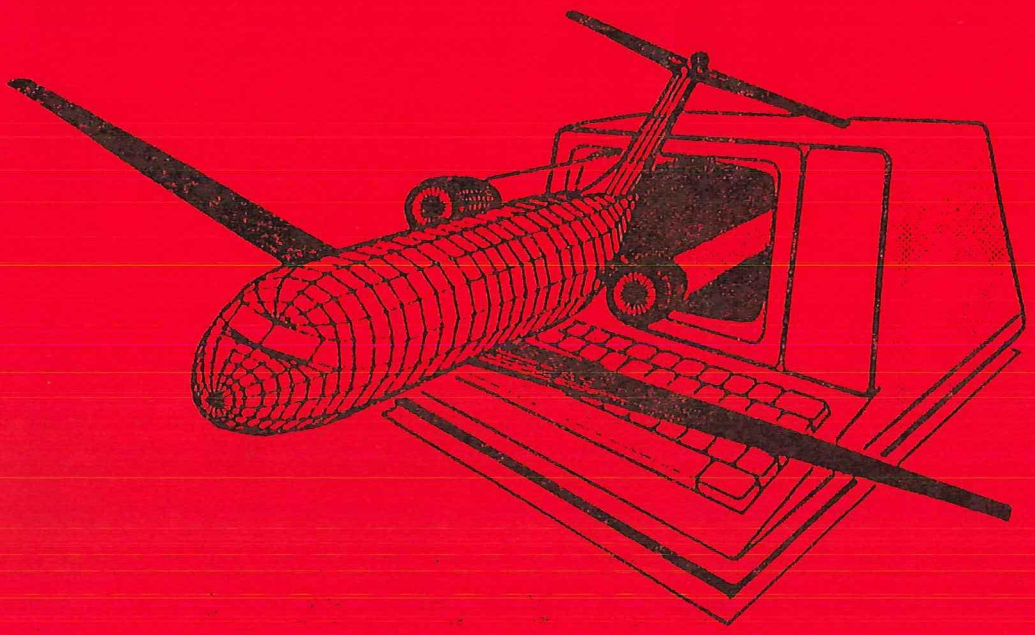
AFSTUDEERVERSLAG

TH Delft

Fokker

VISUALISEREN VAN AERODYNAMISCHE GEGEVENS



Peter van Oosterom

OKTOBER 1985

Afstudeerverslag

VISUALISEREN VAN AERODYNAMISCHE GEGEVENS

P.J.M. van Oosterom

Datum: 30 oktober 1985

De afstudeercommissie:

Prof. dr. D.J. McConalogue (THD)
Ir. D.J. Laan (Fokker)
Ir. F.H. Post (THD)
Dr. ir. H. Stienen (THD)
Ir. N. Voogt (Fokker)

Technische Hogeschool Delft
Onderafdeling der Wiskunde en Informatica
Vakgroep Informatica - Vakgebied Technische Toepassingen
Julianalaan 132
Postbus 356
2600 AJ Delft

VOORWOORD

Bij de vakgroep Informatica van de onderafdeling der Wiskunde en Informatica aan de TH Delft kan men zowel binnen de TH als extern bij een bedrijf een afstudeeropdracht uitvoeren. Als men extern afstudeert, dan kan het eigenlijke afstuderen vooraf worden gegaan door een oriëntatiefase van ongeveer één maand. In deze periode wordt de haalbaarheid van het afstudeerproject onderzocht. Op basis van het verslag van deze oriëntatiefase wordt besloten of de opdracht al of niet zal worden uitgevoerd.

In de informatica opleiding (ref. 1, 2 en 3, deze zijn bij de literatuur te vinden) ontbreekt een periode waarin men enige praktijkervaring bij een bedrijf kan opdoen. Naar mijn mening zou dit moeten gebeuren in de vorm van een stage van $\pm$ 3 maanden halverwege de opleiding. Om toch praktijkervaring op te doen wilde ik graag mijn afstudeeropdracht bij een bedrijf uit te voeren, ondanks het feit dat dit enige extra inspanning met zich mee zou brengen.

