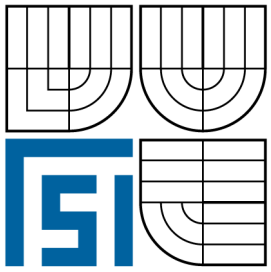
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
LETECKÝ ÚSTAV

FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF AVIATION

POUŽITÍ NELINEÁRNÍ TEORIE NOSNÉ ČÁRY PŘI AERODYNAMICKÉM NÁVRHU KLUZÁKU

NON-LINEAR LIFTING LINE THEORY APPLICATION TO GLIDER AERODYNAMIC DESIGN

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL SCHOŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROBERT POPELA, Ph.D.

BRNO 2011

Abstrakt

Tato diplomová práce naznačuje jakým způsobem je možné použít modifikovanou teorii nosné čáry ři předběžném aerodynamickém návrhu kluzáku. Je ukázáno, že lze dosáhnout poměrně přesných výsledků, při menší výpočetní náročnosti ve srovnání s CFD metodami.
.

Abstract

This master thesis shows, how can be the modified lifting line theory used for preliminary glider design an for wing loads determination. It is shown, that relatively accurate results can be obtained at less computational cost in comparison with CFD methods

Klíčová slova

Teorie nosné čáry, kluzák, zatížení křídla, rychlostní polára

Keywords

Lifting line theory, glider, wing loads, speed polar

Citace

Pavel Schoř: Použití nelineární teorie nosné čáry při aerodynamickém návrhu kluzáku, diplomová práce, Brno, FSI VUT v Brně, 2011

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Roberta Popely Ph.D. s použitím uvedených zdrojů

.....
Pavel Schoř
24. května 2011

Poděkování

Na tomto místě bych rád poděkoval především panu profesorovi Karolu Fiľakovskému, bez jehož cenných rad by tato práce nevznikla. Dále bych rád poděkoval pracovníkům LU-FSI, Robertu Popelovi, Robertu Šošovičkovi a Františku Vaňkovi za cenné připomínky během tvorby této práce.

Obsah

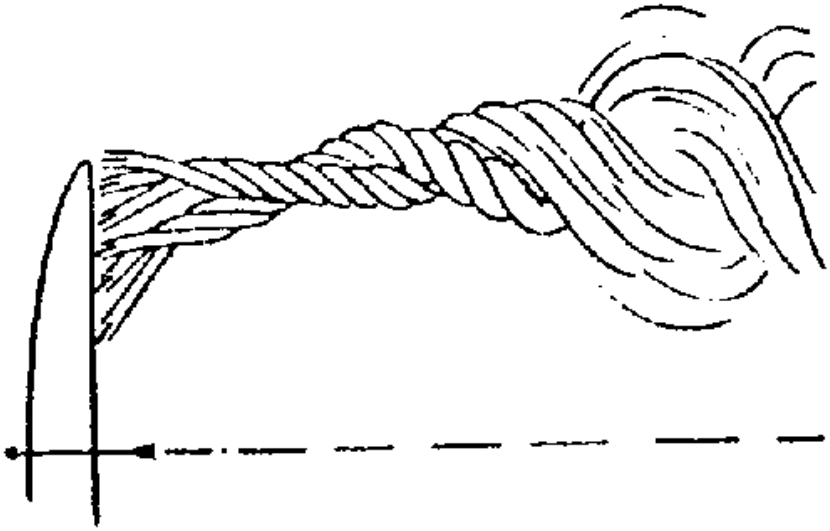
Seznam obrázků

Kapitola 1

Úvod

1.1 Historický vývoj teorie nosné čáry [1]

Vznik a praktické použití lineární teorie nosné čáry se datuje okolo roku 1911. V té době uběhlo již osm let od prvního letu motorového letadla vyrobeného bratři Wrightovými. Ti sice svou konstrukci dimenzovali na šestinásobek celkové váhy avšak přesný průběh zatížení od vzdušných sil jim nebyl znám. V roce 1907 uveřejňuje F.W.Lanchester ve své knize "Aerodynamics"teorii o vírech na koncích křídla.



Obrázek 1.1: Představa k vírové teorii křídla z knihy F.W.Lanchestra [1]

F.W.Lanchester navštívil v roce 1908 L.Prandtla, který poté dále rozvíjel vírovou teorii křídla. První publikaci uveřejnil Prandtl v roce 1911, vývoj této teorie pokračoval do roku 1918. Během první světové války byla tato teorie předmětem válečného tajemství.

Po skončení války vydal Prandtl souhrn svých poznatků, ten byl uveřejněn v roce 1923 jako NACA Report 116.

1.2 Vztah teorie nosné čáry k praxi

Do doby uveřejnění teorie nosné čáry záleželo stanovení zatížení od vzdušných sil více méně na odhadu konstruktéra. Tomu odpovídaly i tvary tehdejších konstrukcí.

S příchodem teorie nosné čáry bylo možné stanovit poměrně přesně hledané zatížení od vzdušných sil. Půdorysné tvary křídel odpovídají předpokladům pro řešení pomocí teorie nosné čáry.

Základní předpoklady pro řešení pomocí teorie nosné čáry

- štíhlost křídla $\lambda > 6$
- spojnice čtvrtinových bodů je přímka kolmá na rovinu symetrie
- proudění je neviskózní, nestlačitelné

S příchodem rychlých letounů za druhé světové války je nutné řešit jevy související se stlačitelností, které teorie nosné čáry nedokáže řešit. Přesto je tato teorie nadále používána a poskytuje poměrně dobré výsledky do $M = 0.6$. [9]

Obrázek 1.2: Letoun P-51 mustang, křídlo lichoběžníkového půdorysu odpovídá předpokladům teorie nosné čáry.[12]

S příchodem proudových letounů se začínají objevovat šípová a později i delta křídla, na která nelze použít teorii nosné čáry. Vznikají proto další analytické metody pro řešení

konkrétních problémů. Později dochází k rozvoji počítačové techniky a původní analytické metody jsou nahrazovány panelovými a CFD metodami. Tyto metody jsou schopny podat informaci o zatížení nejen po rozpětí, nýbrž i po hloubce křídla.Navíc je lze použít i na úplně obecnou geometrii.

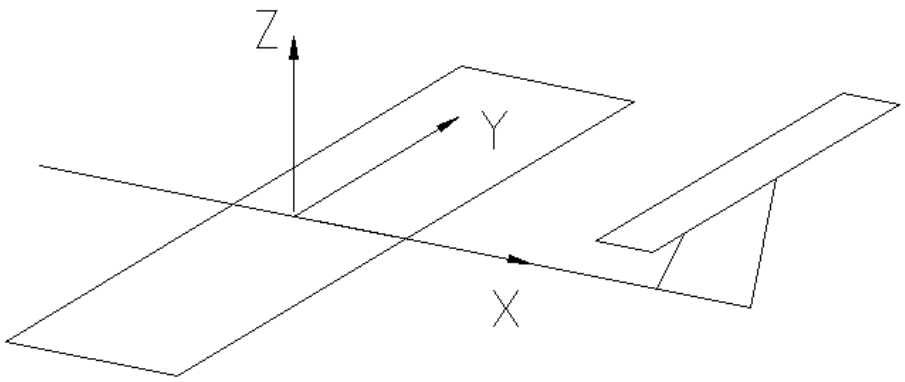
Okolo roku 1950 začíná docházet k intenzivnímu rozvoji vývoje kluzáku. Křídlo kluzáku většinou svým půdorysem a štíhlostí odpovídají předpokladům pro řešení pomocí teorie nosné čáry. I přes značný pokrok v této oblasti během posledních padesáti let zůstávají u kluzáků stále splněny předpoklady pro řešení touto teorii, protože hlavní úsilí rozvoje tvoří vývoj profilů a běžně používané půdorysy křídel kluzáků poskytují dobrou aerodynamickou účinnost.

1.3 Teorie nosné čáry v současnosti

I přes značný rozvoj numerických výpočetních metod neupadla teorie nosné čáry v zapomnění. Dnes představuje základní a léty prověřený postup pro stanovení zatížení u malých sportovních letadel a kluzáků. Hlavní výhodou této teorie je její jednoduchost, nenáročnost a dobrá přesnost. V současnosti se objevují modifikace, které umožňují řešit i obecnější geometrie, nebo uvažovat nelineární vztlakové charakteristiky. [4]

1.4 Souřadný systém

Není-li uvedeno jinak, je použit pravotočivý souřadný systém z následujícího obrázku. Jeho počátek je umístěn do čtvrtinového bodu profilu v rovině symetrie.



Obrázek 1.3: Použitý souřadný systém

Kapitola 2

Úplná formulace zadání

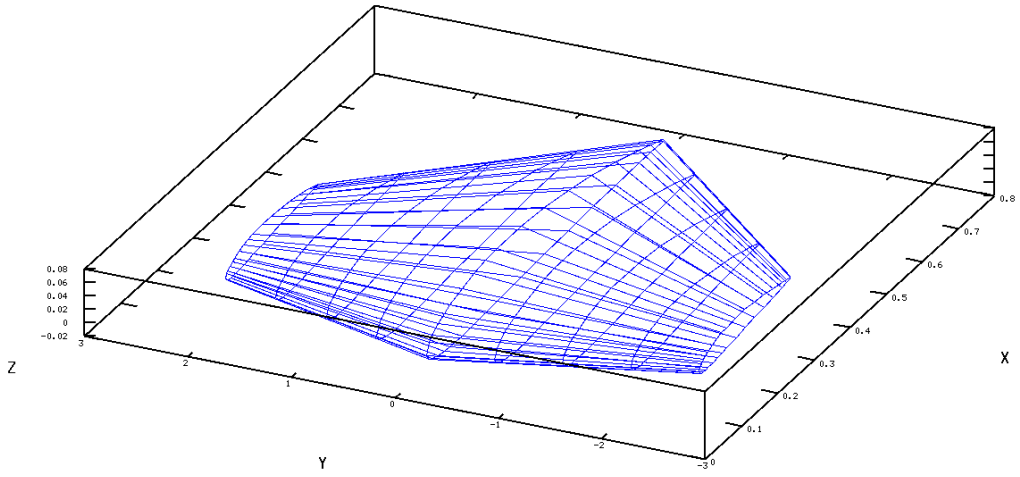
2.1 Nekroucené křídlo

Teorie nosné čáry umožňuje řešit rozložení vztlaku na křídle, jehož spojnice čtvrtinových bodů je přímá čára. Hloubka profilů v jednotlivých řezech se může měnit, tím je možné dosáhnout různých půdorysů křídla. Pokud je na tomto křídle použit jeden profil, který je stejný ve všech řezech, hovoříme o křídle nekrouceném. Taková křídla byla používána především v počátcích letectví, avšak z různých aerodynamických důvodů se později začala používat křídla kroucená.

Křídlo s konstantní hloubkou řezů představuje ideální vstup pro řešení rozložení vztlaku. Protože hloubka je ve všech řezech stejná, bude pro všechny řezy stejné i Reynoldsovo číslo. Potom se vztlaková čára použitého profilu nemění a je stejná pro všechny řezy. Takové křídlo umožňuje jednoduché řešení rozložení vztlaku a jednoduchou konstrukci, ale současně vyniká nejvyšší hodnotou indukovaného odporu.

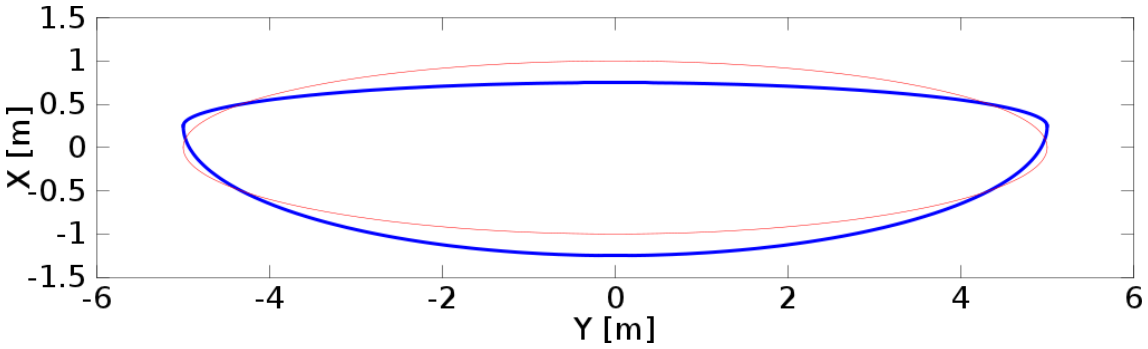
Křídlo s proměnnou hloubkou řezů Snaha o snížení indukovaného odporu přímého křídla s konstantní hloubkou vedla ke konstrukcím křídel s proměnnou hloubkou po rozpětí. Z takových křídel poskytuje nejmenší indukovaný odpor křídlo elipsovitého půdorysu. Prakticky nejrozšířenější je ale křídlo lichoběžníkového půdorysu, protože je výrobně jednodušší, než křídlo elipsovitého půdorysu. O aerodynamických výhodách a nevýhodách jednotlivých řešení pojednávají následující řádky.

Křídlo lichoběžníkového půdorysu Vznikne, pokud je hloubka koncového řezu menší, než hloubka kořenového řezu. V takovém případě se místní hloubky mění po rozpětí lineárně a s tím i Reynoldsovo číslo. Pokud jsou známy vztlakové čáry v kořenovém a koncovém řezu pro odpovídající Reynoldsova čísla, je možné předpokládat, že vztlakové čáry v mezilehlých řezech vzniknou lineární interpolací těchto vztlakových čar.



Obrázek 2.1: Pohled na křídlo lichoběžníkového půdorysu

Křídlo elipsovitého půdorysu znamená, že místní hloubky odpovídají hodnotám, které se určí jako řezy elipsou. Hlavní osa takové elipsy představuje rozpětí, vedlejší pak hloubku v kořenovém řezu. Důležité je, že půdorys sám o sobě nemá tvar elipsy, protože místní hloubky jsou vázány ke čtvrtinovému bodu. Reynoldsovo číslo se po rozpětí mění nelineárně, proto interpolace vztlakových čar v jednotlivých řezech musí zohledňovat tento fakt. Křídlo s takovým půdorysem poskytuje téměř konstantní rozložení vztlaku po rozpětí. To je nevýhodné při překročení maximálního součinitele vztlaku křídla, protože k poklesu vztlaku dojde téměř naráz.



Obrázek 2.2: Elipsovitý půdorys křídla

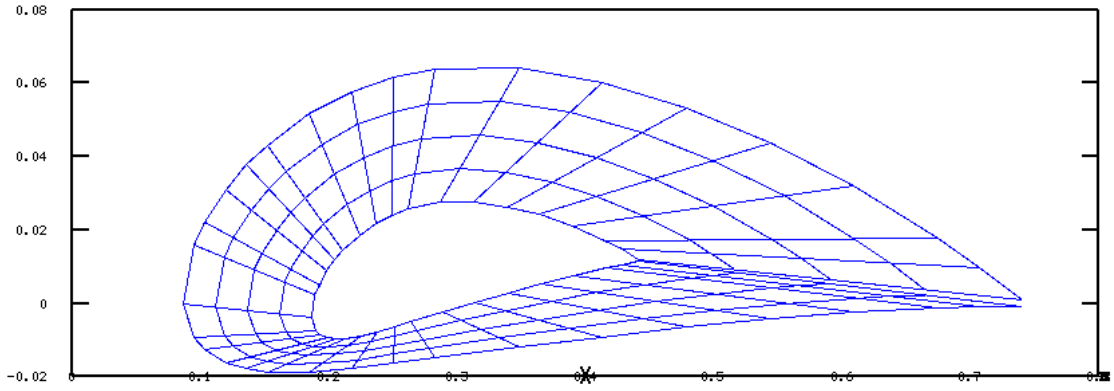
2.2 Aerodynamicky kroucené křídlo

se nazývá takové křídlo, na kterém jsou v různých řezech použity různé profily. To je obvyklé pro lichoběžníková křídla, kdy se v koncovém řezu volí takový profil, který dosáhne svého maximálního profilového součinitele vztlaku při vyšším úhlu náběhu, než profil použitý v kořenovém řezu. Toto opatření dokáže částečně eliminovat nepříjemnou vlastnost nekrou-

cených lichoběžníkových křídel, kdy dochází k dosažení maximálního součinitele vztlaku v řezech, ve kterých bývají obvykle umístěna křidélka. Pokud se tedy dostane nekroucené lichoběžníkové do pádu, obvykle nejsou křidélka účinná. V další kapitole bude popsán způsob získání vztlakové v řezech mezi kořenovým a koncovým řezem.

2.3 Geometricky kroucené křídlo

se používá obvykle v případech, kdy aerodynamické zkroucení nedokáže zajistit vhodné pádové vlastnosti, ale může být použito i samostatně, bez aerodynamického kroucení. Podstata geometrického kroucení spočívá v natočení profilu v daném řezu o určitý úhel. Tento se volí tak, aby koncový profil pracoval na menším úhlu náběhu, než kořenový profil.

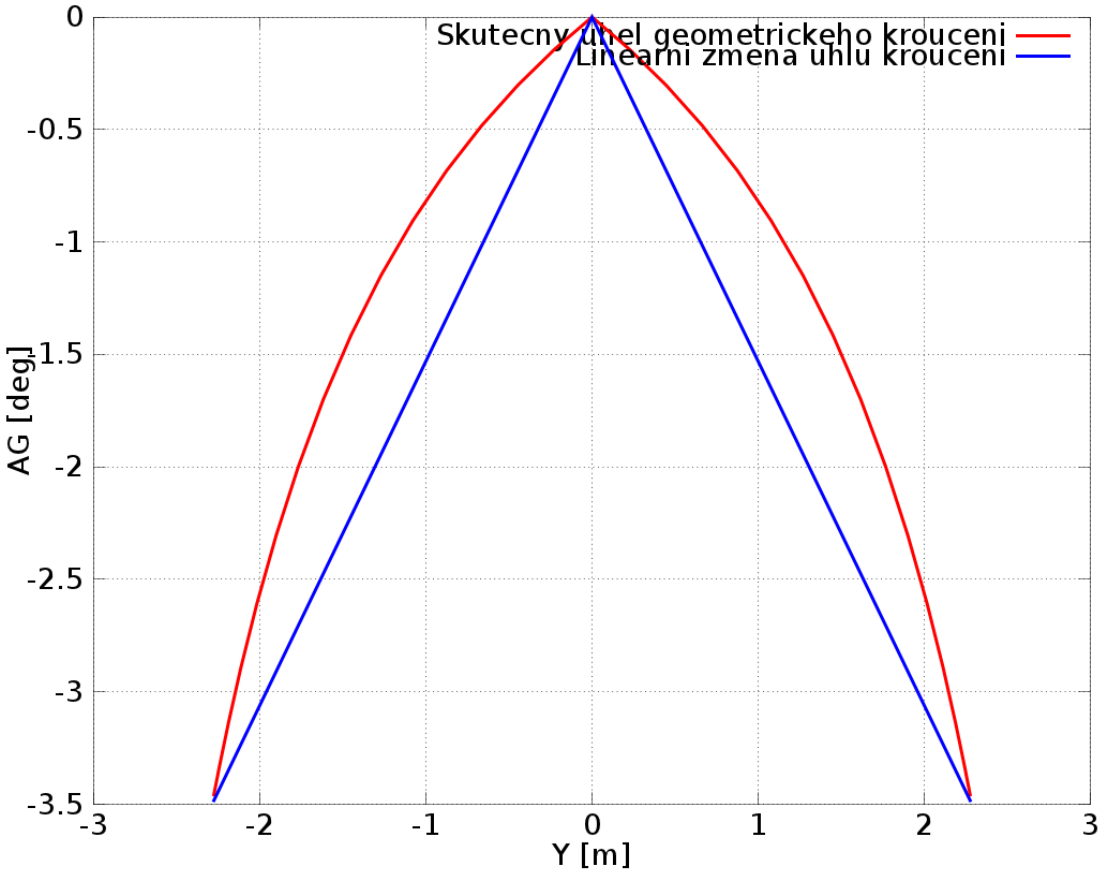


Obrázek 2.3: Boční pohled na křídlo s geometrickým a aerodynamickým kroucením

Je velmi důležité uvážit fakt, že na lichoběžníkovém křídle se úhel geometrického kroucení jednotlivých řezu mění po rozpětí nelineárně. K lineární změně úhlu kroucení po rozpětí dochází pouze u křídel s konstantní hloubkou řezu. Tato nelineární změna úhlu geometrického kroucení může významně ovlivňovat výsledné rozložení vztlaku. Uvažujme křídlo , jehož body jsou zadány v absolutních souřadnicích $[x, y, z]$.

- Spojnice čtvrtinových bodu má souřadnice $[0.25, y, 0]$. Každý řez křídla je škálován a natočen okolo této spojnice.
- Na normalizovaných a neotočených kořenových a koncových profilech se určí pořadové číslo bodu o souřadnicích $[0, 0]$. Čísla těchto bodů budou odpovídat bodům na náběžných hranách u kořenových a koncových řezu v absolutních souřadnicích.
- Určí se průsečíky —spojnice bodů na náběžných hranách— s požadovanými rovinami Glauertových řezů. Tím vzniknou nové body $GK_i = [X, Y, Z]_{GK}$.
- Úhel geometrického kroucení v řezu se potom určí jako $\alpha_G(y) = arctg(\frac{z_{GK}(y)}{0.25 - x_{GK}(y)})$

Toto je nejsnadnější postup stanovení správných úhlu kroucení v Glauertových řezech s využitím výpočetní techniky. Následující obrázek ukazuje, významnost této nelinearity na příkladu dále zkoumaného křídla EXAMPLE WING z [11].

