} (\rho_{\text{spars}})_i (t_{\text{spars}})_i \sum_{j=1}^{(n_{\text{spars}})_i} \int_{l_{ij}} h_{ij}(u) \left\| \frac{d\mathbf{l}_{ij}}{du} \right\| du, \tag{5.11}$$

gdje je:

$(n_{\text{spars}})_i$ – broj ramenjača na *i*-tom klasteru,

$(\rho_{\text{spars}})_i$ – gustoća rebara na *i*-tom klasteru,

$(t_{\text{spars}})_i$ – debljina rebara na *i*-tom klasteru,

$h(u)$ – distribucija debljine aeroprofila na mjestu *j*-te ramenjače *i*-tog klastera i

$\mathbf{l}_{ij}(u)$– krivulja *j*-te ramenjače *i*-tog klastera, koja predstavlja ramenjaču.

Za izračun centra mase ramenjača, uvest će se ista pretpostavka kao i prije, pa će centar mase ramenjača biti:

$$\mathbf{CM}_{\text{spars}} = \sum_{i=1}^{n_c} \frac{(\rho_{\text{spars}})_i (t_{\text{spars}})_i}{m_{\text{spars}}} \sum_{j=1}^{(n_{\text{spars}})_i} \int_{l_{ij}} h_{ij}(u) \mathbf{l}_{ij}(u) \left\| \frac{d\mathbf{l}_{ij}}{du} \right\| du. \tag{5.12}$$

Za tenzora inercije ramenjača, uvest će se ista pretpostavka kao i prije, pa će tenzor inercije ramenjača biti:

$$\mathbf{I}_{\text{spars}} = \sum_{i=1}^{n_c} (\rho_{\text{spars}})_i (t_{\text{spars}})_i \sum_{j=1}^{(n_{\text{spars}})_i} \int_{\mathbf{l}_{ij}} h_{ij}(u) (\tilde{\mathbf{I}}_{\text{spars}})_{ij} du, \tag{5.13}$$

$$(\tilde{\mathbf{I}}_{\text{spars}})_{ij} = \begin{bmatrix} y_{l_{ij}}^2 + z_{l_{ij}}^2 & 0 & -x_{l_{ij}} z_{l_{ij}} \\ 0 & x_{l_{ij}}^2 + z_{l_{ij}}^2 & 0 \\ -x_{l_{ij}} z_{l_{ij}} & 0 & x_{l_{ij}}^2 + y_{l_{ij}}^2 \end{bmatrix}. \tag{5.14}$$

5.4 „Puno“ krilo i spremnici goriva

Za računanje inercijskih značajki krila koje nema nikakvu posebnu konstrukciju, već je napravljeno kao jedna ispunjena cjelina, iskoristit će se generalizirani Pappusov teorem. Za krilo u kojem je smješteno gorivo, inercijske značajke se računaju na isti način. Masa ispunjenog krila, biti će:

$$m_{\text{spars}} = \sum_{i=1}^{n_c} (\rho_{\text{solid}})_i \int_{l_i} A_i(u) \left\| \frac{d\mathbf{l}_i}{du} \right\| du, \tag{5.15}$$

gdje je:
$(\rho_{\text{solid}})_i$ – gustoća krila na *i*-tom klasteru,

$A_i(u)$ – distribucija površine poprečnog presjeka na *i*-tom klasteru i

$\mathbf{l}_{ij}(u)$– krivulja *i*-tog klastera, duž koje se proteže krilo (u ovom slučaju, ona prolazi kroz težište aeroprofila).

Centar mase se dobiva na sličan način:

$$\mathbf{CM}_{\text{spars}} = \sum_{i=1}^{n_c} (\rho_{\text{solid}})_i \int_{l_i} A_i(u)\mathbf{l}_i(u) \left\| \frac{d\mathbf{l}_i}{du} \right\| du. \tag{5.16}$$

Za izračun tenzora inercije, pristupa se na način da se jedan infinitezimalni segment krila smatra kao kutija (što je i dobra pretpostavka kada su u pitanju spremnici), koja ima poprečni presjek jednak površini aeroprofila, dok je debljina jednaka infinitezimalnoj debljini segmenta. U tom će slučaju tenzor inercije oko ishodišta biti:

$$\mathbf{I}_{\text{spars}} = \sum_{i=1}^{n_c} (\rho_{\text{solid}})_i \int_{l_i} A_i(u) \left(\mathbf{R}_i(u) \left(\mathbf{M}^0_{\text{solid}}\right)_i(u) \mathbf{R}_i^T(u) + (\mathbf{M}^s_{\text{solid}})_i(u) \right) \left\| \frac{d\mathbf{l}_i}{du} \right\| du, \tag{5.17}$$

gdje je $\mathbf{R}_i(u)$ matrica rotacije, koja predstavlja kompoziciju rotacije oko *y* i *x*-osi:

$$\mathbf{R}_i(u) = \mathbf{R}_x\left(\theta_{Ai}(u)\right)\mathbf{R}_y\left(\alpha_{Ai}(u)\right) \tag{5.18}$$

Tenzor $\left(\mathbf{M}^0_{\text{solid}}\right)_i(u)$ i predstavlja specifični tenzor inercije kutije (oko centra mase kutije), za slučaj kada je kutija poravnata sa osima k.s., dok tenzor $(\mathbf{M}^s_{\text{solid}})_i(u)$ predstavlja specifični Steinerov dodatak:

$$\left(\mathbf{M}^0_{\text{solid}}\right)_i(u) = \frac{1}{12} \begin{bmatrix} \frac{A_i^2(u)}{c_i^2(u)} & 0 & 0 \\ 0 & c_i^2(u) + \frac{A_i^2(u)}{c_i^2(u)} & 0 \\ 0 & 0 & c_i^2(u) \end{bmatrix}, \tag{5.19}$$

$$(\mathbf{M}^s_{\text{solid}})_i(u) = \begin{bmatrix} y_{l_{ij}}^2 + z_{l_{ij}}^2 & -x_{l_{ij}}y_{l_{ij}} & -x_{l_{ij}}z_{l_{ij}} \\ -x_{l_{ij}}y_{l_{ij}} & x_{l_{ij}}^2 + z_{l_{ij}}^2 & -y_{l_{ij}}z_{l_{ij}} \\ -x_{l_{ij}}z_{l_{ij}} & -y_{l_{ij}}z_{l_{ij}} & x_{l_{ij}}^2 + y_{l_{ij}}^2 \end{bmatrix}. \tag{5.20}$$

5.5 Ukupna masa letjelice

Kada se saznaju inercijske značajke svih komponenti, tada će ukupna masa letjelice u općem slučaju biti:

$$m = m_{\text{skin}} + m_{\text{ribs}} + m_{\text{spars}} + m_{\text{solid}} + +m_{\text{comp.}} \tag{5.21}$$

Centar mase letjelice se računa kao:

$$\mathbf{CM} = \frac{m_{\text{skin}}\mathbf{CM}_{\text{skin}} + m_{\text{ribs}}\mathbf{CM}_{\text{ribs}} + m_{\text{spars}}\mathbf{CM}_{\text{spars}} + m_{\text{solid}}\mathbf{CM}_{\text{solid}} + m_{\text{comp}}\mathbf{CM}_{\text{comp}}}{m}. \tag{5.22}$$

Sada kada se pozna centar mase letjelice, može se izračunati tenzor inercije oko centra mase. Prvo se računa ukupni tenzor inercije svih dijelova, oko ishodišta $\mathbf{I}_0$:

$$\mathbf{I}_0 = \mathbf{I}_{\text{skin}} + \mathbf{I}_{\text{ribs}} + \mathbf{I}_{\text{spars}} + \mathbf{I}_{\text{solid}} + \mathbf{I}_{\text{comp}}. \tag{5.23}$$

Kako bi se dobio tenzor inercije oko centra mase, dodaje se Steinerov dodatak $\mathbf{I}_s$:

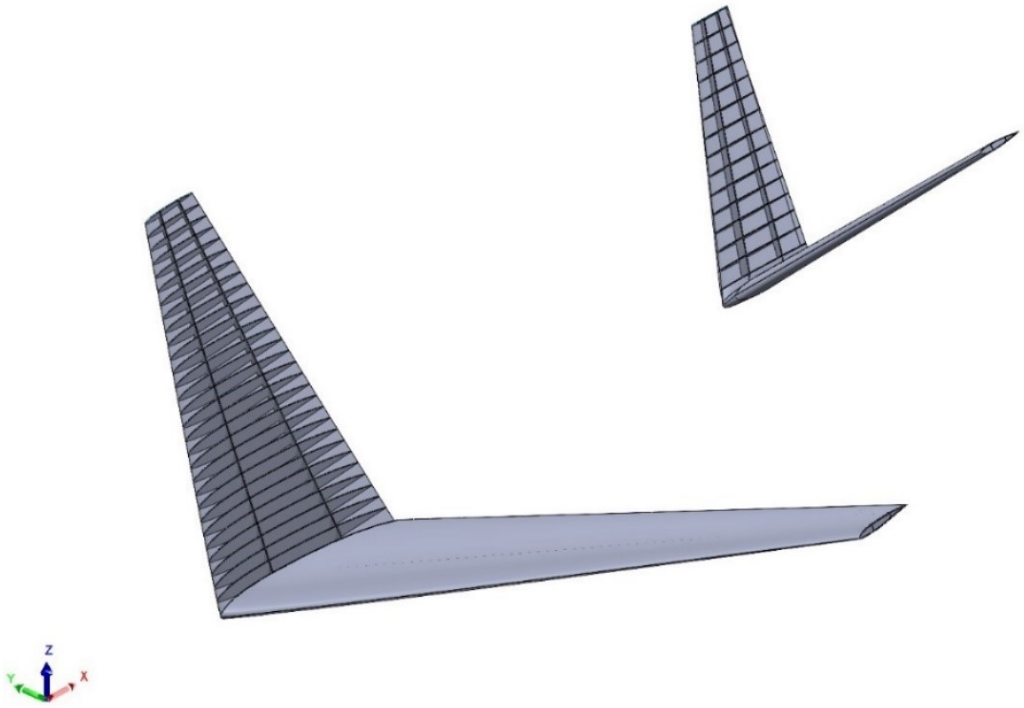
$$\mathbf{I} = \mathbf{I}_0 + \mathbf{I}_s, \tag{5.24}$$

pri čemu je $\mathbf{I}_s$ definiran na sljedeći način:

$$\mathbf{I}_s = \begin{bmatrix} -\left(y_{\text{CM}}^2 + z_{\text{CM}}^2\right)m & 0 & x_{\text{CM}}z_{\text{CM}}m \\ 0 & -(x_{\text{CM}}^2 + z_{\text{CM}}^2)m & 0 \\ x_{\text{CM}}z_{\text{CM}}m & 0 & -\left(x_{\text{CM}}^2 + y_{\text{CM}}^2\right)m \end{bmatrix}. \tag{5.24}$$

5.6 Usporedba inercijskih značajki sa programom SolidWorks

Za usporedbu, uzela se geometrija sa slike 5.1, te su se testirale dvije varijante. Jedna ima klasičnu konstrukciju (ramenjače + rebra + oplata) a druga varijanta letjelice je ispunjena tj. krilo je homogeno.



Slika 5.1 Testna geometrija u SolidWorks-u sa prikazanom konstrukcijom

Prikaz vrijednosti inercijskih značajki za jednu i drugu varijantu geometrije, vide se u tablicama 5.1 odnosno 5.2 te se u istim može pronaći korijen relativne srednje kvadratne pogreške (RRMSE).

Tablica 5.1 Rezultati usporedbe inercijskih značajki za klasičnu konstrukciju

	Masa [kg]	Centar mase [m]	Tenzor inercije [kgm^2]
AVCOPro	8.53	$[0.98 \quad 0 \quad 0.13]$	$\begin{bmatrix} 2.43 & 0 & -0.55 \\ 0 & 4.13 & 0 \\ -0.55 & 0 & 6.35 \end{bmatrix}$
SolidWorks	8.56	$[0.97 \quad 0 \quad 0.13]$	$\begin{bmatrix} 2.49 & 0 & -0.56 \\ 0 & 4.19 & 0 \\ -0.56 & 0 & 6.46 \end{bmatrix}$
RRMSE [%]	0.26	0.83	2.92

Tablica 5.2 Rezultati usporedbe inercijskih značajki za ispunjeno krilo

	Masa [kg]	Centar mase [m]	Tenzor inercije [kgm^2]
AVCOPro	54.54	$[0.82 \quad 0 \quad 0.1]$	$\begin{bmatrix} 12.81 & 0 & -2.58 \\ 0 & 20.15 & 0 \\ -2.58 & 0 & 31.97 \end{bmatrix}$
SolidWorks	46.63	$[0.80 \quad 0 \quad 0.1]$	$\begin{bmatrix} 10.94 & 0 & -2.27 \\ 0 & 17.58 & 0 \\ -2.27 & 0 & 27.65 \end{bmatrix}$
RRMSE [%]	16.95	4.12	26.59

Iz dobivenih rezultata, vidi se da u slučaju klasične konstrukcije, vrijednosti koje se dobivaju iz oba programa se izvrsno slažu za sve veličine, dok za drugu varijantu, slaganje je nešto lošije (osim za centar mase), no i dalje je u prihvatljivim granicama.

6 AERODINAMIKA

Opće strujanje fluida, opisano je nelinearnim parcijalnim diferencijalnim jednadžbama, pa je proces rješavanja tih jednadžbi iznimno složen i vremenski skup. Zbog toga, direktno koristiti te matematičke modele u kontekstu optimizacije je praktički zabranjeno te se pribjegava modelima koji su dovoljno vjerni a da su pri tome „jeftini“. Ukoliko se ograničimo na strujanja ispod 0.3 Macha, moguće je promatrati strujanje kompresibilnih plinova (npr. zrak) kao strujanje nekompresibilnog fluida, što je već značajno pojednostavljenje. Međutim, jednadžbe koje opisuju takvo strujanje (Navier-Stokesove jednadžbe) su i daje vrlo složene. Srećom, postoji način da se Navier-Stokesove jednadžbe svedu na Laplaceovu diferencijalnu jednadžbu, ukoliko se zanemari viskoznost i uvede pretpostavka da je je strujanje potencijalno, čime se drastično pojednostavio problem.

6.1 Formulacija metode panela

Laplaceoviu diferencijalnu jednadžbu, može se riješiti za opće uvjete superponiranjem singularnih elemenata – izvora i dipola. Te singularnosti se distribuiraju po plohi, koja predstavlja granicu domene. Problem rješavanja strujanja se stoga svodi na diskretizaciju plohe (vidi slike 3.5-3.8) te postavljane (i naknadno određivanje) singulariteta nepoznatih intenziteta. Intenziteti tih singulariteta se mogu odrediti pomoću Dirichletovog rubnog uvjeta, koji se postavlja na svaki panel, čime se dobiva sljedeći sustav ([4] i [5]):

$$\mathbf{A}_{\mu}\boldsymbol{\mu} = -\mathbf{A}_{\sigma}\boldsymbol{\sigma}, \tag{6.1}$$

gdje su matrice $\mathbf{A}_{\mu}$ i $\mathbf{A}_{\sigma}$isključivo ovisne o geometriji te predstavljaju međusobnu interakciju između panela, dok je vektor $\boldsymbol{\mu}$, vektor jačine dipola a vektor $\boldsymbol{\sigma}$ predstavlja vektor izvora. Konkretni izrazi za sastavljanje matrica $\mathbf{A}_{\mu}$ i $\mathbf{A}_{\sigma}$, mogu se naći u [4] i [6]. Izvori su ovisni o geometriji ali i uvjetima leta tj. parametrima leta, kao i centru mase (radi rotacije), pa se vektor $\boldsymbol{\sigma}$ može napisati kao:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{CM}, v_{\infty}, \alpha, \beta, p, q, r), \tag{6.2}$$

gdje je:
$\mathbf{CM}$ – centar mase letjelice,

v_{∞} – brzina letjelice,

α – napadni kut letjelice,

β – kut klizanja letjelice,

p – kutna brzina oko osi x k.s. letjelice,

q – kutna brzina oko osi y k.s. letjelice i

r – kutna brzina oko osi z k.s. letjelice.

Imajući na umu sve navedeno, slijedi da će rješenje sustava (6.1) također imati oblik (u smislu varijabli o kojima ovisi) kao i 6.2:

$$\boldsymbol{\mu} = \boldsymbol{\mu}(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r). \tag{6.3}$$

Nakon što su se našle jačine dipola, mogu se izračunati inducirane brzine na svim panelima pomoću slijedećih relacija:

$$\mathbf{v}_{\mathrm{i}\tilde{x}} = -\frac{\partial \boldsymbol{\mu}}{\partial \tilde{x}}, \mathbf{v}_{\mathrm{i}\tilde{y}} = -\frac{\partial \boldsymbol{\mu}}{\partial \tilde{y}}, \mathbf{v}_{\mathrm{i}\tilde{z}} = \boldsymbol{\sigma}, \tag{6.4}$$

gdje su brzine: $\mathbf{v}_{\mathrm{i}\tilde{x}}$, $\mathbf{v}_{\mathrm{i}y}$ i $\mathbf{v}_{\mathrm{i}\tilde{z}}$, brzine izražene u lokalnim koordinatama panele. Derivacije (6.4) predstavljaju kovarijantne derivacije tj. derivacije duž plohe. Zbog toga, potrebno je transformirati te brzine natrag u globalni koordinatni sustav:

$$\mathbf{v}_{\mathrm{i}x} = \mathbf{T}\mathbf{v}_{\mathrm{i}\tilde{x}}, \mathbf{v}_{\mathrm{i}y} = \mathbf{T}\mathbf{v}_{\mathrm{i}\tilde{y}}, \mathbf{v}_{\mathrm{i}z} = \mathbf{T}\mathbf{v}_{\mathrm{i}\tilde{z}}, \tag{6.5}$$

gdje je **T** matrica transformacija. Baš kao i jačine singulariteta i brzine se mogu zapisati u obliku kao što je to napravljeno u jednadžbi (6.3) te je time u potpunosti opisano polje strujanja za određeno stanje leta.

6.2 Izračun sila i momenata

Iz kvazistacionarne Bernoullijeve jednadžbe, mogu se dobiti koeficijenti tlakova na svim panelima pomoću slijedeće relacije:

$$\mathbf{C_p} = \mathbf{v}^{*2} - \mathbf{w}^{*2}, \tag{6.6}$$

gdje je $\mathbf{v}^*$ normalizirana brzina slobodne struje a $\mathbf{w}^*$ je zbroj normalizirane brzine slobodne struje i normalizirane inducirane brzine tj. $\mathbf{w}^* = \mathbf{v}^* + \mathbf{v}_\mathrm{i}^*$.
Tri komponente neviskoznih sila računaju se kao:

$$F_x^{\mathrm{inv}} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(C_\mathrm{p}\right)_i (x_\mathrm{n})_i (A)_i, \tag{6.7}$$

$$F_y^{\mathrm{inv}} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(C_\mathrm{p}\right)_i (y_\mathrm{n})_i (A)_i, \tag{6.8}$$

$$F_z^{\text{inv}} = -\frac{\rho v_\infty}{2}\sum_{i=1}^{n}\left(C_{\text{p}}\right)_i (z_{\text{n}})_i (A)_i, \tag{6.9}$$

gdje su:
ρ – gustoća zraka,

$\left(C_{\text{p}}\right)_i$ – koef. tlaka na i - tom panelu,

$(x_{\text{n}})_i, (y_{\text{n}})_i, (z_{\text{n}})_i$ – komponente normale na i - tom panelu i

$(A)_i$ - površina i - tog panela.

Tri komponente neviskoznih momenata računaju se kao:

$$M_x^{\text{inv}} = -\frac{\rho v_\infty}{2}\sum_{i=1}^{n}\left(C_p\right)_i ((y_{\text{r}})_i (z_{\text{n}})_i - (z_{\text{r}})_i (y_{\text{n}})_i)(A)_i, \tag{6.10}$$

$$M_y^{\text{inv}} = -\frac{\rho v_\infty}{2}\sum_{i=1}^{n}\left(C_{\text{p}}\right)_i ((z_{\text{r}})_i (x_{\text{n}})_i - (x_{\text{r}})_i (z_{\text{n}})_i)(A)_i, \tag{6.11}$$

$$M_z^{\text{inv}} = -\frac{\rho v_\infty}{2}\sum_{i=1}^{n}\left(C_{\text{p}}\right)_i ((x_{\text{r}})_i (y_{\text{n}})_i - (y_{\text{r}})_i (x_{\text{n}})_i)(A)_i, \tag{6.12}$$

gdje su: $(x_{\text{r}})_i, (y_{\text{r}})_i$ i $(z_{\text{r}})_i$, komponente vektora položaja težišta i - tog panela, u odnosu na centar mase. Bitno je reći da se navedena jednadžba za silu otpora F_x ne koristi u obliku koji je naveden u (6.7), već se računa preko Trefftzove ravnine radi stabilnosti i točnosti [7].
Sile i momenti koji su navedeni, dobiveni su uz pretpostavku neviskoznog strujanja. Kako bi se dodao utjecaj viskoznosti, koristit će se metoda trake u kojoj je pretpostavka da se trodimenzionalno strujanje oko krila može svesti na niz dvodimenzionalnih strujanja oko aeroprofila. Ukoliko imamo neki segment krila, tada će otpor uslijed viskoznosti prema teoriji trake biti:

$$F_x^{\text{v}} = \frac{\rho v_\infty}{2}\int_0^1 c_{\text{d}}\left(c_{\text{l}}(u)\right)c(u)\left\|\frac{d\mathbf{g}}{du}\right\| du, \tag{6.13}$$

gdje je $c_{\text{l}}(y)$ koeficijent lokalnog uzgona a c_{d} koeficijent otpora aeroprofila u 2D strujanju. Za slučaj diskretizirane geometrije, gornji izraz će postati:

$$F_x^{\text{v}} = \frac{\rho v_\infty}{2}\sum_{i=1}^{n_{\text{TE}}} c_{\text{d}}\left(c_{\text{l}}(u_i)\right)c(u_i)\Delta l_i, \tag{6.14}$$

gdje je n_{TE} broj panela na izlaznom bridu (gledano ili sa donje ili gornje strane) a Δl_i duljina brida panela na izlaznom bridu. Lokalni koeficijent uzgona c_{l}, proporcionalan je cirkulaciji Γ oko aeroprofila u vidu. Može se pokazati da će c_{l} biti:

$$c_{l(u_i)} = \frac{2\Gamma(u_i)}{c(u_i)} = \frac{2(\mu_{\mathrm{TEd}}(u_i) - \mu_{\mathrm{TEu}}(u_i))}{c(u_i)}, \tag{6.15}$$

gdje je μ_{TEd} intenzitet dipola na panelu s donje strane izlaznog brida a μ_{TEu} intenzitet dipola s gornje strane aeroprofila.

Kako bi se dobio utjecaj kontrolnih ploha, potrebno je riješiti ponovno sustav (6.1), ali sa otklonjenim kontrolnim plohama (jednom sa pozitivnim $(+\delta)$ i jednom sa negativnim $(-\delta)$ otklonom). To se radi kako bi se mogla napraviti kvadratna interpolacija sila i momenata pomoću tri stanja ($-\delta$, neutralno i $+\delta$). Radi jednostavnosti, varijable: α, β, p, q i r se postavljaju na nulu pri otklonima kontrolnih ploha, čime se dobivaju sile i momenti uslijed otklonjenih ploha:

$$F_x^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i (x_{\mathrm{n}})_i (A)_i \right)^{\pm\delta}, \tag{6.16}$$

$$F_y^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i (y_{\mathrm{n}})_i (A)_i \right)^{\pm\delta}, \tag{6.17}$$

$$F_z^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i (z_{\mathrm{n}})_i (A)_i \right)^{\pm\delta}, \tag{6.18}$$

$$M_x^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i ((y_{\mathrm{r}})_i (z_{\mathrm{n}})_i - (z_{\mathrm{r}})_i (y_{\mathrm{n}})_i)(A)_i \right)^{\pm\delta}, \tag{6.19}$$

$$M_y^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i ((z_{\mathrm{r}})_i (x_{\mathrm{n}})_i - (x_{\mathrm{r}})_i (z_{\mathrm{n}})_i)(A)_i \right)^{\pm\delta}, \tag{6.20}$$

$$M_z^{\pm\delta} = -\frac{\rho v_\infty}{2} \sum_{i=1}^{n} \left(\left(C_{\mathrm{p}}\right)_i ((x_{\mathrm{r}})_i (y_{\mathrm{n}})_i - (y_{\mathrm{r}})_i (x_{\mathrm{n}})_i)(A)_i \right)^{\pm\delta}. \tag{6.21}$$

Sada konačno možemo zapisati konačne izraze za ukupne sile i momente:

$$F_x = F_x^{\text{inv}} + F_x^{\text{v}} + \frac{1}{2}\sum_i \left(F_x^{+\delta} + F_x^{-\delta}\right)\delta^2 + \left(F_x^{+\delta} - F_x^{-\delta}\right)\delta + F_{\text{p}}, \tag{6.22}$$

$$F_y = F_y^{\text{inv}} + \frac{1}{2}\sum_i \left(F_y^{+\delta} + F_y^{-\delta}\right)\delta^2 + \left(F_y^{+\delta} - F_y^{-\delta}\right)\delta, \tag{6.23}$$

$$F_z = F_z^{\text{inv}} + \frac{1}{2}\sum_i \left(F_z^{+\delta} + F_z^{-\delta}\right)\delta^2 + \left(F_z^{+\delta} - F_z^{-\delta}\right)\delta, \tag{6.24}$$

$$M_x = M_x^{\text{inv}} + \frac{1}{2}\sum_i \left(M_x^{+\delta} + M_x^{-\delta}\right)\delta^2 + \left(M_x^{+\delta} - M_x^{-\delta}\right)\delta, \tag{6.25}$$

$$M_y = M_y^{\text{inv}} + \frac{1}{2}\sum_i \left(M_y^{+\delta} + M_y^{-\delta}\right)\delta^2 + \left(M_y^{+\delta} - M_y^{-\delta}\right)\delta, \tag{6.26}$$

$$M_z = M_z^{\text{inv}} + \frac{1}{2}\sum_i \left(M_z^{+\delta} + M_z^{-\delta}\right)\delta^2 + \left(M_z^{+\delta} - M_z^{-\delta}\right)\delta, \tag{6.27}$$

gdje je F_{p} parazitni otpor uzrokovan npr. izvučenim podvozjem ili nekim drugim „izvorom“ otpora. Tako ćemo na kraju imati sile u obliku:

$$F_x = F_x(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta), \tag{6.28}$$

$$F_y = F_y(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta), \tag{6.29}$$

$$F_z = F_z(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta), \tag{6.30}$$

$$M_x = M_x(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta), \tag{6.31}$$

$$M_y = M_y(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta), \tag{6.32}$$

$$M_z = M_z(\mathbf{CM}, v_\infty, \alpha, \beta, p, q, r, \delta). \tag{6.33}$$

Gornje relacije za sile i momente su jednostavne algebarske jednadžbe, što znači da se jeftino može evaluirati mnogo različitih stanja leta, što je izrazito bitno kada je u pitanju evaluacija različitih performansi letjelice.