Daar mijn interesse uitgaat naar CAD/CAM en grafische toepassingen in het bijzonder, ligt het voor de hand dat mijn keuze uiteindelijk op het bedrijf Fokker is gevallen. Uit de eerste contacten met Fokker, die in augustus 1984 werden gelegd, bleek dat hier goede mogelijkheden voor een afstudeeropdracht aanwezig waren en bovendien sprak Fokker (en de vliegtuigbouw) mij sterk aan. Om enigszins wegwijs te raken in de terminologie die in de vliegtuigbouw wordt gebruikt, heb ik het college elementaire vliegtuigbouwkunde (ka21, ref. 4) gevolgd, voordat in december 1984 met de oriëntatiefase werd begonnen.

Achteraf gezien denk ik dat het een goede keuze was om bij Fokker af te studeren. Ten eerste bleek de opdracht geschikt en ten tweede werd ik uitstekend geholpen, waarvoor ik de betrokken personen zeer dankbaar ben. Op de TH waren dit: mijn afstudeerdocent de heer McConalogue en mijn begeleiders, de heren Post en Stienen en binnen Fokker waren dit: de heren Laan en Voogt. Bij Fokker zijn er nog een aantal mensen die ik ook zou willen bedanken. Mevr. de Boer van de afdeling Werving zorgde voor de administratieve zaken gedurende het afstudeerwerk. De programmeur, Dhr. Tromp was altijd aanwezig om vragen over diverse programmatuur te beantwoorden. En verder heb ik ondersteuning gekregen van de heren Deegens, Krab en Mandemaker.

In dit afstudeerverslag zal de term "postprocessor" voor een datapresentatieprogramma en de term "afstudeerder" voor verwijzingen naar mijzelf worden gebruikt.

Ik ben bijzonder veel dank verschuldigd aan de mensen die de voorlopige versie van dit afstudeerverslag hebben willen doorlezen. Zonder hun aanwijzingen en kritiek zou dit verslag nooit de vorm gekregen hebben, die het nu heeft. Naast mijn afstudeerdocent en begeleiders waren dit P.M. van Oosterom, M.W. Vos en M. Wendel.

Woudenberg (Utrecht) Peter van Oosterom
Oktober 1985

SAMENVATTING

Bij Fokker wordt binnen de groep Numerieke Aerodynamica gebruik gemaakt van een programma (GAMMA) waarmee geometrische modellen worden gedefinieerd voor aerodynamische analyse. De programma's, waarmee de drie-dimensionale stroming rond deze modellen kan worden berekend, vergen veel rekentijd en leveren zeer veel numerieke uitvoer, die in het relationele database systeem ORACLE wordt opgeslagen. Om de resultaten op een overzichtelijke wijze aan de aerodynamische ontwerper te presenteren is het wenselijk om gebruik te maken van computer graphics. De geometrische en aerodynamische gegevens moeten kunnen worden gecombineerd en gevisualiseerd op een beeldscherm. Er is een systeem ontwikkeld, genaamd VISUAL met functies waarmee op een (drie-dimensionaal) oppervlak de druk- en snelheidsverdeling van de luchtstroming zichtbaar gemaakt kunnen worden. Verder is er een functie waarmee isobaren berekend en afgebeeld kunnen worden.

INHOUD

Hoofdstuk 1 INLEIDING 1

1.1 De afstudeeropdracht 2
1.2 De oriëntatiefase 4
1.3 De resultaten van de oriëntatiefase 7
1.4 De gevolgde software engineering methodiek 9
1.5 De voorlopige planning 10

Hoofdstuk 2 SYSTEEMDEFINITIE 11

2.1 Het doel van de systeemdefinitie 12
2.2 De beschrijving van het aerodynamische ontwerpproces 13
2.3 De ontwikkelingsomgeving 19
2.4 De functionele systeemeisen 23
2.5 De randvoorwaarden 26
2.6 De belangrijkste beslissingen 29