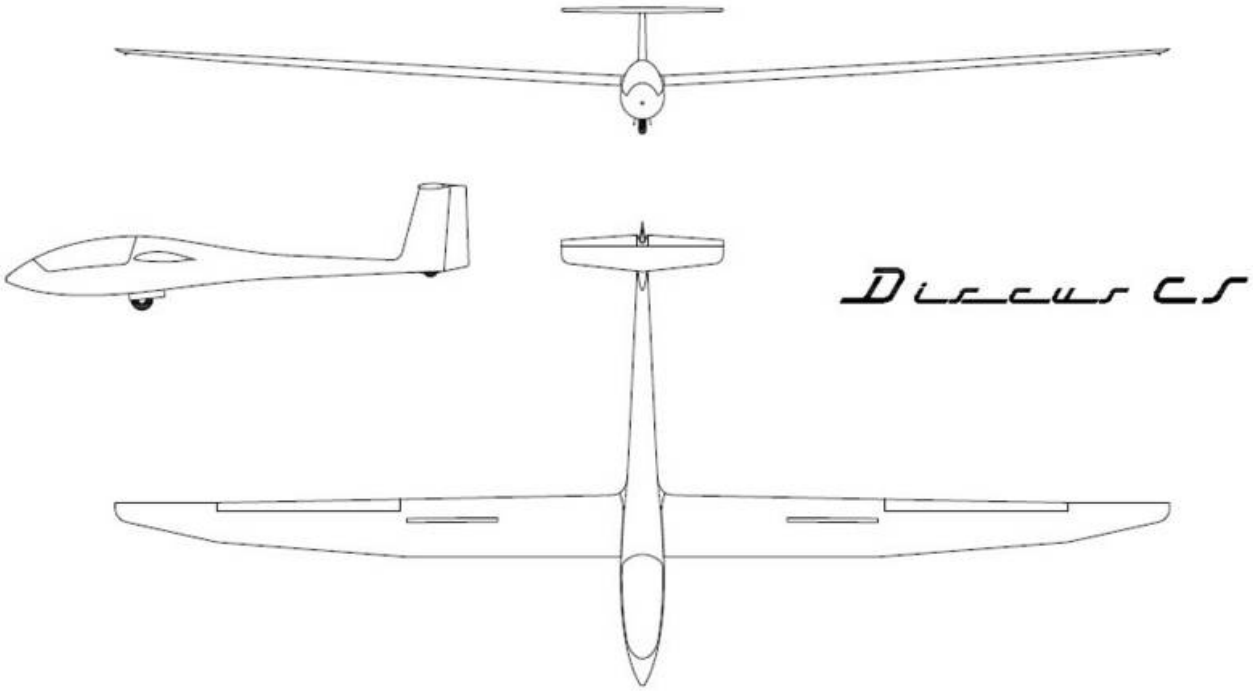


Obrázek 2.4: Porovnání lineární a skutečné změny úhlu geometrického kroucení

Výše uvedený postup zjišťování úhlů geometrického kroucení je vztažený ke kroucení okolo spojnice čtvrtinových bodů profilů. V praxi používaná křídla mohou být z různých konstrukčních důvodů kroucena okolo jiných bodů. Potom je nutné postup náležitě poupravit, aby vypočtené kroucení odpovídalo skutečnosti.

2.4 Křídla současných kluzáků

Při pohledu na půdorys křídla kluzáku Discus je zřejmé, že jeho konstruktérům pro dosažení požadovaných výkonů a vlastností již nedostačovaly výše uvedené tři základní půdorysy křídla a byli nuceni použít půdorys složitější.



Obrázek 2.5: Moderní kluzák DISCUS CS [13]

V tomto konkrétním případě se půdorys křídla skládá ze tří lichoběžníkových segmentů. V praxi se ale vyskytují ještě komplikovanější půdorysy, například různé kombinace elipsovitých a lichoběžníkových segmentů, popřípadě zcela obecné půdorysy.

2.5 Testovací příklad

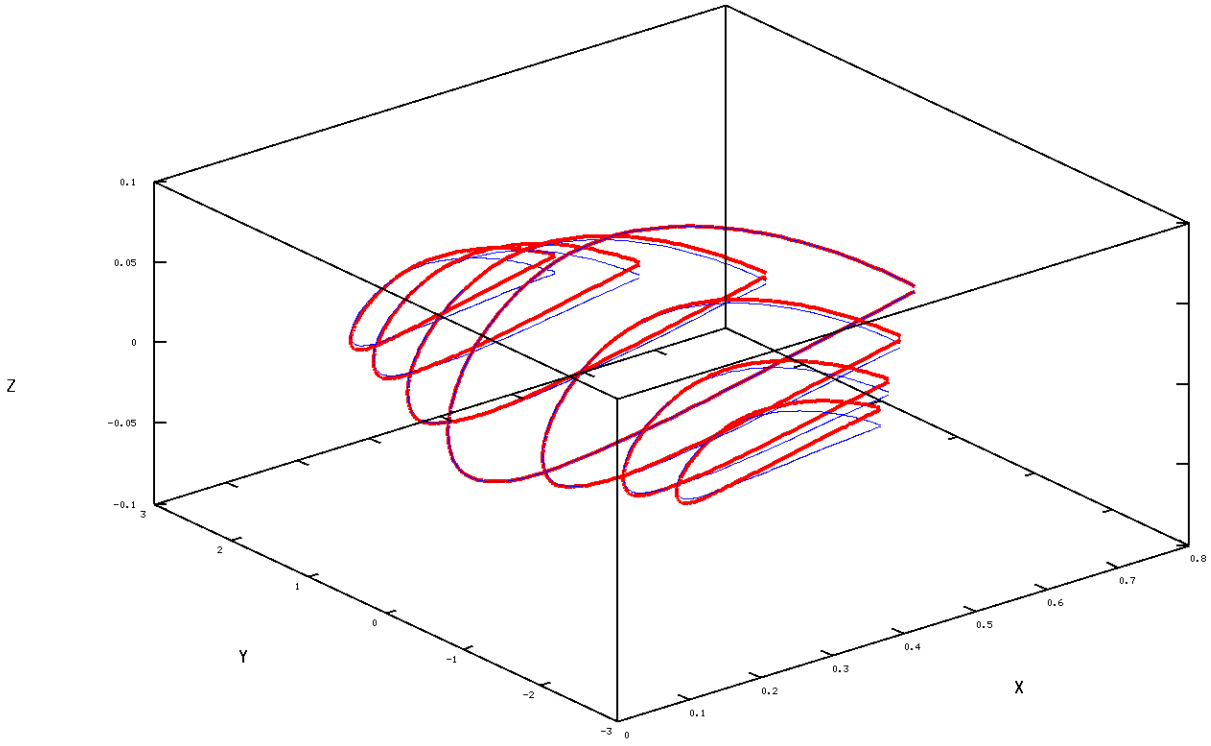
Jako testovací příklad bylo zvoleno křídlo z [11] NACA T.N.1269. Jde o křídlo aerodynamicky i geometricky kroucené, jeho parametry jsou uvedeny v následující tabulce.

Název	EXAMPLE WING
Plocha křídla	$S = 2.0801\ m^2$
Rozpětí	$b = 4.5752\ m$
Hloubka kořenového řezu	$c_R = 0.6536\ m$
Hloubka koncového řezu	$c_T = 0.2557\ m$
Kroucení koncového řezu	$\alpha_{G_T} = -3.5\ ^\circ$
Profil kořenového řezu	NACA 4420
Profil koncového řezu	NACA 4412
Re kořenového řezu	$4.7\ \cdot 10^6$

Pro výpočet rozložení vztlaku na tomto křídle je nutné znát vztlakové čáry ve všech Glauertových řezech křídla. Ty lze určit jedním z následujících postupů:

- určit vztlakové čáry ve všech řezech pro zkroucenou geometrii
- určit vztlakové čáry ve všech řezech pro nezkroucenou geometrii a při výpočtu vztlaku odečítat úhly zkroucení od od úhlů náběhu

- určit vztlakové čáry v krajních řezech segmentů, mezilehlé vztlakové čáry lineárně interpolovat,při výpočtu vztlaku odečítat úhly zkroucení od od úhlů náběhu



Obrázek 2.6: Porovnání zkoumaných geometrií na křídle EXAMPLE WING

2.6 Testovací příklad - zkroucená geometrie

CAD systém pro kreslení trojrozměrných objektů je nutný pro zjištění geometrie v řezech křídla. Takový systém musí umožňovat exportovat geometrii do zvoleného formátu. Prakticky je možné buď psát makra v komerčně dostupných CAD systémech, nebo lze vytvořit soubor funkcí pro geometrické operace v nějakém programovacím jazyku. Jako vhodný jazyk pro tento úkol byl zvolen GNU/octave, zejména pro rozsáhlou knihovnu běžně používaných matematických funkcí. Dopsáním malého počtu funkcí vznikl systém pro generaci trojrozměrné geometrie ve všech řezech křídla. Takto vzniklá geometrie je následně aerodynamicky analyzována pomocí vhodné metody.

Program X-FOIL je svobodný software, 2-D panelová metoda, která je prakticky používán pro analýzu profilů. Jeho velkou výhodou je možnost skriptování, tj. výpočet probíhá automaticky podle předem daných pravidel, bez zásahu uživatele. Takto je možné rychle realizovat výpočty značného množství vztlakových čar.

Postup tedy sestává z následujících kroků:

- zjištění geometrie ve všech požadovaných řezech a export této geometrie

- určení Reynoldsových čísel v požadovaných řezech
- výpočet aerodynamických charakteristik exportované geometrie v programu X-FOIL
- zpracování aerodynamických charakteristik a sestavení vztlakových, odporových a momentových "ploch"
- hledání rozložení vztlaku pomocí metody nelineární nosné čáry

2.7 Testovací příklad - nezkroucená geometrie

Postup je téměř shodný jako v předchozím případě, ale exportovaná geometrie není zkroucená. Navíc je nutné v každém řezu určit úhel geometrického kroucení. Shrnuto:

- zjištění geometrie ve všech požadovaných řezech a export této geometrie
- určení Reynoldsových čísel v požadovaných řezech
- určení úhlů geometrického zkroucení v požadovaných řezech
- výpočet aerodynamických charakteristik exportované geometrie v programu X-FOIL
- zpracování aerodynamických charakteristik a sestavení vztlakových, odporových a momentových "ploch"
- hledání rozložení vztlaku pomocí metody nelineární nosné čáry s uvažováním úhlů geometrického zkroucení

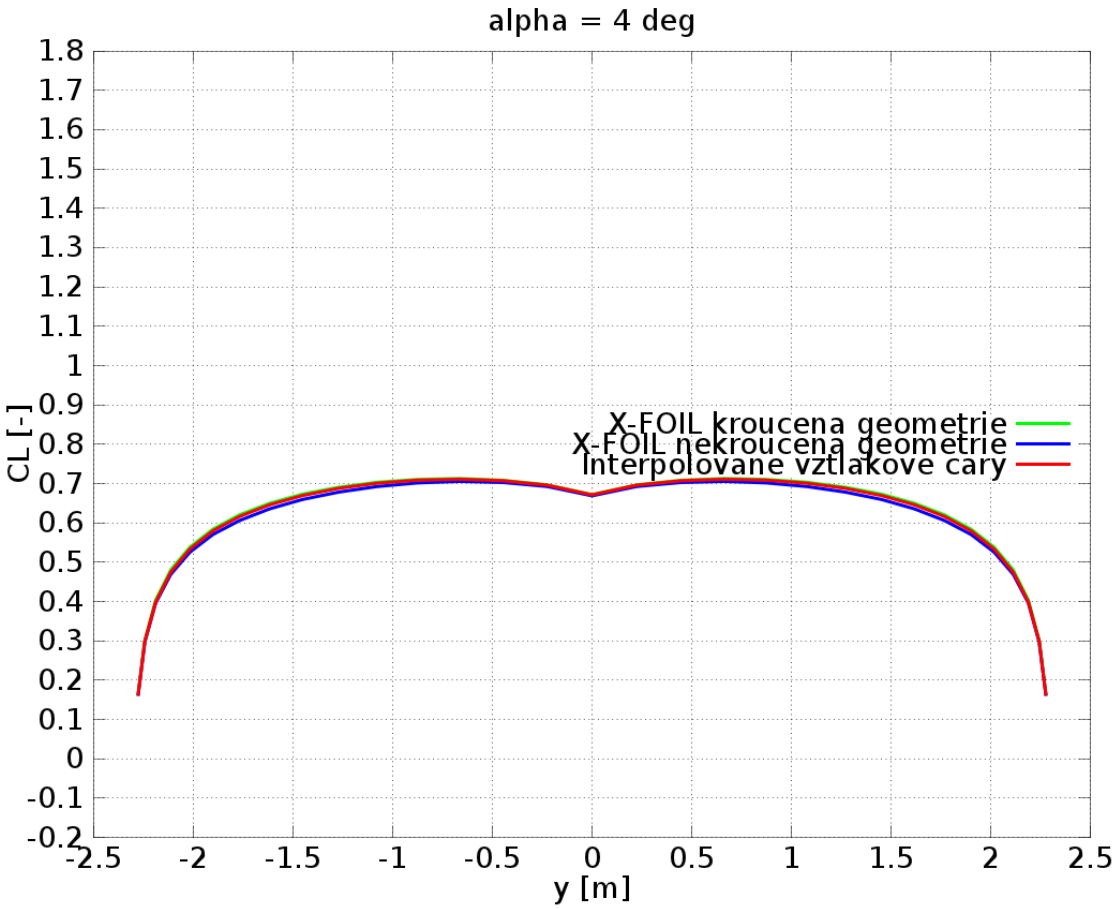
2.8 Testovací příklad - nezkroucená geometrie, lineárně interpolované vztlakové čáry

Oproti předchozím případům se neurčují vztlakové čáry ve všech požadovaných řezech, ale pouze v krajních řezech segmentů. Vztlakové čáry mezi krajními řezy segmentu se určí pomocí lineární interpolace krajních vztlakových čar. Postup je tedy následující:

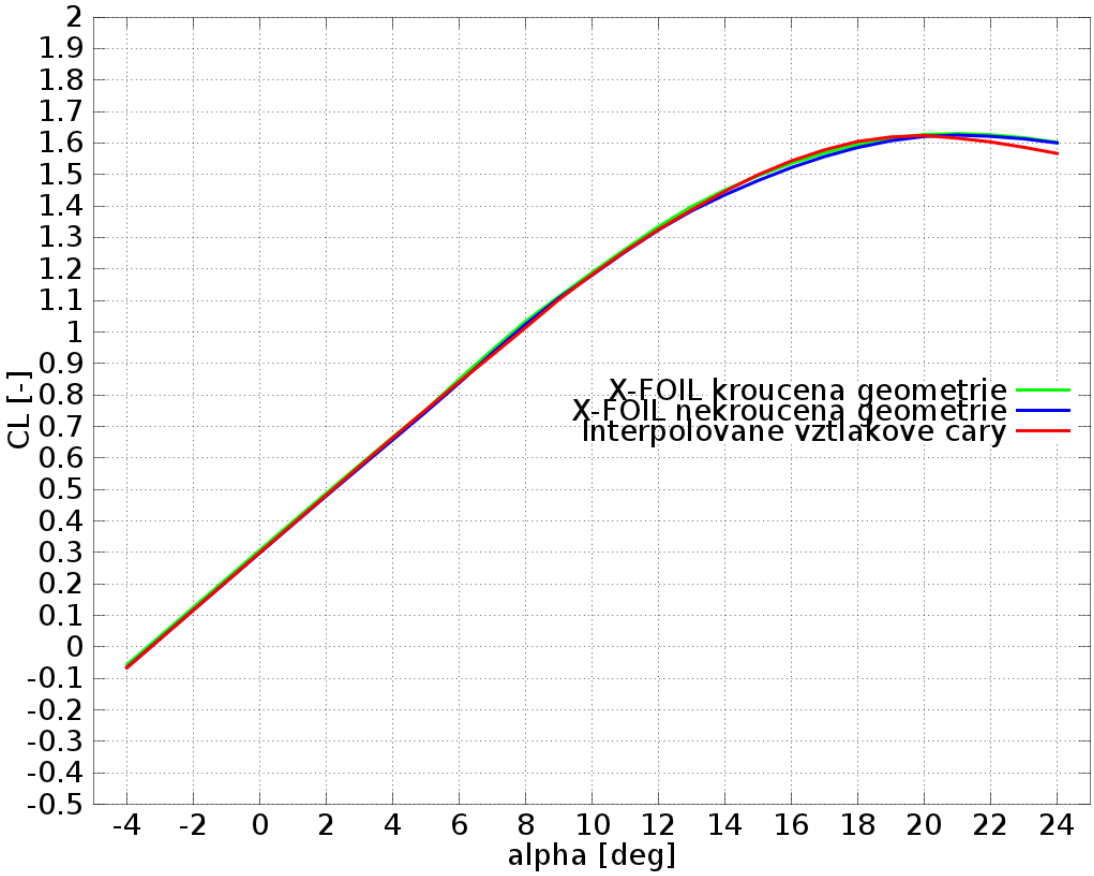
- zjištění geometrie v kořenovém a koncovém řezu a export této geometrie
- určení Reynoldsových čísel v kořenovém a koncovém řezu
- určení úhlů geometrického zkroucení v kořenovém a koncovém řezu
- výpočet aerodynamických charakteristik exportované geometrie v programu X-FOIL
- lineární interpolace vztlakových čar ve všech požadovaných řezech
- zpracování aerodynamických charakteristik a sestavení vztlakových, odporových a momentových "ploch"
- hledání rozložení vztlaku pomocí metody nelineární nosné čáry s uvažováním úhlů geometrického zkroucení

2.9 Testovací příklad - porovnání výsledku

Všechny tří výše popsané způsoby přípravy vstupních dat byly uskutečněny v GNU/octave. Připravená data byla zpracována programem pro výpočet rozložení vztlaku s nelineární korekcí. Výsledky shrnují následující obrázky, z nich je zřejmé, že všechny metody přípravy vstupních dat poskytují přibližně stejná vstupní data.



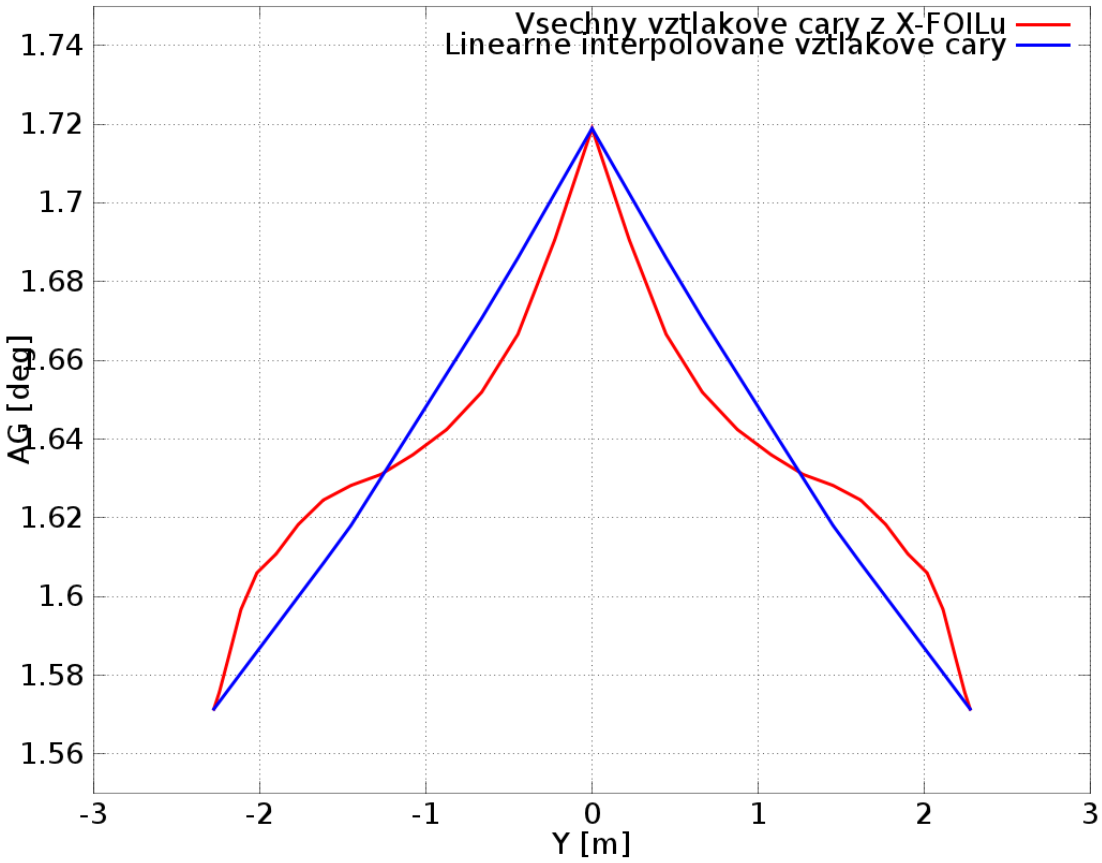
Obrázek 2.7: Ukázka výsledného rozložení vztlaku



Obrázek 2.8: Porovnání jednotlivých vztlakových čar křídla EXAMPLE WING

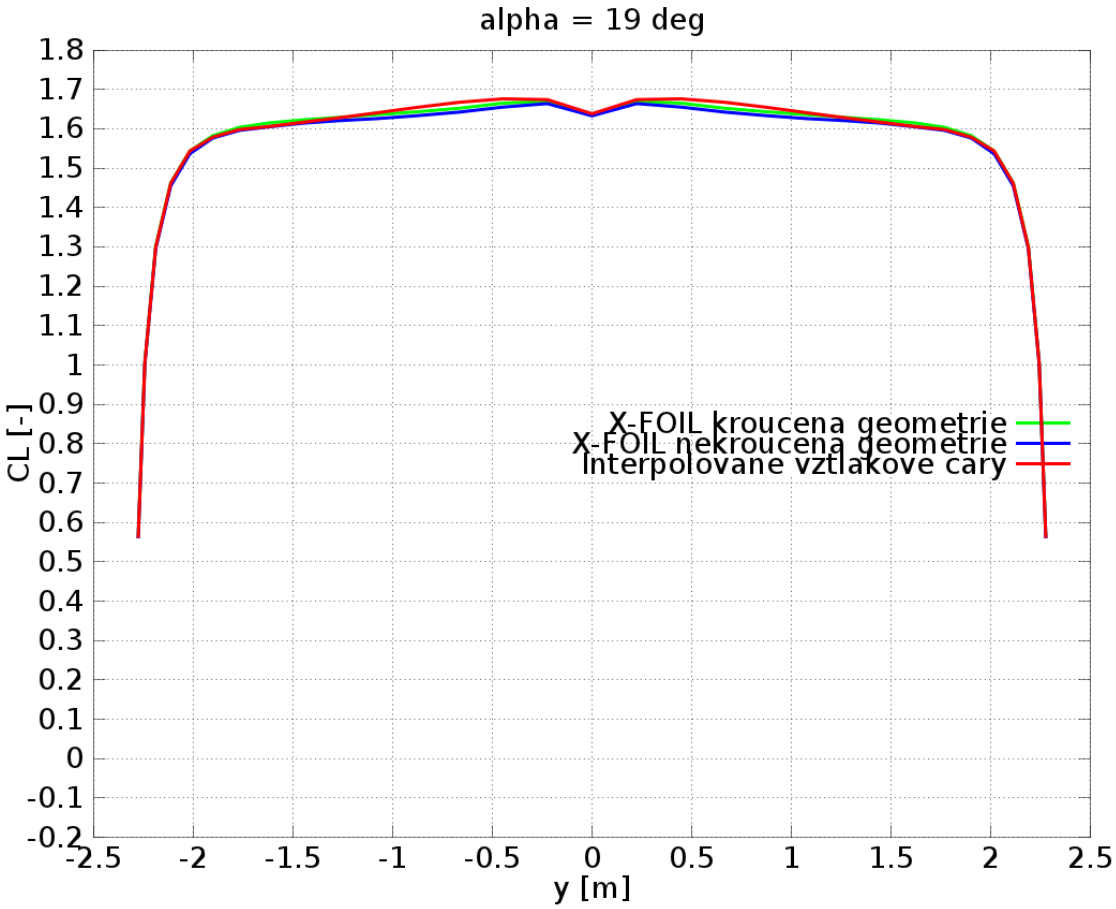
2.10 Závěr

Porovnáním výsledků z obrázku 2.7 a 2.8 je možné považovat všechny tři metody přípravy vstupních dat za rovnocenné pro použití na křídle s lichoběžníkovými segmenty. Pro praktické použití je nejvýhodnější metody s využitím lineární interpolace vztlakových čar. Tato metoda je nenáročná jak na vstupní data, tak na výpočetní výkon. Následující obrázek představuje porovnání maximálních profilových součinitelů vztlaku vzniklých lineární interpolací vztlakových čar a přímým výpočtem z kroucené geometrie v programu X-FOIL.