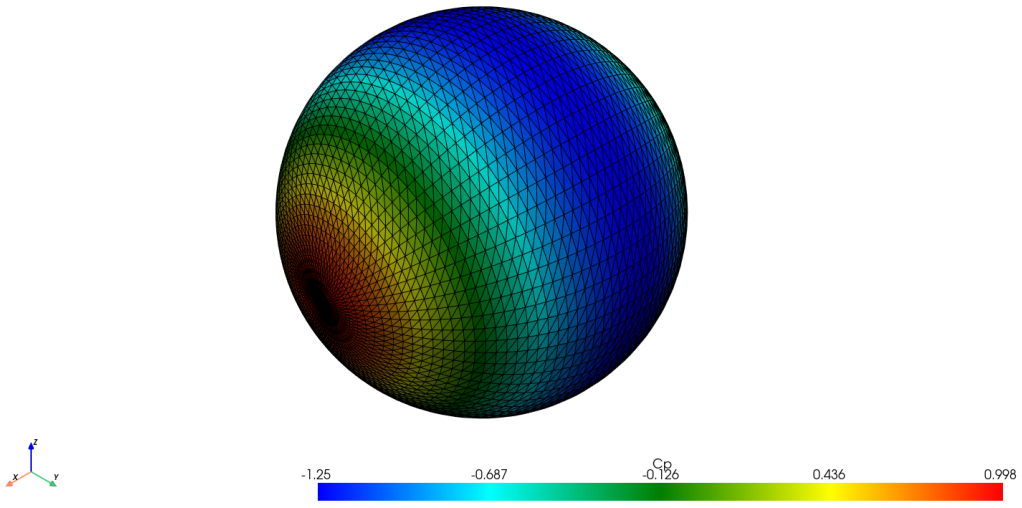
6.3 Provjera aerodinamičkog modela

6.3.1 Neviskozno optjecanje sfere

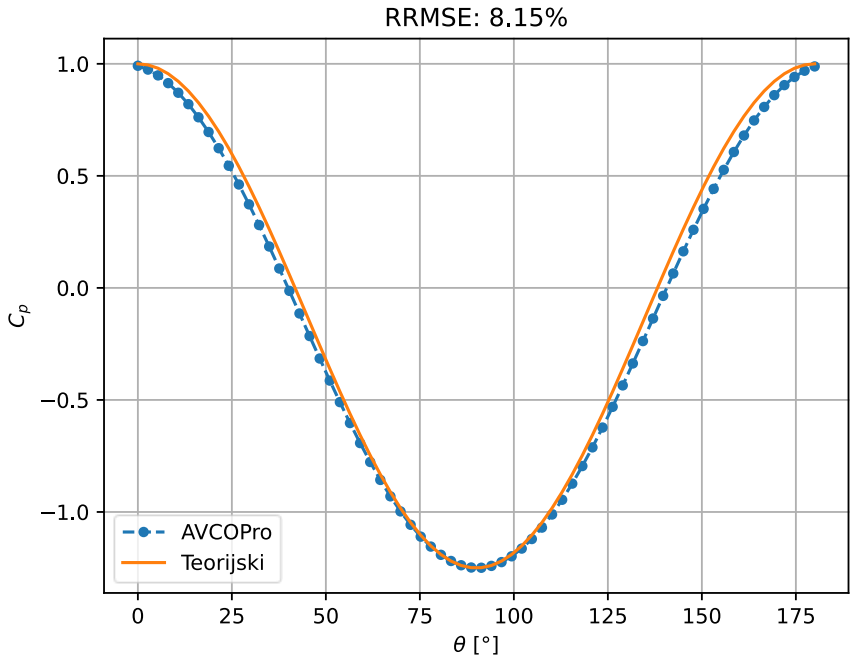
Poznato je da će distribucija koeficijenta tlaka po sferi u potencijalnom strujanju biti ([8] i [9]):

$$C_{\mathrm{p}} = 1 - \frac{9}{4}\sin(\theta)^2, \tag{6.34}$$

gdje kut θ predstavlja longitudinalni položaj na sferi.



Slika 6.1 Distribucija koeficijenta tlaka na sferi



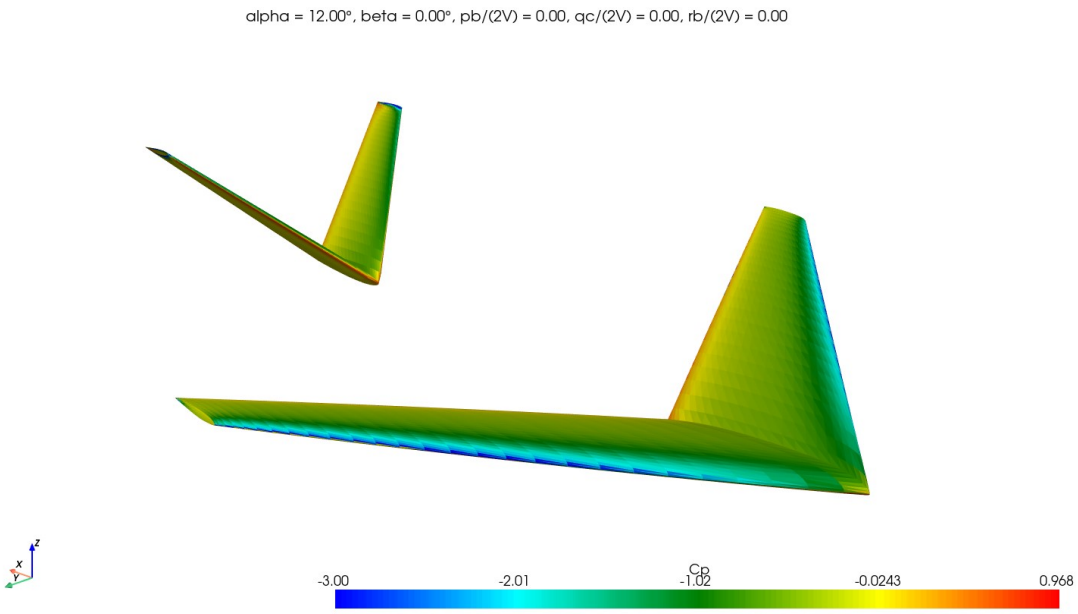
Slika 6.2 Usporedba distribucija tlakova

Na slici 6.1, prikazana je distribucija koeficijenata tlakova na sferi dok se na slici 6.2 vidi usporedba između teorijskog rješenja i rješenja koje se dobiva aerodinamikom u AVCOPro-u.

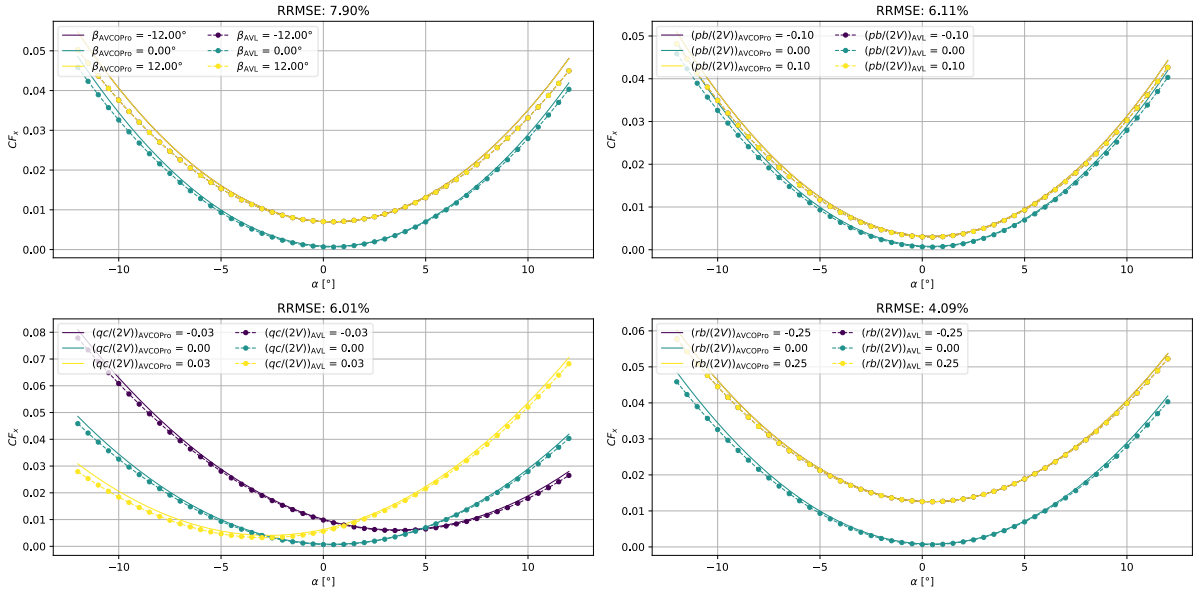
Kao što se vidi, poklapanje je najbolje na mjestu najniže vrijednosti koeficijenta tlaka (mjesta najveće brzine) dok je na prednjem i stražnjem djelu poklapanje nešto lošije, no kao što se i vidi, relativna pogreška je ispod 10%, što je izvrstan rezultat.

6.3.2 Usporedba sila i momenata sa programom AVL

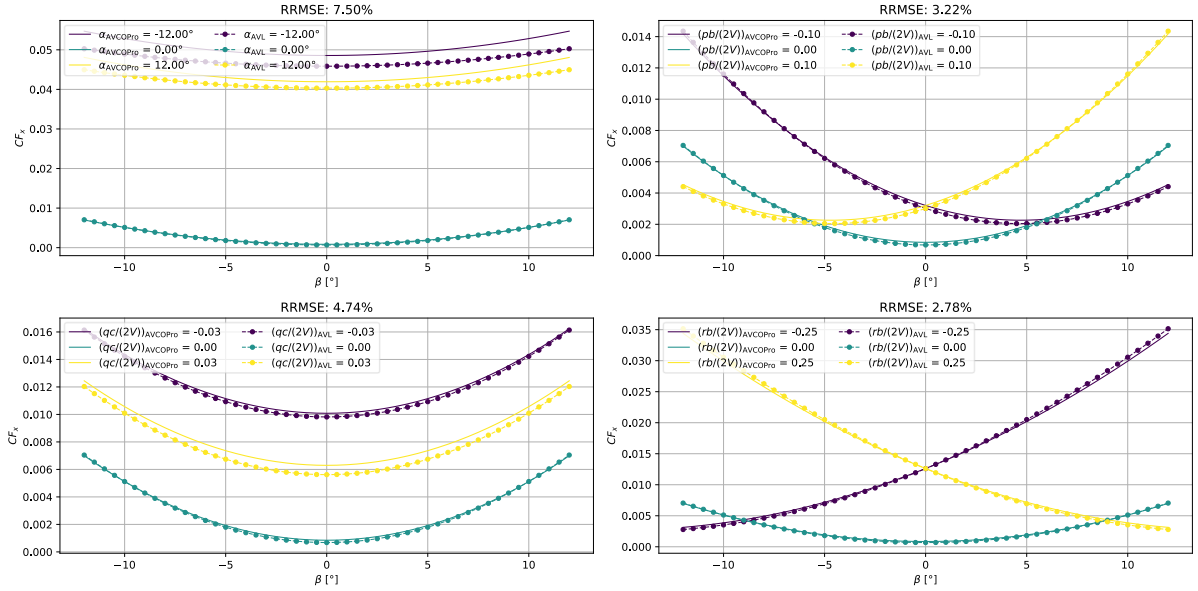
Prethodna usporedba je pokazala da je metoda panela ispravno implementirana te da je uspješno riješen sustav. Slijedeće, potrebno je provjeriti sile i momente na složenijoj geometriji. Kao testna geometrija, uzet će se konfiguracija sa slike 6.3. Napravljeno je niz izračuna sila i momenata za različita stanja leta. Rezultati se vide na slijedećim slikama. Važno je prije toga naglasiti da su otkloni kontrolnih ploha bezdimenzijski tj. kada je $\delta = 1$, to znači da je kontrolna ploha maksimalno otklonjena. Analogno vrijedi za $\delta = -1$. Za promatrane slučajeve, minimalni odnosno maksimalni otkloni su iznosili $\pm 20°$. Sile i momenti koji su promatrani su u potpunosti neviskozni, kako bi se ispitala implementacija tog djela aerodinamike. Što se tiče utjecaja viskoznosti, bilo bi potrebno korišteni model usporediti sa CFD simulacijama, no radi manjka vremena, taj dio se neće ovdje obrađivati.



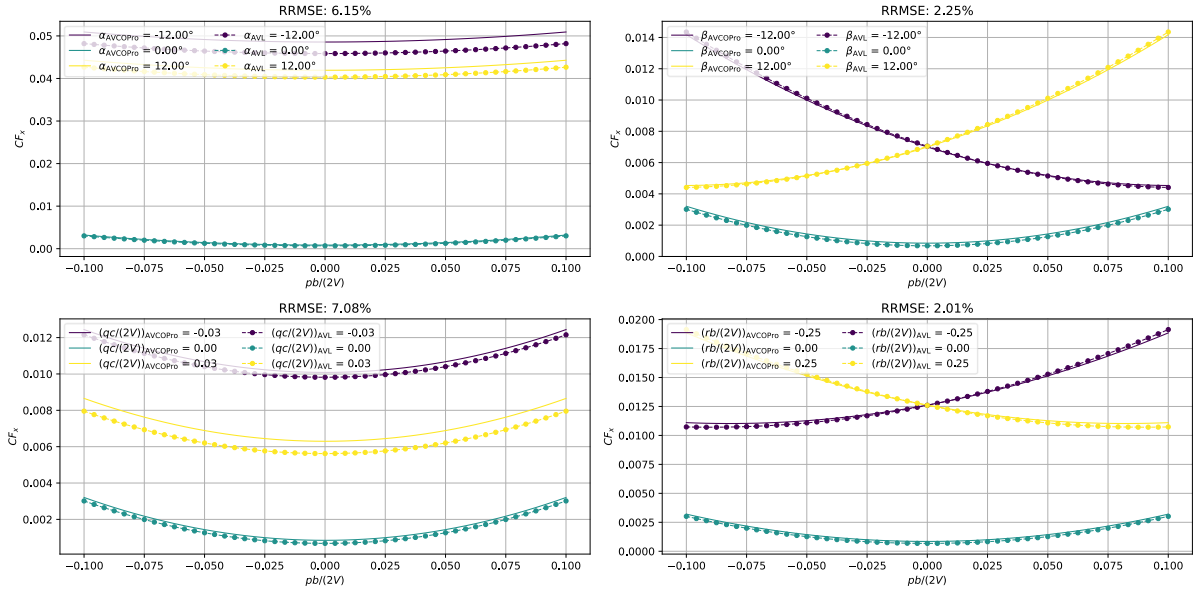
Slika 6.3 Distribucija koeficijenta tlaka na konfiguraciji za usporedbu rezultata sa programom AVL



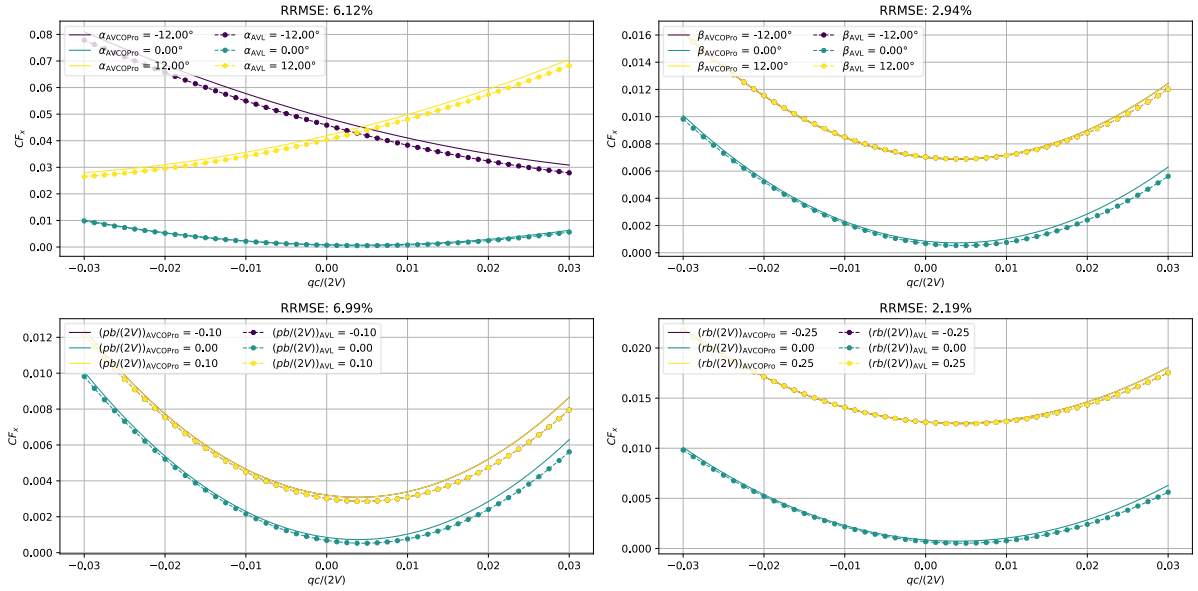
Slika 6.4 Koeficijent sile u smjeru osi *x*, u ovisnosti o *α*



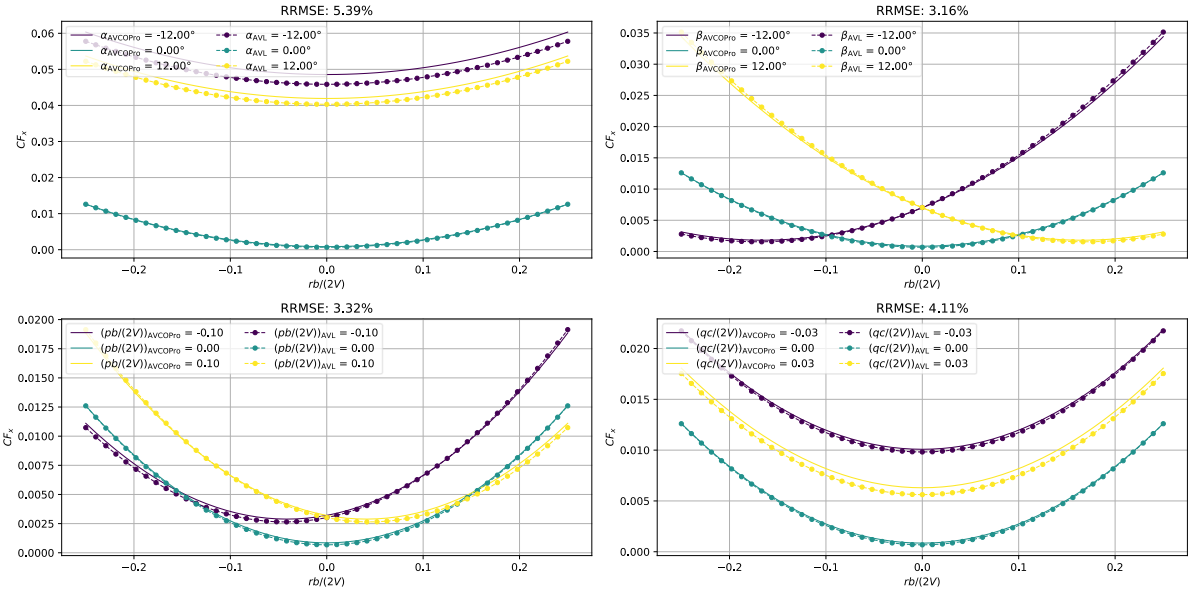
Slika 6.5 Koeficijent sile u smjeru osi *x*, u ovisnosti o *β*



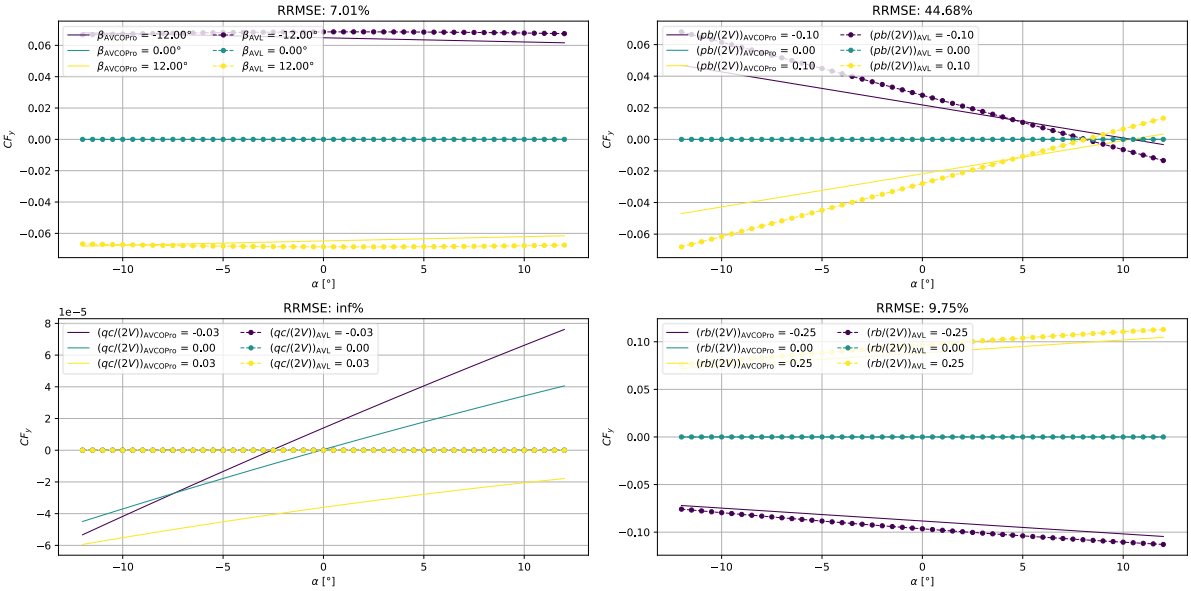
Slika 6.6 Koeficijent sile u smjeru osi *x*, u ovisnosti o *pb*/(2*V*)



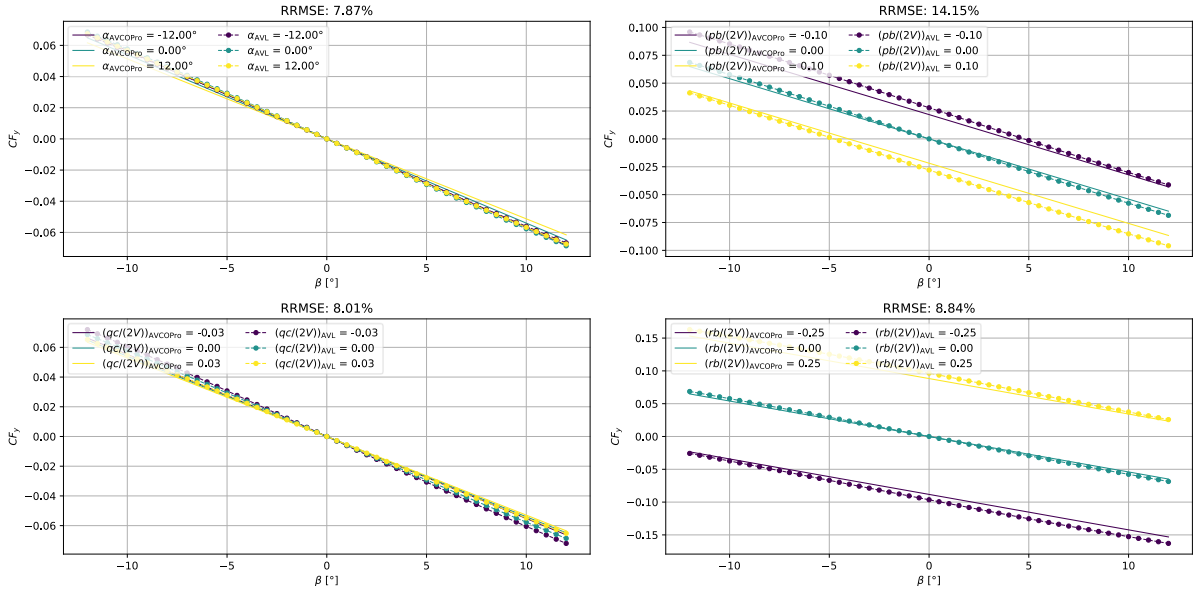
Slika 6.7 Koeficijent sile u smjeru osi *x*, u ovisnosti o *qc*/(2*V*)



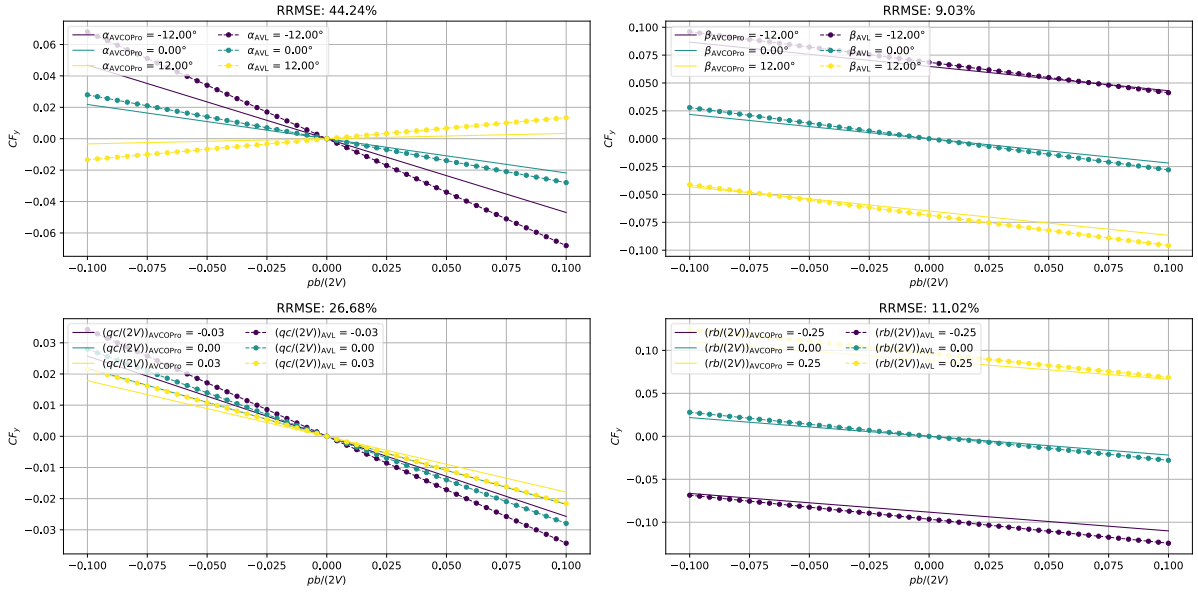
Slika 6.8 Koeficijent sile u smjeru osi *x*, u ovisnosti o *rb*/(2*V*)