Hoofdstuk 3 SYSTEEMONTWERP 31

3.1 Het doel van het systeemontwerp 32
3.2 De geraadpleegde literatuur voor mogelijke oplossingen 33
3.3 Het ontwerp van de datastructuur 36
3.4 De commandotaal COLAS 45
3.5 Het ontwerp van de globale systeemstructuur 47
3.6 De snelheidsaspecten 52

Hoofdstuk 4 IMPLEMENTATIE 57

4.1 Het doel van de implementatie 58
4.2 De gevolgde richtlijnen en standaarden 59
4.3 Het selecteren in het deelsysteem DIRSEL 62
4.4 Het deelsysteem 3D-FUNCTIE 65
4.5 De SET-UP van VISUAL 72
4.6 De opzet van de gebruikershandleiding 74

Hoofdstuk 5 AANBEVELINGEN VOOR VERDERE ONTWIKKELING VAN VISUAL 75

5.1 De functies CONTOUR en CARPET in het deelsysteem 3D-FUNCTIE 76
5.2 De ontwikkeling van de deelsystemen 1D- en 2D-FUNCTIE 80
5.3 Mogelijke verbeteringen van een aantal functies 84
5.4 Een aantal nieuwe functies 88
5.5 Een andere opzet van de database 91
5.6 Algemene aanbevelingen en opmerkingen 95

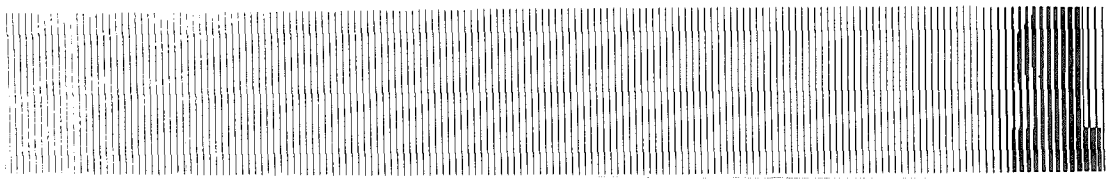
Hoofdstuk 6 EVALUATIE 97

6.1 Evaluatie van de gebruikte hulpmiddelen 98
6.2 De kwaliteit van VISUAL 102
6.3 De driehoeksrelatie "THD-Fokker-afstudeerder" 103
6.4 Enige aspecten betreffende de planning 104
6.5 Enkele kwantitatieve aspecten rondom VISUAL 106
6.6 De conclusie 107

LITERATUUR 108

APPENDIX A: Kleurenappendix 115
APPENDIX B: Tekeningen Z/W 120
APPENDIX C: Commando overzicht VISUAL 129
APPENDIX D: Verklarende woordenlijst 133
APPENDIX E: De PROGRAM MODES van VISUAL 140
APPENDIX F: Overzicht van de files voor VISUAL 145
APPENDIX G: Implementatie van de datastructuur 151
APPENDIX H: Ontwerp van de database 158

BIJLAGE: Gebruikershandleiding VISUAL



Inleiding

Deze inleiding begint in 1.1 met een letterlijke weergave van de afstudeeropdracht. Voorts zullen de doelstellingen, de gevolgde aanpak en resultaten van de oriëntatiefase worden uitgewerkt in 1.2 en 1.3. De gevolgde software engineering methodiek wordt besproken in 1.4. Tenslotte wordt in 1.5 een overzicht gegeven van de voorlopige planning zoals deze in de oriëntatiefase werd opgesteld.

1.1 - De afstudeeropdracht

Hier volgt de letterlijke weergave van het voorstel voor een afstudeeropdracht zoals deze in november 1984 door Fokker werd geformuleerd:

"Voorstel afstudeeropdracht voor Dhr. van Oosterom van de vakgroep informatica, T.H.-Delft,
uit te voeren bij de groep numerieke aerodynamica, Fokker

Opdracht
Maak aan de hand van een gebruikersspecificatie een postprocessor voor de panelenmethode, een programma voor de analyse van 3D-stromingen.