Obrázek 2.9: Porovnání maximálních profilových součinitelů vztlaku

Jednotlivé maximální profilové součinitele vztlaku se sice liší, avšak při porovnání rozložení vztlaku při maximálním vztlaku křídla je zřejmé, že tento rozdíl je téměř zanedbatelný, na integrální charakteristice dle 2.8 téměř neznatelný.



Obrázek 2.10: Výsledné rozložení vztlaku při maximálním součiniteli vztlaku křídla EXAMPLE WING

Jistý problém představuje nadhodnocování součinitele vztlaku v programu X-FOIL, avšak tyto vstupy lze nahradit vstupy z měření v aerodynamických tunelech. Pro nezvyklé kombinace profilů, například z různých profilových rodin, je přesto vhodné provést kontrolu správnosti interpolovaných vztlakových čar s využitím programu X-FOIL.

Kapitola 3

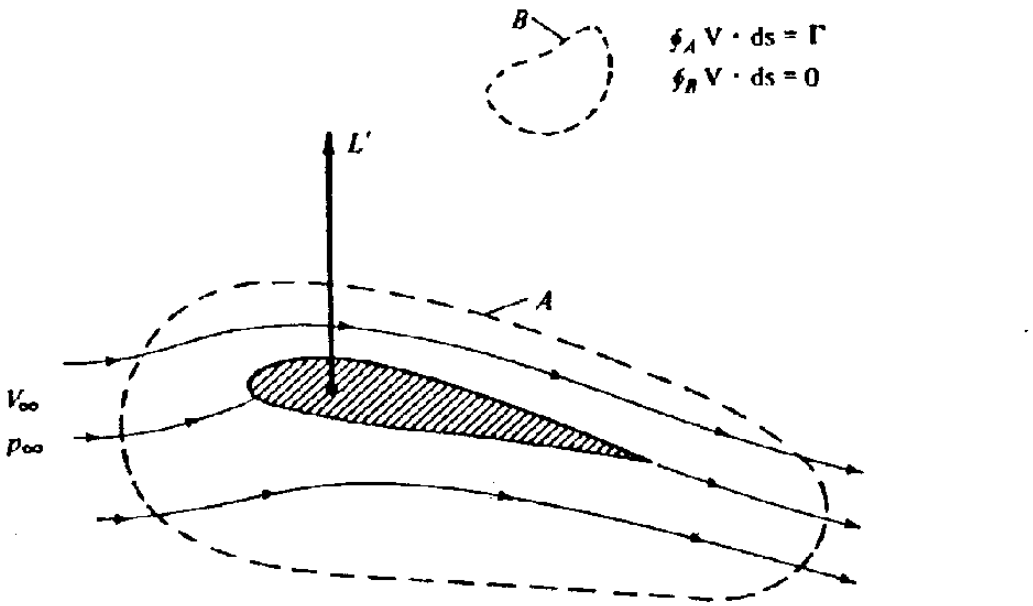
Teorie nosné čáry - lineární řešení

Tato kapitola si neklade za cíl detailní vysvětlení problému teorie nosné čáry křídla. To lze nalézt v literatuře [3] , [1], nebo [5]. Cílem této kapitoly je připomenout nejdůležitější pojmy, které jsou nutné pro výpočet rozložení vztlaku na křídle.

3.1 2D Cirkulace

Cirkulace je matematický způsob popisu ideálního,nevazkého, nestlačitelného proudového pole okolo tělesa. Je definována jako

$$\Gamma = V_\infty \oint_A \cdot ds \tag{3.1.1}$$



Obrázek 3.1: Cirkulace okolo profilu [1]

Důležitý je především vztah mezi cirkulací a výslednou aerodynamickou silou. Pro ten se vžil název "Věta Kutta-Žukovského". Protože se uvažuje proudění nevazké a nestlačitelné, představuje vztlaková síla výslednou aerodynamickou sílu.

Věta Kutta-Žukovského

$$L = \Gamma \cdot \rho \cdot v_\infty \tag{3.1.2}$$

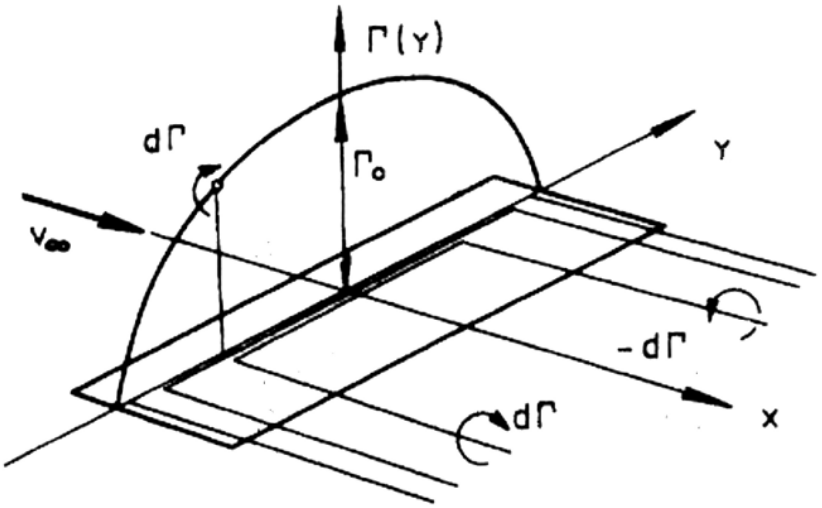
3.2 3D cirkulace

Vírové vlákno konstantní cirkulace Představuje myšlenou křivku v prostoru, která představuje spojnice míst s konstantní cirkulací. Při výpočtu rozložení vztlaku křídla je toto vlákno obvykle umísťováno do spojnice čtvrtinových bodů místních hloubek, která musí být po celém rozpětí přímá.

Biot-Savartův zákon byl původně odvozen pro popis elektromagnetického pole okolo vodiče. Později byl tento vztah použit pro popis rychlostního pole okolo křídla, ve formě vztahu

$$V = \frac{\Gamma}{4\pi} \cdot \frac{dl \times r}{|r|^3} \tag{3.2.1}$$

Superpozice vírových vláken se provádí, aby bylo možné přesně popsat rychlostní pole okolo křídla. Použití vírového vlákna s konstantní cirkulací by vedlo k nekonečně velkým indukovaným rychlostem na koncích křídla. Navíc by takto nebylo možné popsat složitější půdorysy křídla.



Obrázek 3.2: Superpozice vírových vláken, převzato a upraveno z [3]

3.3 Prandtlova rovnice nosné čáry křídla

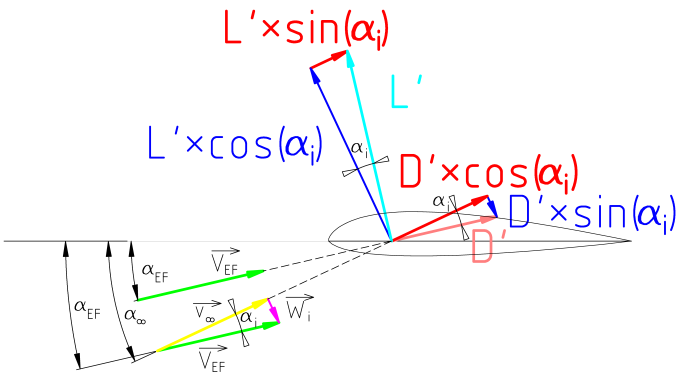
Prandtl odvodil vztah popisující vazbu mezi hledaným rozložením cirkulace $\Gamma(y)$ a vstupními parametry křídla. Ty sestávají z:

- $c(y)$ místní hloubky v závislosti na poloze řezu
- $a(y)$ místní sklon vztlakové čáry v závislosti na poloze řezu
- $\alpha_A(y)$ místní aerodynamický úhel náběhu v závislosti na poloze řezu

Prandtlova rovnice má tvar

$$\Gamma = \frac{1}{2} \cdot V_\infty \cdot c \cdot a \cdot [\alpha_A + \frac{1}{4 \cdot \pi \cdot V_\infty} \int_{\frac{-l}{2}}^{\frac{l}{2}} \frac{d\Gamma}{dy_1} \cdot \frac{dy_1}{y_1 - y}] \tag{3.3.1}$$

Tato rovnice popisuje rozložení indukovaných rychlostí W_i po rozpětí. Tyto indukované rychlosti snižují místní úhly náběhu z hodnoty α_{WING} na hodnotu α_{EF}



Obrázek 3.3: Určení efektivního úhlu náběhu α_{EF}

Aerodynamický úhel náběhu α_A je dán jako rozdíl úhlu náběhu a úhlu náběhu při nulovém vztlaku.

$$\alpha_A(y) = \alpha(y) - \alpha_0(y) \tag{3.3.2}$$

Místní součinitel vztlaku je přímo úměrný aerodynamickému úhlu náběhu:

$$C'_L(y) = \frac{\mathrm{d}\, C_L}{\mathrm{d}\, \alpha}(y) \cdot \alpha_A(y) \tag{3.3.3}$$

3.4 Glauertovo řešení Prandtlovy rovnice [3]

Glauert použil v rovnici 3.3.1 náhradu

$$y = \frac{l}{2} \cdot cos(\theta) \tag{3.4.1}$$

Hledaný průběh cirkulace po rozpětí aproximoval pomocí části Fourierovy řady

$$\Gamma(\theta) \;=\; 2 \cdot l \cdot V_\infty \cdot \sum_n A_n \cdot sin(n \cdot \theta) \tag{3.4.2}$$

Po dalších úpravách dle [3] má výsledná rovnice tvar:

$$\sum_n (sin(\theta) + \mu \cdot n) \cdot A_n \cdot sin(n \cdot \theta) \;=\; \mu \cdot \alpha_A \cdot sin(\theta) \tag{3.4.3}$$

kde μ představuje:

$$\mu \;=\; \frac{t \cdot a}{4 \cdot l} \tag{3.4.4}$$

Pro určení poloh jednotlivých řezů je možné použít dělení s konstantním přírůstkem úhlu. Pokud máme křídlo rozdělit na n řezů, pak poloha i-tého řezu bude dána vztahem:

$$\theta_i \;=\; \frac{\pi \cdot i}{n+1} \tag{3.4.5}$$

Toto dělení je ve zbytku této práce použito jako jediné. Pokud má být známa hodnota součinitele vztlaku v rovině symetrie křídla je nutné, aby počet řezů byl lichý.

3.5 Maticové řešení Prandtlovy rovnice

Rovnici 3.4.3 je možné pro požadovaný počet řezů n zapsat v maticové podobě jako:

$$|\mathbb{S}|_{n,n} \times |\mathbb{A}|_{n,1} \;=\; |\mathbb{B}|_{n,1} \tag{3.5.1}$$

Matice $|\mathbb{S}|_{n,n}$ je určena jako

$$|\mathbb{S}|_{n,n} \;=\; \begin{pmatrix} (\sin(\theta_1) + 1 \cdot \mu_1) \cdot \sin(1 \cdot \theta_1) & \cdots & (\sin(\theta_1) + n \cdot \mu_1) \cdot \sin(n \cdot \theta_1) \\ (\sin(\theta_2) + 1 \cdot \mu_2) \cdot \sin(1 \cdot \theta_2) & \cdots & (\sin(\theta_2) + n \cdot \mu_2) \cdot \sin(n \cdot \theta_2) \\ \vdots & \ddots & \vdots \\ (\sin(\theta_n) + 1 \cdot \mu_n) \cdot \sin(1 \cdot \theta_n) & \cdots & (\sin(\theta_n) + n \cdot \mu_n) \cdot \sin(n \cdot \theta_n) \end{pmatrix} \tag{3.5.2}$$

Matice $|\mathbb{A}|_{n,1}$ je určena jako

$$|\mathbb{A}|_{n,1} \;=\; \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix} \tag{3.5.3}$$

Matice $|\mathbb{B}|_{n,1}$ je určena jako

$$|\mathbb{B}|_{n,1} \;=\; \begin{pmatrix} (\mu_1 \cdot \alpha_{A_1}) \cdot \sin\theta_1 \\ \vdots \\ (\mu_n \cdot \alpha_{A_n}) \cdot \sin\theta_n \end{pmatrix} \tag{3.5.4}$$

Úkolem je nalezení neznámých koeficientů Fourierova rozvoje $A_1 \ldots A_n$ To se provede pomocí maticového počtu jako:

$$|\mathbb{A}|_{n,1} \;=\; |\mathbb{S}|^{-1}_{n,n} \times |\mathbb{B}|_{n,1} \tag{3.5.5}$$

Ze známých koeficientů Fourierova rozvoje $A_1 \cdots A_n$ je možné určit místní indukované úhly náběhu každého Glauertova řezu jako:

$$\alpha_i(\theta_i) \;=\; \sum_{j=1}^{i} j \cdot A_j \cdot \frac{\sin(j \cdot \theta_j)}{sin(\theta_j)} \tag{3.5.6}$$

Pro určení místních součinitelů vztlaku v každém Glauertově řezu je nutné doplnit vztahy pro jejich závislost na aerodynamickém úhlu náběhu:

$$\alpha_{EF} \;=\; \alpha_{WING} - \alpha_i \tag{3.5.7}$$

$$\alpha_A \;=\; \alpha_{EF} - \alpha_0 \tag{3.5.8}$$

Potom je možné určit místní součinitel vztlaku v Glauertově řezu jako:

$$C'_L(\alpha_A) \;=\; \alpha_A \cdot a \tag{3.5.9}$$

Kapitola 4

Oprava na nelinearitu

4.1 Nutné vstupy do nelineárního výpočtu

Při lineárním výpočtu je nutné znát v každém Glauertově řezu tři parametry:

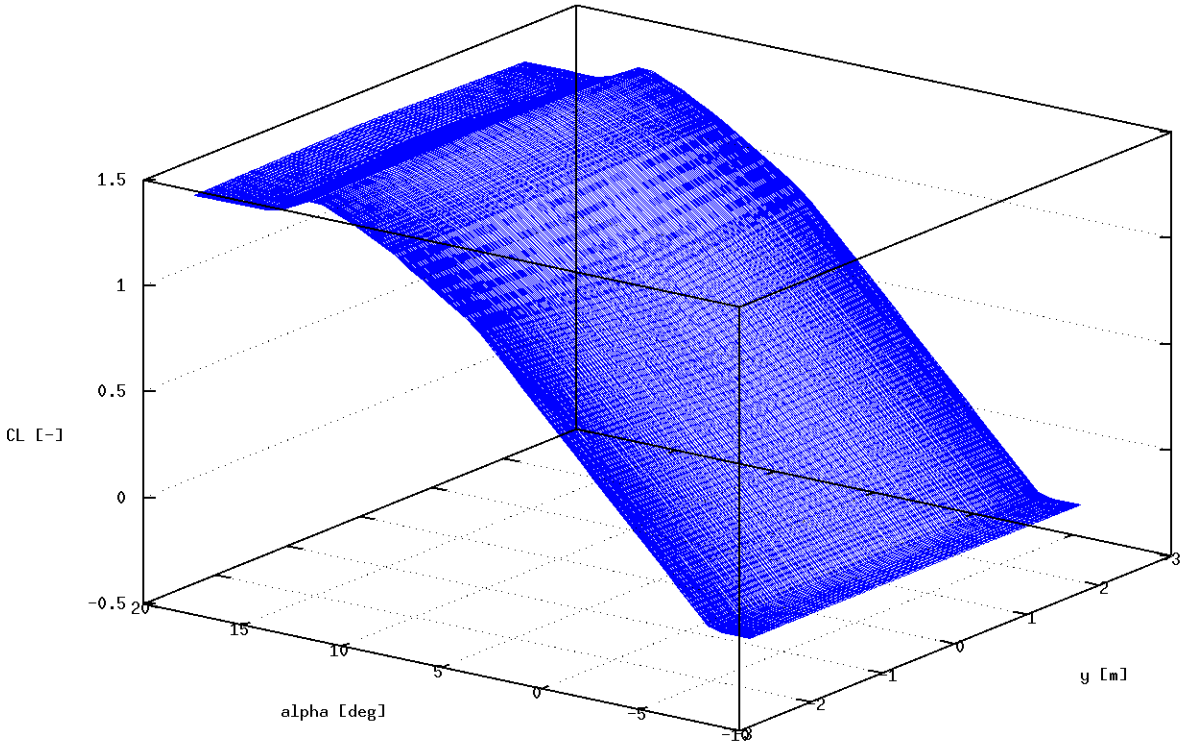
- úhel nulového vztlaku profilu
- sklon vztlakové čáry profilu
- maximální součinitel vztlaku profilu

Obvykle se tyto parametry zadávají v místech, kde se mění profiláž křídla, například v kořenovém a koncovém řezu. Pokud potřebné hodnoty nejsou známy, získávají se pomocí lineární interpolace.

Při nelineárním výpočtu je nutné přiřadit každému Glauertovu řezu vlastní vztlakovou čáru. To se opět uskuteční použitím lineární interpolace. Tentokrát se ale provádí tříroz-měrná interpolace. Interpolují se hodnoty součinitele vztlaku v závislosti na poloze řezu a úhlu náběhu. Můžeme uvažovat kartézský souřadný systém s osami:

- y
- α
- C_L

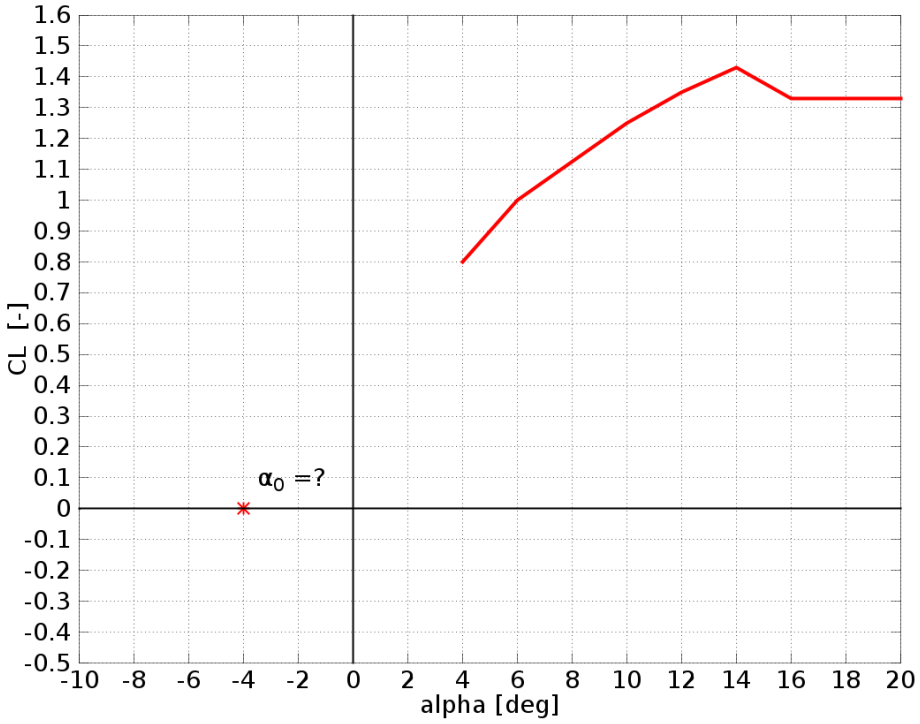
Potom vstup do nelineárního výpočtu bude tvořit jakousi plochu v tomto souřadném systému. Pro každý Glauertův řez je možné jednoduše určit součinitel vztlaku, který odpovídá danému úhlu náběhu řezu. Pro praktické použití se jedná o vztlakové čáry, jejichž poloha odpovídá poloze řezu ve výše uvedeném souřadném systému.



Obrázek 4.1: Vstup nutný pro nelineární řešení - vztlaková plocha

4.2 Omezení vstupních dat

Vstupní data podléhají určitým omezením, vstupy se do výpočtu zadávají jako souřadnice bodů. Ty jsou zadávány od nejmenší hodnoty úhlu náběhu po největší.Pokud je potřeba znát hodnotu mimo zadané souřadnice, dopočte pomocí lineární interpolace. Jejich dělení může být zadáno nepravidelně, avšak během výpočtu je dělení převedeno na pravidelné, kvůli rychlejšímu provádění interpolací.
Především je nutno vzpomenout na rovnici 3.3.3 Z ní vyplývá, že v každém řezu je nutno znát aerodynamický úhel náběhu, resp. úhel nulového vztlaku řezu. Je tedy nutné, aby ve všech řezech byl zadán úhel nulového vztlaku, nebo aby bylo možné jej jednoznačně dopočíst pomocí interpolace.

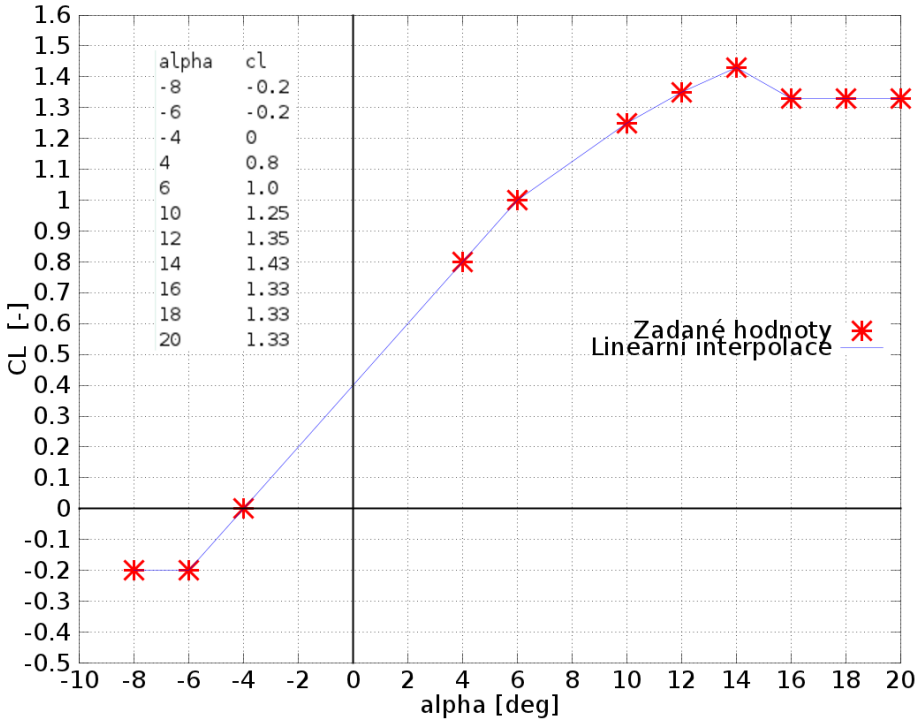


Obrázek 4.2: Příklad špatně zadaného vstupu, nelze jednoznačně určit úhel nulového vztlaku

Pokud je úhel náběhu větší nebo menší, než ve vstupním zadání, provádí se lineární extrapolace podle derivace $\frac{\mathrm{d}\,C_L}{\mathrm{d}\,\alpha}$. Experimentálně se ukazuje jako nejvýhodnější zadat derivace v krajních bodech jako nulové, toho lze dosáhnout například přidáním dalšího bodu se stejnou hodnotou C_L vně koncového bodu. Je možné použít i takové profilové charakteristiky, kde existuje i více než jeden úhel nulového vztlaku, například pro úhel náběhu $\alpha \in 0° \ldots 360°$ Je ale nutné upravit algoritmus pro vyhledávání úhlu nulového vztlaku. To lze uskutečnit například postupným procházením dané vztlakové čáry z bodu $\alpha = 0°$. Běžně použitý vyhledávací algoritmus pouze provádí lineární interpolaci vztlakové čáry pro bod $C_L = 0$, v případě, že by existovalo více takových bodů, algoritmus selže.