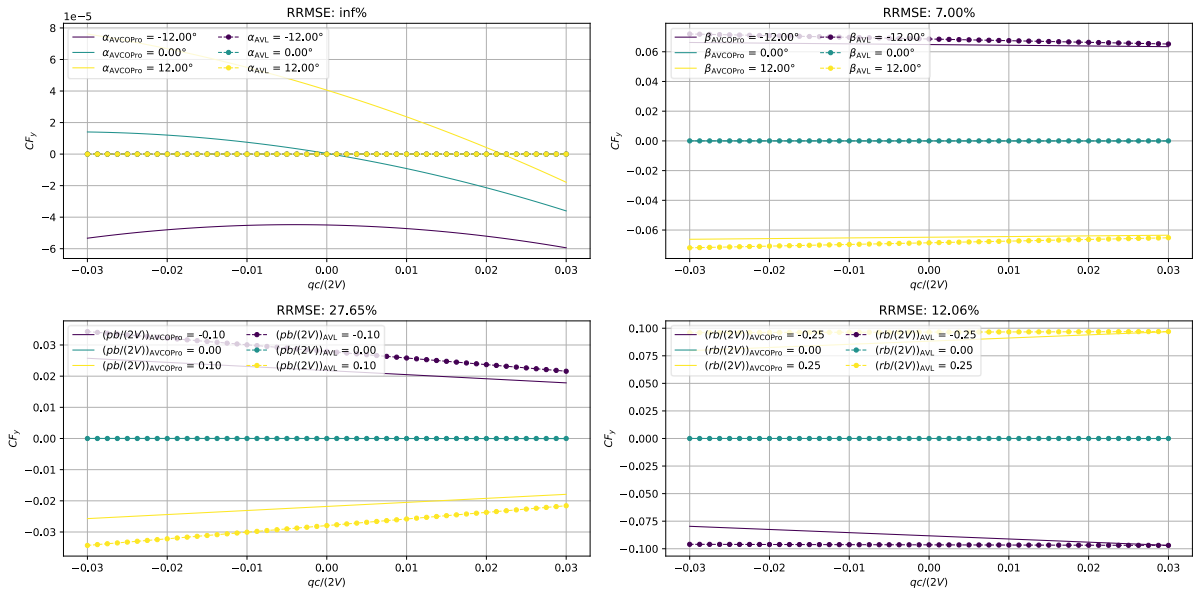
Slika 6.9 Koeficijent sile u smjeru osi *y*, u ovisnosti o *α*



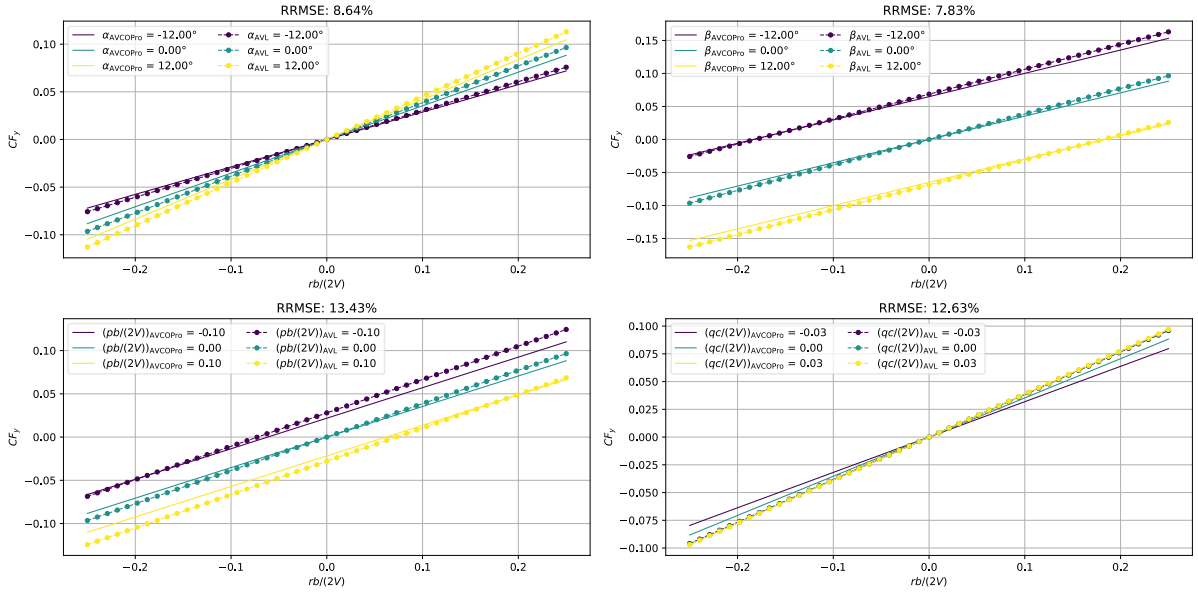
Slika 6.10 Koeficijent sile u smjeru osi *y*, u ovisnosti o *β*



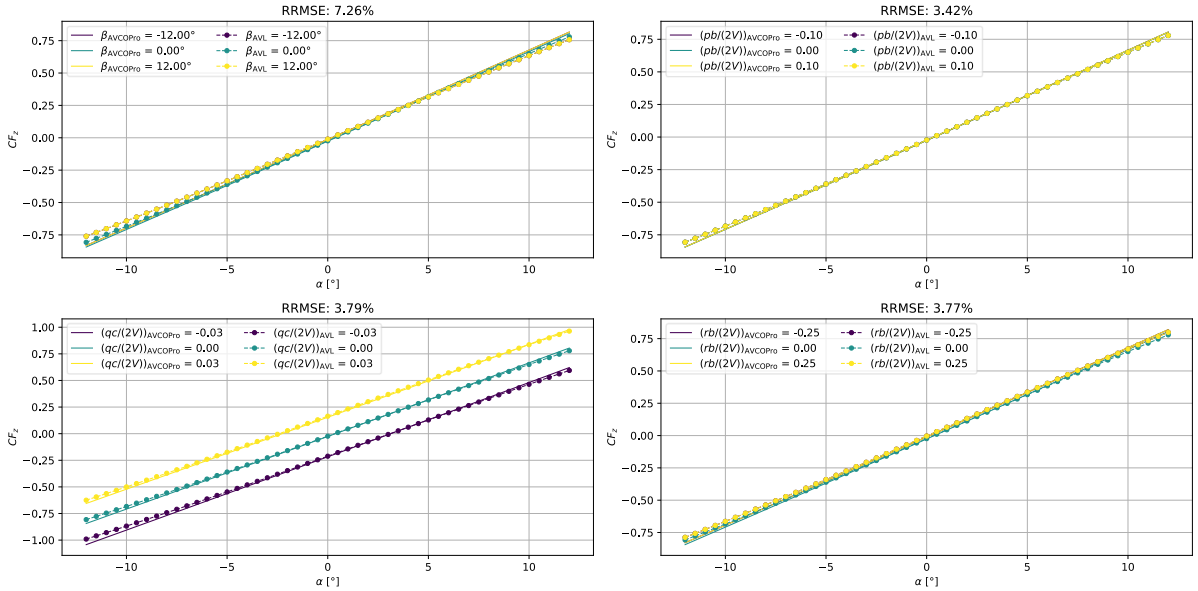
Slika 6.11 Koeficijent sile u smjeru osi *y*, u ovisnosti o *pb*/(2*V*)



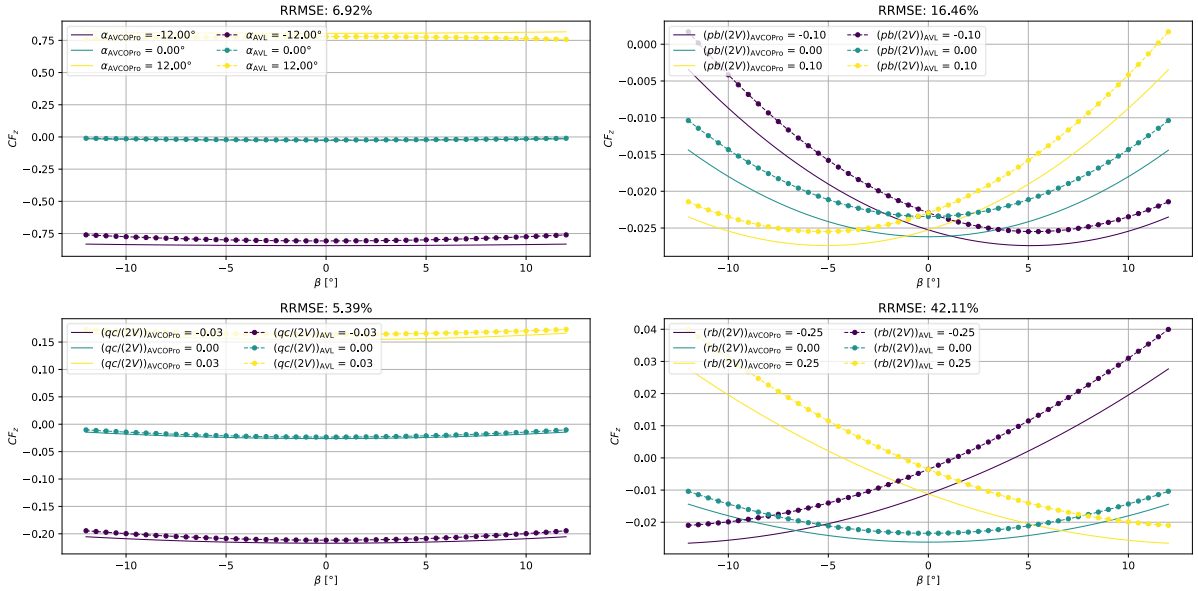
Slika 6.12 Koeficijent sile u smjeru osi *y*, u ovisnosti o *qc*/(2*V*)



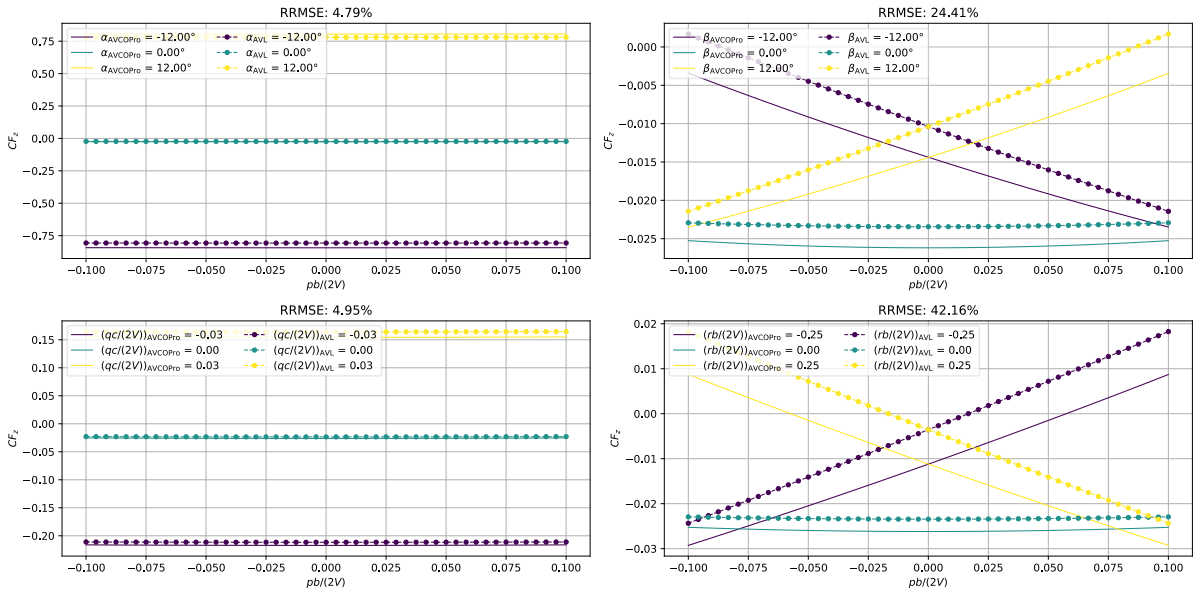
Slika 6.13 Koeficijent sile u smjeru osi *y*, u ovisnosti o *rb*/(2*V*)



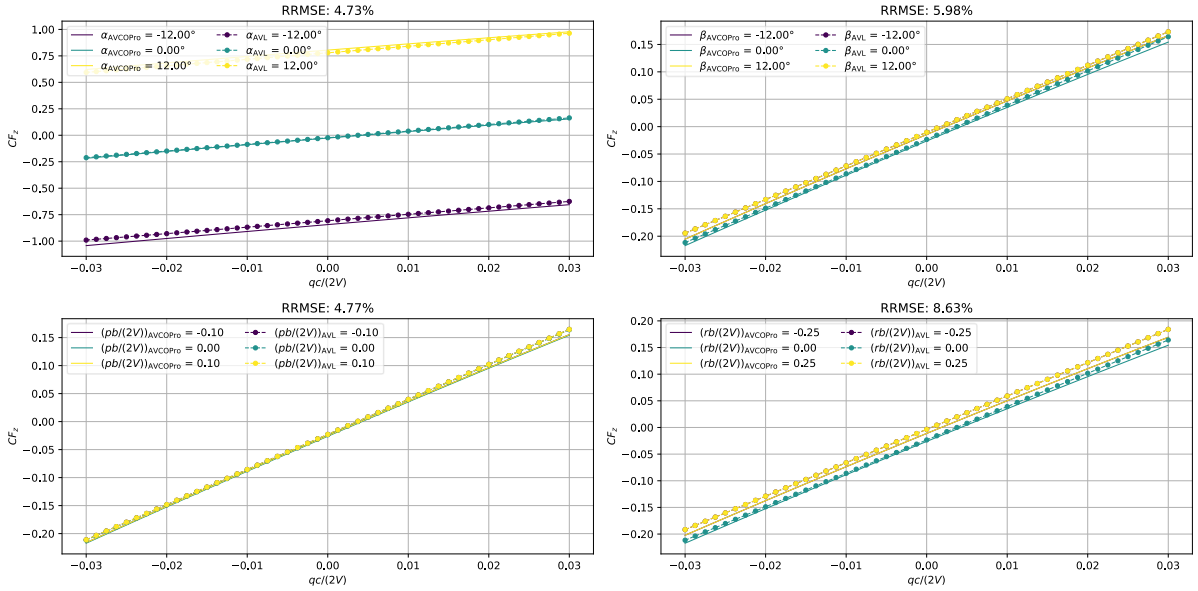
Slika 6.14 Koeficijent sile u smjeru osi *z*, u ovisnosti o *α*



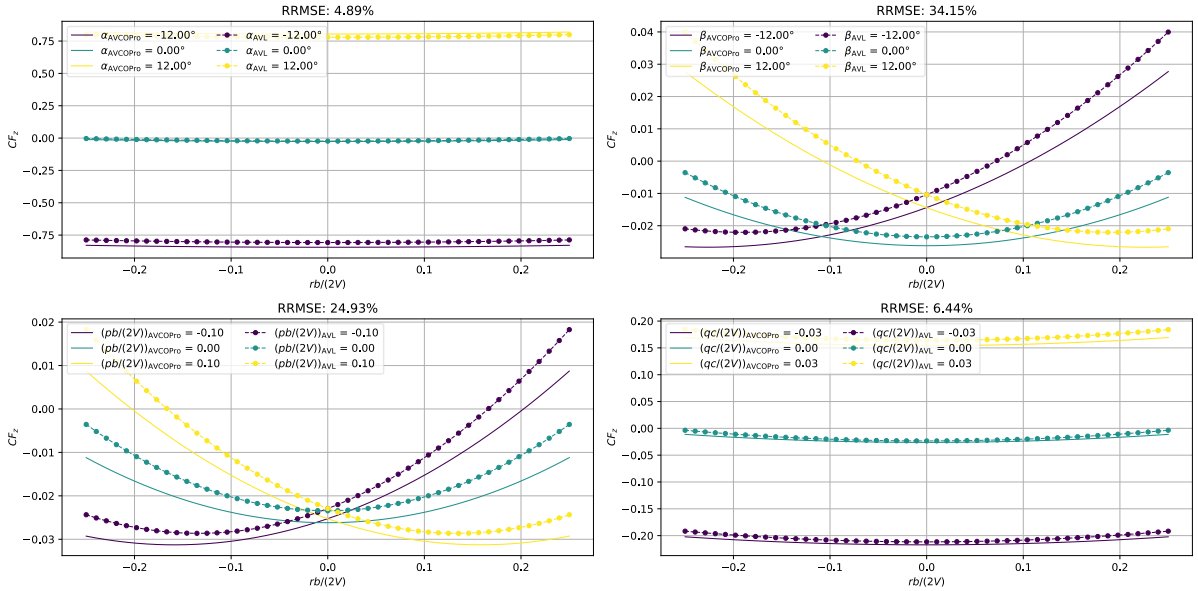
Slika 6.15 Koeficijent sile u smjeru osi *z*, u ovisnosti o *β*



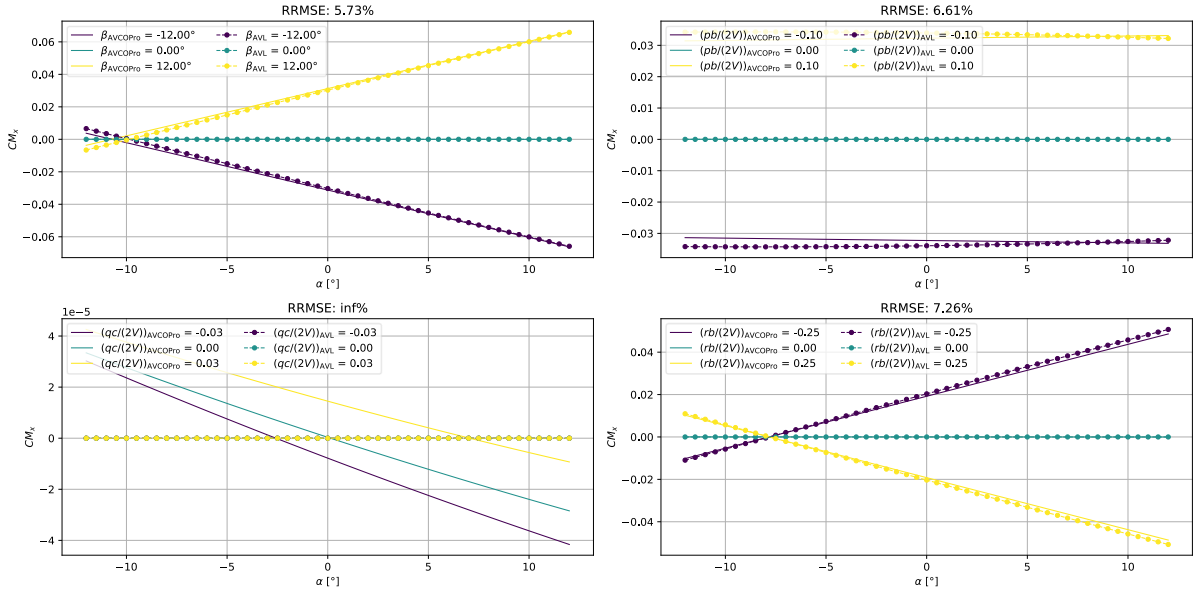
Slika 6.16 Koeficijent sile u smjeru osi *z*, u ovisnosti o *pb*/(2*V*)



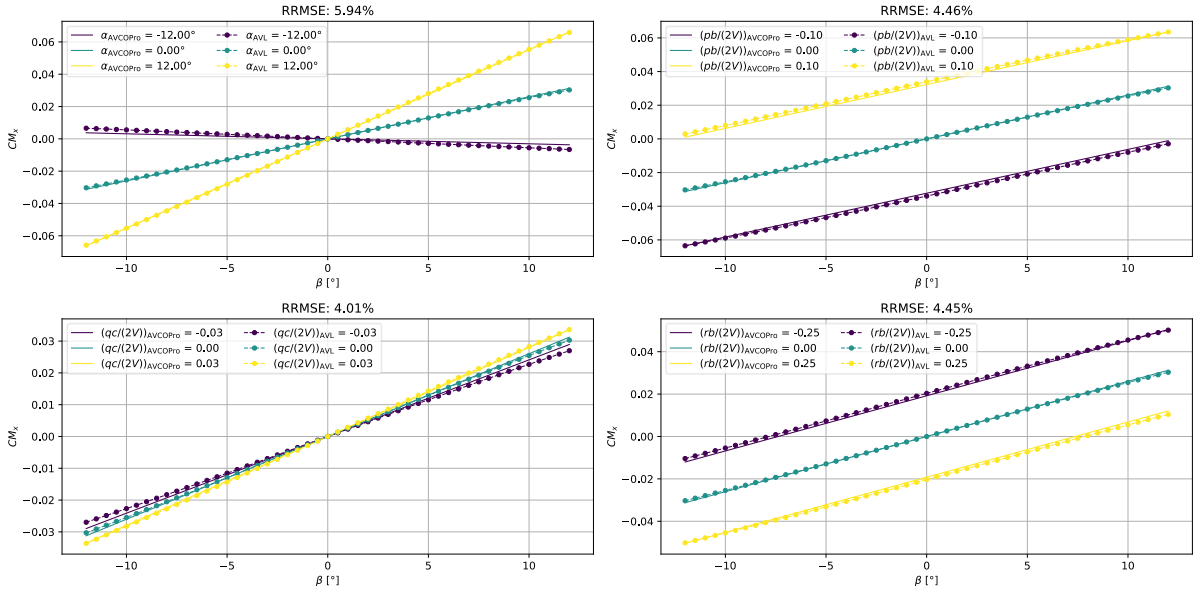
Slika 6.17 Koeficijent sile u smjeru osi *z*, u ovisnosti o *qc*/(2*V*)



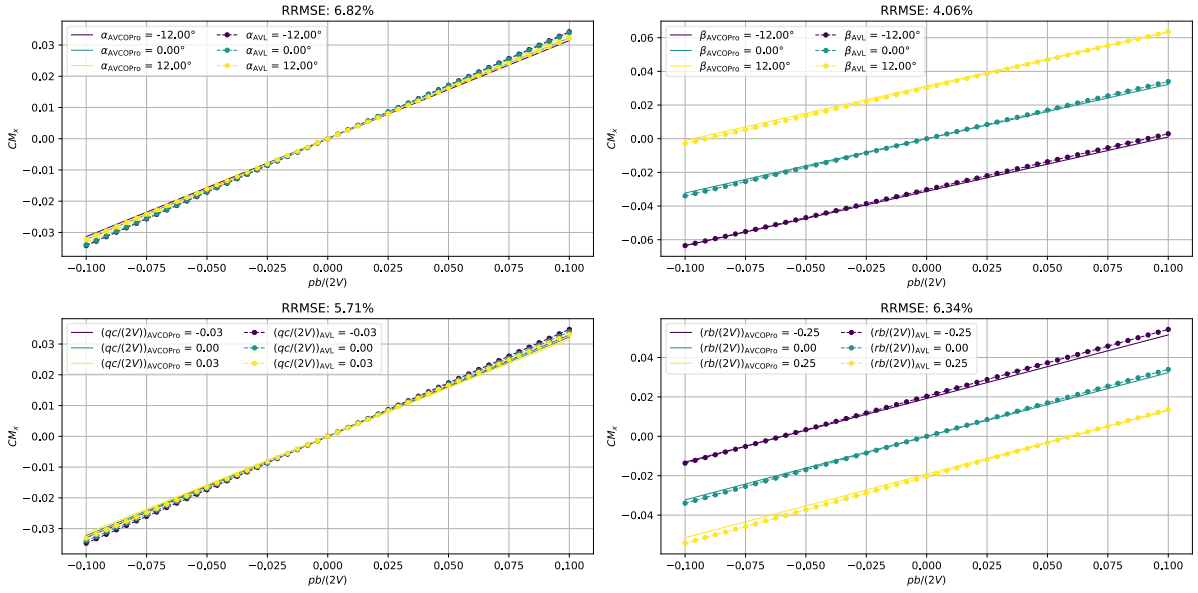
Slika 6.18 Koeficijent sile u smjeru osi *z*, u ovisnosti o *rb*/(2*V*)



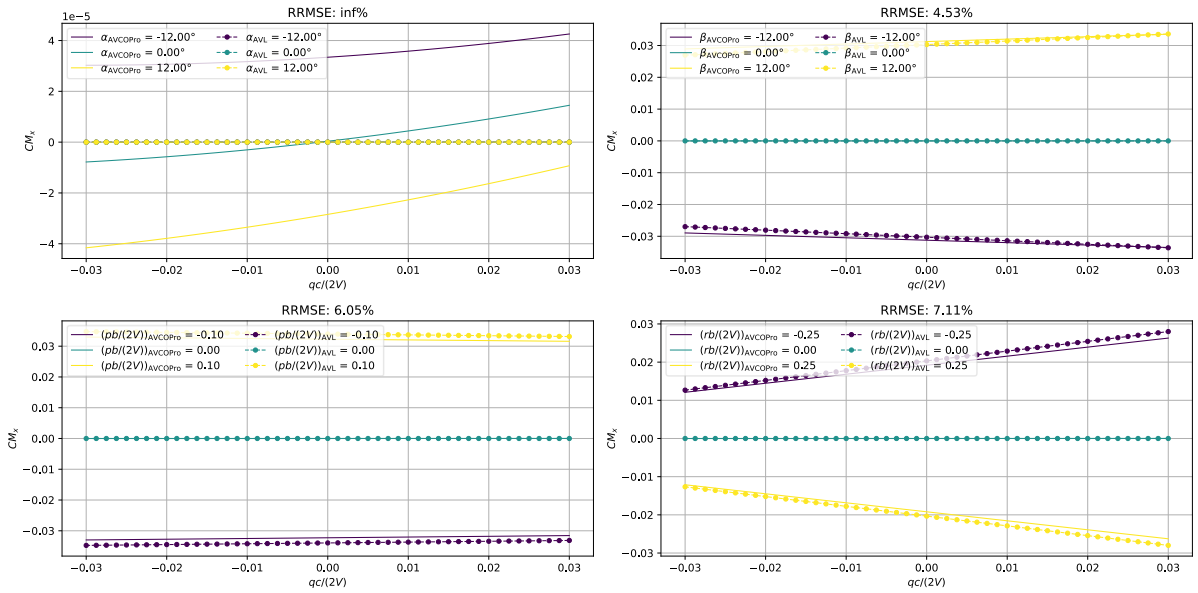
Slika 6.19 Koeficijent momenta oko osi *x*, u ovisnosti o *α*



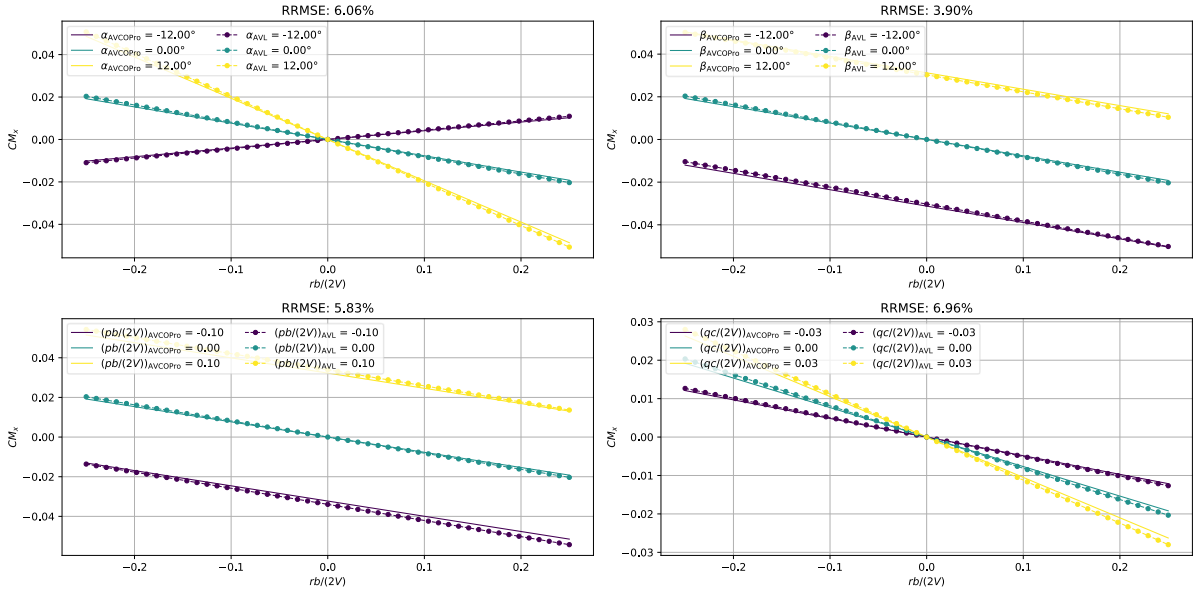
Slika 6.20 Koeficijent momenta oko osi *x*, u ovisnosti o *β*