Toelichting
Binnen de groep numerieke aerodynamica wordt het geometriepakket GAMMA gebruikt voor de definitie van aerodynamische modellen. De drie-dimensionale, niet-visceuse stroming rond deze modellen kan worden berekend met de NLR panelenmethode. Om de resultaten te analyseren is het nodig om geometrie gegevens en aerodynamica gegevens te combineren. Gezien de zeer grote hoeveelheden gegevens is het zeer wenselijk gebruik te maken van computer graphics. De postprocessor moet het gereedschap vormen waarmee de analyse efficiënt kan plaatsvinden. In de gebruikersspecificatie zullen de aan de postprocessor te stellen eisen, nader worden uitgewerkt.

Aspecten die aan de orde komen zijn :

- Databeheer: opzetten van een consistente opslag van de gegevens, mogelijk gebruik makend van een relationeel database management systeem.

- Computer-graphics: toepassen van computer-graphics technieken met behulp van het grafisch pakket GINO op Tektronix 4115 beeldschermen.

- Software engineering aspecten: systeemdefinitie, systeemontwerp, implementatie."

Tot zover de letterlijke weergave van de opdracht.

Opmerking: Fokker bestaat uit de volgende vier takken:

- Personeels en sociale zaken;
- Technologie en ontwikkeling (TO);
- Marketing en verkoop;
- Financiën en ekonomie.

Hierboven staat de Raad van Bestuur. De groep Numerieke Aerodynamica is een onderdeel van de afdeling TOAP; hierin staan de letters "AP" voor Aerodynamica en Prestaties.

De in de opdracht genoemde gebruikersspecificatie is een document van D. Laan; ref. 5. Deze gebruikersspecificatie is het resultaat van de probleemdefinitie (zie 1.4) welke door Fokker werd opgesteld. De doelstellingen van de probleemdefinitie zijn o.a.: het verkrijgen van een goed inzicht in het probleem en het formuleren van de eisen waaraan een te ontwikkelen systeem moet voldoen.

In het kort geformuleerd luidde de afstudeeropdracht: "ontwikkel een systeem waarmee aerodynamische gegevens gevisualiseerd kunnen worden". Het systeem moest natuurlijk ook een naam krijgen en niets lag meer voor de hand om hieraan de naam VISUAL te geven.

1.2 - De oriëntatiefase

Zoals eerder vermeld werd het eigenlijke afstudeerwerk voorafgegaan door een oriëntatiefase. De doelstellingen van deze oriëntatiefase waren:

1) Het vertrouwd raken met de industriële organisatie van Fokker;
2) Een goed beeld krijgen inzake de wensen van de gebruikers;
3) Een overzicht krijgen van de aanwezige hard- en software die ter ondersteuning beschikbaar is.
4) Enig inzicht krijgen in het niveau en de omvang van het werk dat verricht moet worden tijdens de afstudeeropdracht;

Dit alles had uiteindelijk als voornaamste doel: om over de haalbaarheid van het afstudeerproject te kunnen beslissen. Voorts ook om de opdracht meer gedetailleerd te formuleren.

Tijdens de oriëntatiefase is de gebruikersspecificatie als uitgangspunt gekozen; ref. 5. Door middel van formele en ook vooral informele interviews met de toekomstige gebruikers is getracht de wensen uit de gebruikersspecificatie te verduidelijken, uit te breiden en van een prioriteits-volgorde te voorzien. Naast deze interviews zijn er ook een groot aantal documenten doorgenomen, waarmee de hierboven genoemde doelstellingen zijn getracht te bereiken. De belangrijkste documenten zijn geweest:

- Program Panel (version 1.0) user guide; ref. 8. Dit is het programma waarvoor VISUAL oorspronkelijk als postprocessor moest fungeren. Later is de taak van VISUAL wat algemener geworden, wat wil zeggen, dat VISUAL ook als postprocessor voor andere, soortgelijke programma's voor analyse van stromingen is te gebruiken. Ook voor windtunnelmetingen en voor vliegproeven moet VISUAL als postprocessor kunnen dienen.

- Documentatie over VAX/VMS: de primer, de user introduction en de editors EDT en SED; zie respectievelijk ref. 9, 10, 11 en 12.