Správně zadaný vstup pro praktické použití tedy musí vyhovovat těmto podmínkám:

- Zadává ve sloupcích úhel náběhu profilu a tomu odpovídající součinitel vztlaku
- Hodnoty jsou setříděny podle úhlu náběhu a to od nejmenšího po největší
- Existuje jeden bod, nebo jeho jednoznačná interpolace, pro nějž platí $C_L = 0$
- Derivace $\frac{\mathrm{d}\,C_L}{\mathrm{d}\,\alpha}$ v krajních bodech jsou nulové



Obrázek 4.3: Příklad správně zadaného vstupu se vstupními daty

4.3 Podstata korekce na nelinearitu

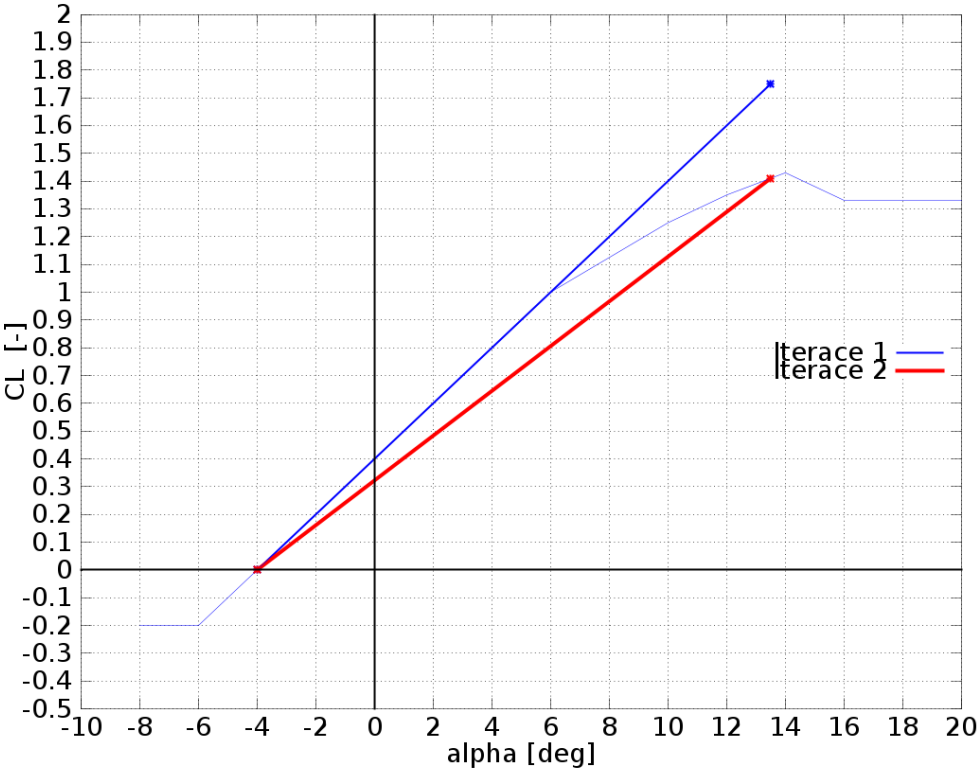
Nelineární výpočet využívá standardního lineárního řešení, které je popsáno v kapitole 3 - Teorie nosné čáry - lineární řešení , toto lineární řešení tvoří základ níže popsané iterační smyčky, která provádí nelineární korekce. Postup byl publikován v literatuře [11], nebo [4]. Zde uvedený postup výpočtu vzchází z těchto zdrojů.

Před začátkem vlastního výpočtu je ve všech řezech nutno určit hodnoty sklonů vztlakových čar, ty jsou dány hodnotami z lineární oblasti, například jako hodnota derivace mezi body $\alpha = 0°$ a $\alpha = 2°$. Matematicky zapsáno :

$$a_{iter=0}(y) = \frac{\mathrm{d}\, C_L}{\mathrm{d}\, \alpha} = \frac{C_L(2) - C_L(0)}{2 - 0} \tag{4.3.1}$$

Z podstaty nelineární korekce je možné zadat sklony vztlakových čar libovolně, ale pokud bude zadaný sklon blízký sklonu v lineární oblasti, ušetří se takto značný počet iterací. Pokud budou vztlakové čáry ve všech Glauertový řezech ve svých lineárních oblastech, provede se iterační smyčka pouze jednou a se stejným výsledkem, jako při klasickém lineárním výpočtu.

Iterační smyčka provede lineární výpočet a v každém Glauertově řezu určí hodnoty efektivních úhlů náběhu α_{EF} a hodnoty místních součinitelů vztlaku. podle toho se každému řezu přiřadí nová hodnota sklonu vztlakové čáry, která se pro požadovaný efektivní úhel náběhu odečte ze vztlakové čáry daného řezu. Viz. následující obrázek.

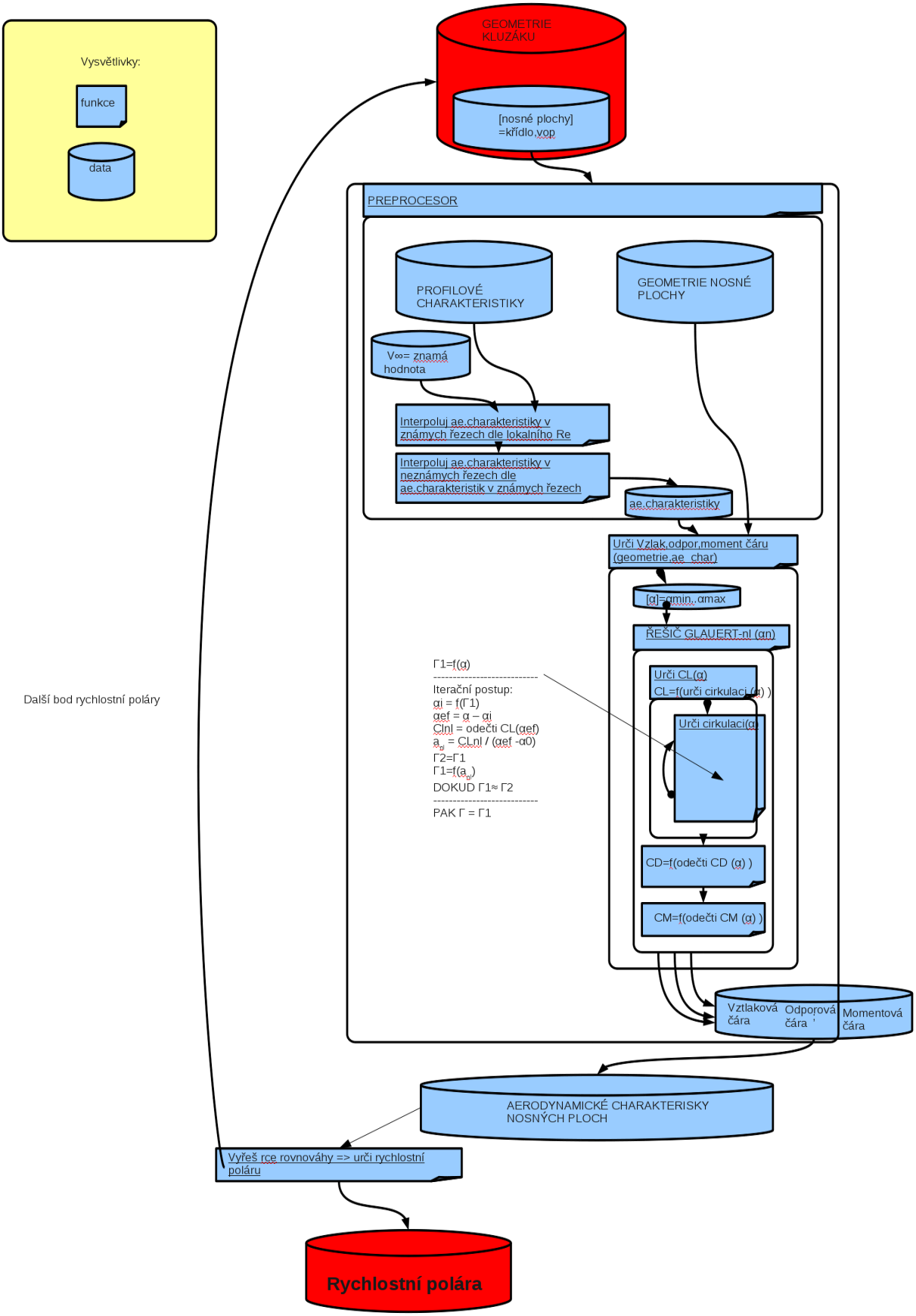


Obrázek 4.4: Určení nového sklonu vztlakové čáry pro Glauertův řez

Iterační smyčka se opakuje dokud se místní součinitele vztlaku pro daný úhel náběhu křídla v každém řezu neodlišují od součinitelů vztlaku daných příslušnou vztlakovou čárou daného řezu o předem danou chybu.

Podstata nelineární korekce tedy spočívá v hledaní nových sklonů vztlakových čar $\frac{\mathrm{d}\,C_L}{\mathrm{d}\,\alpha}(y)$ dokud ve všech řezech nebudou součinitele vztlaku přibližně odpovídat součinitelům vztlaku, které se pro odpovídající efektivní úhly náběhu odečtou z jednotlivých vztlakových čar. Tomu tedy odpovídá stav, kdy se hodnoty cirkulace z aktuálního a předešlého výpočtu ve všech řezech od sebe odlišují maximálně o hodnotu předem dané chyby.

Celkovou představu o funkci nelineární korekce lze získat z následujícího schématu, který ukazuje její využití při výpočtu rychlostní poláry:

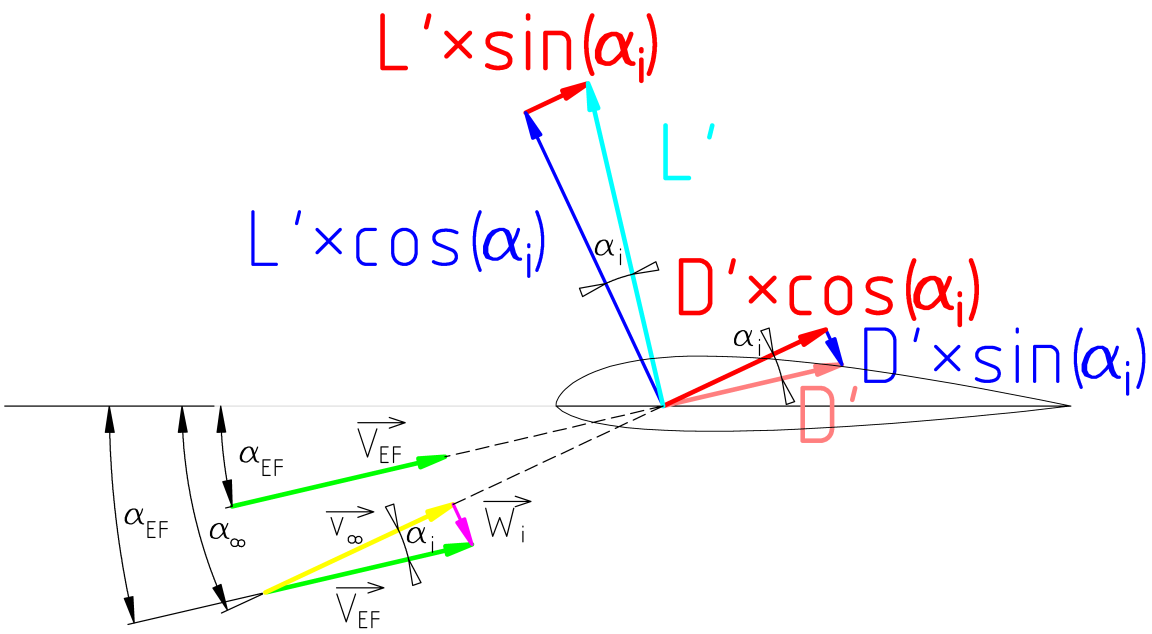

Vysvětlivky:
funkce
data
GEOMETRIE KLUZÁKU
[nosné plochy] =křídlo.vop
PREPROCESOR
PROFILOVÉ CHARAKTERISTIKY
GEOMETRIE NOSNÉ PLOCHY
V∞= známá hodnota
Interpoluj ae.charakteristiky v známých řezech dle lokalního Re
Interpoluj ae.charakteristiky v neznámych řezech dle ae.charakteristik v známých řezech
ae.charakteristiky
Urči Vzlak,odpor,moment čáru (geometrie,ae_char)
[α]=αmin..αmax
ŘEŠIČ GLAUERT-nl (αn)
Γ1=f(α)
Iterační postup:
αi = f(Γ1)
αef = α – αi
Clnl = odečti CL(αef)
ai = Clnl / (αef -α0)
Γ2=Γ1
Γ1=f(ai)
DOKUD Γ1≈ Γ2
PAK Γ = Γ1
Urči CL(α)
CL=f(urči cirkulaci (α))
Urči cirkulaci(α)
CD=f(odečti CD (α))
CM=f(odečti CM (α))
Vztlaková čára
Odporová čára '
Momentová čára
AERODYNAMICKÉ CHARAKTERISKY NOSNÝCH PLOCH
Vyřeš rce rovnováhy => urči rychlostní poláru
Rychlostní polára
Další bod rychlostní poláry

4.4 Určení sil působících v Glauertových řezech

Koncové víry způsobují změnu vektoru rychlosti z $\vec{V_\infty}$ na $\vec{V_{ef}}$ dle vztahu

$$\vec{V_{ef}} = \vec{V_\infty} + \vec{W_i} \tag{4.4.1}$$

Velikost vektoru $\vec{V_{ef}}$ je tedy větší než velikost vektoru $\vec{V_\infty}$ z toho vyplývá, že Reynoldsova čísla jednotlivých řezů jsou různá od Reynoldsových čísel, která by odpovídala součinu rychlosti $\vec{V_\infty}$ a místních hloubek. Korektní postup by tedy spočíval v odečtu vztlakových čar, které by odpovídaly Reynoldsovým číslům pro součiny rychlosti $\vec{V_{ef}}$ a místních hloubek. Rychlost $\vec{V_{ef}}$ je ale funkcí indukovaných úhlů náběhu α_i ,které jsou na počátku výpočtu neznámé. Bylo by tedy nutné výpočet několikrát opakovat, pro jejich iterační určení. Pro praktické účely je ale rozdíl v těchto Reynoldsových číslech zanedbatelný, blíží se chybě měření. Proto lze tento efekt bez dalších následků jednoduše zanedbat.



Obrázek 4.5: Určení sil v Glauertově řezu

Významnější efekt koncových vírů představuje otočení místní výslednice vztlaku. Dle věty Kutta-Žukovského 3.1.2 je působící vztlak kolmý na rychlost nabíhajícího proudu. Díky vlivům koncových víru se ale nebude jednat o rychlost $\vec{V_\infty}$, ale o rychlost $\vec{V_{ef}}$. Místní výslednice vztlaku $\vec{L'}$ tedy bude pootočena o úhel α_i. Situace je podrobně znázorněna na obrázku 4.5

Výpočet rozložení vztlaku a odporu musí tedy uvážit pootočení místní výslednice vztlaku, protože vztlak křídla se uvažuje vždy kolmý k vektoru rychlosti nerozrušeného proudu $\vec{V_\infty}$.

Dle obrázku 4.5 se určí síly působící v Glauertových řezech křídla jako:

$$L = L' \cdot \cos(\alpha_i) - D' \cdot \sin(\alpha_i) \tag{4.4.2}$$

$$D = D' \cdot \cos(\alpha_i) - L' \cdot \sin(\alpha_i) \tag{4.4.3}$$

Výpočet normálného a tečného zatížení se provede obdobně jako výpočet rozložení vztlaku a odporu. Indukovaný úhel α_i se nahradí efektivním úhlem náběhu α_{EF}. Rovnice tedy budou mít následující tvar.

$$N = L' \cdot \cos(\alpha_{EF}) - D' \cdot \sin(\alpha_{EF}) \tag{4.4.4}$$

$$T = D' \cdot \cos(\alpha_{EF}) - L' \cdot \sin(\alpha_{EF}) \tag{4.4.5}$$

Z těchto rovnic vyplývá důležitý poznatek: Pro správné určení vztlaku křídla je nutné znát nejen vztlakové, ale i odporové charakteristiky použitých profilů. Podobně jako pro správné určení odporu křídla je nutné znát nejen odporové, ale i vztlakové charakteristiky. Vztlak se obvykle u profilu v určitém rozsahu úhlu náběhu mění lineárně, avšak odpor se většinou mění zcela obecně. Proto je nejjednodušší uvažovat zadávání aerodynamických charakteristik pomocí výše uvedeného způsobu s pevným rozdělením úhlu náběhu a mezilehlé hodnoty určovat lineární interpolací.

Kapitola 5

Motivace pro použití oprav na nelinearitu

Tato kapitola představuje porovnání metody s běžně používanými řešeními nosné čáry. Vztahy dle rovnic 4.4.2 a 4.4.3 , které upřesňují místní rozložení vztlaku nejsou uvažovány, jelikož je běžně používané metody rovněž neuvažují.

5.1 Výpočet rozložení vztlaku metodou lineárního řešení

Celkové rozložení vztlaku Klasické postupy výpočtu rozložení vztlaku na křídle vzniky v době, kdy nebyla k dispozici výkonná výpočetní technika. Tomu jsou přizpůsobeny výpočetní postupy pro výpočet obecného rozložení vztlaku. Výsledné rozložení vztlaku se určuje jako součet tzv. normálného rozložení a rozložení,která modifikují tvar. Tento součet je poté vynásoben požadovaným součinitelem vztlaku křídla.

$$C_{L_W}(y) = C_{L_N}(y) \cdot C_{L_{REQ}} \; + \; C_{L_M}(y) \tag{5.1.1}$$

Normálné rozložení vztlaku je takové rozložení, které dává výslednou hodnotu součinitele vztlaku rovnu jedné. Matematicky je možno zapsat

$$C_{L_N} = \int_{\frac{-l}{2}}^{\frac{l}{2}} C'_L(y) \; \mathrm{d}y = 1 \tag{5.1.2}$$

Při určování normálného rozložení se využije faktu, že sklon vztlakové čáry křídla je konstantní. Takto stačí spočítat výsledné hodnoty součinitele vztlaku pro dva libovolné úhly náběhu křídla, například 0 a 1 radián. Obdržíme rovnici přímky, ze které je pak možné určit úhel náběhu křídla, který odpovídá součiniteli vztlaku křídla $C_{L_W} = 1$ .Zároveň můžeme zjistit úhel náběhu, který odpovídá nulovému vztlaku křídla a především velikost sklonu vztlakové čáry křídla. Ta je důležitá například při výpočtu poryvového zatíženi. Pokud potřebujeme znát rozložení vztlaku pro jiný součinitel vztlaku křídla, stačí nyní vynásobit hodnoty místních součinitelů vztlaku ve všech Glauertových řezech hodnotou požadovaného součinitele vztlaku. Tento postup přináší značnou úsporu práce při stanovení rozložení vztlaku na křídle, zároveň však přináší i některé problémy, o kterých bude pojednáno níže.

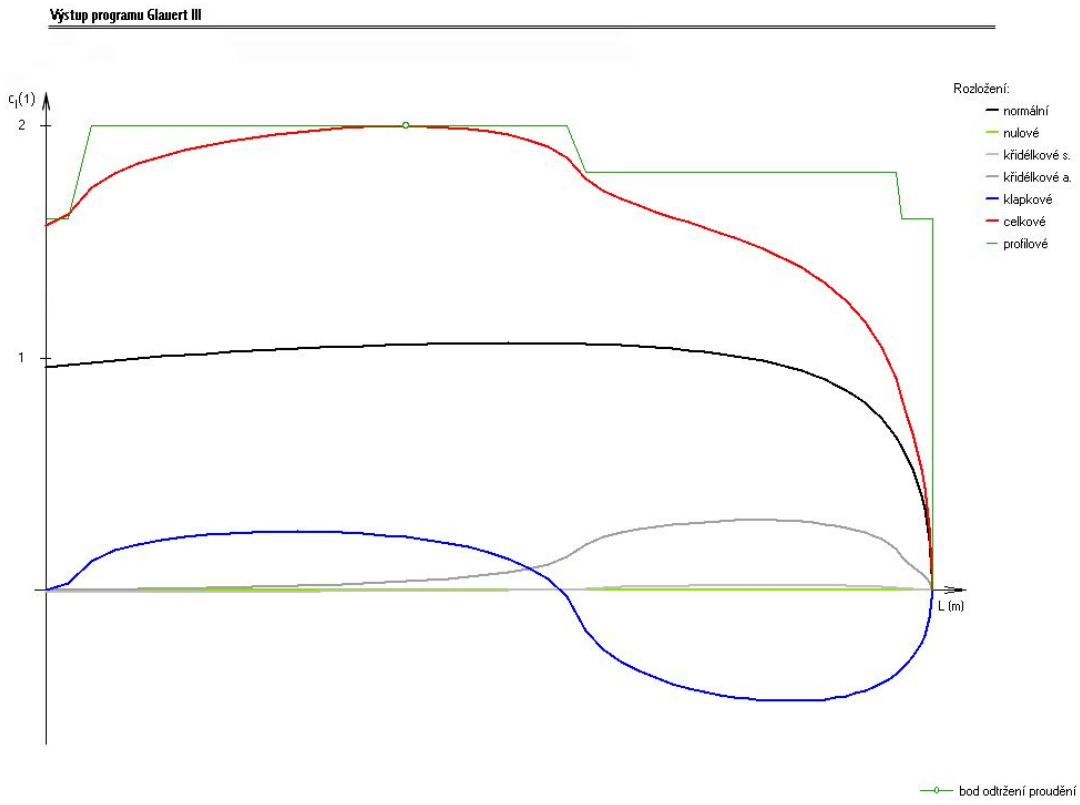
Rozložení modifikující tvar je takové rozložení, které dává výslednou hodnotu součinitele vztlaku rovnu nule. Matematicky je možno zapsat

$$C_{L_M} = \int_{\frac{-l}{2}}^{\frac{l}{2}} C'_L(y) \ \mathrm{d}y = 0 \tag{5.1.3}$$

Tyto rozložení se k normálnému rozložení přičítají v případě, že:

- křídlo je geometricky krouceno - nulové rozložení
- je vychýleno křidélko - křidélkové rozložení symetrické, antisymetrické, od tlumení
- je vychýlena klapka - klapkové rozložení

Společným znakem těchto rozložení je fakt, že výsledná hodnota jejich součinitele vztlaku je nulová. To je důležité proto, aby zůstala zachována snadnost výpočtu pro požadovaný součinitel vztlaku křídla dle rovnice 5.1.1. K objasnění výpočtu celkového rozložení vztlaku je uveden následující obrázek z programu GLAUERT III.