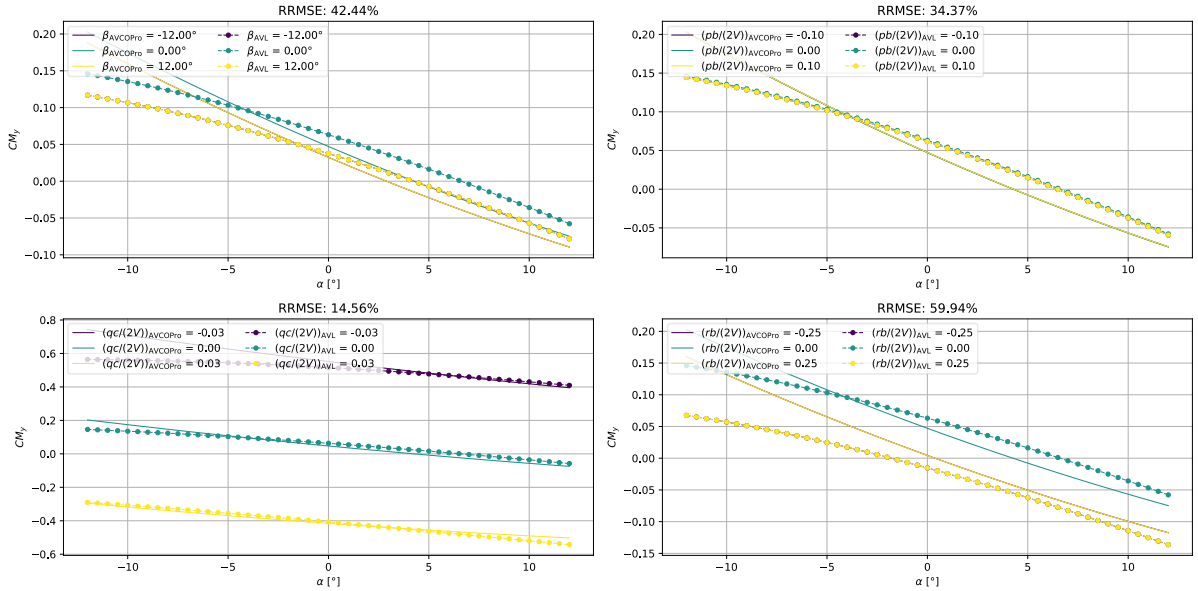
Slika 6.21 Koeficijent momenta oko osi *x*, u ovisnosti o *pb*/(2*V*)



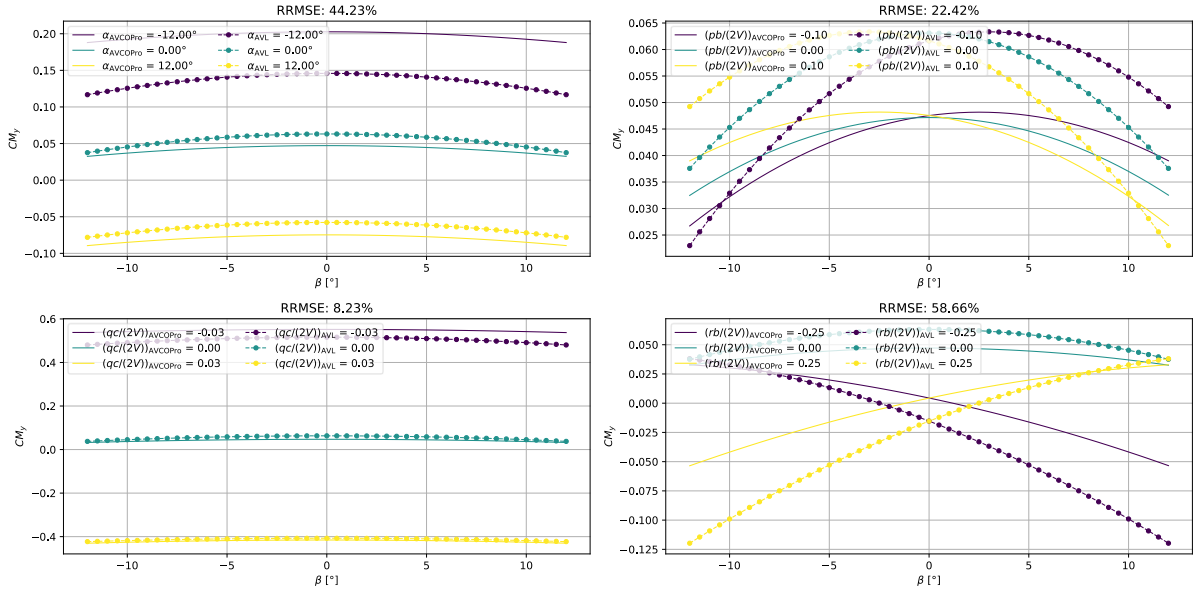
Slika 6.22 Koeficijent momenta oko osi *x*, u ovisnosti o *qc*/(2*V*)



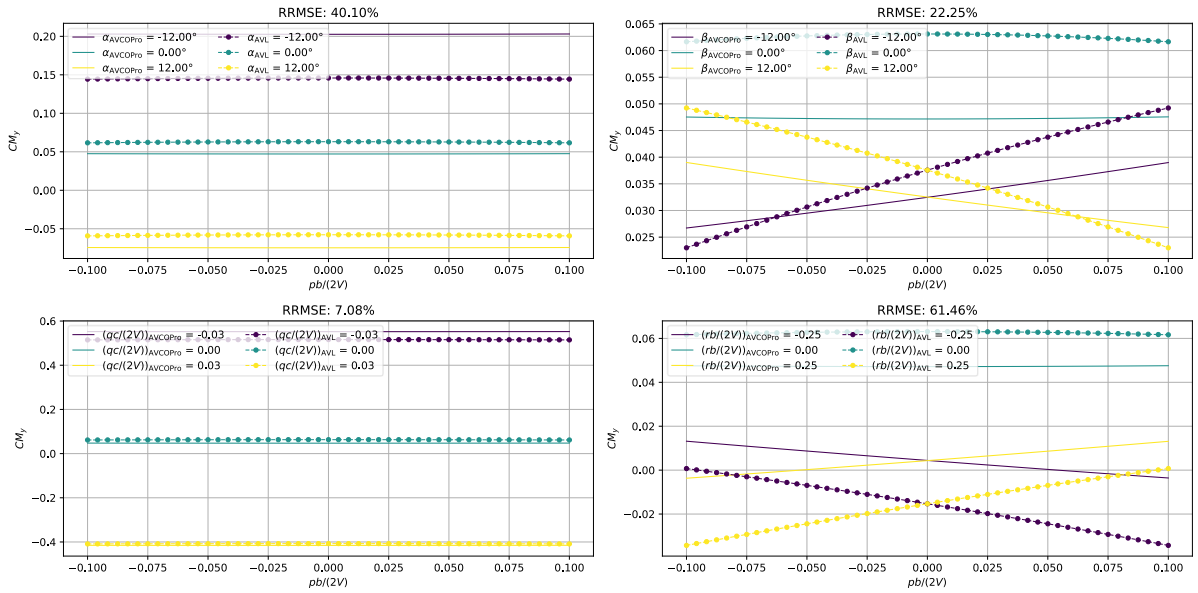
Slika 6.23 Koeficijent momenta oko osi *x*, u ovisnosti o *rb*/(2*V*)



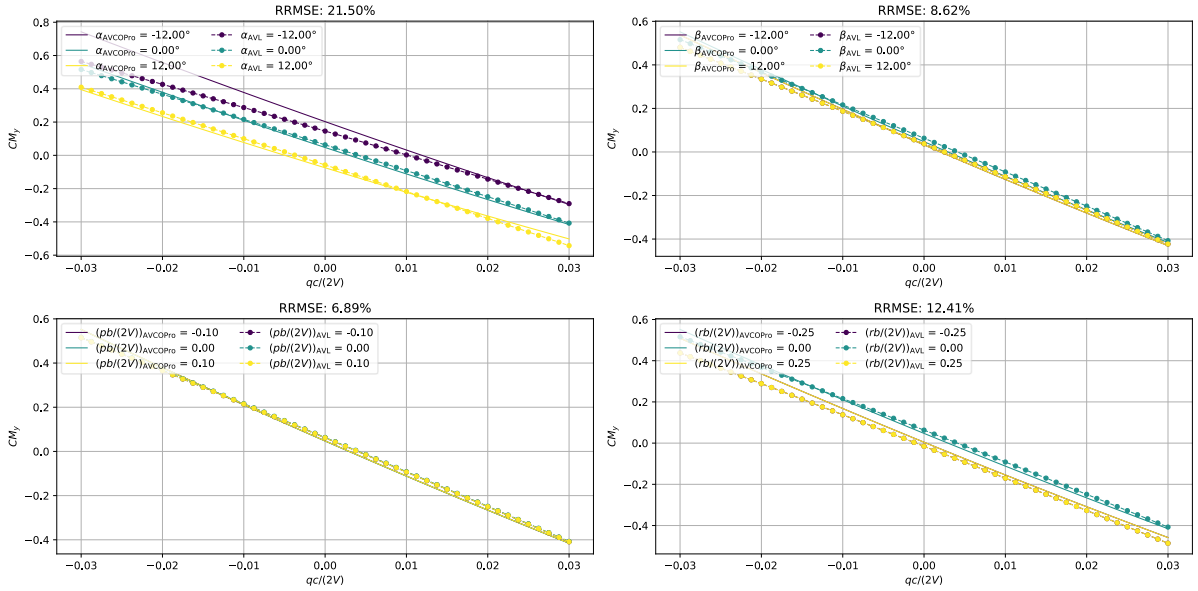
Slika 6.24 Koeficijent momenta oko osi *y*, u ovisnosti o *α*



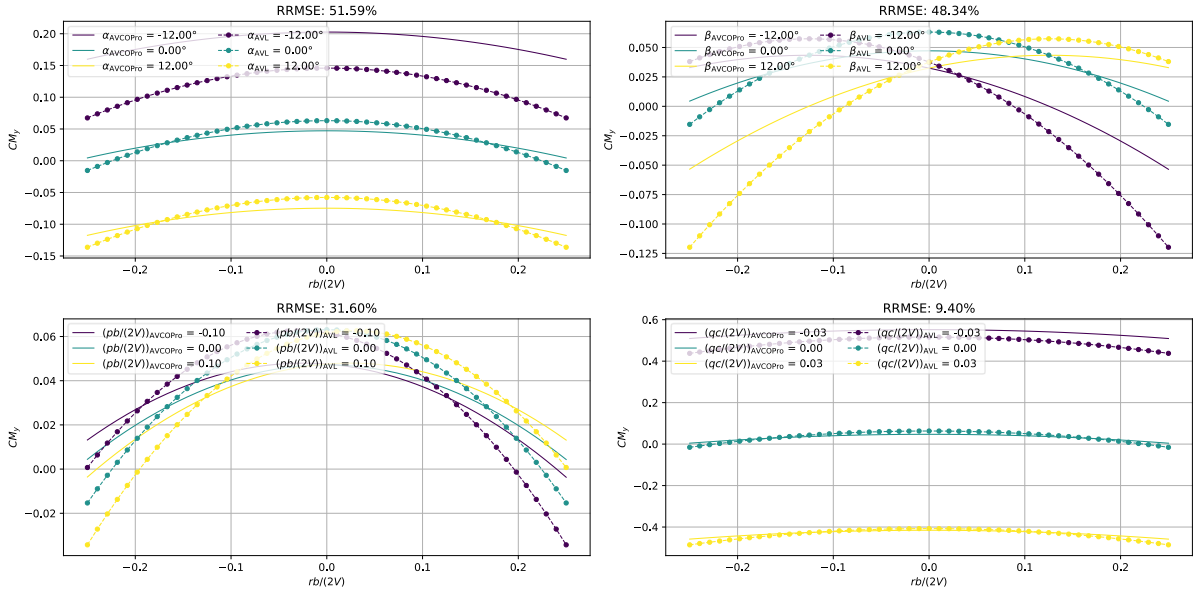
Slika 6.25 Koeficijent momenta oko osi *y*, u ovisnosti o *β*



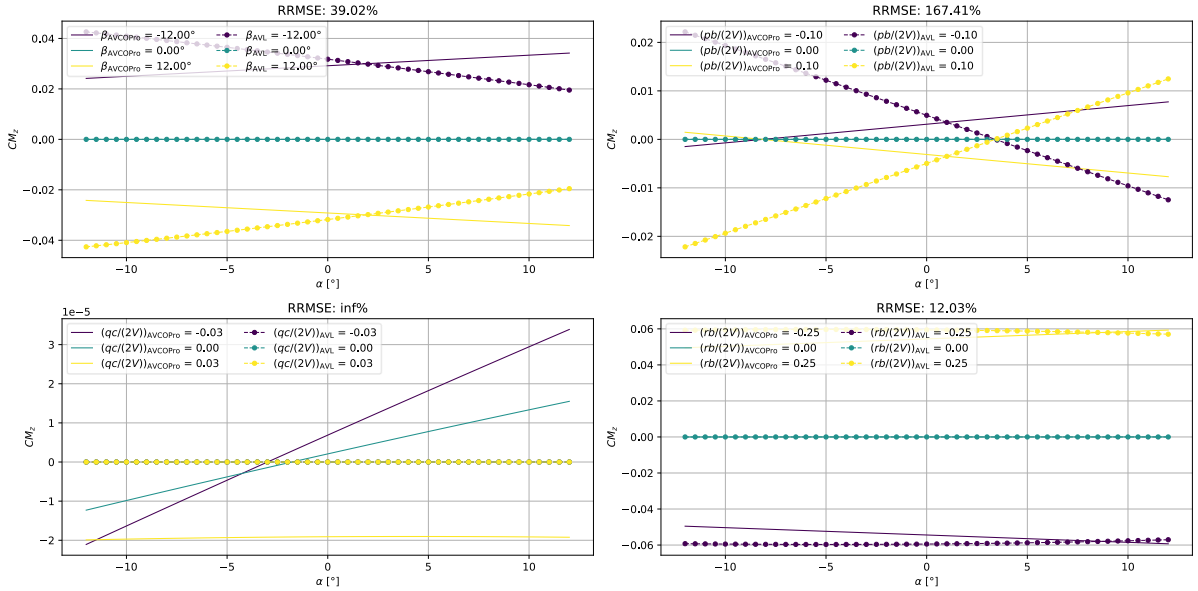
Slika 6.26 Koeficijent momenta oko osi *y*, u ovisnosti o *pb*/(2*V*)



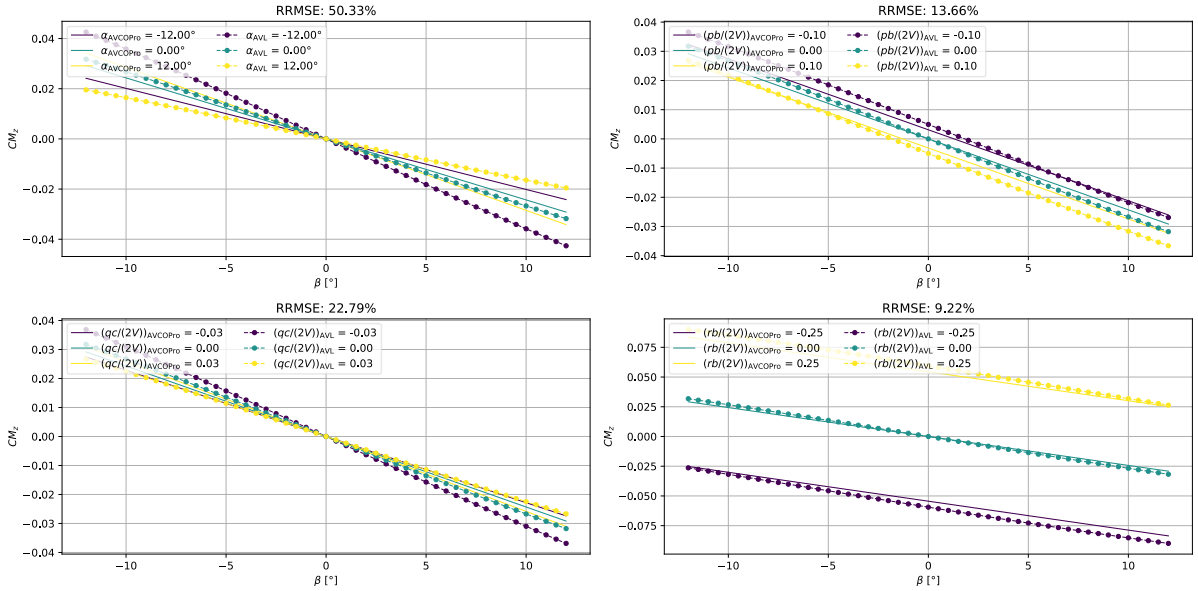
Slika 6.27 Koeficijent momenta oko osi *y*, u ovisnosti o *qc*/(2*V*)



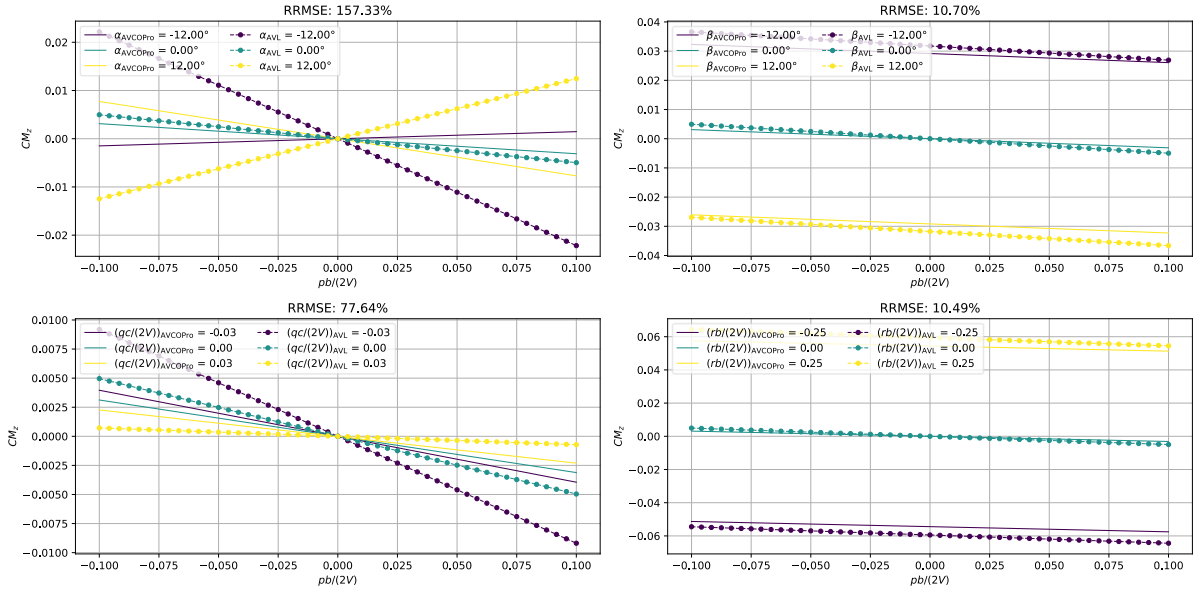
Slika 6.28 Koeficijent momenta oko osi *y*, u ovisnosti o *rb*/(2*V*)



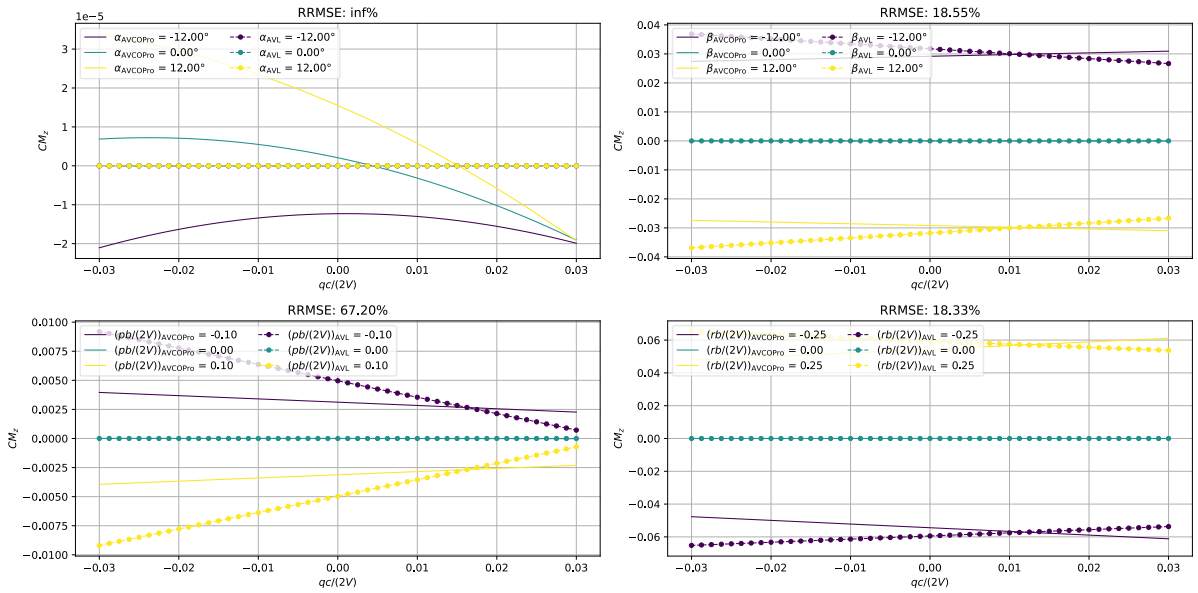
Slika 6.29 Koeficijent momenta oko osi *z*, u ovisnosti o *α*



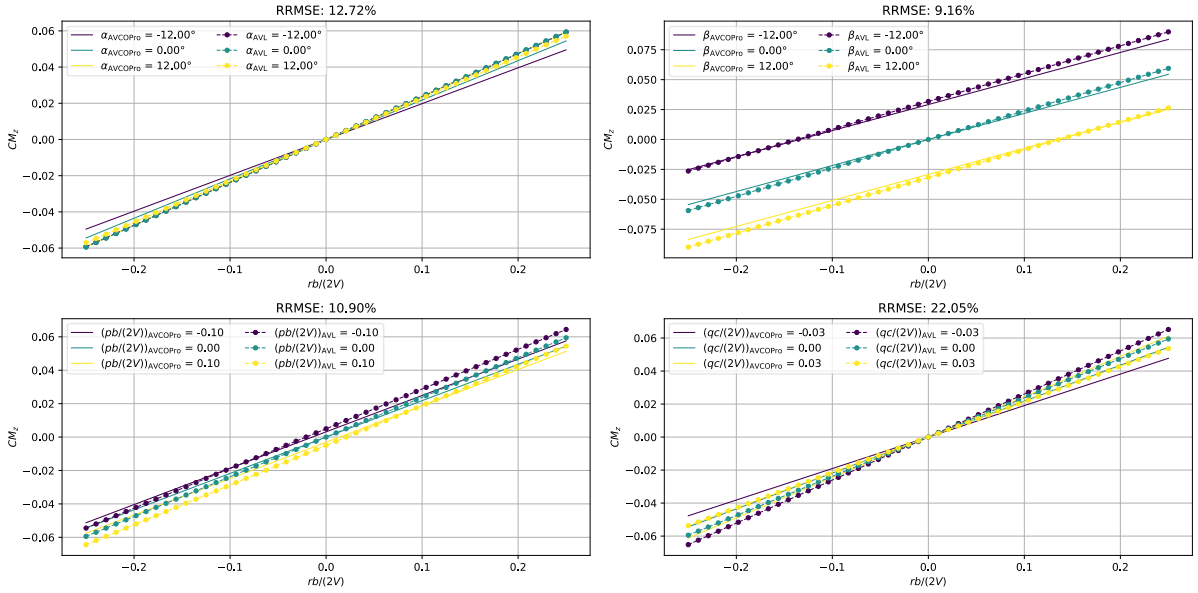
Slika 6.30 Koeficijent momenta oko osi *z*, u ovisnosti o *β*



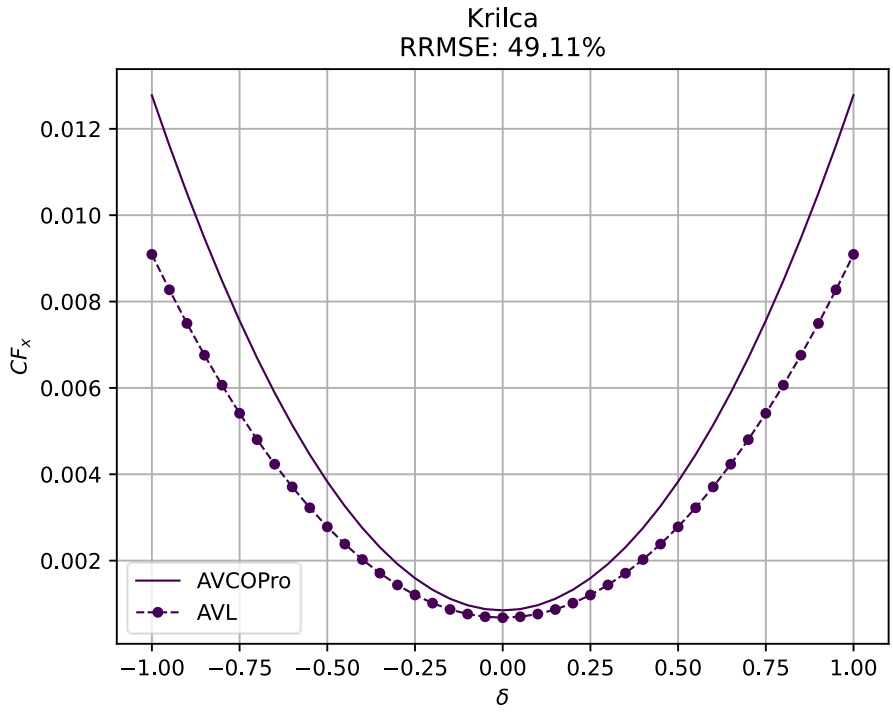
Slika 6.31 Koeficijent momenta oko osi *z*, u ovisnosti o *pb*/(2*V*)