- Documentatie over het relationele database systeem ORACLE; ref. 13 en 14 en documentatie over databases in het algemeen; ref. 15, 16, 17 en 18. Verder zijn er nog een aantal documenten doorgenomen over het database ontwerp, welke niet in de lijst met referenties zijn opgenomen, geschreven door M. Cornelissen en F. Tromp.

- Documentatie over het grafische pakket GINO-F; ref. 19 en 20 en documentatie over de verschillende grafische terminals; ref. 21 tot en met 24.

- Documentatie over het aerodynamische ontwerpproces; ref. 25 en 26. Dit betreft het grotere geheel waarin ook VISUAL een rol zal moeten gaan spelen.

Tenslotte zijn er, naast het afnemen van interviews en het doornemen van documentatie, ook nog een aantal programma's bekeken welke als mogelijke inspiratiebron voor VISUAL zouden kunnen dienen. De volgende programma's werden bekeken:

- GAMMA: Geometry Analysis & Manipulation for Multiple Application. Dit is een interactief grafisch pakket waarmee geometrie gecreëerd en gewijzigd kan worden. Bovendien beschikt GAMMA ook over een aantal visualisatie functies.

- CODE: Een interactief grafisch programma voor het ontwerpen van 2D-profielen.

- CATIA: Een 3D CAD/CAM pakket.

- PATRAN: Een systeem voor modelling, eindige elementen generatie, analyse en voor resultaten evaluatie, postprocessing. Dit programma was een zeer goed voorbeeld voor VISUAL.

De eerste twee genoemde programma's, GAMMA en CODE, zijn door Fokker zelf ontwikkeld en de laatste twee, CATIA en PATRAN, zijn onder licentie bij Fokker in gebruik.

Om een beter inzicht te krijgen in de omgeving waarin VISUAL moest gaan functioneren, heb ik tijdens de oriëntatiefase de ontwerpcyclus eenmaal zelf doorlopen. Globaal ziet dit er als volgt uit:

1) Definieren van een geometriemodel met behulp van GAMMA; ref. 27.

2) Voorbereiden van de invoer voor de stromingsberekeningen met behulp van een preprocessor genaamd FOKSYS. De invoer bestaat uit geometrie en aanstromingscondities, zoals machgetal, invalshoek, sliphoek, etc.

3) Uitvoeren van de stromingsberekeningen, bijvoorbeeld met de NLR panelenmethode; ref. 8.

4) Opslaan van de uitvoer van de stromingsberekeningen. De uitvoer wordt in files, via bepaalde bewerkingen met het programma 0215DB, opgeslagen.

5) Verrichten van enige postprocessing met behulp van GAMMA.

Opmerking : Dit is de situatie zoals deze was in december 1984. Het is de bedoeling dat in de nieuwe situatie de geometrie en de uitvoer van de stromingsberekeningen, zie stap 4, in de ORACLE database zullen worden opgeslagen. VISUAL zal dan de postprocessing, stap 5, gaan verrichten.

1.3 - De resultaten van de oriëntatiefase

De resultaten van de oriëntatiefase zijn beschreven in ref. 28. Elke paragraaf in dit document is opgesplitst in twee delen. In het eerste deel staat letterlijk de overeenkomstige paragraaf uit de gebruikersspecificatie en in het tweede deel staan de uitgewerkte toelichtingen.

Na de oriëntatiefase was het duidelijk geworden wat de afstudeeropdracht precies inhield en in welke omgeving het project uitgevoerd moest worden en welke middelen beschikbaar waren. Het was tijdens de oriëntatiefase ook duidelijk geworden dat het te veel en ook wetenschappelijk gezien minder interessant werk zou worden om alle gebruikerswensen, zoals deze tijdens de probleemdefinitie naar voren waren gekomen te implementeren. Er zijn toen een aantal aspecten van VISUAL aangegeven waarop de nadruk gelegd zou kunnen worden; ref. 29. De volgende zeven aspecten zijn genoemd:

1) Visualisatie aspecten. Het opzetten van een geometrisch model dat speciaal geschikt is voor het snel genereren van afbeeldingen van dat model.