Obrázek 5.1: Rozložení vztlaku v programu Glauert III

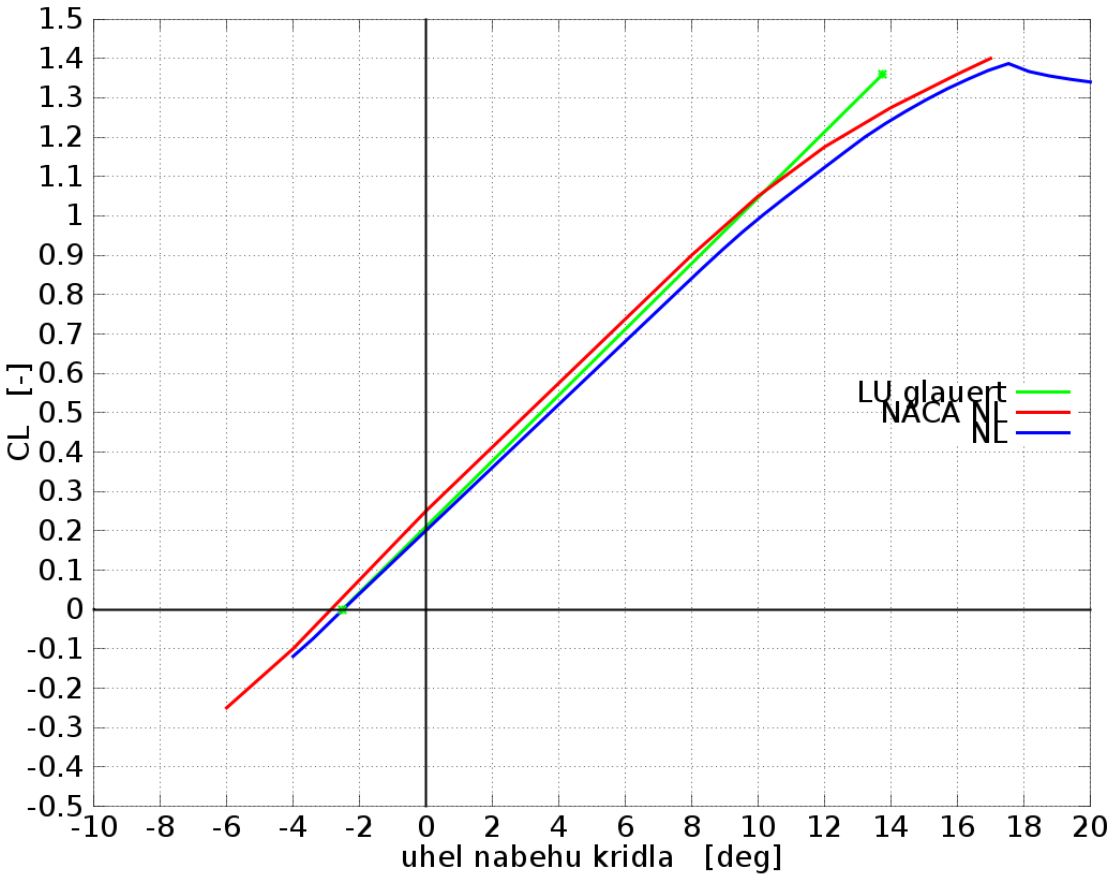
5.2 Výpočet rozložení vztlaku metodou nelineárního řešení

Korekce na nelinearitu na první pohled vnáší do výpočtu řadu komplikací. Již není možné postupovat dle rovnice 5.1.1, protože vztlakové charakteristiky každého Glauertova

řezu se mění s úhlem náběhu nelineárně. Navíc je nutné pro každou novou konfiguraci křídla sestavit nové zadání vstupních parametrů. ("vztlaková plocha") Při výpočtu je tedy nutno spočítat celou vztlakovou čáru křídla. Ta se spočte v daném rozsahu úhlů náběhu s pevně daným dělením. Například $\alpha_{MIN} = -20° \ldots \alpha_{MAX} = 20°$ s krokem po jednom stupni. Pokud je známa vztlaková čára křídla, je nyní možné určit úhel náběhu křídla, který odpovídá požadovanému součiniteli vztlaku. To lze provést například pomocí lineární interpolace. Pro přesnější určení hodnoty maximálního součinitele vztlaku
Výpočet rozložení vztlaku se zopakuje pro tento požadovaný úhel náběhu a tím se získá rozložení vztlaku pro požadovaný součinitel vztlaku křídla. Tento postup je nepochybně mnohem komplikovanější a bez výkonné výpočetní techniky velmi obtížně uskutečnitelný, avšak přináší i některé výhody.

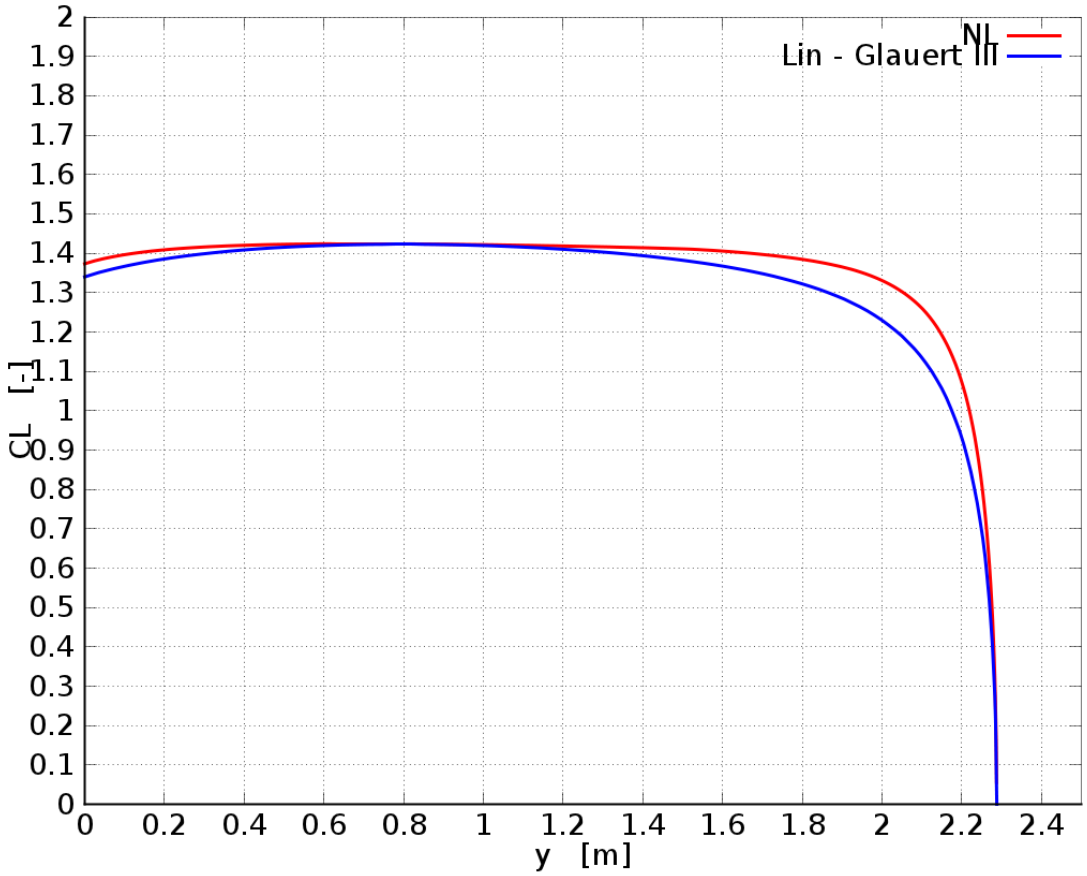
5.3 Porovnání lineárního výpočtu s nelineární korekcí

Vhodný příklad pro porovnání lze nalézt například v [11] NACA T.N.1269, který popisuje jeden ze způsobů výpočtu rozložení vztlaku v nelineární oblasti. V tomto reportu je také uveden modelový příklad výpočtu. Jeho výsledky jsou poté konfrontovány s tunelovým měřením a výpočtem bez uvažování nelinearit z programu Glauert III.



Obrázek 5.2: Porovnání metod výpočtu vztlaku křídla

Porovnáním integrálních aerodynamických veličin zjistíme, že při použití nelineární korekce dojde k určitému zvýšení maximálního dosažitelného součinitele vztlaku. Současně však dojde i poměrně značnému zvýšení úhlu náběhu, při kterém bude tohoto součinitele dosaženo. Výraznější rozdíly lze vypozorovat na rozložení součinitelů vztlaku po rozpětí.



Obrázek 5.3: Porovnání metod výpočtu vztlaku křídla

Kapitola 6

Rozložení vztlaku po rozpětí

Tato kapitola tato kapitola již uvažuje vztahy dle rovnic 4.4.2 a 4.4.3 resp. 4.4.4 a 4.4.5 , tak aby výsledná rozložení byla co nejblíže experimentálně získaným výsledkům.

6.1 Report NACA L5G10

Porovnává experimentálně určené rozložení normálného zatížení s teoretickým odhadem. Tento report původně vznikl za účelem výzkumu pádových vlastností křídla za vysokých podzvukových rychlostí.

Podstata tohoto reportu spočívá v rozmístění velkého množství tlakový snímačů (děr) na povrch křídla a následném určení normálného zatížení po rozpětí pomocí numerické integrace. Křídlo je umístěno v aerodynamickém tunelu, kde je měněn jeho úhel náběhu a rychlost obtékání. Schématicky je křídlo znázorněno na následujícím obrázku.

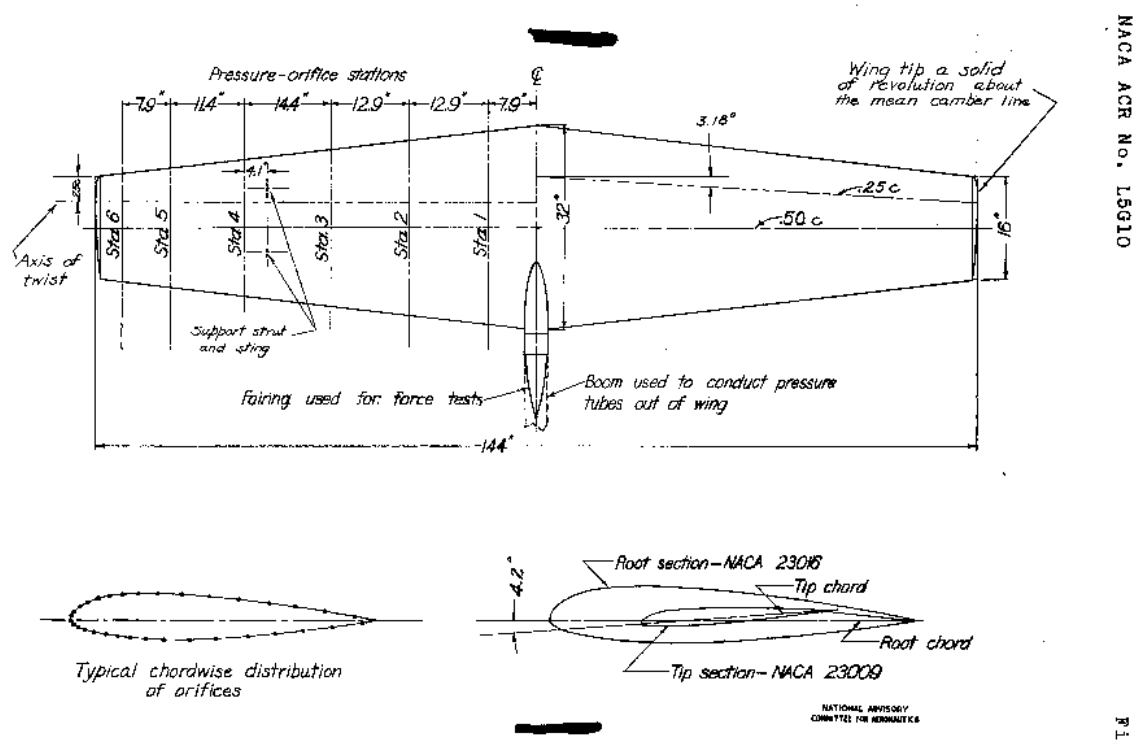


Figure 1.–Principal wing dimensions and locations of pressure orifices.

NACA ACR No. L5G10

Fig. 1

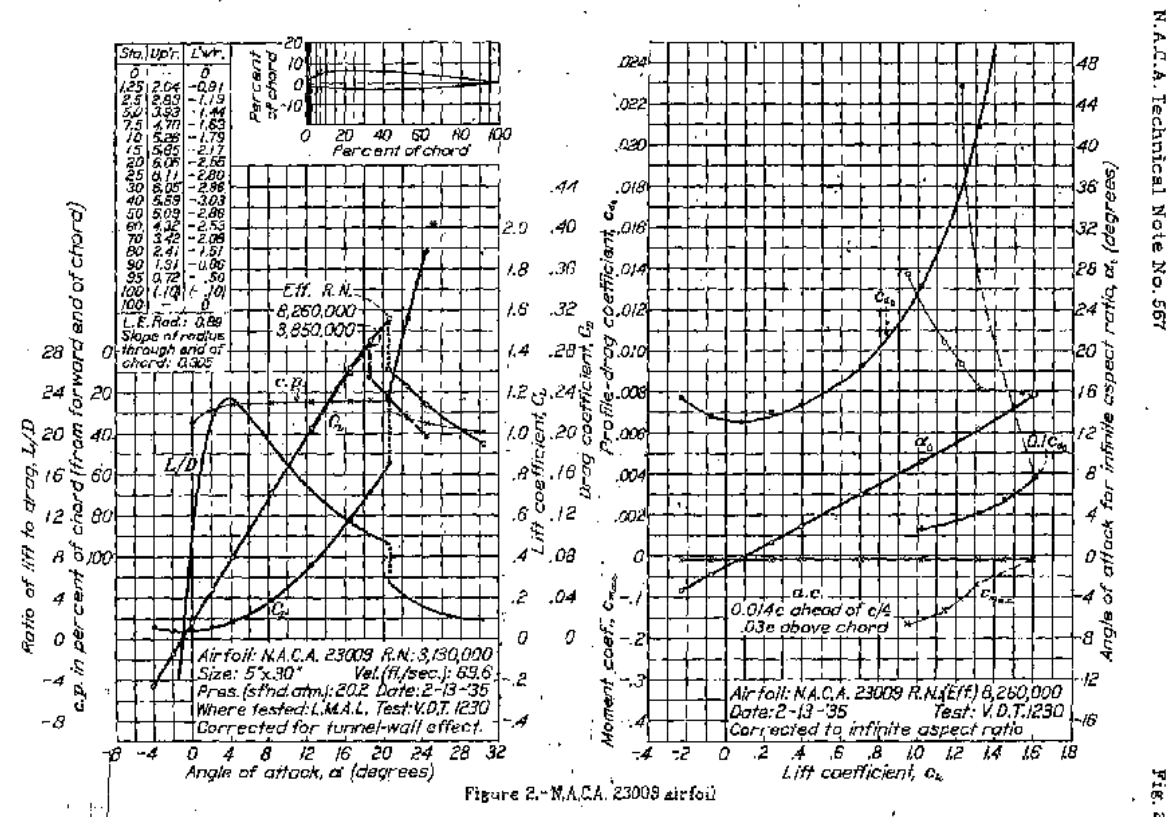
Obrázek 6.1: Křídlo dle NACA L5G10 [9]

Podrobnější popis poskytuje následující tabulka:

Název	FIGHTER WING
Plocha křídla	$S = 2.0801\ m^2$
Rozpětí	$b = 3.6576\ m$
Hloubka kořenového řezu	$c_R = 0.8128\ m$
Hloubka koncového řezu	$c_T = 0.4064\ m$
Kroucení koncového řezu	$\alpha_{G_T} = -4.2\ °$
Profil kořenového řezu	NACA 23016
Profil koncového řezu	NACA 23009
Re kořenového řezu	$7.2\ \cdot 10^6$

Referenční případ byl volen s ohledem na dostupné profilové podklady, odpovídá rychlosti nerozrušeného proudu $v_\infty = 135\frac{m}{s}$. Bohužel není k dispozici tunelové měření profilu NACA 23016, proto je místo něj uvažován kořenový profil NACA 23015, pro který jsou dostupné podklady z tunelového měření. Pomocí výše popsané lineární interpolace vztlakových čar byl sestaven vstup do výpočtu, jehož výsledky jsou dále porovnávány s výpočtem dle GLAUERT III a experimentálně získanými výsledky.

Aerodynamické charakteristiky byly odečteny z následujících podkladu:



Obrázek 6.2: Profil NACA 23009

Rozložení součinitelů vztlaku po rozpětí je porovnáváno pouze s výsledky z GLAUERT III,v reportu toto není zmíněno. GLAUERT III neuvažuje rovnice 4.4.2 a 4.4.3 tudíž pro stanovení rozložení součinitelů vztlaku je nutné zpětně dopočíst indukované úhly $\alpha_i(y)$. To lze provést pomocí koeficientů $A_1 \dots A_n$, které lze z programu vyexportovat. Indukovaný úhel náběhu se potom určí dle vztahu:

$$\alpha_i(y) \;=\; \sum_n n \cdot A_n \cdot \frac{\sin(n \cdot \theta(y))}{sin(\theta(y))} \tag{6.1.1}$$

Potom se odpovídající součinitele vztlaku určí jako

$$CL(y) \;=\; CL_{GLAUERT}(y) \cdot cos(\alpha(y)) \tag{6.1.2}$$

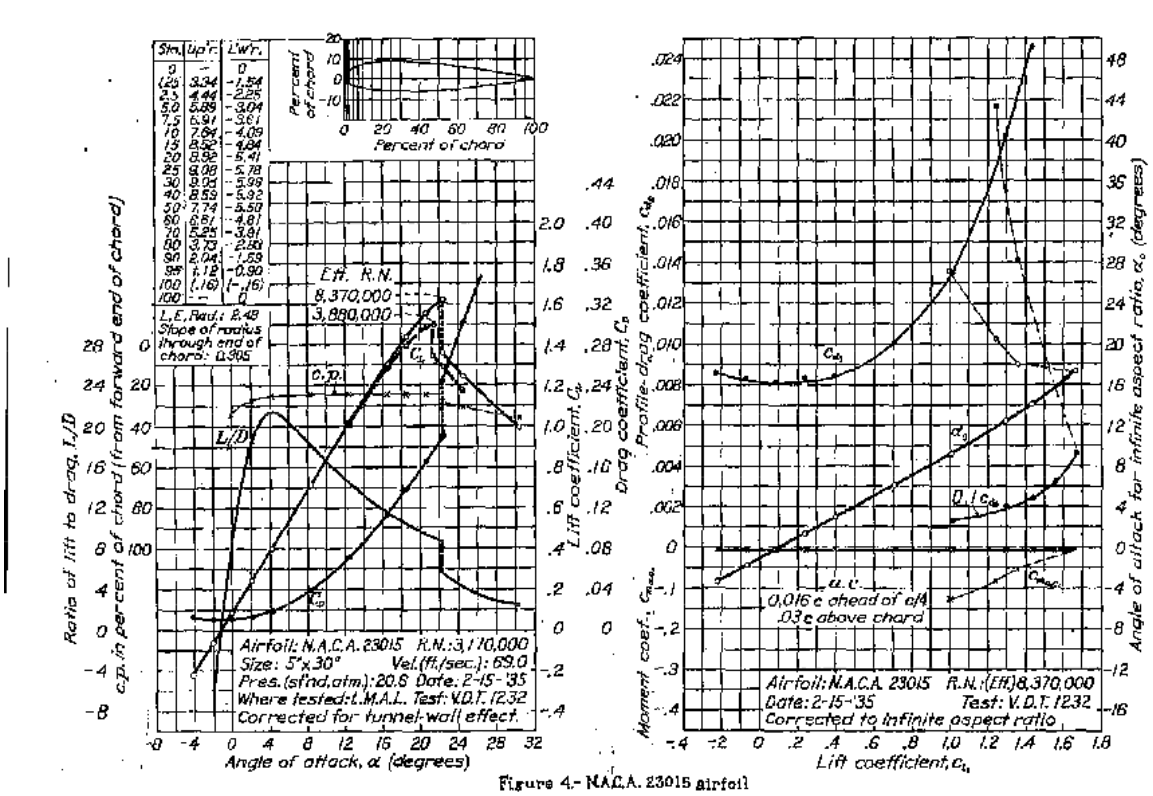
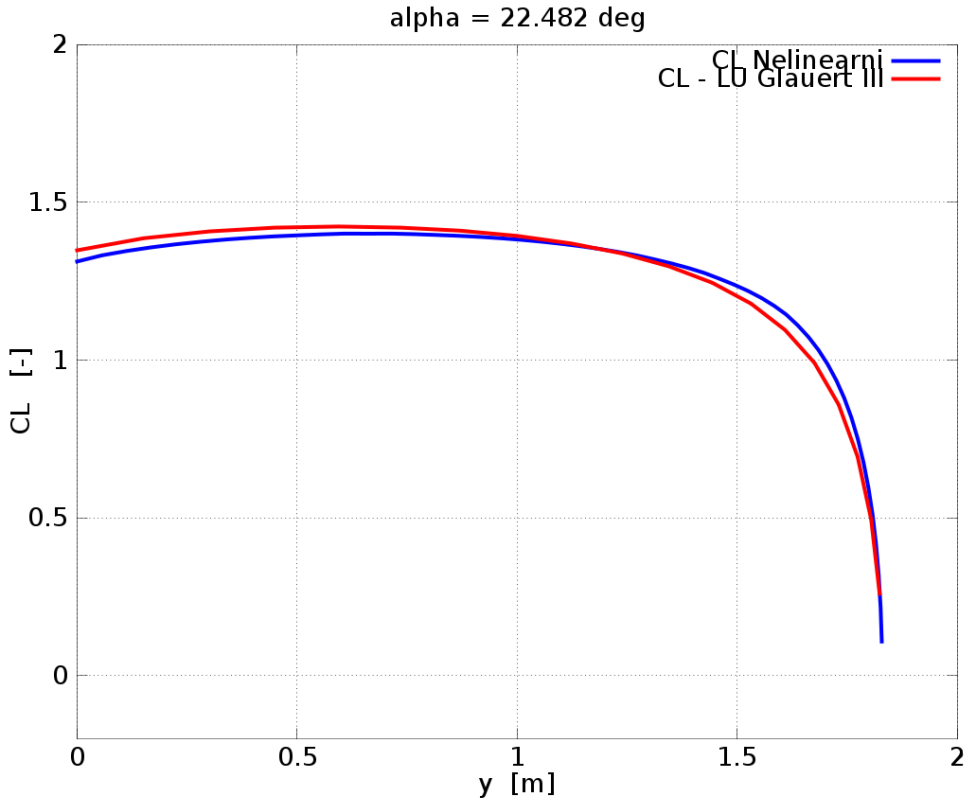


Figure 4.- N.A.C.A. 23015 airfoil

Obrázek 6.3: Profil NACA 23015

Následující obrázek porovnává výsledky nelineární metody s výsledky z programu GLAUERT III pro maximální součinitel vztlaku křídla. Z obrázku je vidět, že výsledky jsou téměř identické.