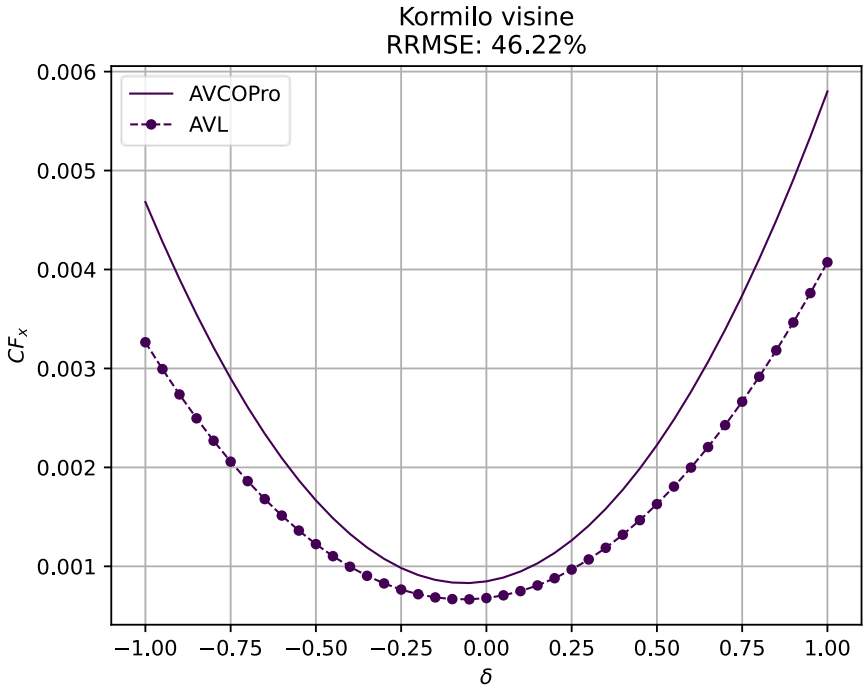
Slika 6.32 Koeficijent momenta oko osi *z*, u ovisnosti o *qc*/(2*V*)



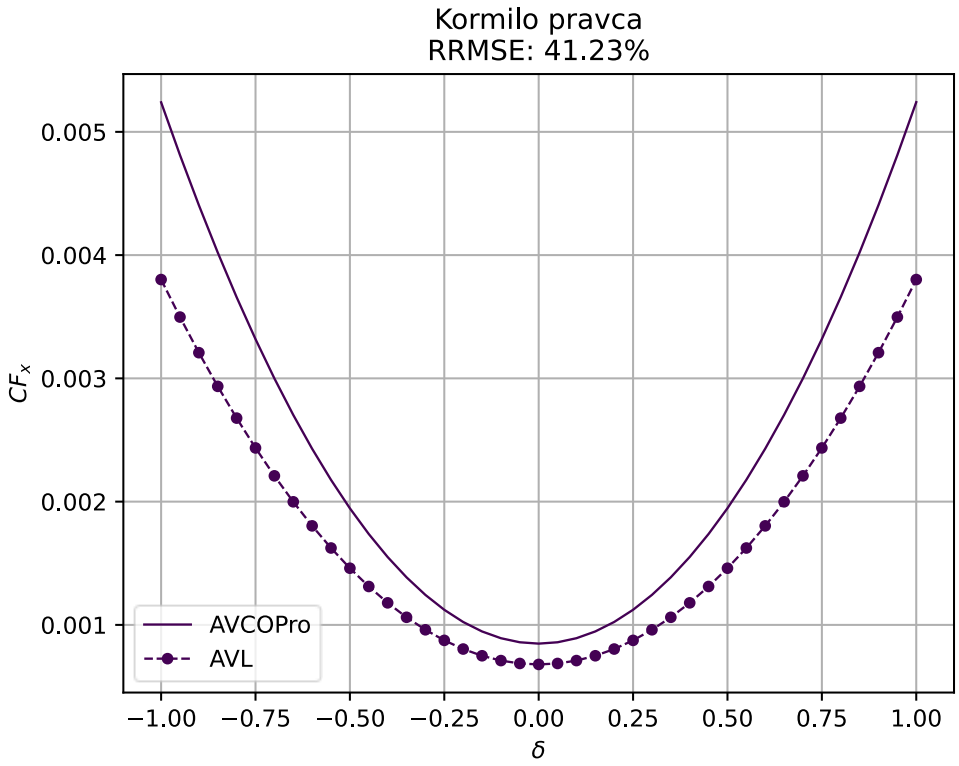
Slika 6.33 Koeficijent momenta oko osi *z*, u ovisnosti o *rb*/(2*V*)



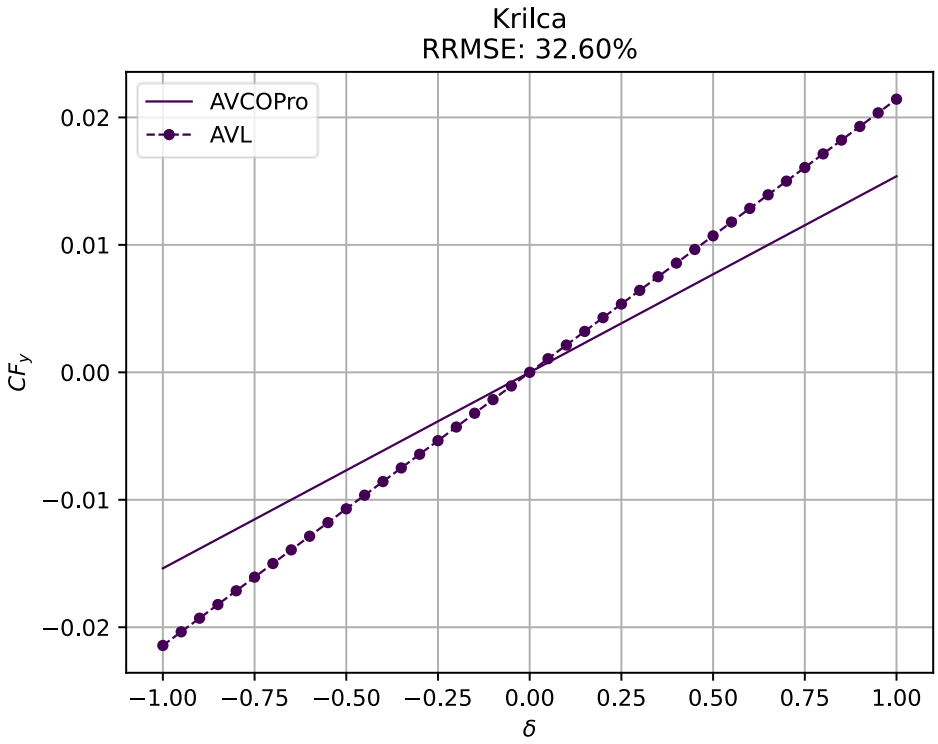
Slika 6.34 Koeficijent sile u smjeru osi *x*, u ovisnosti otklona krilaca



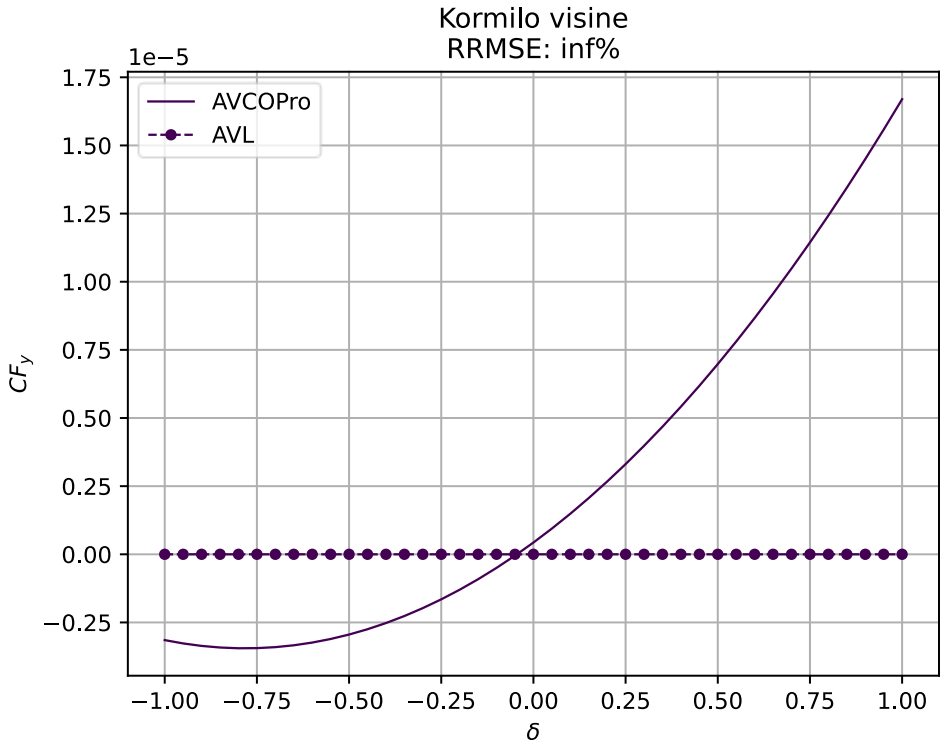
Slika 6.35 Koeficijent sile u smjeru osi *x*, u ovisnosti otklona kormila visine



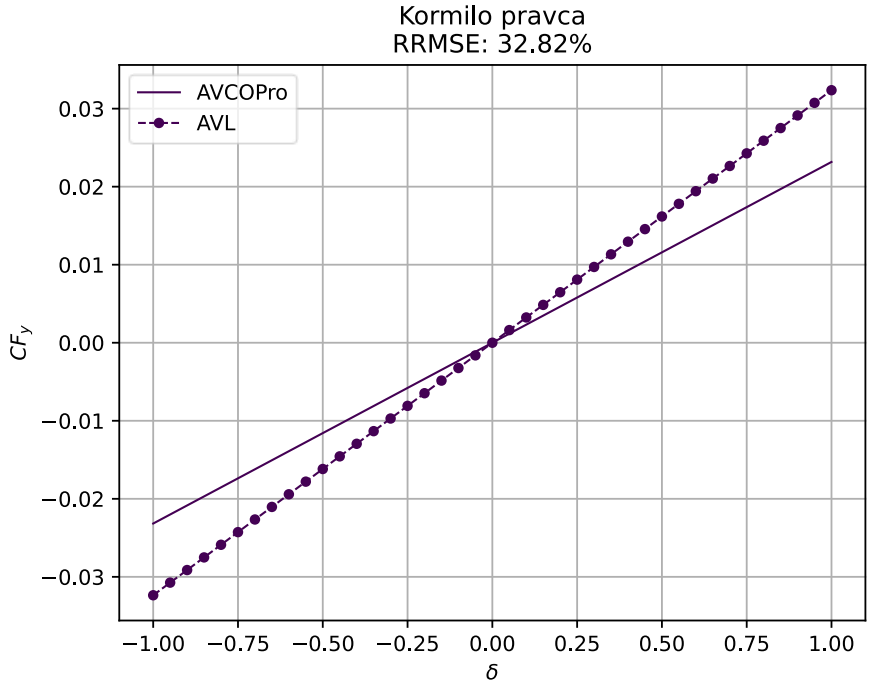
Slika 6.36 Koeficijent sile u smjeru osi *x*, u ovisnosti otklona kormila pravca



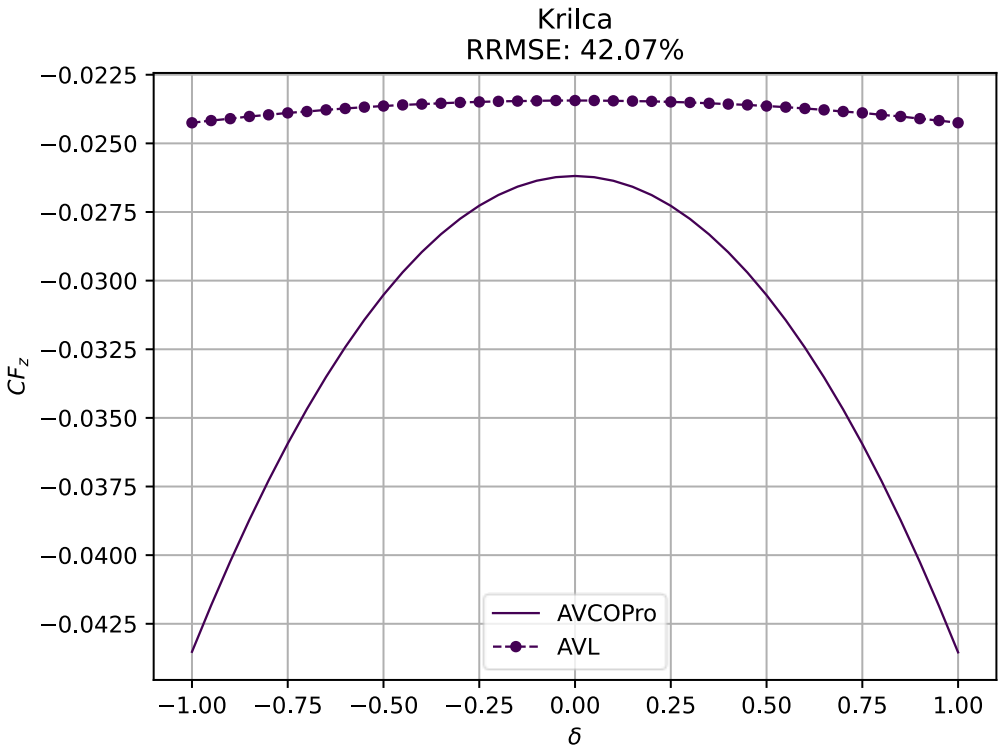
Slika 6.37 Koeficijent sile u smjeru osi *y*, u ovisnosti otklona krilaca



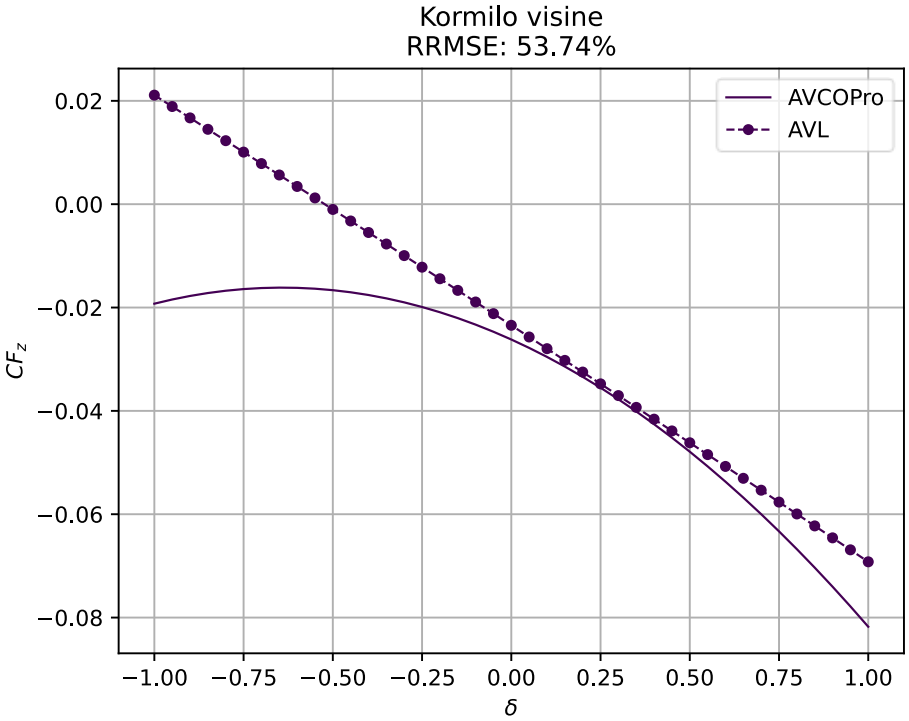
Slika 6.38 Koeficijent sile u smjeru osi *y*, u ovisnosti otklona kormila visine



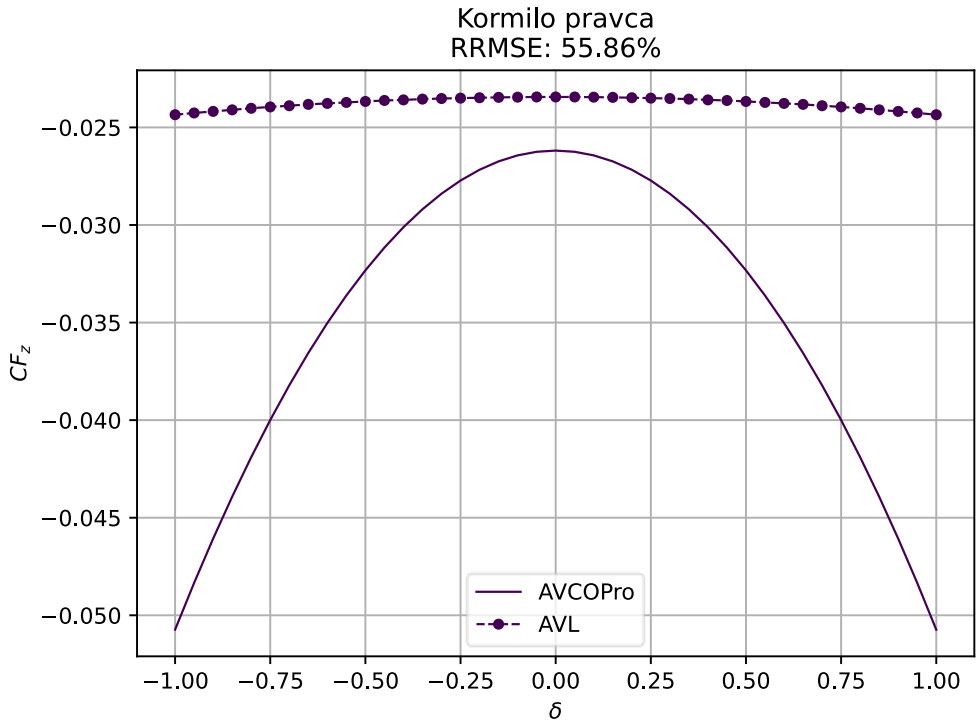
Slika 6.39 Koeficijent sile u smjeru osi *y*, u ovisnosti otklona kormila pravca



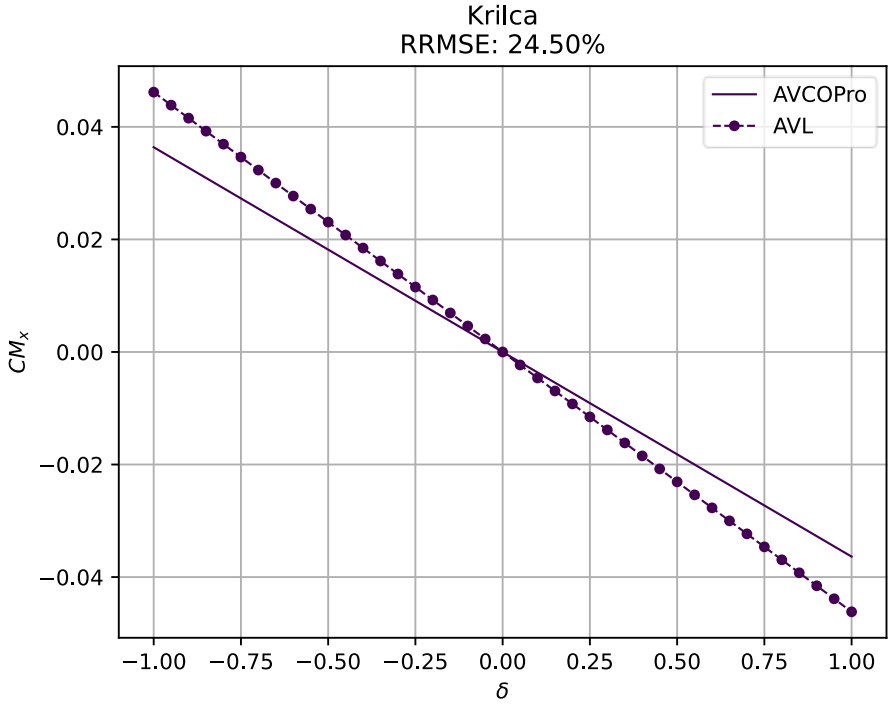
Slika 6.40 Koeficijent sile u smjeru osi *z*, u ovisnosti otklona krilaca



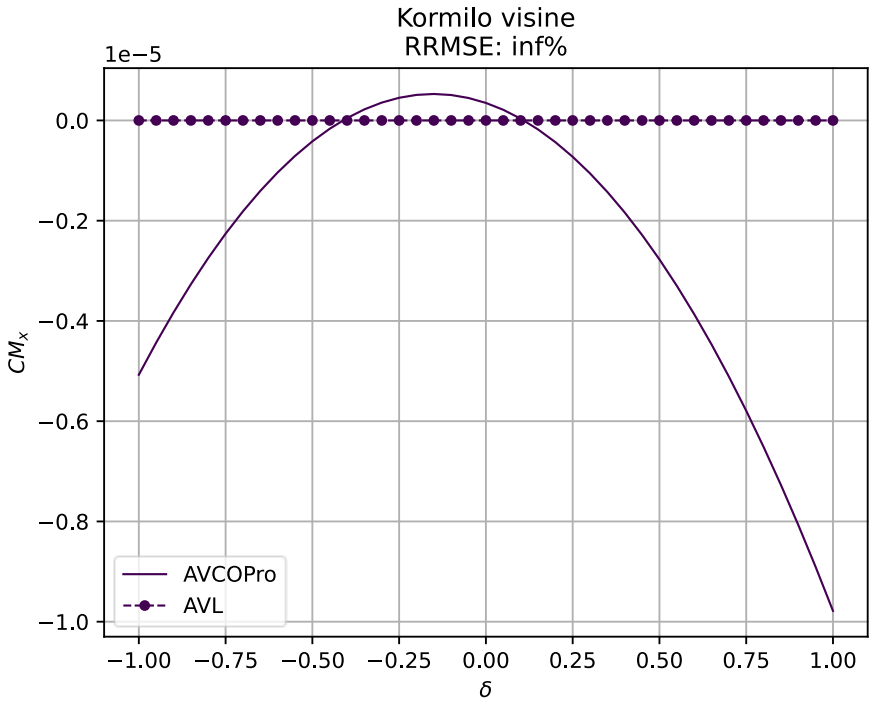
Slika 6.41 Koeficijent sile u smjeru osi *z*, u ovisnosti otklona kormila visine



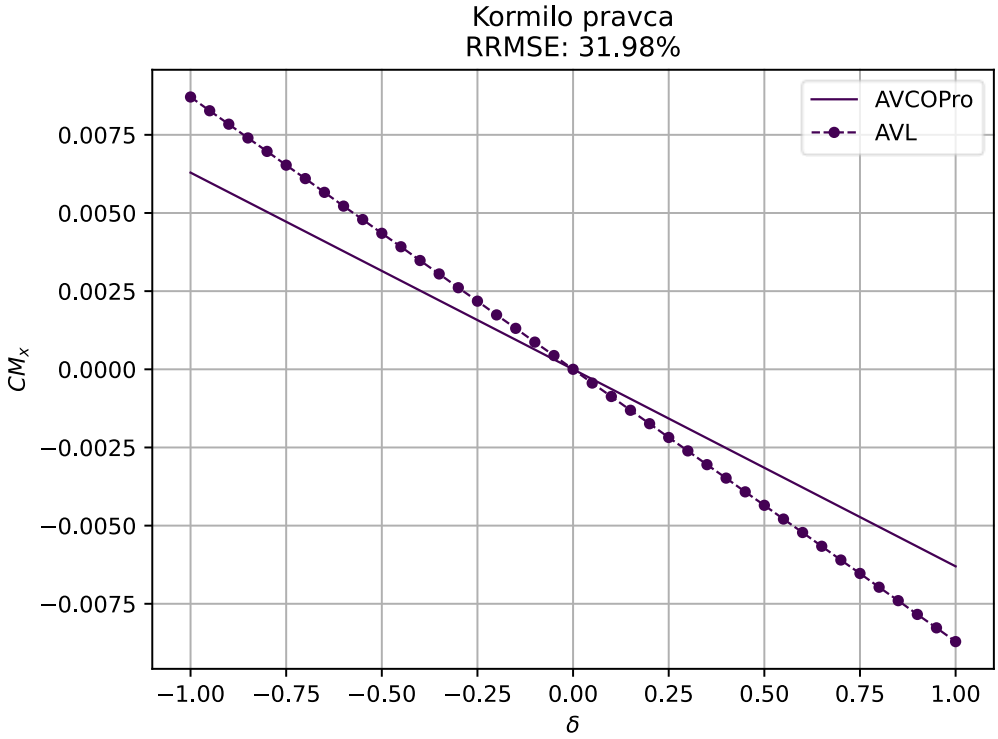
Slika 6.42 Koeficijent sile u smjeru osi *z*, u ovisnosti otklona kormila pravca



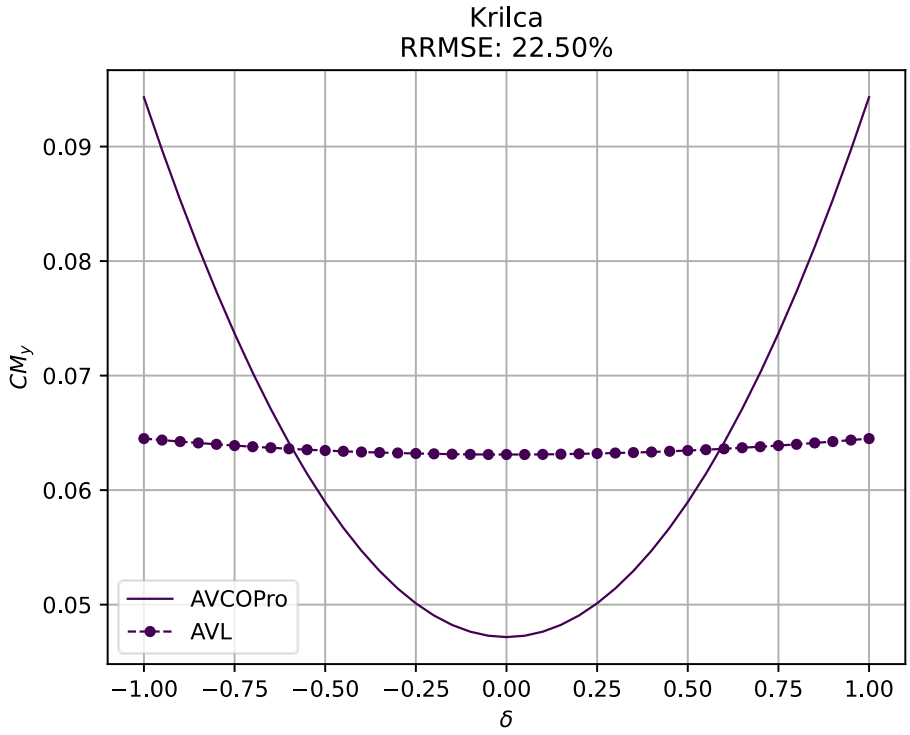
Slika 6.43 Koeficijent momenta oko osi *x*, u ovisnosti otklona krilaca