Obrázek 6.4: NACA L5G10, porovnání CL(y)

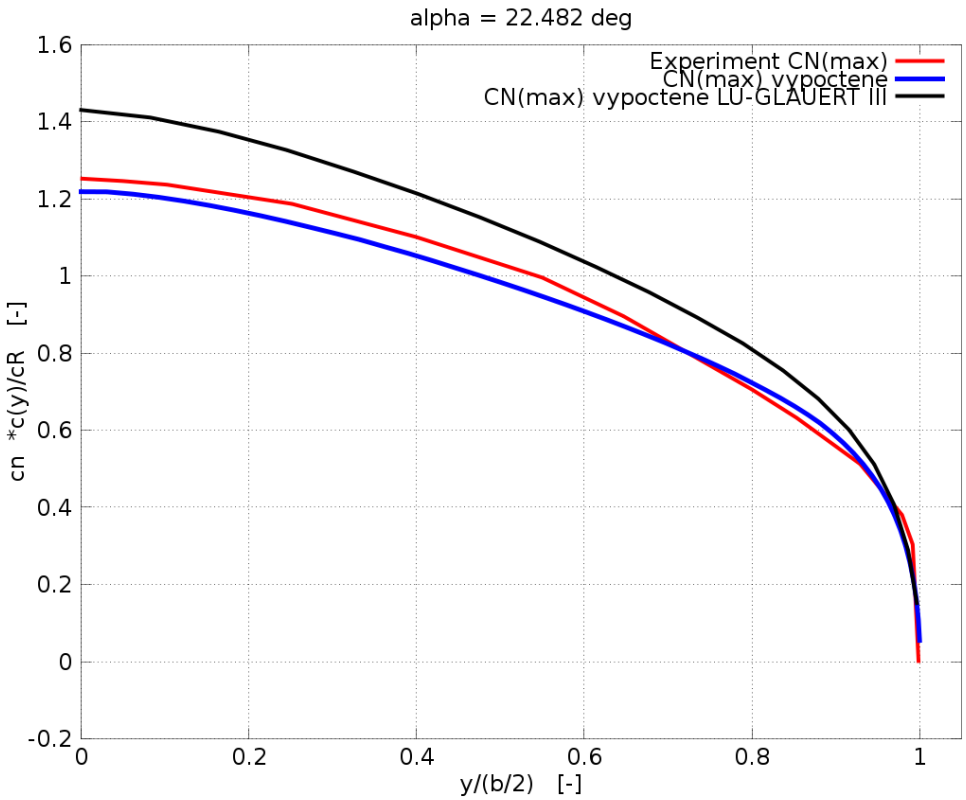
Rozložení normálného zatížení po rozpětí je prováděno s ohledem na report, který uvádí rozložení normálného zatížení ve tvaru

$$C_N(y) \cdot \frac{c(y)}{c_{ROOT}} \tag{6.1.3}$$

a pro rozpětí potom

$$\frac{y}{b/2} \tag{6.1.4}$$

Určení hodnot $CN(y)$ se provede dle rovnice 4.4.4. Bohužel pro výsledky z programu GLAUERT III nejsou dostupné informace o odporu, což značně ovlivní výsledky. Ty jsou zobrazeny na následujícím obrázku.



Obrázek 6.5: NACA L5G10, porovnání CN(y)

Závěrem lze konstatovat, že použití nelineární korekce s odečtem odporu v závislosti na úhlu náběhu dokáže podat poměrně přesné výsledky v porovnání s experimentálně zjištěnými daty. Největší problém představuje zajištění potřebných vstupních dat.

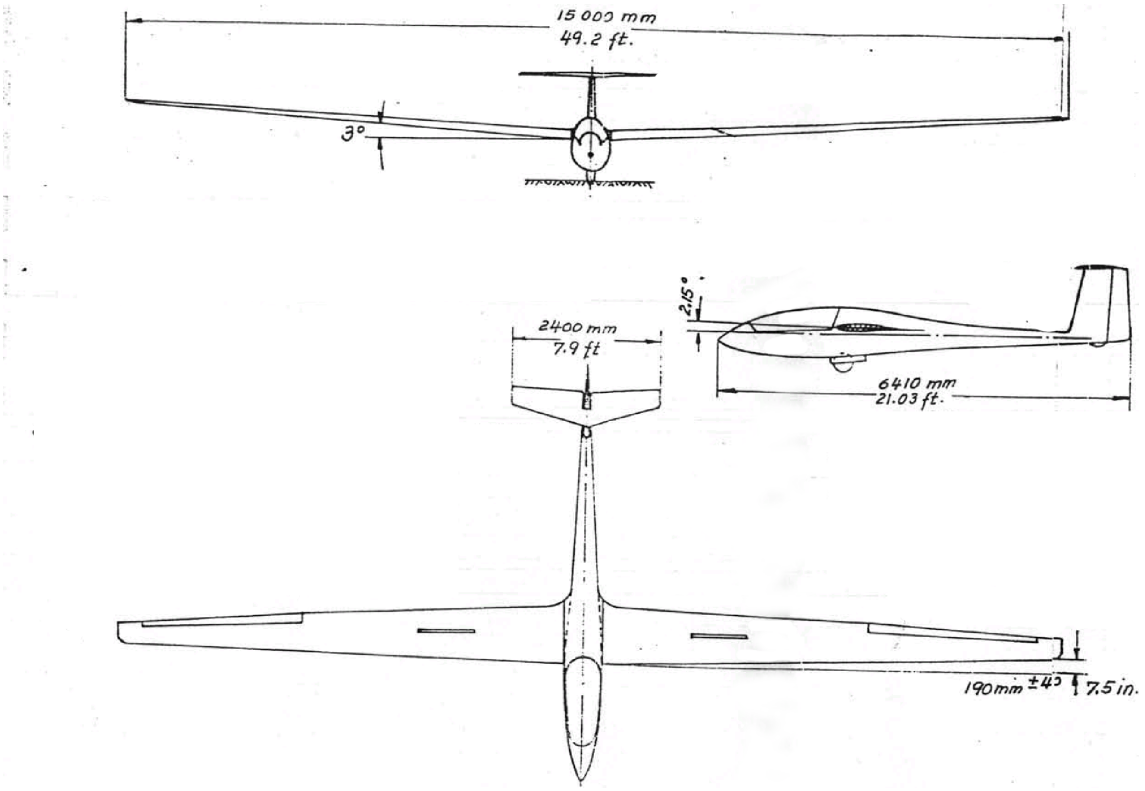
Kapitola 7

Rychlostní polára kluzáku Standard Cirrus

Odhad rychlostní poláry představuje pro konstruktéra kluzáku důležitý úkon. Se správně odhadnutou rychlostní je potom schopen odhadnout, v čem je nový kluzák lepší než stávající typy. Pokud je výpočet odhadu rychlostní poláry dostatečně rychlý, je možné spočítat několik variant křídla a poté vybrat nejvhodnější řešení pro zamýšlené použití kluzáku. Tato kapitola představuje jednu z možností, jak odhadnout rychlostní poláru kluzáku s uvažováním vlivu Reynoldsova čísla. Tento výpočet je následně konfrontován s výsledky letových měření.

7.1 Kluzák Standard Cirrus

byl vyráběn jako závodní kluzák pro třídu standard. Tím je dáno rozpětí jeho křídla 15 metrů a to že křídlo není vybaveno vztlakovou mechanizací. Rozměry kluzáku udává následující obrázek.



Obrázek 7.1: Standard Cirrus - muška [13]

Geometrie křídla a další údaje potřebné pro výpočet rychlostní poláry jsou uvedeny v následující tabulce.

Název	STANDARD CIRRUS
Plocha křídla	$S = 10\ m^2$
Rozpětí	$b = 15\ m$
Hloubka kořenového řezu	$c_R = 0.94\ m$
Hloubka koncového řezu	$c_T = 0.37\ m$
Kroucení koncového řezu	$\alpha_{G_T} = -0.75\ °$
Profil kořenového řezu	FX S02-196
Profil koncového řezu	FX 66-17AII-182
Letová hmotnost	330 kg

Tento kluzák vybrán pro porovnání, protože jsou k němu volně dostupná data z více nezávislých měření letových výkonů. Navíc je tento kluzák v tuzemských podmínkách velmi oblíbený a rozšířeny.

7.2 Postup výpočtu rychlostní poláry

sestává z určení požadovaných rychlostí letu, opakovaného volání funkce pro výpočet rozložení vztlaku a odporu křídla a řešení rovnic rovnováhy. Zjednodušeně lze postup zapsat takto:

- Zadaní aerodynamických charakteristik použitých profilů v závislosti na Reynoldsově čísle
- Zadání geometrie křídla
- Zadání letové hmotnosti
- Zadání předpokládaných rychlostí letu $v(i)$
- Výpočet smyčky pro výpočet rovnic rovnováhy klouzavého letu
- Sestavení rychlostní poláry

Vnitřní smyčka potom sestává z následujících úkonu:

- Určení konkrétní rychlosti letu $v = v(i)$
- Interpolace aerodynamických charakteristik pro Reynoldsova čísla odpovídající rychlosti letu v
- Výpočet odpovídající aerodynamické poláry kluzáku
- Řešení rovnic rovnováhy ustáleného klouzavého letu.
- Stanovení rychlostí Vx a Vz

Tato smyčka se provede pro všechny zadané rychlosti letu $v(i)$.

Aerodynamická polára kluzáku se vypočítá z aerodynamické poláry křídla. Pro jednoduchost je zanedbán vliv trupu a VOP na vztlak. Odpor trupu, ocasních ploch a interferenční odpor je uvažován jako konstantní pro všechny úhly náběhu. Zapsáno v rovnicích:

$$c_{L_{GLD}}(\alpha) = 0.9 \cdot c_{L_{WING}}(\alpha) \tag{7.2.1}$$

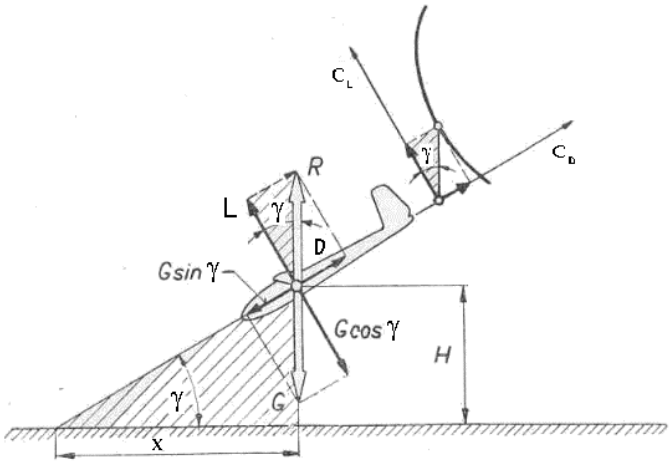
$$c_{D_{GLD}}(\alpha) = c_{D_{WING}}(\alpha) + D0 \tag{7.2.2}$$

Konstanta $D0$ byla pro kluzák Standard Cirrus odhadnuta jako:

$$D0 = 0.0042 \tag{7.2.3}$$

Tento odhad byl proveden na základě CFD výpočtů z literatury [8]

Řešení rovnic rovnováhy ustáleného klouzavého letu je prováděno dle následujícího obrázku:



Obrázek 7.2: Rovnováha ustáleného klouzavého letu [7]

Rovnice popisující rovnováhu ustáleného klouzavého letu jsou:

$$\frac{1}{2} \cdot \rho \cdot v^2 \cdot S \cdot c_L = m \cdot g \cdot cos(\gamma) \tag{7.2.4}$$

$$\frac{1}{2} \cdot \rho \cdot v^2 \cdot S \cdot c_D = m \cdot g \cdot sin(\gamma) \tag{7.2.5}$$

$$c_D = f(c_L) \tag{7.2.6}$$

Vlivem obecné závislosti součinitel odporu na součiniteli vztlaku v rovnici 7.2.6 není možné uvedenou soustavu rovnic řešit analytickými metodami. Jako způsob řešení bylo zvoleno řešení iterační metodou. Ta je popsána následujícími rovnicemi:

$$\frac{c_L = 2 \cdot m \cdot g \cdot cos(\gamma)}{\rho \cdot v^2 \cdot S} \tag{7.2.7}$$

$$c_D = f(c_L) \tag{7.2.8}$$

Tato závislost se určuje pomocí lineární interpolace.

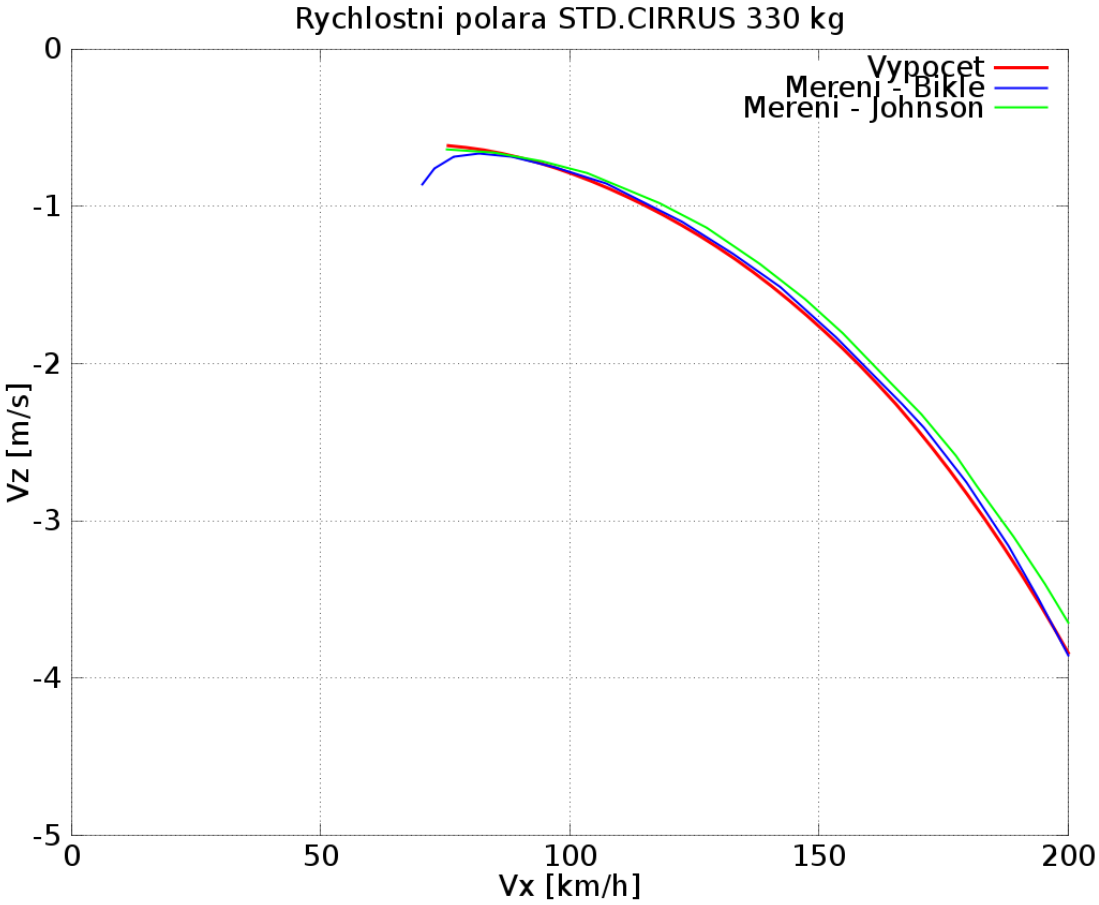
$$\gamma = arctg(\frac{c_D}{c_L}) \tag{7.2.9}$$

Iterace se provádí tak dlouho, dokud není dosaženo rozdílu mezi současnou a předchozí iteracemi menší než $1 \cdot 10^{-8}$

Sestavení rychlostní poláry Během řešení rovnic rovnováhy je ověřováno, zda není potřebný součinitel vztlaku větší, než maximální součinitel vztlaku křídla pro danou rychlost. Pokud tento případ nastane, uloží se do výsledné matice nulové hodnoty pro dopřednou a klesací rychlost. Při sestavování rychlostní poláry se potom z této matice kopírují pouze nenulové řádky.

7.3 Porovnání vypočtené rychlostní poláry s letovými měřeními

Letová měření výkonů kluzáku Standard Cirrus byly prováděny nezávisle na sobě P.Bicklem [2] a R.Johnsonem [6] . Následující obrázek porovnává tyto naměřené rychlostní poláry s vypočtenou.



Obrázek 7.3: Porovnání rychlostních polár

Závěrem lze říci, že se podařilo dosáhnout relativně dobré shody výpočtu s praktickým měřením. A to nejen v závislosti rychlosti opadání na rychlosti letu, ale i v určení pádové rychlosti.

Kapitola 8

Závěr

8.1 Teorie nosné čáry křídla s nelineární korekcí

poskytuje poměrně jednoduchý, výpočetně nenáročný a přesný nástroj pro určování vztlaku na křídle.

Existují některé problémy, které jsou touto metodou velmi obtížně řešitelné. Například vliv interference křídlo-trup, nebo křídlo z velkým vzepětím a úhlem šípu. V takových případech je nutné použít složitější výpočetní metody, například panelové metody, nebo CFD.

Většina současných kluzáků však odpovídá podmínkám pro výpočet pomocí teorie nosné čáry křídla. To je dáno především konstrukcí jejich křídel, kdy je použit jeden přímý nosník v přibližně jedné čtvrtině místních hloubek.

Pro takové případy se podařilo prokázat poměrně dobré shody teorie s experimentem a to jak pro integrální aerodynamické charakteristiky, tak pro rozložení vztlaku a odporu po rozpětí.

Po detailní verifikaci pomocí tunelových měření by mohla nelineární korekce přinést i určité úspory hmotnosti konstrukce křídla, vzhledem k přesnějšímu stanovení zatížení křídla, než při použití teorie nosné čáry křídla bez nelineární korekce.

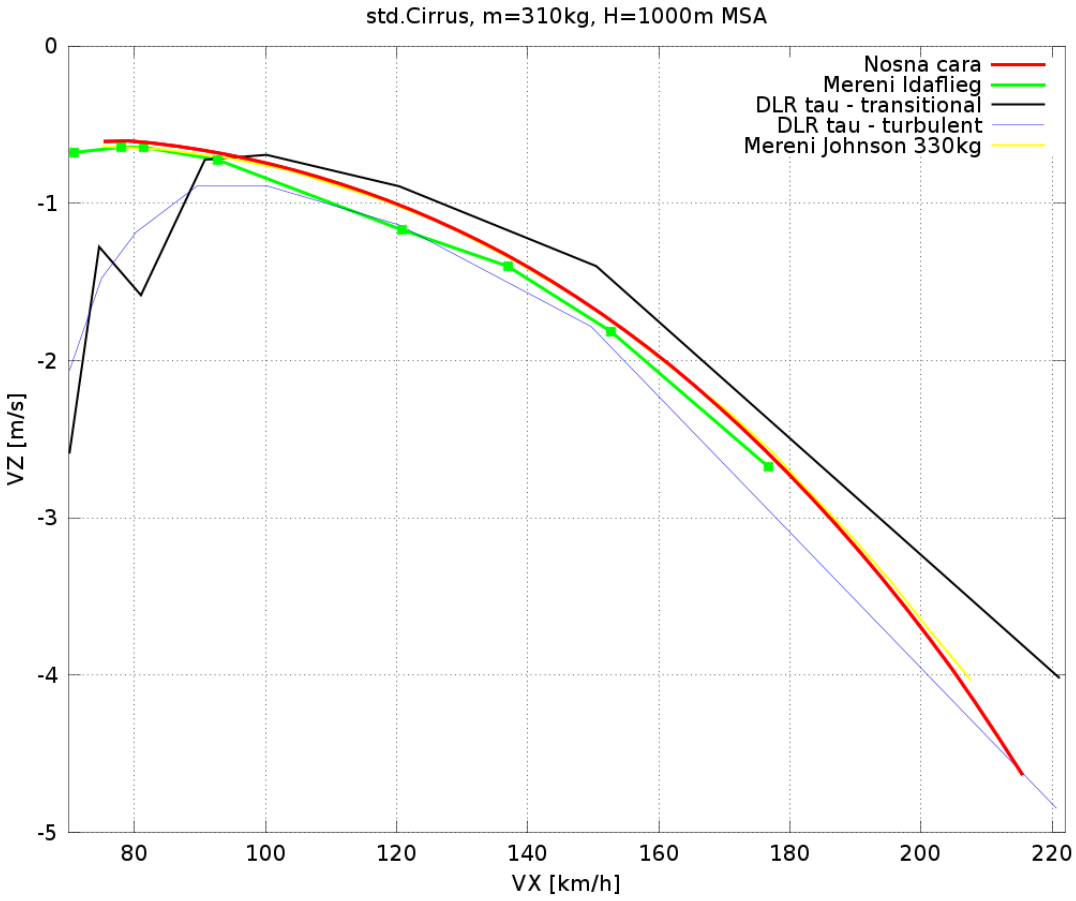
Nízká výpočetní náročnost a snadnost zadávání vstupů spolu s relativně vysokou přesností dělají z této metody vhodný nástroj pro předběžné aerodynamické optimalizace. Díky tomu, že vstupní problém je určen malým počtem parametrů je možné vstupy jednoduchým způsobem automatizovat. Díky krátkému výpočetnímu času je možné nechat spočítat více variant křídel a z nich následně vybrat tu nejvhodnější variantu pro požadovaný účel kluzáku.