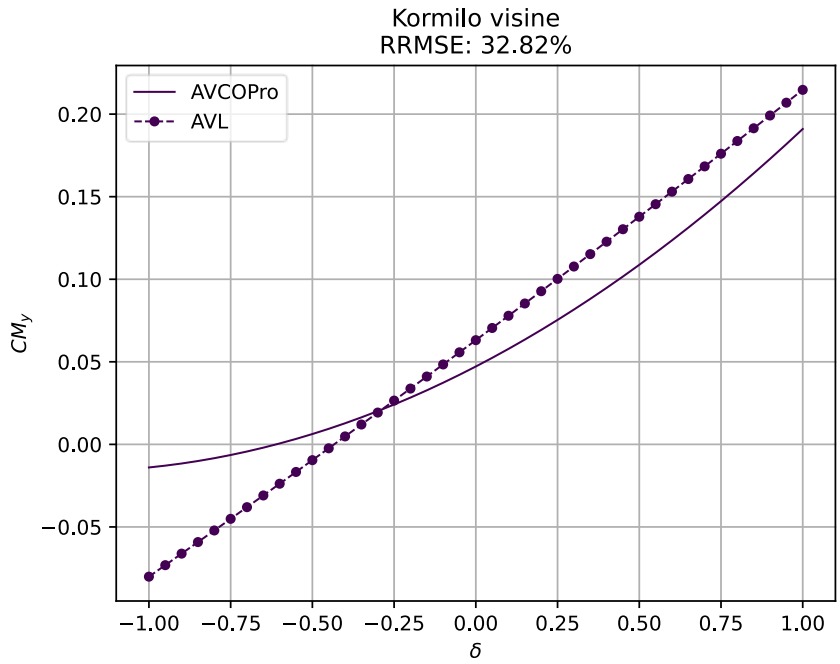
Slika 6.44 Koeficijent momenta oko osi *x*, u ovisnosti otklona kormila visine



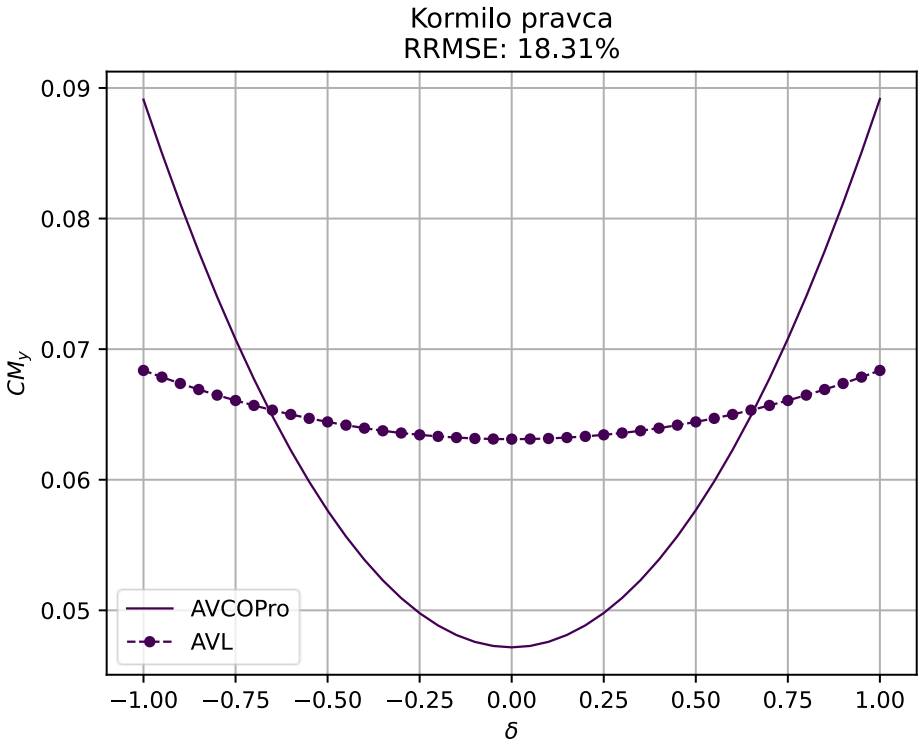
Slika 6.45 Koeficijent momenta oko osi *x*, u ovisnosti otklona kormila pravca



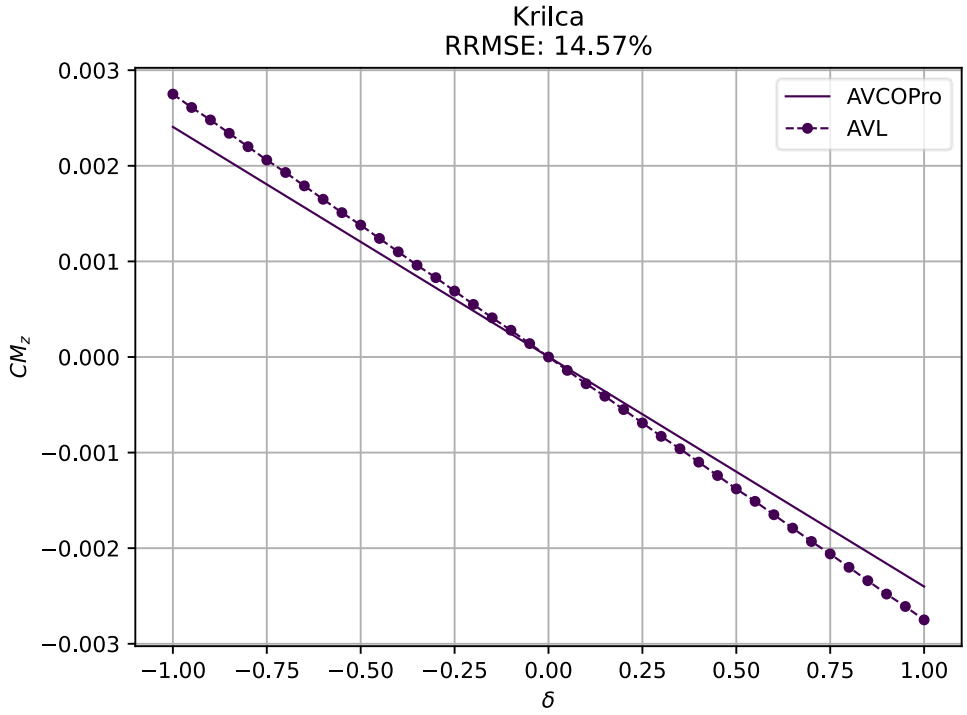
Slika 6.46 Koeficijent momenta oko osi *y*, u ovisnosti otklona krilaca



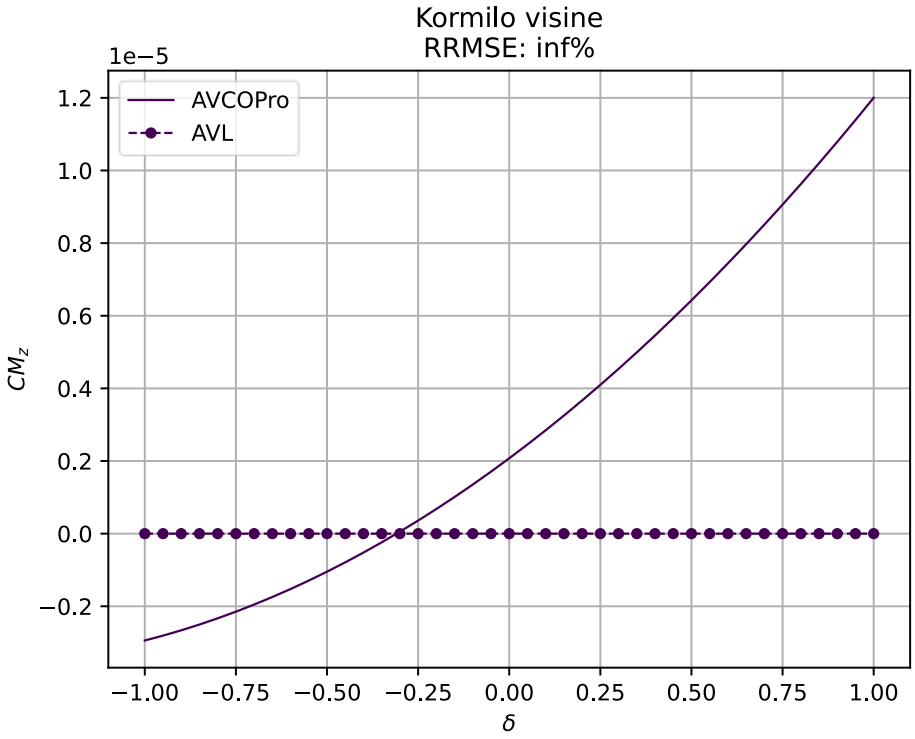
Slika 6.47 Koeficijent momenta oko osi *y*, u ovisnosti otklona kormila visine



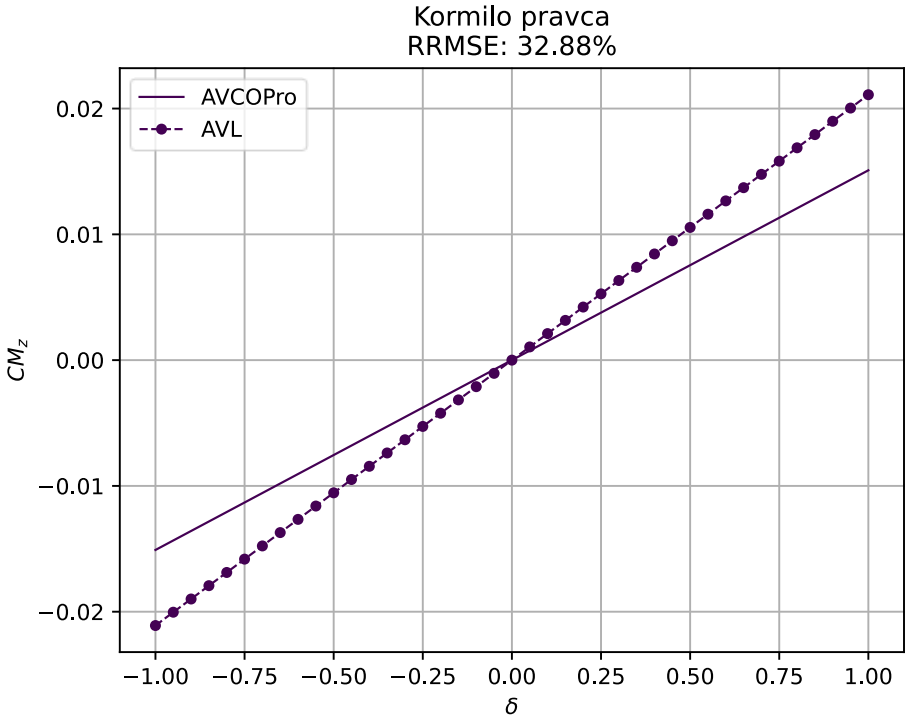
Slika 6.48 Koeficijent momenta oko osi *y*, u ovisnosti otklona kormila pravca



Slika 6.49 Koeficijent momenta oko osi *z*, u ovisnosti otklona krilaca



Slika 6.50 Koeficijent momenta oko osi *z*, u ovisnosti otklona kormila visine



Slika 6.51 Koeficijent momenta oko osi *z*, u ovisnosti otklona kormila pravca

Iz dobivenih rezultata, vidi se da se dobiva zadovoljavajuće poklapanje. Egzaktno poklapanje nije niti za očekivati, pošto AVL koristi metodu vrtložne rešetke, dok se ovdje koristi metoda panela. Relativna pogreška je nešto veća u slučajevima, gdje ulazni argument ima slabi utjecaj na izlazni argument (npr. utjecaj napadnog kuta na moment oko *x*-osi). Što se tiče modeliranja utjecaja kontrolnih ploha, i ovdje su poklapanja zadovoljavajuća, iako su razlike nešto izraženije. Postoje dva razloga zbog čega ovdje dolaze razlike više do izražaja. Prvo, AVL modelira utjecaj kontrolnih ploha na način da samo zarotira vektore normale na mjestu gdje se nalaze kontrolne plohe (normale se koriste za ostvarivanje uvjeta nepromočivosti). Drugo, u AVCOPro-u, kao što je već opisano, utjecaj otklona kontrolnih ploha se računa na temelju kvadratne interpolacije referentnih stanja a ne rješavanjem čitavog sustava (6.1) za različite otklone.

7 DINAMIKA LETA

Kako bi se mogao izraditi dinamički model leta zrakoplova, zrakoplov se može promatrati kao kruto tijelo na koje djeluju različite sile što znači da će dinamika zrakoplova biti opisana Newton-Eulerovim jednadžbama. Pomoću Newton-Eulerovih jednadžbi će se moći pronaći stanje zrakoplova u koordinatnom sustavu letjelice [10]:

$$m\left(\dot{\mathbf{V}} + \tilde{\boldsymbol{\omega}}\mathbf{V}\right) = \mathbf{F}_{\mathrm{A}} + \mathbf{F}_{\mathrm{T}} + \mathbf{F}_{\mathrm{G}}, \tag{7.1}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}\mathbf{I}\boldsymbol{\omega} = \mathbf{M}_{\mathrm{A}} + \mathbf{M}_{\mathrm{T}}, \tag{7.2}$$

$$\dot{\boldsymbol{\Phi}} = \mathbf{H}\boldsymbol{\omega}, \tag{7.3}$$

gdje je:

m – masa zrakoplova u nekom vremenskom trenutku

$\mathbf{V} = [u \quad v \quad w]^T$ – vektor brzine leta u k.s. letjelice,

$\boldsymbol{\omega} = [p \quad q \quad r]^T$ – vektor kutne brzine zrakoplova u k.s. letjelice,

$\tilde{\boldsymbol{\omega}}$ – koso-simetrična matrica vektora $\boldsymbol{\omega}$,

$\mathbf{F}_{\mathrm{A}} = [F_{\mathrm{A}x} \quad F_{\mathrm{A}y} \quad F_{\mathrm{A}z}]^T$ – vektor aerodinamičke sile u k.s. letjelice,

$\mathbf{F}_{\mathrm{T}} = T[F_{\mathrm{T}x} \quad F_{\mathrm{T}y} \quad F_{\mathrm{T}z}]^T$ – vektor pogonske sile u k.s. letjelice,

$\mathbf{F}_{\mathrm{G}} = [F_{\mathrm{G}x} \quad F_{\mathrm{G}y} \quad F_{\mathrm{G}z}]^T$ – vektor gravitacijske sile u k.s. letjelice,

$\mathbf{I}$ – matrica tenzora inercije zrakoplova u k.s. letjelice,

$\mathbf{M}_{\mathrm{A}} = [M_{\mathrm{A}x} \quad M_{\mathrm{A}y} \quad M_{\mathrm{A}z}]^T$ – vektor aerodinamičkog momenta u k.s. letjelice,

$\mathbf{M}_{\mathrm{T}} = T[M_{\mathrm{T}x} \quad M_{\mathrm{T}y} \quad M_{\mathrm{T}z}]^T$ – vektor pogonskog momenta u k.s. letjelice,

$\boldsymbol{\Phi} = [\phi \quad \theta \quad \psi]^T$ – vektor koji sadrži kutove stava (Eulerove kuteve) i

$\mathbf{H}$ – matrica koja povezuje vektore $\boldsymbol{\omega}$ i $\dot{\boldsymbol{\Phi}}$ (Eulerova kinematička matrica).

Važno je naglasiti da se konvencionalno k.s. letjelice poklapa sa glavnim osima tromosti, no ovdje, smatra se da se x-os k.s. letjelice proteže duž trupa i gleda prema naprijed.

Iz gore navedenog sustava, moguće je izračunati sve relevantne performanse zrakoplova kao što je horizontalni let, penjanje, valjanje, polijetanje itd.

7.1 Linearizacija 6 DOF modela

Ono što je ovdje zanimljivije je određivanje dinamičke stabilnosti letjelice. Da bi se to napravilo, potrebno je sustav jednadžbi (7.1) – (7.3) linearizirati oko neke referentne točke. Ta točka je obično stanje u ravnotežnom letu. Gore navedeni sustav će u linearizirano obliku glasiti:

$$\begin{aligned}\Delta\dot{V} = \frac{1}{m}\left(\left(\frac{\partial F_{\mathrm{A}x}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 F_{\mathrm{T}x}\right)\Delta V + \frac{1}{m}\left(\frac{\partial F_{\mathrm{A}x}}{\partial \beta}\right)_0 \Delta\beta + \frac{1}{m}\left(\frac{\partial F_{\mathrm{A}x}}{\partial \alpha}\right)_0 \Delta\alpha \\ + \frac{1}{m}\left(\frac{\partial F_{\mathrm{A}x}}{\partial p}\right)_0 \Delta p + \left(\frac{1}{m}\left(\frac{\partial F_{\mathrm{A}x}}{\partial q}\right)_0 - V_0\alpha_0\right)\Delta q + \frac{1}{m}\left(\frac{\partial F_{\mathrm{A}x}}{\partial r}\right)_0 \Delta r \\ - g\cos(\theta_0)\Delta\theta,\end{aligned} \tag{7.4}$$

$$\begin{aligned}\Delta\dot{\beta} = \frac{1}{mV_0}\left(\left(\frac{\partial F_{\mathrm{A}y}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 F_{\mathrm{T}y}\right)\Delta V + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}y}}{\partial \beta}\right)_0 \Delta\beta + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}y}}{\partial \alpha}\right)_0 \Delta\alpha \\ + \left(\frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}y}}{\partial p}\right)_0 + \alpha_0\right)\Delta p + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}y}}{\partial q}\right)_0 \Delta q \\ + \left(\frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}y}}{\partial r}\right)_0 - 1\right)\Delta r + \frac{g}{V_0}\cos(\theta_0)\,\Delta\phi,\end{aligned} \tag{7.5}$$

$$\begin{aligned}\Delta\dot{\alpha} = \frac{1}{mV_0}\left(\left(\frac{\partial F_{\mathrm{A}z}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 F_{\mathrm{T}z}\right)\Delta V + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}z}}{\partial \beta}\right)_0 \Delta\beta + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}z}}{\partial \alpha}\right)_0 \Delta\alpha \\ + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}z}}{\partial p}\right)_0 \Delta p + \left(\frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}z}}{\partial q}\right)_0 + 1\right)\Delta q + \frac{1}{mV_0}\left(\frac{\partial F_{\mathrm{A}z}}{\partial r}\right)_0 \Delta r \\ - \frac{g}{V_0}\sin(\theta_0)\Delta\theta,\end{aligned} \tag{7.6}$$

$$\begin{aligned}\Delta\dot{p} = \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\left(\frac{\partial M_{\mathrm{A}x}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 M_{\mathrm{T}x}\right) - I_{xz}\left(\left(\frac{\partial M_{\mathrm{A}z}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 M_{\mathrm{T}z}\right)\right)\Delta V \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial \beta}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}z}}{\partial \beta}\right)_0\right)\Delta\beta \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial \alpha}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}z}}{\partial \alpha}\right)_0\right)\Delta\alpha \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial p}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}z}}{\partial p}\right)_0\right)\Delta p \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial q}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}z}}{\partial q}\right)_0\right)\Delta q \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{zz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial r}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}z}}{\partial r}\right)_0\right)\Delta r,\end{aligned} \tag{7.7}$$

$$\begin{aligned}\Delta\dot{q} = \frac{1}{I_{yy}}\left(\left(\frac{\partial M_{\mathrm{A}y}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 M_{\mathrm{T}y}\right)\Delta V + \frac{1}{I_{yy}}\left(\frac{\partial M_{\mathrm{A}y}}{\partial \beta}\right)_0 \Delta\beta + \frac{1}{I_{yy}}\left(\frac{\partial M_{\mathrm{A}y}}{\partial \alpha}\right)_0 \Delta\alpha \\ + \frac{1}{I_{yy}}\left(\frac{\partial M_{\mathrm{A}y}}{\partial p}\right)_0 \Delta p + \frac{1}{I_{yy}}\left(\frac{\partial M_{\mathrm{A}y}}{\partial q}\right)_0 \Delta q + \frac{1}{I_{yy}}\left(\frac{\partial M_{\mathrm{A}y}}{\partial r}\right)_0 \Delta r,\end{aligned} \tag{7.8}$$

$$\begin{aligned}\Delta\dot{r} = \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\left(\frac{\partial M_{\mathrm{A}z}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 M_{\mathrm{T}z}\right) - I_{xz}\left(\left(\frac{\partial M_{\mathrm{A}x}}{\partial V}\right)_0 + \left(\frac{\partial T}{\partial V}\right)_0 M_{\mathrm{T}x}\right)\right)\Delta V \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\frac{\partial M_{\mathrm{A}z}}{\partial \beta}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial \beta}\right)_0\right)\Delta\beta \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\frac{\partial M_{\mathrm{A}z}}{\partial \alpha}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial \alpha}\right)_0\right)\Delta\alpha \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\frac{\partial M_{\mathrm{A}z}}{\partial p}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial p}\right)_0\right)\Delta p \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\frac{\partial M_{\mathrm{A}z}}{\partial q}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial q}\right)_0\right)\Delta q \\ + \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\left(I_{xx}\left(\frac{\partial M_{\mathrm{A}z}}{\partial r}\right)_0 - I_{xz}\left(\frac{\partial M_{\mathrm{A}x}}{\partial r}\right)_0\right)\Delta r,\end{aligned} \tag{7.9}$$

$$\Delta\dot{\phi} = \Delta p + \tan(\theta_0)\,\Delta r, \tag{7.10}$$

$$\Delta\dot{\theta} = \Delta q, \tag{7.11}$$

gdje je:

V_0 – Brzina u ravnotežnom letu,

α_0 – napadni kut u ravnotežnom letu i

$\theta_0 = \alpha_0$ – kut propinjanja, koji je u ravnotežnom letu jednak napadnom kutu.

Varijabla ψ se nije razmatrala, pošto nema utjecaj na dinamiku letjelice.

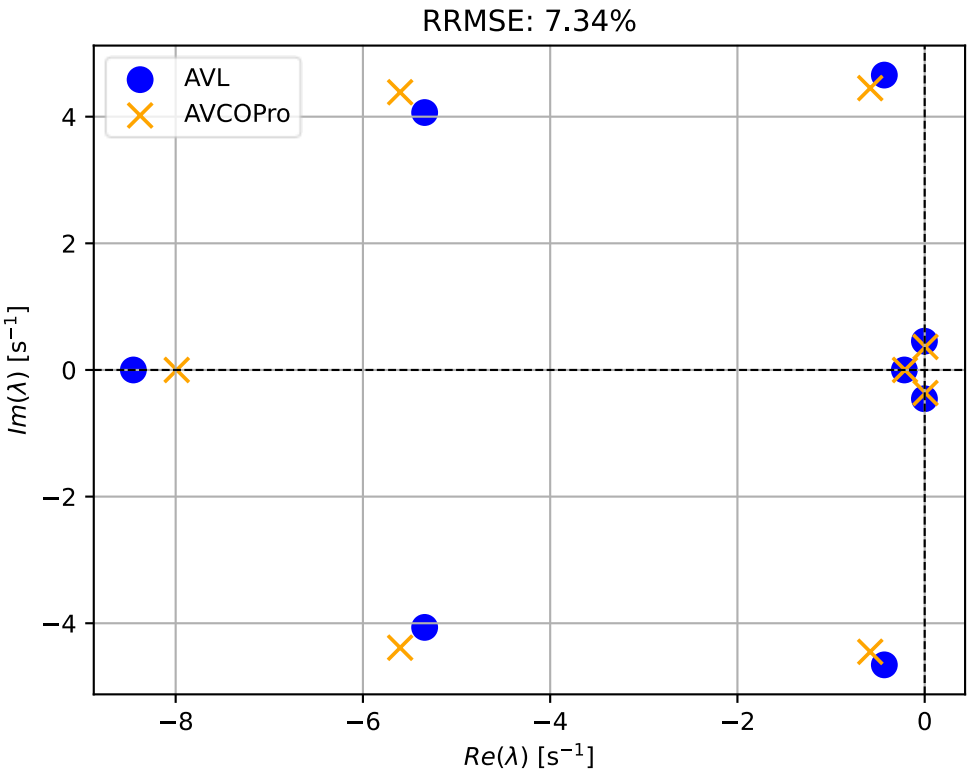
Linearizirani sustav se sada može zapisati u obliku jednadžbe stanja:

$$\Delta\dot{\mathbf{X}} = \mathbf{A}\Delta\mathbf{X}. \tag{7.12}$$

Svojstvene vrijednosti matrice **A** su tzv. korijeni sustava, te njihove vrijednosti govore stabilnosti i dinamičkom ponašanju letjelice.

7.2 Usporedba dinamičkog modela sa programom AVL

Za provjeru ispravnosti ovog modula, ponovno se iskoristila ista testna geometrija (vidi sliku 5.1 ili 6.3), kako bi se usporedila analiza dinamičke stabilnosti u programu AVCOPro i programu AVL. Rezultat se mogu vidjeti na slici 7.1.

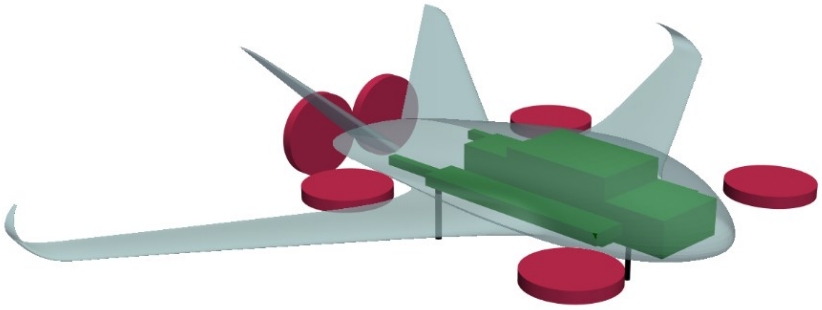


Slika 7.1 Rezultat usporedbe vrijednosti korijena

Iz priložene slike se vidi da je poklapanje izvrsno, što upućuje na to da je ispravno integrirana aerodinamika u 6 DOF model te da su jednadžbe ispravno linearizirane.

8 POSTAVLJANJE I RJEŠAVANJE OPTIMIZACIJSKOG PROBLEMA

Problem koji će se rješavati kao primjer je projektiranje bespilotne letjelice za dostavu organa. Odabrana konfiguracija će biti *Blended Wing Body* sa uvlačivim VTOL propelerima (vidi sliku 8.1). Pogon je u potpunosti električni.



Slika 8.1 Početna geometrija letjelice

8.1 Formulacija optimizacijskog problema

Projektne varijable su parametri za definiranje geometrije krila stopljenog sa trupom, koji su opisani u 3. poglavlju. Osim navedenih varijabli, javlja se visina leta kao varijabla te varijable dodatnih komponenti letjelice (promjer propelera, gabaritne dimenzije baterije i motora te položaji komponenti u trupu). Modeli pogonskih sustava (motora, propelera i baterije), dobiveni su stvaranjem surogat modela na temelju skupljanja podataka od različitih proizvođača, te kao takav, predstavlja jednostavan primjer pa se zbog toga neće detaljno objašnjavati. Sveukupno, koristilo se 84 projektnih varijabli. Ciljevi, i zahtjevi (ograničenja jednakosti i nejednakosti), mogu se vidjeti u sljedećim tablicma. Korišteni hiperparametri optimizatora su: $\alpha = 0.001$, $\beta_1 = 0.9$ i $\beta_2 = 0.999$, dok su početni penalizacijski koeficijenti bili: $\mu_{\mathrm{h}} = 10$, $\mu_{\mathrm{g}} = 10$, $\mu_{\mathrm{gI}} = 2 \cdot 10^{-4}$, te su se u svakoj iteraciji mijenjali na slijedeći način: $\Delta\mu_{\mathrm{h}} = 2$, $\Delta\mu_{\mathrm{g}} = 2$, $\Delta\mu_{\mathrm{gI}} = -99 \cdot 10^{-6}$.

Tablica 8.1 Funkcije cilja

	Funkcija	Opis
$f_1(\mathbf{x})$	m	Minimizirati masu letjelice
$f_2(\mathbf{x})$	t	Minimizirati vrijeme koje je potrebno da letjelica stigne do odredišta
$f_3(\mathbf{x})$	V_a	Minimizirati brzinu prilaza

Tablica 8.2 Ograničenja jednakosti

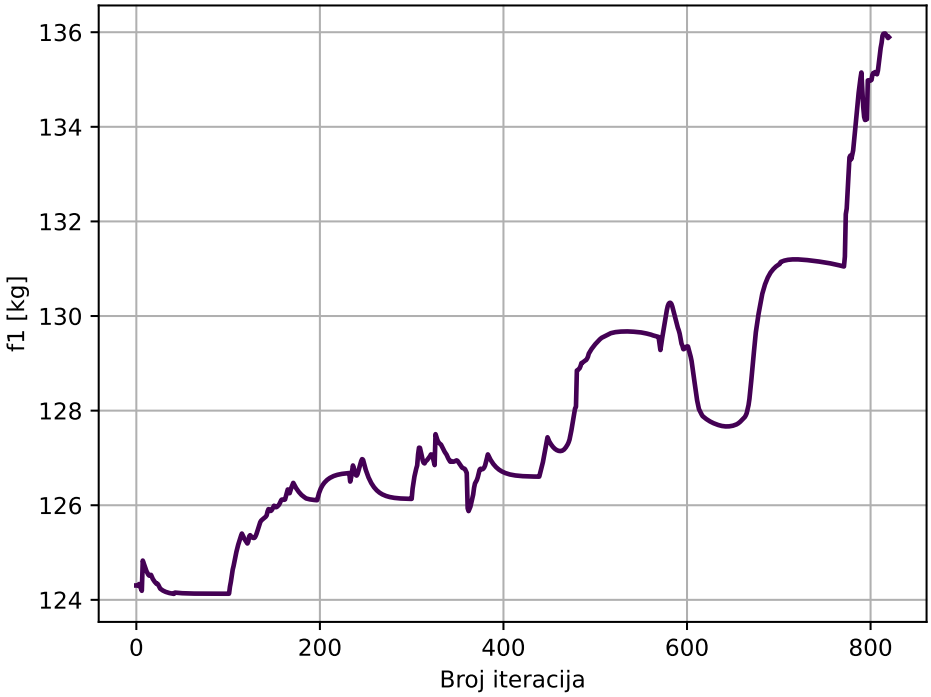
	Funkcija	Opis
$h_1(\mathbf{x})$	$e_1 + 8.2700$	Elementarni simetrični polinomi dinamičkog sustava letjelice, moraju odgovarati onima koji su dobiveni iz korijena, koji predstavljaju željenu dinamiku. Ciljani korijeni su [s^{-1}]: • -1.72 + 2.46i • -1.72 - 2.46i • -0.03 + 0.075i • -0.03 - 0.075i • -0.36 + 0.17i • -0.36 - 0.17i • -4.0 • -0.05
$h_2(\mathbf{x})$	$e_2 - 29.1924$	
$h_3(\mathbf{x})$	$e_3 + 56.8031$	
$h_4(\mathbf{x})$	$e_4 - 35.7274$	
$h_5(\mathbf{x})$	$e_5 + 9.4836$	
$h_6(\mathbf{x})$	$e_6 - 0.9274$	
$h_7(\mathbf{x})$	$e_7 + 0.0641$	
$h_8(\mathbf{x})$	$e_8 - 0.0019$	

Tablica 8.2 Ograničenja nejednakosti

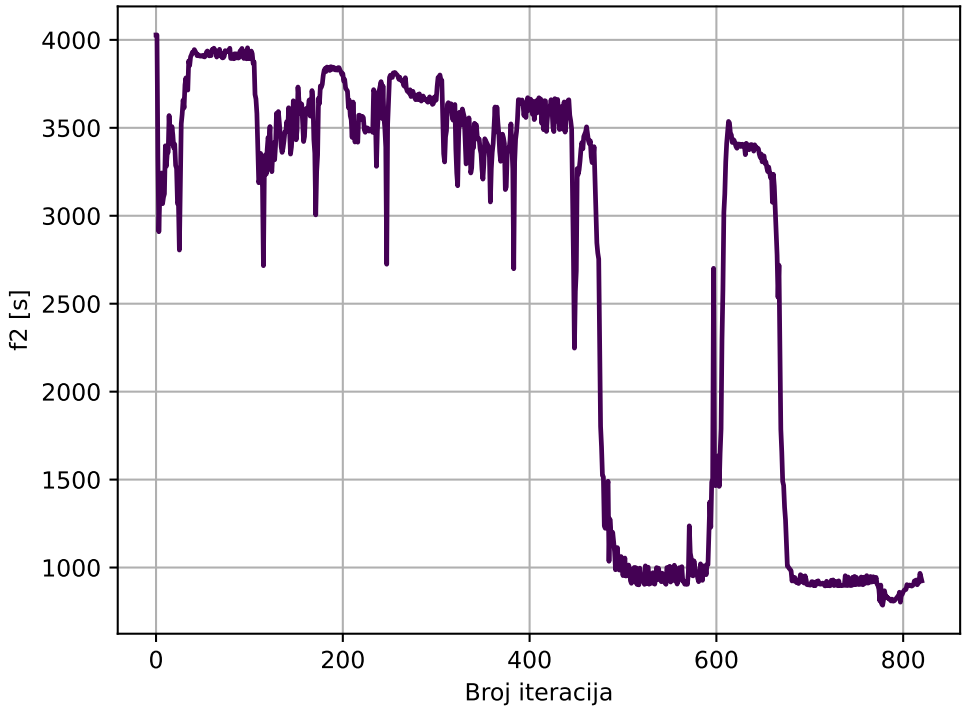
	Funkcija	Opis
$g_1(\mathbf{x})$	$x_{\mathrm{CM}} - x_{\mathrm{G1}}$	*x* koordinata centra mase, mora biti veća od *x* koordinate prednje noge (radi se u konstrukcijskom k.s.)
$g_2(\mathbf{x})$	$x_{\mathrm{G2}} - x_{\mathrm{CM}}$	*x* koordinata stražnje noge, mora biti veća od *x* centra mase (radi se u konstrukcijskom k.s.)
$g_3(\mathbf{x})$	$(T_{\mathrm{VTOL}})_{\max} - T_{\mathrm{TO}}$	Maksimalni potisak jednog VTOL propelera mora biti veći od zahtijevanog potiska za polijetanje
$g_4(\mathbf{x})$	$s - 100$	Domet mora biti veći od 100km
$g_5(\mathbf{x})$	$2 - t$	Trajanje leta ukupnog leta mora trajati ispod dva sata
$g_6(\mathbf{x})$	$w_{\mathrm{c}} - 13$	Minimalni bočni vjetar na kojeg letjelica mora biti otporna pri krstarenju, iznosi 13 m/s
$g_7(\mathbf{x})$	$\phi - 30$	U roku od 1.3s, letjelica se mora pri brzini prilaza zavaljati za minimalno 30°
$g_8(\mathbf{x})$	$90 - \phi$	U roku od 1.3s, letjelica se ne smije pri brzini prilaza zavaljati za više od 90°

8.2 Rezultati optimizacije

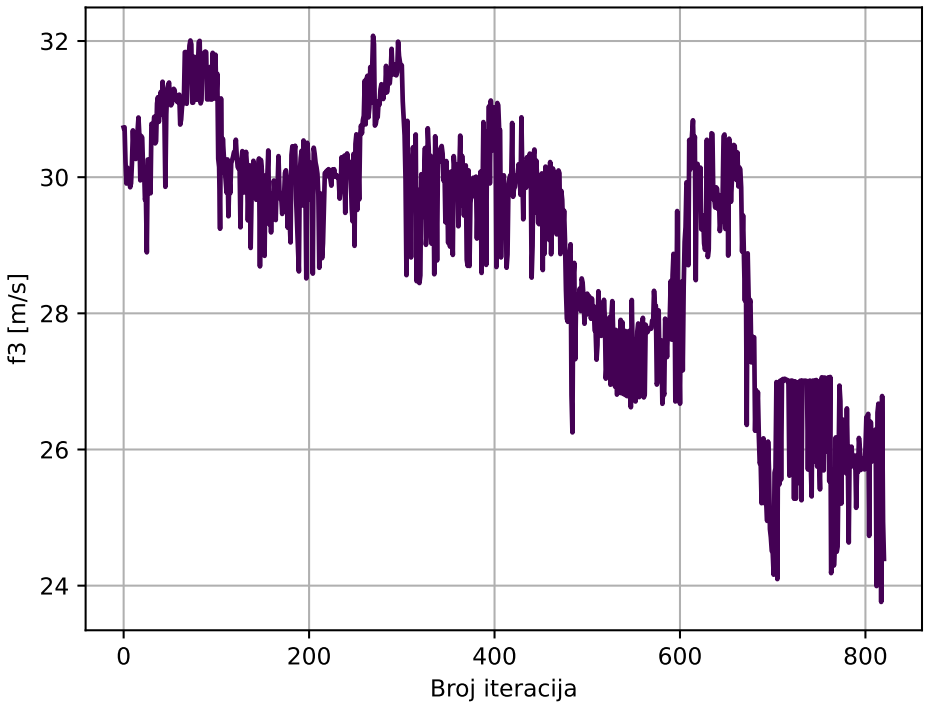
Na sljedećim slikama se vidi razvoj optimizacije tj. razvoj funkcija: cilja, ograničenja jednakosti i ograničenja nejednakosti.



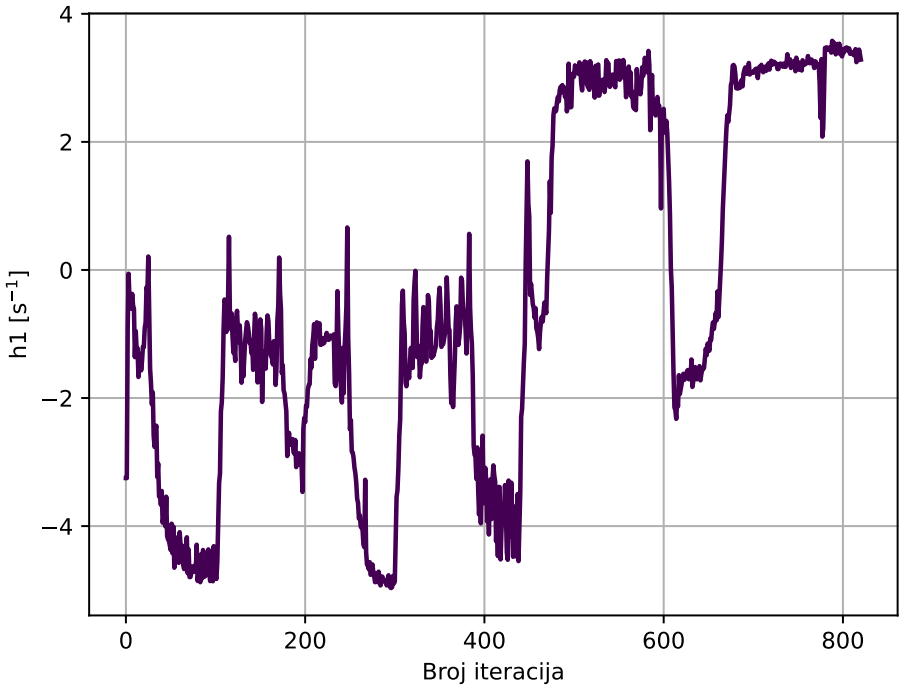
Slika 8.2 Razvoj funkcije cilja 1



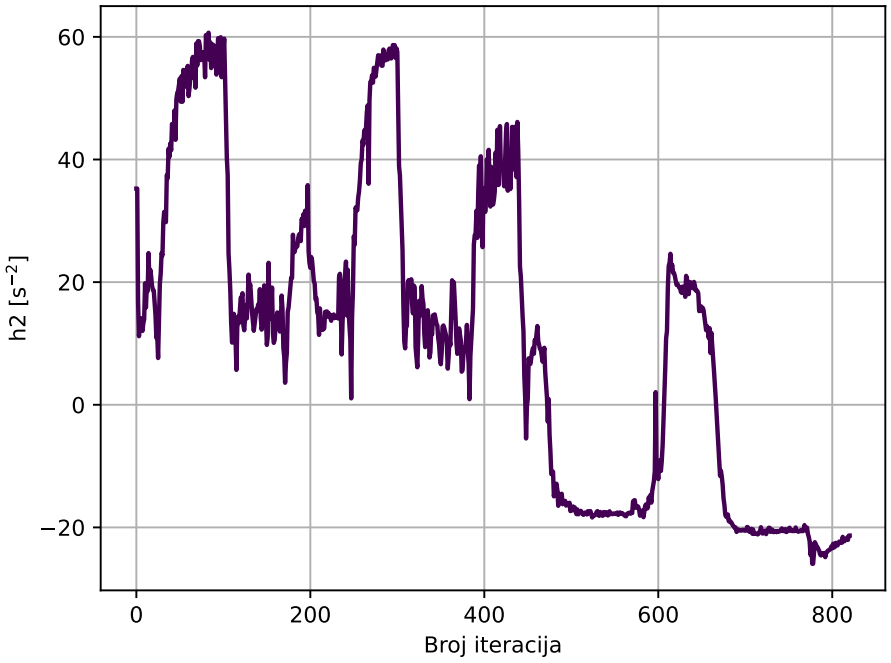
Slika 8.3 Razvoj funkcije cilja 2



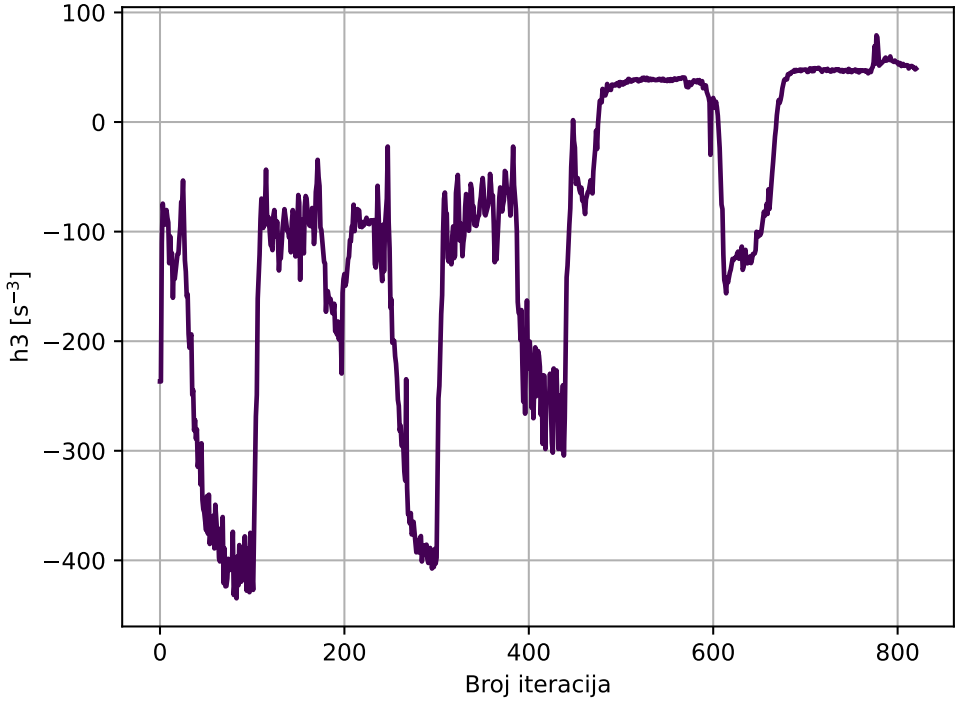
Slika 8.4 Razvoj funkcije cilja 3



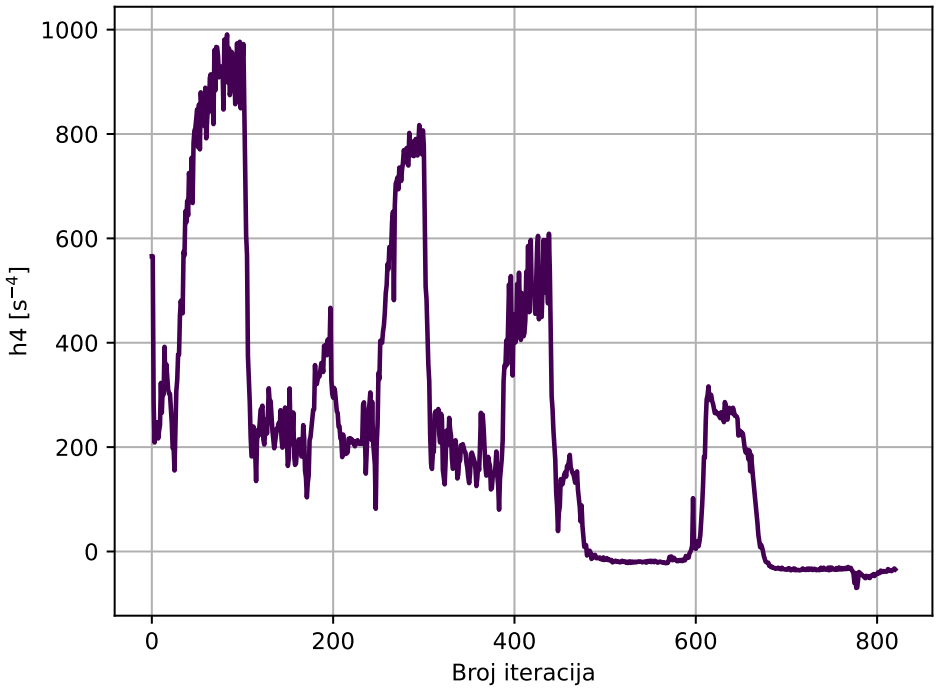
Slika 8.5 Razvoj funkcije ograničenja jednakosti 1



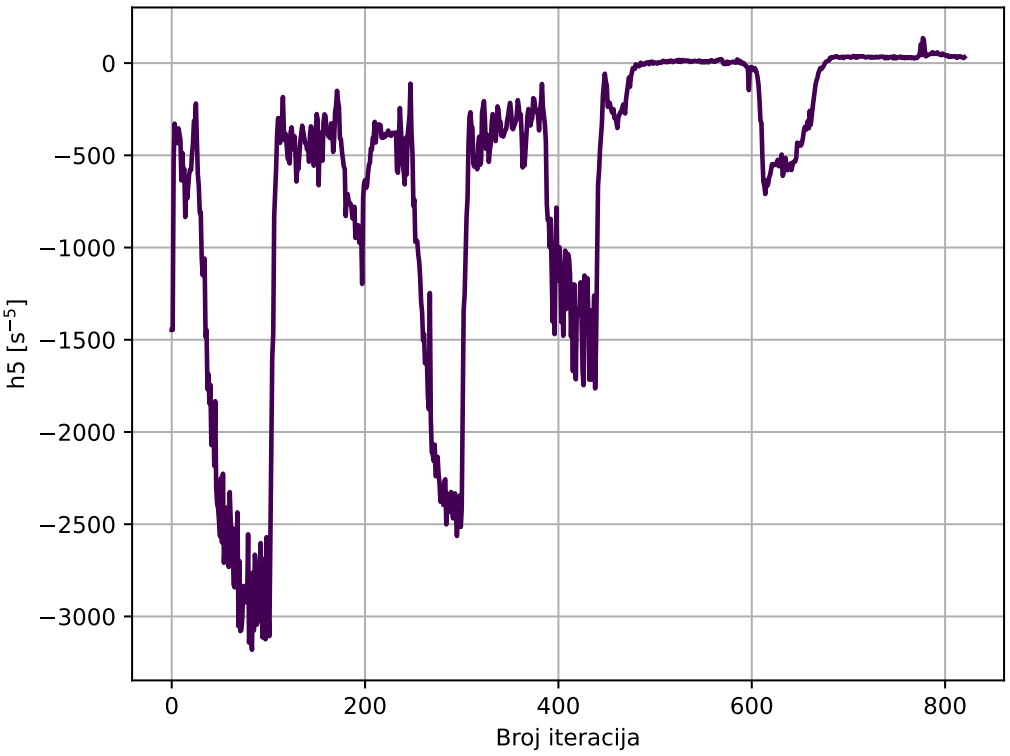
Slika 8.6 Razvoj funkcije ograničenja jednakosti 2



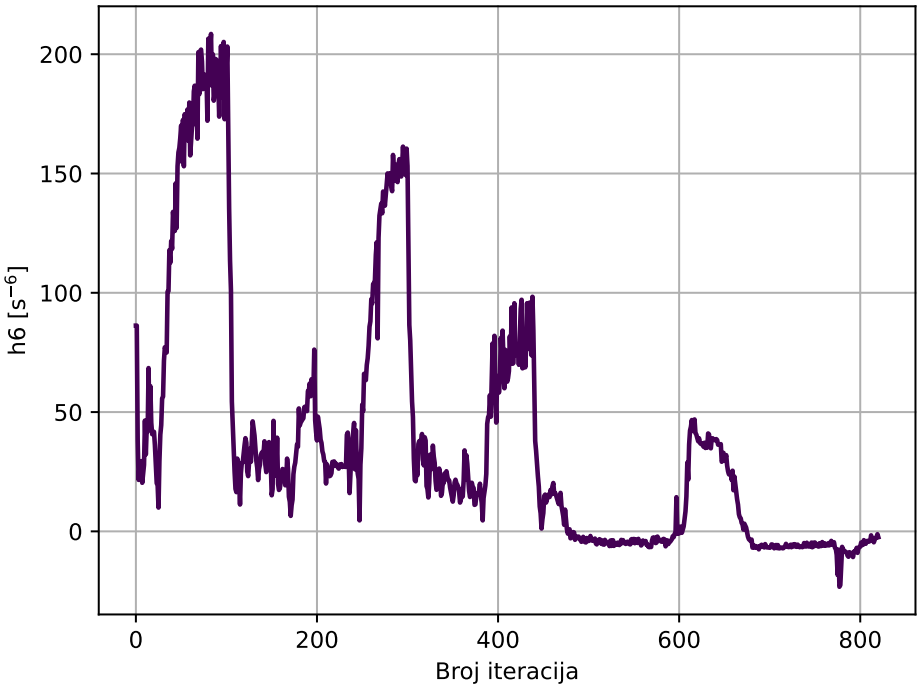
Slika 8.7 Razvoj funkcije ograničenja jednakosti 3



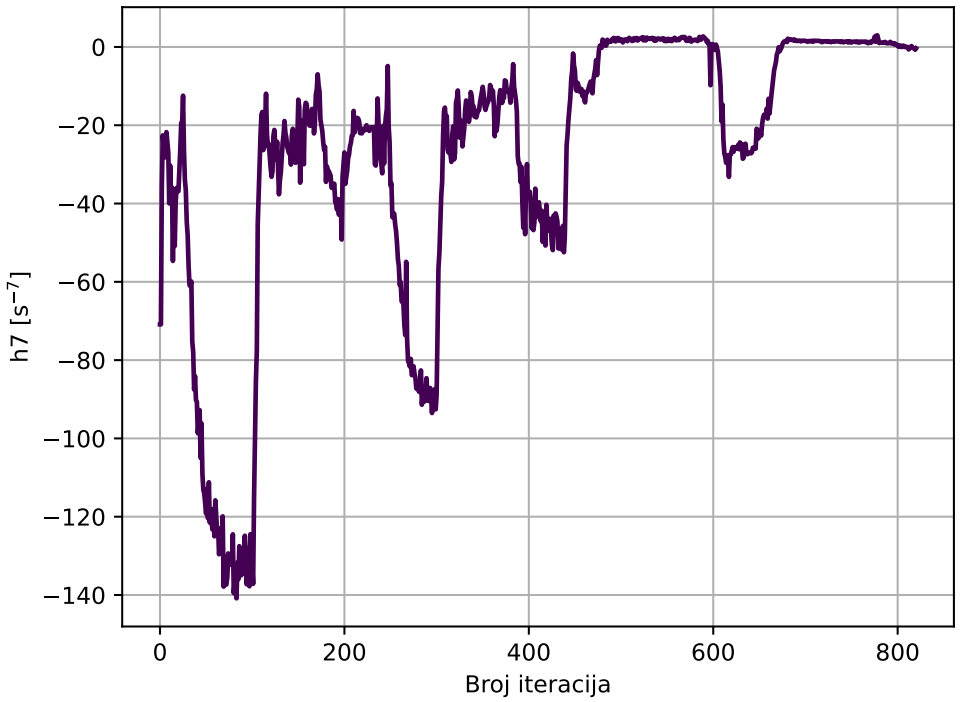
Slika 8.8 Razvoj funkcije ograničenja jednakosti 4



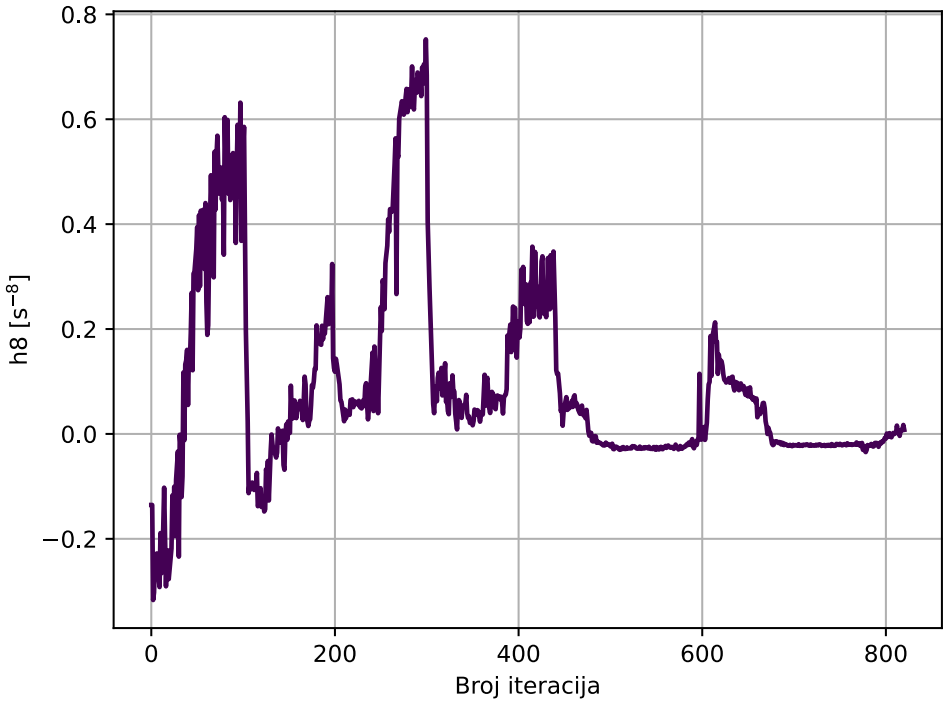
Slika 8.9 Razvoj funkcije ograničenja jednakosti 5