8.2 Přímé porovnání teorie nosné čáry a CFD metod

Pro kluzák Standard Cirrus je dostupný výpočet rychlostní poláry pomocí CFD metody [10] , je použit kód DLR-TAU. Při tomto výpočtu je řešena geometrie křídlo + trup + svislé ocasní plochy. Vliv vodorovné ocasní plochy je dopočten pomocí empirických koeficientů. Výpočet je proveden pro následující podmínky:

Letová hmotnost	m	310 kg
Výška letu	H	1000 m MSA
Hustota vzduchu	ρ	1.112 $\frac{kg}{m^3}$
Kinematická vizkozita	μ	$0.1555 \cdot 10^{-4}$ $\frac{m^2}{s}$

Tyto vstupy jsou vloženy do výpočtu pomocí nelineární teorie nosné čáry, výsledek shrnuje následující obrázek, na kterém jsou kromě výsledků z DLR-TAU a teorie nosné čáry uvedeny také výsledky z letových měření v IDA-FLIEG pro hmotnost 310kg a [6] pro hmotnost 330kg. Při výpočtu pomocí nelineární teorie nosné čáry byl požit stejný postup jako v kapitole 7.2



Obrázek 8.1: Porovnání rychlostních polár

Z dostupných podkladů lze provést následující porovnání náročnosti a možností použitých metod pro případ výpočtu rychlostní poláry kluzáku Standard Cirrus:

Porovnání metod	CFD	Nosná čára
Doba formulace zadání	cca. 1 den	cca. 2 hod.
Rozsáhlost řešení	5.5 milionů uzlů sítě	70 řezů křídla
Řešených případů	8	42
Výpočetní doba	dny	5 minut
Omezení metody	není	pouze křídlo, musí vyhovovat 1.2

Z výše uvedeného vyplývá, že nelineární teorie nosné čáry je vhodná pro základní aerodynamický návrh kluzáku a optimalizaci základních parametrů křídla, jako jsou plocha křídla, místní hloubky, geometrické kroucení, použité profily. Dále může být tato metoda vhodná pro výpočet případů zatížení v letové obálce. To především díky krátkému času výpočtu.

Kapitola 9

Seznam zkratek a symbolů

SYMBOL	JEDNOTKA	VÝZNAM
$a = \frac{dC_L}{d\alpha}$	$\frac{1}{rad}$	sklon vztlakové čáry
b	m	rozpětí křídla
c	m	hloubka křídla v daném řezu
C'_D	-	součinitel odporu v daném řezu
C'_L	-	součinitel vztlaku v daném řezu
C'_M	-	součinitel klopivého momentu v daném řezu
C_D	-	součinitel odporu obecně
C_L	-	součinitel vztlaku obecně
C_M	-	součinitel klopivého obecně
C'_N	-	součinitel normálné síly v daném řezu
C'_T	-	součinitel tečné v daném řezu
g	$\frac{m}{s^2}$	gravitační zrychlení
m	kg	hmotnost
S	m^2	plocha křídla
V	$\frac{m}{s}$	rychlost obecně
V_∞	$\frac{m}{s}$	rychlost nerozrušeného proudu
α	rad	úhel náběhu obecně
α_A	rad	aerodynamický úhel náběhu
α_0	rad	úhel náběhu při nulovém vztlaku
α_i	rad	indukovaný úhel náběhu
α_{EF}	rad	efektivní úhel náběhu
α_G	rad	úhel geometrického kroucení
$\alpha_\infty = \alpha_{wing}$	rad	úhel náběhu křídla
Γ	-	cirkulace
γ	-	úhel klouzání
θ	rad	úhel Glauertova řezu
ρ	$\frac{kg}{m^3}$	hustota vzduchu
$[X, X, Z]$	$[-, -, -]$	souřadný systém
x	m	vzdálenost
y	m	vzdálenost
z	m	vzdálenost

Literatura

[1] Anderson, J. D.: *Fundamentals of aerodynamics*. McGrav-Hill, Inc., 1984, ISBN 0-07-001656-9.

[2] Bickle, P.: Polars of eight. `http://www.standardcirrus.org/`.

[3] Brož, V.: *Aerodynamika nízkých rychlostí*. ČVUT Praha, 1995, ISBN 80-01-01367-7.

[4] Fiľakovský, K.: *Předprojektová studie o větroni stavebnicové řady*. Moravan, 1968.

[5] Houghton E.L. ,Carpenter P.W.: *Aerodynamics for Engineering Students*. Butterworth-Heinemann, 1960, ISBN 0 7506 5111 3.

[6] Johnson, R. E.: Flight evulation of Standard Cirrus. `http://www.standardcirrus.org/`.

[7] Kolektiv autorů: *Aerodynamika a mechanika letu pro plachtaře*. Naše Vojsko, 1960.

[8] Krmela, L.: *AERODYNAMICKÝ NÁVRH A VÝPOČET KLUZÁKU TWIN SHARK*. Diplomová práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010.

[9] Pearson E.O , Evans A.J., West F.E. : *NACA Report L5G10*. NACA, 1945, Langley Memorial Aeronautical Laboratory.

[10] S. Melber-Wilkending, G. Schrauf, M. Rakowitz: *Aerodynamic Analysis of Flows with Low Mach- and Reynolds-Number under Consideration and Forecast of Transition on the Example of a Glider*. STAB-Tagung, 2004, Notes on Numerical Fluid Mechanics and Multidisciplinary Design.

[11] Sivells J.C. , Neely R.H. : *NACA Technical Note No. 1269*. NACA, 1947, Langley Memorial Aeronautical Laboratory.

[12] WWW stránky: Letecké fotografie Airliners.net. `http://www.airliners.net/`.

[13] WWW stránky: Schempp-Hirth. `http://www.schempp-hirth.com/`.

Příloha A

Disk CD

Disk CD obsahuje zdrojové kódy řešiče nelinární teorie nosné čáry, včetně zdrojových kódů použitých knihoven. Dále obsahuje pomocné skripty v GNU/octave a obslužné skripty v Unixovém shellu Bash.

Příloha B

Zdrojový kód řešiče

```
1 #include "../headers/LVP_headers.h"
2 #include "../headers/LVP_solver.h"
3 #include "lapacke.h"
4 #include <plplot/plplot.h>
5 #include <stdio.h>
6 #include <math.h>
7 #include <malloc.h>
8 #include <stdlib.h>
9 #include <ctype.h>
10 #include <stdlib.h>
11
12
13
14 int main (void)
15 {
16 int i,j,n_cuts,n,n_alpha;
17 double dPi = 4.0 * atan(1.0);
18 double*CL,*CD,*CM,*ALPHA;
19
20     FILE *fp = fopen("data/filters/alpha_res.txt", "r");
21     n_alpha = LVP_lines_count(fp);
22     ALPHA = (double*) malloc(sizeof(double) * (n_alpha));
23     LVP_read1COL(fp, ALPHA,  n_alpha);
24     fclose(fp);
25
26
27     CL = (double*) malloc(sizeof(double) * (n_alpha));
28     CD = (double*) malloc(sizeof(double) * (n_alpha));
29     CM = (double*) malloc(sizeof(double) * (n_alpha));
30
31
32 struct solver_ae_surfs AERO;
33 AERO=create_LIFT_SURF();
34
35
36 n_cuts=AERO.n_cuts;
37
38 struct solver_geometry GEO;
39 GEO= create_GS(n_cuts);
40
41 struct solver_angles AOAT;
42 AOAT= create_AOATS(n_cuts);
43
```

```
44 struct solver_ae_coefs COEFS;
45 COEFS = create_AE_COEFS(n_cuts);
46
47
48 ////////////////////////////////////////////////////////////////////////
49 fopen ("data/results/lift_dist.txt", "w");
50   for (j=0; j<n_alpha; j++)
51   {
52           for (i=0; i<n_cuts; i++)
53                   {
54     AOAT.Aw[i]=ALPHA[j];
55                   }
56
57
58 GET_LIFT(n_cuts,AERO, GEO,AOAT,COEFS);
59 GET_DRAG(n_cuts,AERO, GEO,AOAT,COEFS);
60 GET_MOMENT(n_cuts,AERO, GEO,AOAT,COEFS);
61
62   CL[j]=EVAL_LIFT(n_cuts,AERO, GEO,AOAT,COEFS);
63   CD[j]=EVAL_DRAG(n_cuts,AERO, GEO,AOAT,COEFS);
64   CM[j]=EVAL_MOMENT(n_cuts,AERO, GEO,AOAT,COEFS);
65
66   LVP_awrite3file(COEFS.n_cuts,"data/results/lift_dist.txt",AOAT.Aw,GEO.yi,COEFS.CL_nl);
67   LVP_awrite3file(COEFS.n_cuts,"data/results/drag_dist.txt",AOAT.Aw,GEO.yi,COEFS.CD);
68         LVP_awrite3file(COEFS.n_cuts,"data/results/moment_dist.txt",AOAT.Aw,GEO.yi,COEFS.C
69
70         LVP_awrite3file(COEFS.n_cuts,"data/results/angles.txt",AOAT.Aw,AOAT.Aef,AOAT.Ai);
71
72
73   LVP_awrite3file(COEFS.n_cuts,"data/results/normal_dist.txt",AOAT.Aw,GEO.yi,COEFS.CN);
74   LVP_awrite3file(COEFS.n_cuts,"data/results/tangent_dist.txt",AOAT.Aw,GEO.yi,COEFS.CT);
75
76   }
77
78 LVP_write2file(n_alpha,"data/results/lift_line.txt",ALPHA,CL);
79 LVP_write2file(n_alpha,"data/results/drag_line.txt",ALPHA,CD);
80 LVP_write2file(n_alpha,"data/results/moment_line.txt",ALPHA,CM);
81
82
83 WRITE_SOLVER_LOG("data/results/log.txt",n_cuts,AERO, GEO,AOAT,COEFS);
84
85 return 0;
86 }
```

Příloha C

Knihovna řešiče

```
1  #include "../headers/LVP_headers.h"
2  #include "../headers/LVP_solver.h"
3  #include "lapacke.h"
4  #include <stdio.h>
5  #include <math.h>
6  #include <malloc.h>
7  #include <stdlib.h>
8  #include <ctype.h>
9
10
11 //////////// CREATES STRUCTURE WITH GEOMETRICAL INPUTS FOR SOLVER ///////
12 //////////// ALSO PROVIDES ALLOCATION OF REQUIRED INPUTS
13         struct solver_ae_surfs create_LIFT_SURF(void)
14     {
15                 int alpha_lines,lift_surf_lines,n_elements,n_cuts;
16     double*alpha;
17     double*lift_surface;
18     double*drag_surface;
19     double*moment_surface;
20                 const char *lift_file="data/lift/lift_surf.txt";
21           const char *drag_file="data/lift/drag_surf.txt";
22     const char *moment_file="data/lift/moment_surf.txt";
23           const char *alpha_file="data/lift/alpha.txt";
24                 FILE *fp = fopen(lift_file, "r");
25                 FILE *fp2 = fopen(alpha_file, "r");
26     FILE *fp3 = fopen(drag_file, "r");
27     FILE *fp4 = fopen(moment_file, "r");
28                 lift_surf_lines = LVP_lines_count(fp);
29                 alpha_lines = LVP_lines_count(fp2);
30                 n_elements = LVP_elements_count(fp);
31     alpha= (double*) malloc(alpha_lines * sizeof(double));
32     lift_surface=(double*) malloc(lift_surf_lines * sizeof(double));
33     drag_surface=(double*) malloc(lift_surf_lines * sizeof(double));
34     moment_surface=(double*) malloc(lift_surf_lines * sizeof(double));
35     LVP_read1COL(fp, lift_surface, lift_surf_lines);
36     LVP_read1COL(fp3, drag_surface, lift_surf_lines);
37     LVP_read1COL(fp4, moment_surface, lift_surf_lines);
38           LVP_read1COL(fp2, alpha, alpha_lines);
39     fclose(fp);
40     fclose(fp2);
41     fclose(fp3);
42     fclose(fp4);
43
```

```
44                  n_cuts=lift_surf_lines/alpha_lines;
45    printf("%s\t%d","lift surf lines = ",lift_surf_lines);
46
47    struct solver_ae_surfs tmp;
48    tmp.n_cuts=n_cuts;
49    tmp.alpha_lines=alpha_lines;
50    tmp.lift_surf_lines=lift_surf_lines;
51    tmp.alpha=alpha;
52    tmp.lift_surface=lift_surface;
53    tmp.drag_surface=drag_surface;
54    tmp.moment_surface=moment_surface;
55
56      return tmp;
57
58      }
59
60
61
62 ////////////// CREATES STRUCTURE WITH GEOMETRICAL INPUTS FOR SOLVER ///////
63 ////////////// ALSO PROVIDES ALLOCATION OF REQUIRED INPUTS
64          struct solver_geometry create_GS(int n_cuts)
65          {
66          int i,n_elements;
67          double dPi = 4.0 * atan(1.0);
68          double b;
69    double *yi;
70          double *fi;
71          double *t;
72    double *t2;
73    double *ty;
74          double *AG;
75
76          yi= (double*)  malloc(n_cuts * sizeof(double));
77          fi= (double*)  malloc(n_cuts * sizeof(double));
78    t= (double*)  malloc(n_cuts * sizeof(double));
79    t2= (double*)  malloc(n_cuts * sizeof(double));
80    ty= (double*)  malloc(n_cuts * sizeof(double));
81    const char *span_file="data/geometry/span.txt";
82    FILE *fp2 = fopen(span_file, "r");
83    const char *chords_file="data/geometry/chords.txt";
84    FILE *fp3 = fopen(chords_file, "r");
85          const char *twist_file="data/geometry/wing_twist.txt";
86          FILE *fp4 = fopen(twist_file, "r");
87    n_elements = LVP_elements_count(fp3);
88    t= (double*)  malloc(n_elements * sizeof(double));
89    AG= (double*)  malloc(n_cuts * sizeof(double));
90    LVP_read1COL(fp3, t,  n_elements);
91    LVP_read1COL(fp4, AG,  n_elements);
92    fclose(fp3);
93    fclose(fp4);
94
95    LVP_read1COL(fp2, &b, 1);
96
97                  for (i=0; i<n_cuts; i++)
98                  {
99                  fi[i]=dPi*( (double)(i+1)/(n_cuts+1)   );
100                 yi[i]=b*cos(fi[i]);
101       t2[i]=t[i]*t[i];
102       ty[i]=t[i]*yi[i];
```

```
103         }
104
105
106         struct solver_geometry tmp;
107
108         tmp.n_cuts=n_cuts;
109   tmp.S=fabs(LVP_trapz(n_cuts,yi,t));
110         tmp.b=b;
111   tmp.cSAT=-(LVP_trapz(n_cuts,yi,t2))/fabs(LVP_trapz(n_cuts,yi,t)) ;
112   tmp.ySAT= 1.0; //-(LVP_trapz(n_cuts,yi,ty))/fabs(LVP_trapz(n_cuts,yi,t)) ;
113         tmp.yi=yi;
114         tmp.fi=fi;
115         tmp.t=t;
116   tmp.AG=AG;
117         return tmp;
118         }
119
120
121
122
123 ////////////// CREATES STRUCTURE WITH ANGLE OF ATTACK INPUT FOR SOLVER ///////
124 ////////////// PROVIDES ALLOCATION OF REQUIRED INPUTS
125 struct solver_angles create_AOATS(int n_cuts)
126   {
127   double *Aw;
128   double *A;
129   double *Ai;
130   double *Aef;
131
132   Aw= (double*)  malloc(n_cuts * sizeof(double));
133   A= (double*)  malloc(n_cuts * sizeof(double));
134   Ai= (double*)  malloc(n_cuts * sizeof(double));
135   Aef= (double*)  malloc(n_cuts * sizeof(double));
136
137   struct solver_angles tmp;
138
139   tmp.n_cuts=n_cuts;
140   tmp.Aw=Aw;
141   tmp.A=A;
142   tmp.Ai=Ai;
143   tmp.Aef=Aef;
144   return tmp;
145   }
146
147
148 ////////////// CREATES STRUCTURE WITH AERODYNAMICAL COEFFICIENTS AS OUTPUT
        FOR SOLVER ///////
149 ////////////// PROVIDES ALLOCATION FOR ALL OUTPUTS
150 struct solver_ae_coefs create_AE_COEFS(int n_cuts)
151         {
152         double *CL_nl;
153         double *CD;
154         double *CM;
155   double *CN;
156   double *CT;
157
158         CL_nl= (double*)  malloc(n_cuts * sizeof(double));
159         CD= (double*)  malloc(n_cuts * sizeof(double));
160         CM= (double*)  malloc(n_cuts * sizeof(double));
```

```
161         CN= (double*)  malloc(n_cuts * sizeof(double));
162         CT= (double*)  malloc(n_cuts * sizeof(double));
163
164         struct solver_ae_coefs tmp;
165
166         tmp.n_cuts=n_cuts;
167         tmp.CL_nl=CL_nl;
168         tmp.CD=CD;
169         tmp.CM=CM;
170   tmp.CN=CN;
171   tmp.CT=CT;
172         return tmp;
173         }
174
175
176 /////////////  GET G = circulation in all cuts ////////
177 void LVP_get_G(int n,double *G,  double *fi, double *B)
178         {
179                 int i,j;
180                 double *soucet =(double*) malloc(n * sizeof(double));
181                 for (i=0; i<n; i++)
182                 {
183                         for (j=1; j<n; j++)
184                         {
185                                 soucet[0]=B[0]*sin(1.0*fi[i]);
186                                 soucet[j]=soucet[j-1]+B[j]*sin((double)
                                        (1+j)*fi[i]);
187                                 G[i]=(double)2*soucet[j];
188                         }
189                 }
190   free(soucet);
191         }
192
193
194
195 /////////////  GET Ai = induced angle of attack ////////
196 void LVP_get_Ai(int n,double *Ai,  double *fi, double *B)
197   {
198     int i,j;
199     double *soucet = (double*) malloc(n * sizeof(double));
200     for (i=0; i<n; i++)
201     {
202       for (j=1; j<n; j++)
203       {
204         soucet[0]=B[0]*sin(1.0*fi[i])/sin(fi[i]);
205         soucet[j]=soucet[j-1]+(double) (1+j)*B[j]*sin((double)
              (1+j)*fi[i])/sin(fi[i]);
206         Ai[i]=soucet[j];
207       }
208     }
209    free(soucet);
210   }
211
212
213
214
215 /////////////  GET CL FROM KNOWN CIRCULATION G0 ///////////
216 void LVP_get_CL(int n, double b, double*G0, double *t,  double *CL)
217     {
```

```
218   int i;
219
220   for (i=0; i<n; i++)
221   {CL[i]=4*b*G0[i]/t[i];}
222   }
223
224
225
226
227
228 ////////////////////// THIS FILLS MATRIX ni /////////////
229 void LVP_fill_ni(int n,double *ni,double *t,double *c_lin,double b)
230         {
231   int i;
232     for (i=0; i<n; i++){
233                 *(ni +i)=((t[i])*c_lin[i] )/8/b;
234                 }
235         }
236
237
238
239 void LVP_fill_B(int n, double *B, double *ni, double *fi, double *Aa)
240   {
241     int i;
242     for (i=0; i<n; i++)
243     {
244        *(B + i) =(double)(ni[i]*sin(fi[i])*Aa[i]);
245     }
246   }
247
248 ////////////////// FILLS MATRIX S, must be square!! n*n ///////
249 void LVP_fill_S(int n, double *S, double* fi, double*ni)
250 {
251   int i,j;
252   for (j=0; j<n; j++)
253   {
254     for (i=0; i<n; i++)
255     {
256     *(S +(j*n+i))= (   (sin(fi[i])+((double)(1+j))*ni[i]) * (   sin( fi[i]*
          (double)(1+j) )   )      );
257     }
258   }
259
260 }
261
262
263 void LVP_get_COE_nl(int n,int alpha_lines,double *COE_nl,double
      *lift_surface,double *alpha,double*Aef)
264
265   {
266   double alpha_min;
267   double alpha_max;
268   int i,j;
269   alpha_min=alpha[0];
270   alpha_max=alpha[alpha_lines-1];
271
272
273   for (i=0; i<n; i++)
274     {
```

```
275     *(COE_nl +i) = LVP_interp_fast( &lift_surface[ i*alpha_lines]
            ,*(Aef+i), alpha_min,alpha_max, alpha_lines);
276     }
277
278   }
279
280
281
282
283 void LVP_get_alpha0(int n,int alpha_lines,double *alpha0,double
        *lift_surface,double *alpha)
284 // To find out zero angle of attack in all cuts. Returns array [n] in radians
285         {
286         double alpha_min;
287         double alpha_max;
288         int i,j;
289         alpha_min=alpha[0];
290         alpha_max=alpha[alpha_lines-1];
291
292         for (i=0; i<n; i++)
293                 {
294                 *(alpha0 +i) = LVP_interp_gen(alpha_lines, &lift_surface[
                        i*alpha_lines] ,alpha ,0.0);
295                 }
296
297         }
298
299 void LVP_get_Aef(int n,double *Aef,double* A,double*Ai)
300   {
301   int i;
302     for (i=0; i<n; i++)
303                 {
304     *(Aef +i) =( A[i] - Ai[i]);
305     }
306
307   }
308
309 void LVP_get_c_nl(int n ,double *c_nl ,double *CL_nl,double *Aef ,double *A0)
310   {
311   int i;
312   for (i=0; i<n; i++)
313     {
314     c_nl[i]=(CL_nl[i] / (Aef[i] - A0[i]));
315     }
316   }
317
318 void LVP_modify_CL_by_Ai(int n, double *CL, double *Ai)
319   {
320   int i;
321   for (i=0; i<n; i++){
322     CL[i]=CL[i]*cos(Ai[i]);
323     }
324   }
325
326 ////////////////////////// LAPACK SOLVER FOR GENERAL LINEAR SYSTEM OF
        EQUATION A*X=B  ; RESULT STORED IN input MATRIX B !!!
327 int LVP_solveGE(int N,double *A, double *b)
328   {
329       int nrhs = 1;
```

```
330         int lda = N;
331         int* ipiv = malloc(N * sizeof(int));
332         int ldb = N;
333         int info;
334 
335         dgesv_(&N, &nrhs, A, &lda, ipiv, b, &ldb, &info);
336 
337         if(info == 0) /* succeed */
338       printf("\n DGESV: == OK == \n");
339         else
340       fprintf(stderr, "DGESV: == FAILED ==, error: %d\n", info);
341 
342       free(ipiv);
343         return info;
344     }
345 
346 void GET_LIFT (int m,struct solver_ae_surfs AERO,struct solver_geometry
        geo,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
347 {
348 int alpha_lines=AERO.alpha_lines;
349 int lift_surf_lines= AERO.lift_surf_lines;
350 double* alpha= AERO.alpha;
351 double* lift_surface= AERO.lift_surface;
352 double* drag_surface= AERO.drag_surface;
353 double* moment_surface= AERO.moment_surface;
354 double*t=geo.t;
355 double*fi=geo.fi;
356 double*yi=geo.yi;
357 double*AG=geo.AG;
358 double b=geo.b;
359 double*Aw=AOAT.Aw;
360 double*A=AOAT.A;
361 double*Aef=AOAT.Aef;
362 double*Ai=AOAT.Ai;
363 
364 double*CL_nl=COEFS.CL_nl;
365 double*CD_nl=COEFS.CD;
366 double*CM_nl=COEFS.CM;
367 
368 
369 double dPi;
370 dPi = 4.0 * atan(1.0);
371 int i,j;
372 int n=m;
373 double cllin=(double)2*3.3643;
374 double iter_precs=0.0001;
375 double iter_diff;
376 
377 
378 
379 double *S, *B, *G0, *G1, *CL,*c_nl, *c_lin,*A0, *Aa,  *soucet, *ni;
380 
381   S = (double*) malloc(sizeof(double) * (n*n));
382   B = (double*) malloc(sizeof(double) * (n));
383   G0=(double*)   malloc(n * sizeof(double));
384   G1= (double*)  malloc(n * sizeof(double));
385   CL= (double*)  malloc(n * sizeof(double));
386   c_lin= (double*)malloc(n * sizeof(double));
387         c_nl= (double*)malloc(n * sizeof(double));
```

```
388          A0= (double*)malloc(n * sizeof(double));
389    Aa= (double*)malloc(n * sizeof(double));
390    soucet = (double*)malloc(n * sizeof(double));
391    ni= (double*)malloc(n * sizeof(double));
392
393
394 LVP_get_alpha0(n,alpha_lines,A0,lift_surface,alpha);
395
396    for (i=0; i<n; i++)
397    {
398    A[i]=Aw[i]+AG[i];
399    c_lin[i]=cllin;
400     Aa[i]=((A[i] - A0[i]));
401    }
402
403 LVP_fill_ni(n,ni,t,c_lin,b);
404
405 LVP_fill_B(n,B,ni,fi,Aa);
406
407 LVP_fill_S(n,S,fi,ni);
408
409 /////////////// GET SOLUTION OF LINEAR SYSTEM S*X=B ; SOLUTION STORED IN
        INPUT MATIX B !!!!
410 LVP_solveGE(n,S,B);
411 LVP_get_G(n,G0,fi,B);
412
413 ////////////////////////////     ITERATIVE LOOP
       ////////////////////////////////////////////////
414    int n_iter=0;
415 do
416    {
417    n_iter++;
418    LVP_get_Ai(n,Ai,fi,B);
419
420    LVP_get_Aef(n,Aef,A,Ai);
421
422    LVP_get_COE_nl(n,alpha_lines,CL_nl,lift_surface,alpha,Aef);
423
424    LVP_get_CL(n, b,G0,t, CL);
425
426    LVP_get_c_nl(n,c_nl,CL_nl,Aef,A0);
427
428    LVP_fill_ni(n,ni,t,c_nl,b);
429
430    LVP_fill_B(n,B,ni,fi,Aa);
431
432    LVP_fill_S(n,S,fi,ni);
433
434    for (i=0; i<n; i++) {G1[i]=G0[i]; }
435
436    LVP_solveGE(n,S,B);
437    LVP_get_G(n,G0,fi,B);
438
439    iter_diff=LVP_diff_2arr(n,G0,G1);
440          printf("\n%s%20.20f\n","== ITER DIFFERENCE =  ",iter_diff);
441          printf("%s%d\n","========ITERATION ======= ",n_iter);
442     printf("%s%f\n","========alfa======= ",Aw[0]);
443    ////////////////////////// END OF ITERATIVE REFINEMENT LOOP
          ////////////////////////////////////
```

```
444     }
445   while (LVP_diff_2arr(n,G0 ,G1)>iter_precs /*& n_iter<100 USE FOR HARD
        DEBUGING ONLY!!*/);
446
447   LVP_get_COE_nl(n, alpha_lines , CD_nl, drag_surface , alpha , Aef);
448   LVP_get_COE_nl(n, alpha_lines , CM_nl, moment_surface , alpha , Aef);
449
450
451
452
453   //   LVP_2D_plot(yi , CL_nl,m );
454
455   double *TMP;
456
457           TMP = (double*) malloc(sizeof(double) * (n));
458     for (i=0; i<n; i++)
459       {
460       TMP[i]=CL[i]*t[i];
461       }
462
463
464
465
466     free (TMP);
467     free(S);
468     free(B);
469     free(G0);
470     free(G1);
471     free(CL);
472     free(c_lin);
473           free(c_nl);
474           free(A0);
475     free(Aa);
476     free(soucet);
477     free(ni);
478
479
480   }
481
482
483
484   void GET_DRAG (int m,struct solver_ae_surfs AERO,struct solver_geometry
        geo,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
485   {
486
487
488   int alpha_lines=AERO.alpha_lines;
489   double* alpha= AERO.alpha;
490   double* drag_surface= AERO.drag_surface;
491   double*yi=geo.yi;
492   double*t=geo.t;
493   double*CL_nl=COEFS.CL_nl;
494   double*CD=COEFS.CD;
495
496   double *TMP;
497   TMP = (double*) malloc(sizeof(double) * (m));
498   int i;
499
500
```

```
501
502
503
504 LVP_get_COE_nl(m, alpha_lines ,TMP, drag_surface , alpha ,AOAT. Aef);
505   for (i=0; i<m; i++)
506     {
507     CD[i]= CL_nl[i]*sin(AOAT. Ai[i]) + TMP[i]*cos(AOAT. Ai[i]);
508     CL_nl[i]=CL_nl[i]*cos(AOAT. Ai[i])-TMP[i]*sin(AOAT. Ai[i]);
509
510     COEFS.CN[i]=CL_nl[i]*cos(AOAT. Aef[i]+geo.AG[i]) -
          CD[i]*sin(AOAT. Aef[i]+geo.AG[i]);
511     COEFS.CT[i]=CL_nl[i]*sin(AOAT. Aef[i]+geo.AG[i]) +
          CD[i]*cos(AOAT. Aef[i]+geo.AG[i]);
512
513
514     }
515
516 }
517
518
519 void GET_MOMENT (int m,struct solver_ae_surfs AERO,struct solver_geometry
      geo ,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
520 {
521
522
523 int alpha_lines=AERO. alpha_lines;
524 double* alpha= AERO. alpha;
525 double* moment_surface= AERO. moment_surface;
526 double*Aef=AOAT. Aef;
527 double *TMP;
528
529 TMP = (double*) malloc(sizeof(double) * (m));
530 int i;
531
532 LVP_get_COE_nl(m, alpha_lines ,COEFS.CM, moment_surface , alpha , Aef);
533
534 }
535
536
537
538 double EVAL_DRAG (int m,struct solver_ae_surfs AERO,struct solver_geometry
      geo ,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
539 {
540
541 double*yi=geo.yi;
542 double*t=geo.t;
543
544 double*CD=COEFS.CD;
545 double *TMP;
546 TMP = (double*) malloc(sizeof(double) * (m));
547 int i;
548
549
550
551         for (i=0; i<m; i++)
552                 {
553                 TMP[i]=CD[i]*t[i];
554                 }
555
```

```
556
557 return (LVP_trapz(m,yi,TMP)/-geo.S);
558
559
560         free (TMP);
561
562
563 }
564
565
566 double EVAL_LIFT (int m,struct solver_ae_surfs AERO,struct solver_geometry
        geo,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
567 {
568
569 double*yi=geo.yi;
570 double*t=geo.t;
571
572 double*CL_nl=COEFS.CL_nl;
573 double *TMP;
574 TMP = (double*) malloc(sizeof(double) * (m));
575 int i;
576
577
578
579         for (i=0; i<m; i++)
580                 {
581                 TMP[i]=CL_nl[i]*t[i];
582                 }
583
584
585 return (LVP_trapz(m,yi,TMP)/-geo.S);
586
587
588         free (TMP);
589
590
591 }
592
593
594
595 double EVAL_MOMENT (int m,struct solver_ae_surfs AERO,struct solver_geometry
        geo,struct solver_angles AOAT ,struct solver_ae_coefs COEFS )
596 {
597
598 double *TMP;
599 TMP = (double*) malloc(sizeof(double) * (m));
600 int i;
601 double bb;
602
603
604
605         for (i=0; i<m; i++)
606                 {
607                 TMP[i]=COEFS.CM[i]*geo.t[i]*geo.t[i];
608                 }
609
610
611 return (-LVP_trapz(m,geo.yi,TMP)/geo.S/geo.cSAT );
612
```

```
613
614         free (TMP);
615
616
617 }
618
619
620
621 void WRITE_SOLVER_LOG (const char *res_file,int m,struct solver_ae_surfs
        AERO,struct solver_geometry geo,struct solver_angles AOAT ,struct
        solver_ae_coefs COEFS )
622 {
623 int i;
624 double wing_area;
625
626 FILE *fp = fopen(res_file, "w");
627
628 wing_area=fabs(LVP_trapz(m,geo.yi,geo.t));
629
630
631         fprintf(fp,"%s\n","========= NLG     v. 0.35 ==========");
632   fprintf(fp,"%s\n\n","========= RESULT SUMMARY ==========");
633   fprintf(fp,"%s\t%d\n","- nodes:",m );
634         fprintf(fp,"%s\t\t%f\n","- span:",2.0*geo.b );
635   fprintf(fp,"%s\t%f\n","- wing area:",geo.S );
636         fprintf(fp,"%s\t%f\n","- cSAT :",geo.cSAT );
637         fprintf(fp,"%s\t%f\n","- ySAT :",geo.ySAT );
638         fprintf(fp,"%s\t%f\n","- cROOT :",geo.t[(int) (m/2+0.5) ] );
639   fprintf(fp,"%s\t%f\n","- cTIP :",geo.t[0] );
640
641         fprintf(fp,"\n\n\t%s\n\n","========= GEOMETRY ==========");
642
643   fprintf(fp,"\t%s\t\t%s\t\t%s\n","| y |","| t |","| ag |" );
644
645   for (i=0; i<geo.n_cuts; i++)
646     {
647           fprintf(fp,"\t%f\t%f\t%f\n",geo.yi[i],geo.t[i],geo.AG[i] );
648     }
649
650
651
652
653
654 fclose(fp);
655
656 }
```

Příloha D

Výpočet rychlostní poláry

D.1 polara.m

```
g=9.8;
source get_LD.m
source set_GEO.m
source set_sections.m
source set_lift_surf.m


nn=42;
vi=linspace(19,60,nn);
for j=1:nn
set_SRFS(vi(j));
disp(["rychlost v = ",num2str(vi(j))]);
system(["./set_lift.sh"],"sync");




load("data/data.txt");
CL=load("data/lift_line.txt");
CD=load("data/drag_line.txt");

CD=CD.+0.0042;


a(j,:)=get_LD(vi(j),CL(:,2),CD(:,2),mtow,g,rho,S);

endfor

i=0;
for j=1:nn
  if a(j,1) > 0
    i++;
    VX(i,1)=a(j,2);
    VZ(i,1)=-a(j,3);
  endif
endfor

dlmwrite("rp_pol.txt",[VX,VZ]);
clf;
```

```
41 hold on;
42 plot(VX.*3.6,VZ,"linewidth",3,"r");
43 set (gca,'fontsize',20);
44 grid on;
45 title("Rychlostni polara STD.CIRRUS 330 kg");
46 xlabel("Vx [km/h]");
47 ylabel("Vz [m/s]");
48
49 axis([0,200]);
50
51 load CIRRUS_BIKLE.txt
52 plot(3.6.*CIRRUS_BIKLE(:,1),CIRRUS_BIKLE(:,2),"linewidth",2);
53
54 load CIRRUS.txt
55 plot(3.6.*CIRRUS(:,1),CIRRUS(:,2),"g","linewidth",2);
56
57 legend(["Vypocet"],["Mereni - Bikle"],["Mereni - Johnson"])
58
59 png_path = 'png/';
60 eval(['print("' png_path 'rychl_polara.png", "-dpng");']);
```

D.2 get LD.m

```
1  function AE=get_LD(v,CL,CD,mtow,g,rho,S)
2
3  gm=1;gm_old=0;
4  j=0;
5
6     while (    j<20 | gm-gm_old>0.00000000001 )
7       j++;
8       l=2*mtow*g/rho /S/v^2 *cos(gm);
9       d=interp1(CL,CD,l,"extrap");
10      gm=atan2(d,l);
11      gm_old=gm;
12    endwhile
13
14 cl=2*mtow*g/rho /S/v^2 *cos(gm);
15 cd=2*mtow*g/rho /S/v^2 *sin(gm);
16
17 vx=v*cos(gm);
18 vz=v*sin(gm);
19
20 if ((cl <= (max(CL) )) & vz >0)
21    AE(1,1)=v;
22    AE(1,2)=vx;
23    AE(1,3)=vz;
24    AE(1,4)=cl;
25    AE(1,5)=cd;
26    AE(1,6)=gm;
27 else
28
29    AE(1,1)=0;
30          AE(1,2)=0;
31          AE(1,3)=0;
32          AE(1,4)=0;
33          AE(1,5)=0;
34    AE(1,6)=0;
```

```
35 endif
36
37 endfunction
```

D.3 set GEO.m

```
1  source bin/Bcad.m
2
3  disp("**********************************")
4  disp("Interaktivni zadani geometrie")
5  disp("      +---------->[Geometrie kridla]")
6  disp("      +   ")
7  disp("**********************************")
8  disp(" ")
9  mtow=input("zadej hmotnost  ");
10 rho=input("zadej hustotu vzduchu  ");
11 m=input("zadej pocet nodu  ");
12 b=input("zadej rozpeti [m]");
13 cr=input("zadej korenovou hloubku [m]");
14 ct=input("zadej koncovou hloubku [m]");
15 t_tw=input("zadej krouceni koncoveho profilu");
16
17 t_tw=t_tw*pi/180;
18 b=0.5*b;
19
20
21 if mod (m,2) == 0
22 m=m+1;
23 endif
24
25 for i = 1:m
26 fi(i)=(pi*i)/(m+1);
27 endfor
28
29
30 for i = 1:m
31 yi(i,1)=b*cos(fi(i));
32 endfor
33
34 for i = 1:m
35 cx(i)=( (cr-ct)/(0.5*b)*( (0.5*b)-abs(0.5*b*cos(fi(i) ) )));
36 endfor
37
38 for i = 1:m
39 t(i,1)=ct+cx(i);
40 endfor
41
42 S=abs(trapz(yi,t))
43
44
45
46 tt(1,1)=0.25*t(1,1).*cos(t_tw);
47 tt(1,2)=yi(1,1);
48 tt(1,3)=0.25*t(1,1).*sin(t_tw);
49
50 tt(2,1)=0.25*t(m/2+0.5,1);
51 tt(2,2)=yi(m/2+0.5,1);
```

```
52 tt(2,3)=0;
53
54 tt(3,1)=0.25*t(m,1).*cos(t_tw);
55 tt(3,2)=yi(m,1);
56 tt(3,3)=0.25*t(m,1).*sin(t_tw);
57
58
59 for i = 1:m
60   if yi(i,1) >= 0
61   twl(i,:)=Bcad_intersect_by_plane(tt(1,:),tt(2,:),[0,yi(i,1),0; 10,yi(i,1)
         ,0; 0,yi(i,1),10;]);
62   endif
63
64   if yi(i,1) < 0
65   twl(i,:)=Bcad_intersect_by_plane(tt(2,:),tt(3,:),[0,yi(i,1),0; 10,yi(i,1)
         ,0; 0,yi(i,1),10;]);
66   endif
67 endfor
68
69 for i=1:m AGG(i,1)=atan2(twl(i,3), twl(i,1) ); end
70
71 dlmwrite ("data/geometry/yi.txt",[yi], "delimiter", "\t",'precision', '%.32f'
       );
72 dlmwrite ("data/geometry/span.txt",[b], "delimiter", "\t",'precision', '%.32f
       ');
73 dlmwrite ("data/geometry/wing_twist.txt",[AGG], "delimiter", "\t",'precision'
       , '%.32f');
74 dlmwrite ("data/geometry/chords.txt",[t], "delimiter", "\t",'precision', '
       %.32f');
75 dlmwrite ("data/geometry/chords.txt",[t], "delimiter", "\t",'precision', '
       %.32f');
76
77
78
79 hold on;
80 axis([-(0.3*b+b) (0.3*b+b) -b b]);
81 for i = 1:m-1
82 plot([yi(i,1),yi(i+1,1)],[0.25*t(i,1),0.25*t(i+1,1)])
83 plot([yi(i,1),yi(i+1,1)],[-0.75*t(i,1),-0.75*t(i+1,1)])
84
85
86 endfor
87 plot(0,0,"@x3;KRIDLO;")
88 plot([b*cos(fi(1)),b*cos(fi(1))],[t(i,1),0])
89 plot([b*cos(fi(m)),b*cos(fi(m))],[t(m,1),0])
90
91
92 save "data/data.txt"
```

D.4 set sections.m

```
1
2
3 yi=load("data/geometry/yi.txt");
4 m=rows(yi);
5
6
```

```
for i=1:m
sections{i,1}="N/A";
endfor


sections{m/2+0.5,1}=input("korenovy profil ??  ","s");

sections{m,1}=input("koncovy profil ??  ","s");
sections{1,1}=sections{m,1};

save("prep/sections.txt","sections");
```

D.5 set AEC.m

```
function L= set_AEC(afd,airfoil,filters,RE)
filter=load(filters);
aa=dir([afd,airfoil] );
bb=struct2cell(aa);

n_files=columns(bb)-2;


for i=1:n_files
tmp(i).a=load([afd,airfoil,"/",bb{1,i+2}]);
endfor


for i=1:n_files
re_us(i)=tmp(i).a(1,1);
endfor

[a,b]=sort(re_us);

for i=1:n_files
tmp_s(i)=tmp(b(i));
endfor

for i=1:n_files

s_tab(i,1)=tmp_s(i).a(1,1);

  for j=1:rows(filter)
  tmp_i(i).a(j,1)=tmp_s(i).a(1,1);
  endfor

        tmp_i(i).a(:,2)=filter(:,1);

        tmp_i(i).a(:,3)=interp1( tmp_s(i).a(:,2),tmp_s(i).a(:,3),filter,"
            extrap");

        tmp_i(i).a(:,4)=interp1( tmp_s(i).a(:,2),tmp_s(i).a(:,4),filter,"
            extrap");

        tmp_i(i).a(:,5)=interp1( tmp_s(i).a(:,2),tmp_s(i).a(:,5),filter,"
            extrap");

endfor
```

```
41
42
43 for i=2:rows(s_tab)
44 if ( (s_tab(i,1) > RE) & (s_tab(i-1,1) <= RE) ) ix=i; endif
45 endfor
46
47 if (RE < s_tab(1,1) )
48
49   L(:,:)=tmp_i(1).a(:,:);
50
51 elseif (RE > s_tab(n_files,1))
52
53   L(:,:)=tmp_i(n_files).a(:,:);
54 else
55 disp(["interpolating at RE = ", num2str(RE)]);
56
57    for i=1:rows(filter)
58    L(i,1)=RE;
59    L(i,2)=filter(i,1);
60    L(i,3)=interp1( [tmp_i(ix-1).a(i,1),tmp_i(ix).a(i,1)], [tmp_i(ix-1).a(i,3),
          tmp_i(ix).a(i,3)] , RE);
61    L(i,4)=interp1( [tmp_i(ix-1).a(i,1),tmp_i(ix).a(i,1)], [tmp_i(ix-1).a(i,4),
          tmp_i(ix).a(i,4)] , RE);
62    L(i,5)=interp1( [tmp_i(ix-1).a(i,1),tmp_i(ix).a(i,1)], [tmp_i(ix-1).a(i,5),
          tmp_i(ix).a(i,5)] , RE);
63    endfor
64 endif
65 endfunction
```

D.6 set lift surf.m

```
1 source set_AEC.m
2
3
4 function set_SRFS(v)
5 ni=0.1555e-4
6 filters="filter/alpha_deg.txt"
7 afd=["airfoils/"];
8 filter=load(filters);
9
10
11 yi=load("data/geometry/yi.txt");
12 t=load("data/geometry/chords.txt");
13 m=rows(yi);
14 n=rows(filter);
15
16 load("prep/sections.txt");
17
18 j=0;
19 i=1;
20
21 for i=1:m
22 RE(i,1)=v.*t(i,1)./ni;
23    if (strcmp(sections{i,1},"N/A") == 0 )
24    j++;
25    yj(j,1)=yi(i,1);
26    TMP(j).L=set_AEC(afd,sections{i,1},filters,RE(i,1));
```

```
27    endif
28  endfor
29
30
31  for j=1:rows(yj)
32  LI(:,j)=TMP(j).L(:,3);
33  DI(:,j)=TMP(j).L(:,4);
34  MI(:,j)=TMP(j).L(:,5);
35  endfor
36
37  for i=1:n
38  L(i,:)=interp1(yj,LI(i,:)',yi);
39  D(i,:)=interp1(yj,DI(i,:)',yi);
40  M(i,:)=interp1(yj,MI(i,:)',yi);
41  endfor
42
43
44
45  for j=1:m
46    for i=1:n
47    LSRF(n*(j-1)+ i )=L(i,j);
48    DSRF(n*(j-1)+ i )=D(i,j);
49    MSRF(n*(j-1)+ i )=M(i,j);
50    endfor
51  endfor
52
53
54  dlmwrite ("data/lift/lift_surf.txt",[LSRF'], "delimiter", "\t",'precision', '
      %.32f');
55  dlmwrite ("data/lift/drag_surf.txt",[DSRF'], "delimiter", "\t",'precision', '
      %.32f');
56  dlmwrite ("data/lift/moment_surf.txt",[MSRF'], "delimiter", "\t",'precision',
       '%.32f');
57  dlmwrite ("data/lift/alpha.txt",[filter.*pi/180], "delimiter", "\t",'
      precision', '%.32f');
58  endfunction
```

D.7 set lift.sh

```
1  #!/bin/bash
2  TEST_PATH="/home/$(whoami)/0DP/tests/polara"
3  SOLVER_PATH="/home/$(whoami)/0DP/code/nl"
4  RESULT_PATH="/home/$(whoami)/0DP/code/nl/data/results"
5  ##################################################
6  cd $TEST_PATH
7  cp $TEST_PATH/data/lift/* $SOLVER_PATH/data/lift
8  cp $TEST_PATH/data/geometry/* $SOLVER_PATH/data/geometry
9
10 cd $SOLVER_PATH
11 rm data/results/lift_dist.txt
12 rm data/results/drag_dist.txt
13 rm data/results/moment_dist.txt
14
15 ./get_ae.out  # RUN THE SOLVER
16
17 cd $RESULT_PATH
18 cp lift_dist.txt $TEST_PATH/data/
```

```
19 cp lift_line.txt $TEST_PATH/data/
20 cp drag_dist.txt $TEST_PATH/data/
21 cp drag_line.txt $TEST_PATH/data/
22 cp moment_dist.txt $TEST_PATH/data/
23 cp moment_line.txt $TEST_PATH/data/
```