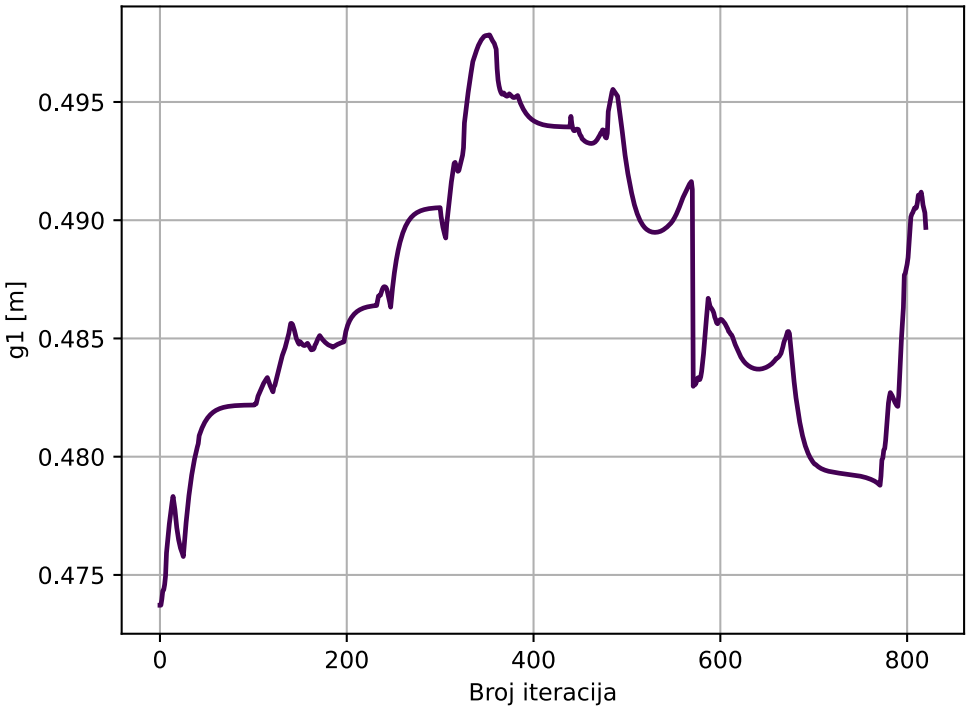
Slika 8.10 Razvoj funkcije ograničenja jednakosti 6



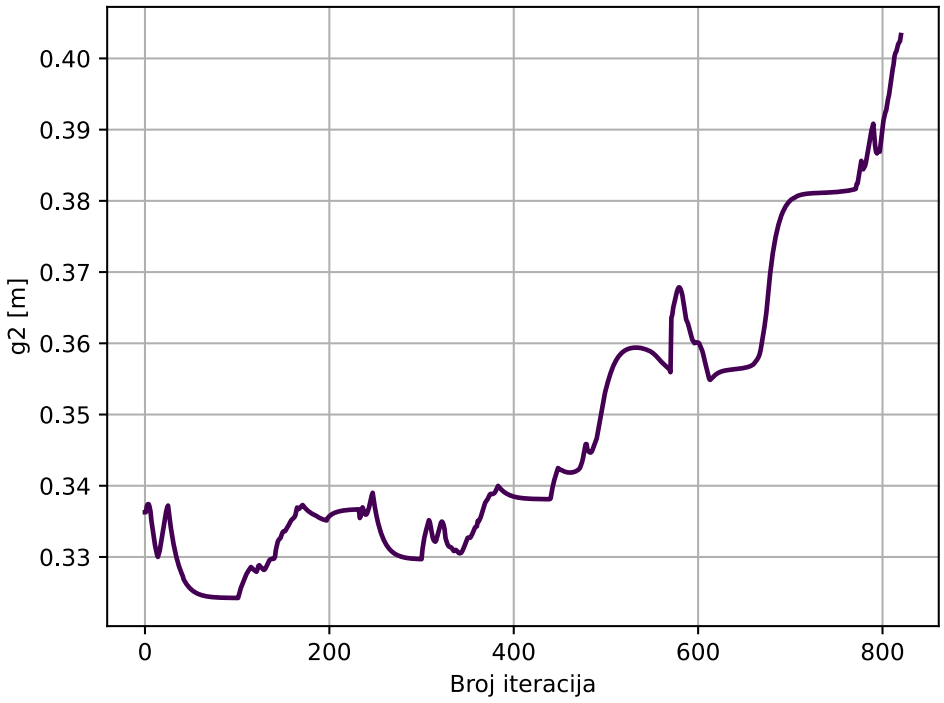
Slika 8.11 Razvoj funkcije ograničenja jednakosti 7



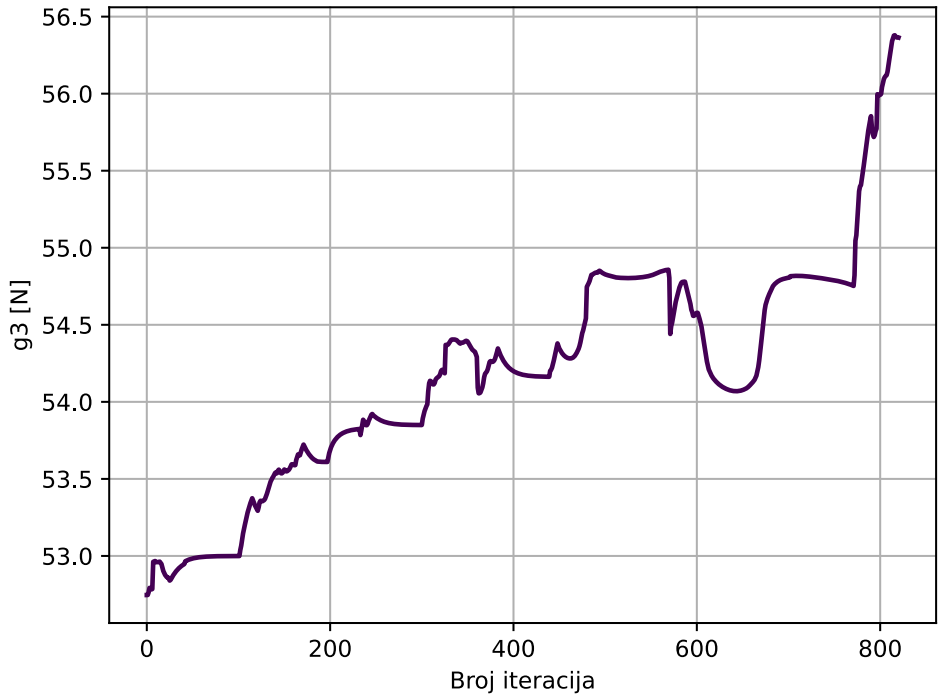
Slika 8.12 Razvoj funkcije ograničenja jednakosti 8



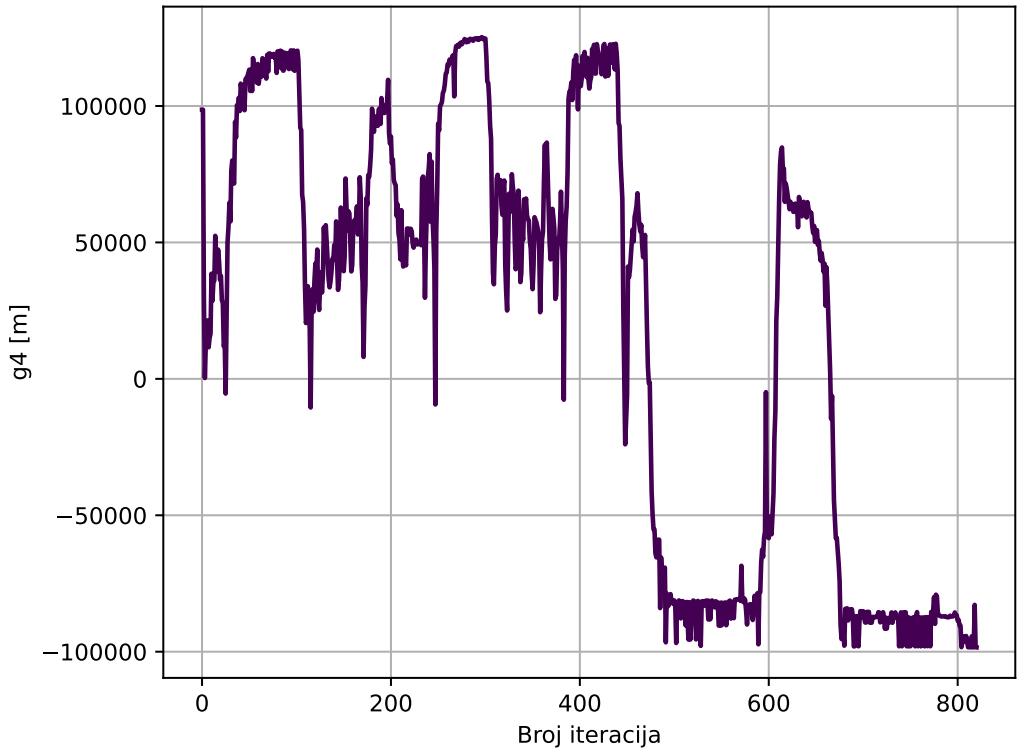
Slika 8.13 Razvoj funkcije ograničenja nejednakosti 1



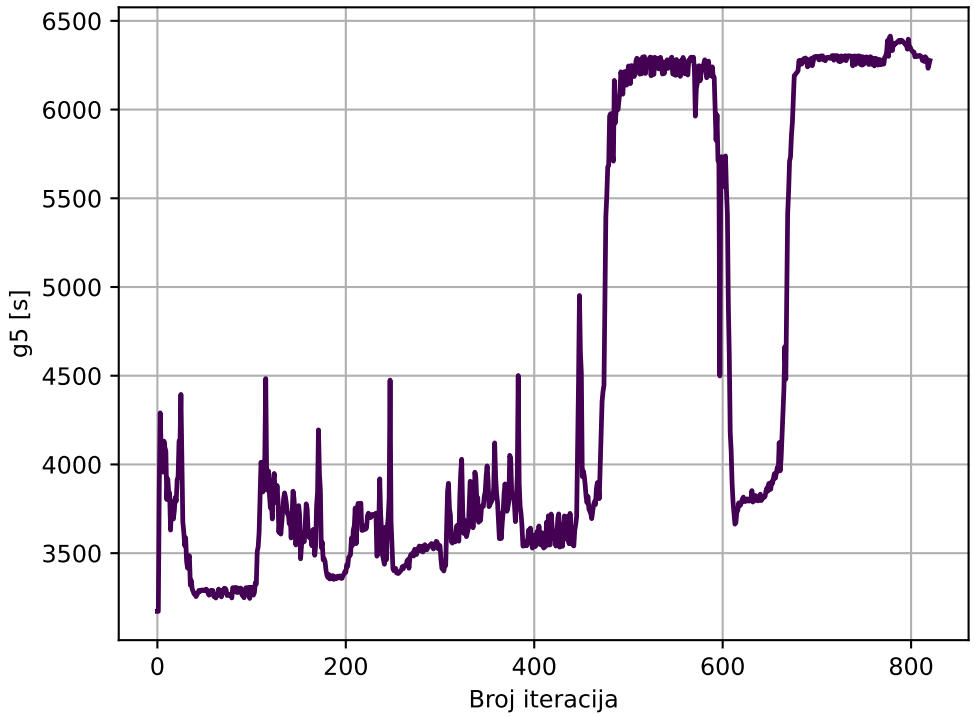
Slika 8.14 Razvoj funkcije ograničenja nejednakosti 2



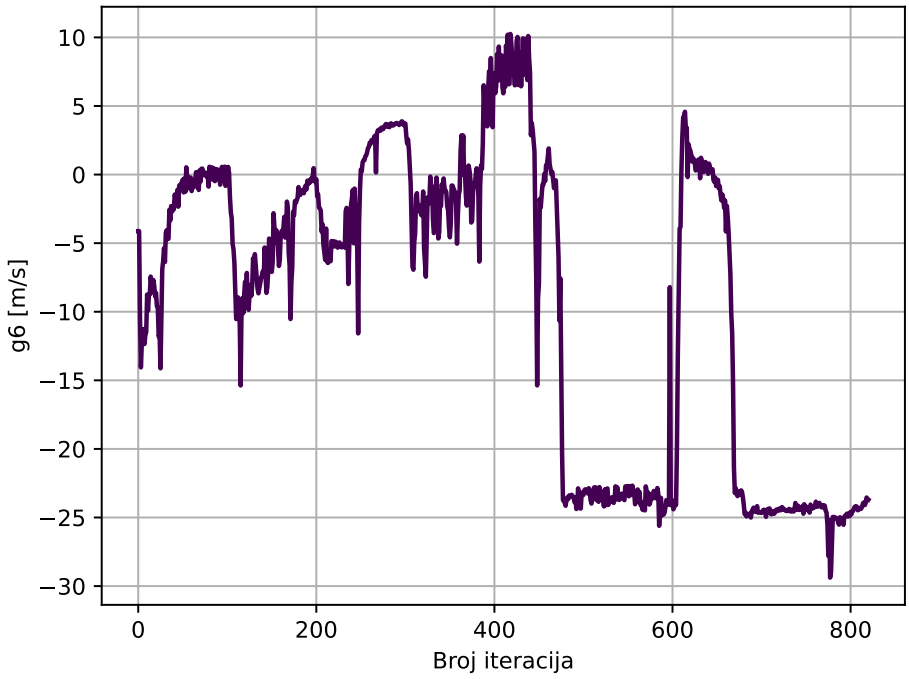
Slika 8.15 Razvoj funkcije ograničenja nejednakosti 3



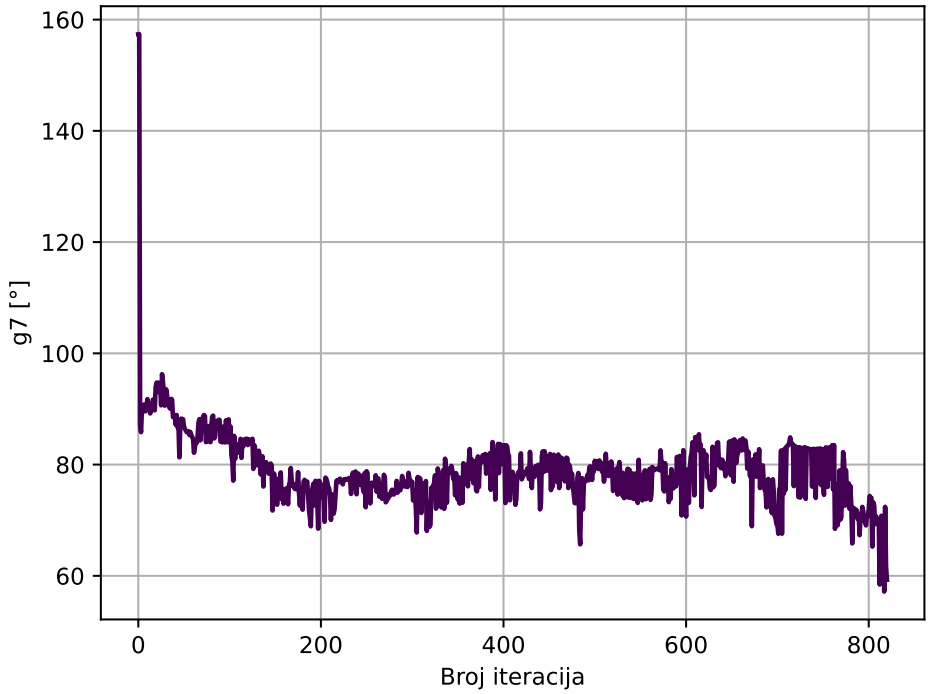
Slika 8.16 Razvoj funkcije ograničenja nejednakosti 4



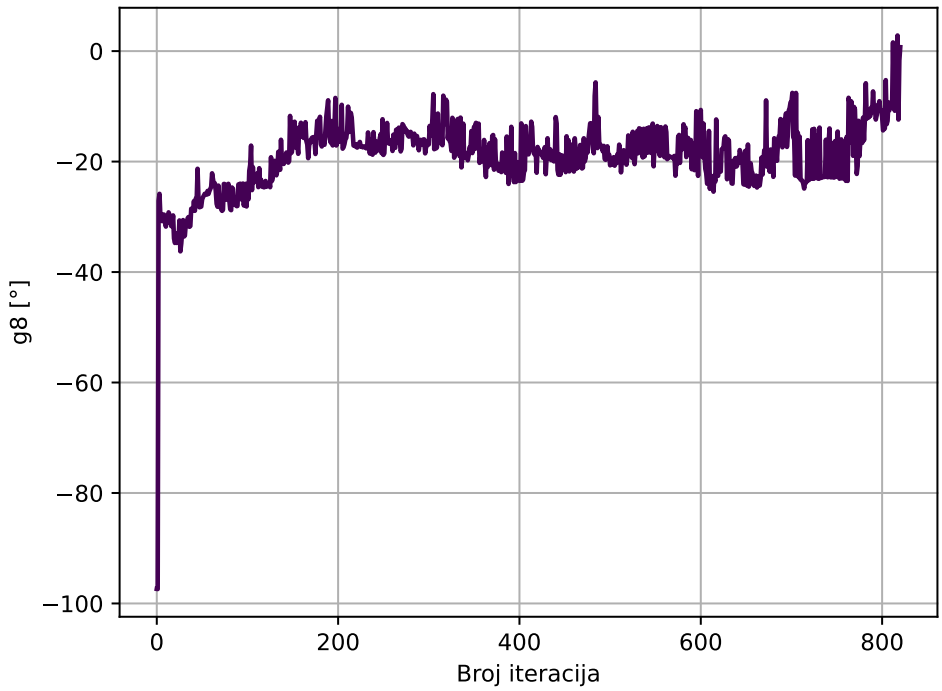
Slika 8.17 Razvoj funkcije ograničenja nejednakosti 5



Slika 8.18 Razvoj funkcije ograničenja nejednakosti 6



Slika 8.19 Razvoj funkcije ograničenja nejednakosti 7



Slika 8.20 Razvoj funkcije ograničenja nejednakosti 8

Iz prethodnih slika, vidi se da optimizacija nije konvergirala. Ono što se može primijetiti uspoređujući neke grafove međusobno jest da kada se jedno ograničenje popravi, drugo se pokvari. To može ukazivati na to da fizikalno nije moguće ostvariti sve zahtjeve simultano. Matematički, to bi značilo da nivo-plohe, koje predstavljaju ograničenja, nemaju presječenih točaka. Isto tako, moguće je da je jednostavno geometrija proširene funkcije cilja izrazito teška za navigirati (rješenja osciliraju), pa gradijent treba puno više iteracija a bi pronašao podobno rješenje. Upravo ta se oscilatorna konvergencija može najbolje vidjeti na grafovima koji prikazuju razvoj ograničenja jednakosti. Iako to može upućivati na složen reljef proširene funkcije cilja, takvo ponašanje može upućivati i na pre veliki odabir koraka (*learning rate*). Ovdje je još interesantno obratiti pažnju na razliku u šumu (oscilacije visoke frekvencije i male amplitude) prije i nakon 360. iteracije. Naime, do 360. iteracije, koristila se čista metoda penalizacije, a nakon 360. iteracije, ubačeni su Lagrangeovi multiplikatori. Vidi se da inicijalno (do 400. iteracije), šum se povećava, no ubrzo se smanjuje na razinu koja je vidljivo niža od prijašnje. Ta razlika u šumu je najizraženija kod ograničenja jednakosti, što i ima smisla, pošto ta ograničenja stvaraju „kanjone“, koji mogu destabilizirati gradijent.

Ovi rezultati potencijalno ukazuju i na lošu početnu točku. Ta nedoumica se može riješiti korištenjem algoritama koji ne rade sa gradijentima u prvih par iteracija, ne bi li se pronašla pogodna početna točka. PSO (*Particle Swarm Optimization*) algoritam je ovdje najzgodniji jer automatski daje najbolju pronađenu točku.

Konačno, poznato je da korišteni algoritam (Adam) nije toliko precizan kod finalne konvergencije kao npr. L-BFGS, stoga bi bilo korisno isprobati L-BFGS algoritam u finalnoj fazi optimizacije kada velike promjene projektnih varijabli iščeznu. Algoritam L-BFGS nije samo pogodan radi preciznosti, već i zbog činjenice da je to algoritam koji spada pod tzv. kvazi-Newtonove algoritme, što znači da tehnički radi sa informacijama drugog reda, no ne računa ih direktno, već ih dobiva na temelju informacija prvog reda (gradijenata).

U daljnjem istraživanju, isprobati će se algoritmi koji direktno rade sa ograničenjima (npr. sekvencijalno kvadratično programiranje), pošto su ti algoritmi neupitno stabilniji od trenutno korištene metode. Nedostatak je da je integracija tih algoritama sa trenutnom arhitekturom programa složenije.

ZAKLJUČAK

U ovom radu, izrađen je Python modul za optimizaciju projektiranja letjelica sa sjedinjenim trupom i krilom. Formuliran je optimizacijski problem koji uključuje projektiranje oblika letjelice. Korišteni su i opisani prethodno razvijeni matematički modeli za definiranje geometrije letjelice, određivanje aerodinamičkih karakteristika, izračun inercijskih značajki te provjeru dinamičke stabilnosti. Ovi modeli, implementirani u Python module, uspoređeni su s dostupnim rezultatima na nekoliko reprezentativnih modela, čime je potvrđena njihova točnost.

Ključna komponenta razvijenog modula je korištenje biblioteke PyTorch za automatsko deriviranje koristeći pristup stvaranja računalnog grafa i propagacije unatrag (*backpropagation*). Ovaj pristup omogućava efikasno računanje gradijenata svih parametara sustava, što je ključno za optimizaciju u složenim inženjerskim problemima s mnogo varijabli i ograničenja. Korištenje optimizacijskog algoritma Adam u kombinaciji s automatskim deriviranjem omogućilo je integraciju svih komponenti u jedinstveni optimizacijski okvir.

Razvijeni modul primijenjen je za optimizaciju oblika besposadne letjelice za dostavu organa za presađivanje. Tijekom optimizacije uočeni su izazovi vezani uz velik broj zahtjevnih i često konfliktnih ograničenja. Kompleksnost problema otežala je dobivanje projektnih rješenja letjelice koja istovremeno zadovoljavaju sve specificirane zahtjeve. U pojedinim dijelovima optimizacijskog procesa uspješno su zadovoljena ograničenja vezana uz dinamičku stabilnost, dok su u drugim dijelovima postignute zahtijevane performanse poput doleta. Međutim, unutar okvira ovog rada nije postignuto rješenje koje istovremeno zadovoljava sve kriterije.

Unatoč ovim izazovima, upotreba PyTorch-a i njegovih mogućnosti automatskog deriviranja, pokazala se vrlo korisnom za efikasno provođenje optimizacije. Ovaj pristup olakšava implementaciju složenih matematičkih modela i upravljanje velikim brojem varijabli i ograničenja, što je ključno u ovoj vrsti optimizacijskog problema.

Mogući daljnji koraci uključuju detaljnu analizu uzroka otežanog zadovoljavanja svih ograničenja. Glavni korak će biti zamjena trenutnog algoritma sa hibridnim algoritmom. U tom algoritmu, prvih nekoliko iteracija će se odraditi sa PSO (*Particle Swarm Optimization*) algoritmom, ne bi li se pronašla optimalna početna točka iz koje kreće algoritam Adam, nakon kojeg će se finalni dio konvergencije odraditi sa L-BFGS algoritmom.

Budući rad na programu će se fokusirati na implementaciju modula za proračun čvrstoće i poboljšanje aerodinamičkog modela pomoću neuronskih mreža PINNs (*Physics Informed Neural Networks*). Time bi se dobio sveobuhvatan alat, koji omogućava projektiranje letjelica koje u potpunosti zadovoljavaju sve specificirane zahtjeve i ograničenja.

LITERATURA

[1] Joaquium R.R.A. Martins and Andrew Ning, Engineering Design optimization, Cambridge University Press, 2021.

[2] Kingma, D. P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." *arXiv preprint arXiv:1412.6980* (2014).

[3] L., Piegl, W. Tiller, The NURBS Book

[4] J., Katz and A., Plotkin, Low-speed aerodynamics, Cambridge University Press, 2001.

[5] Erickson, Larry L. *Panel Methods: An Introduction*. NASA Technical Paper 2995, NASA Ames Research Center, 1990.

[6] Birk, L. (2021). A comprehensive and practical guide to the Hess and Smith constant source and dipole panel. *Ship Technology Research*, *69*(1), 50–62.

[7] M, Drela, Flight Vehicle Aerodynamics, MIT Press, 2014.

[8] Virag, Z., Šavar, M., Džijan, I., Mehanika fluida II, Fakultet strojarstva i brodogradnje, Zagreb 2018.

[9] Janković, S., Virag, Z., Vrdoljak, M., Aerodinamika I, Fakultet strojarstva i brodogradnje, Zagreb 2016.

[10] S. Janković, Mehanika leta zrakoplova , Fakultet strojarstva i brodogradnje, Zagreb, 2005.

[11] K., Karamcheti, Principles of Ideal-Fluid Aerodynamics, 1966.

[12] Kesić, P., Osnove aerodinamike, Fakultet strojarstva i brodogradnje, Zagreb 2003.

[13] Williams J. E. Vukelich S. R. Air Force Flight Dynamics Laboratory & McDonnell Douglas Astronautics Company-St Louis. (1979). The usaf stability and control digital datcom. Air Force Flight Dynamics Laboratory Air Force Wright Aeronautical Laboratories Air Force Systems Command Wright-Patterson Air Force Base ; NTIS distributor

[14] Janković, S., Vrdoljak, M., Performanse zrakoplova, Fakultet strojarstva i brodogradnje, Zagreb 2016.

[15] Michael D. Greenberg, Foundations of applied mathematics, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1978.

[16] H. Goldstein, C. Poole, J. Safko, Classical mechanics

[17] https://pytorch.org - zadnje posjećeno 4.12.2024.

[18] https://numpy.org - zadnje posjećeno 20.11.2024.

[19] https://scipy.org - zadnje posjećeno 2.12.2024.

[20] https://matplotlib.org – zadnje posjećeno 1.11.2024.

[21] https://docs.pyvista.org – zadnje posjećeno 3.12.2024

[22] https://numba.pydata.org – zadnje posjećeno 10.3.2024.

[23] http://airfoiltools.com – zadnje posjećeno 1.11.2024.

[24] https://web.mit.edu/drela/Public/web/avl - zadnje posjećeno 30.11.2024.

[25] http://www.mh-aerotools.de/airfoils/index.htm - zadnje posjećeno 10.11.2024.