2) Commandotaal aspecten. Belangrijke punten hierbij zijn onder andere: kiezen van geschikte commando's, gebruikersvriendelijkheid, robuustheid, opslag van de door de gebruiker gegeven commando's, structuur van de commandotaal, etc.

3) Globale opzet. Dit betreft het specificeren, ontwerpen en implementeren van het hoofdprogramma. Hierbij komt de algemene systeem opzet van VISUAL heel duidelijk naar voren.

4) Uitwerken van één specifieke ingewikkelde functie. Hierbij wordt bijvoorbeeld gedacht aan de functies "heuvellandschapwerking" en "tekenen van de drukverdeling op de contouren van een segment in 3D aanzicht"; ref. 28.

5) Database ontwerp. Dit houdt in het bedenken en opzetten van een database. Hiervoor is een diepgaande informatieanalyse nodig, niet alleen bij de afdeling TOAP, Aerodynamica en Prestaties, maar ook bij de afdeling TOVO, Voorontwerp en eventueel nog andere afdelingen.

6) Uitwerken van een groep van functies. Hiervoor zouden bijvoorbeeld in aanmerking komen de deelsystemen 2D-FUNCTIE en 3D-FUNCTIE. Er kan gebruik worden gemaakt van de overeenkomsten die zich binnen één deelsysteem voordoen.

7) Overdraagbaarheid. Dit is een gecompliceerd probleem, waarbij onderscheid gemaakt moet worden tussen machine-onafhankelijkheid en apparaat- (of terminal-) onafhankelijkheid.

Er werd besloten (op een bijeenkomst van de begeleiders in januari 1985) dat de aspecten 1, 3 en 6 extra aandacht zouden krijgen tijdens het afstudeerwerk. Om later enig werkend resultaat te krijgen kon er niet aan ontkomen worden een globale systeem opzet te ontwikkelen (aspect 3). Om de opdracht voldoende wetenschappelijke kwaliteit te geven zou het visualisatie aspect diep moeten worden uitgewerkt (aspect 1). En in verband met het visualisatie aspect lag het voor de hand om dan ook het deelsysteem 3D-FUNCTIE uit te werken (aspect 6). Voorts zouden in verband met de gewenste gebruikersvriendelijkheid, iets dat bij de ontwikkeling van VISUAL zeer zwaar heeft gewogen, de directory (of overzichts) functie's geïmplementeerd worden. De gebruiker kan hiermee, alvorens een selectie te maken, eerst een overzicht van beschikbare modellen en gegevens opvragen.

1.4 - De gevolgde software engineering methodiek

Het gebruik van een software engineering methodiek vergroot de kans op een goed eindresultaat. De ontwikkeling van VISUAL moest gebeuren aan de hand van een bij Fokker verplichte software engineering standaard, ref. 6. Deze standaard lijkt erg veel op de bekende PRODOSTA methodiek, ref. 7.

De methodiek onderscheidt zeven fasen in de levenscyclus van software en geeft aan wat er moet gebeuren tijdens elk van deze fasen. Elke fase wordt afgerond met een document dat is samengesteld aan de hand van een checklist. De volgende zeven fasen worden er onderscheiden:

1) Probleemdefinitie;
2) Systeemdefinitie;
3) Systeemontwerp;
4) Detailontwerp en implementatie;
5) Integratie en test;
6) Acceptatie;
7) Gebruik.

De eerste fase was reeds door Fokker verricht voor de aanvang van de afstudeeropdracht (zie 1.1), maar is verder uitgewerkt door de afstudeerder tijdens de oriëntatiefase van de afstudeeropdracht; zie 1.2. De fasen 4 en 5 zijn wegens de relatief kleine omvang van het project samengenomen. De gebruikershandleiding is een onderdeel van de documentatie die hoort bij fase 5. De acceptatie en het gebruik (onderhoud en uitbreiding) vielen buiten het kader van de afstudeeropdracht.

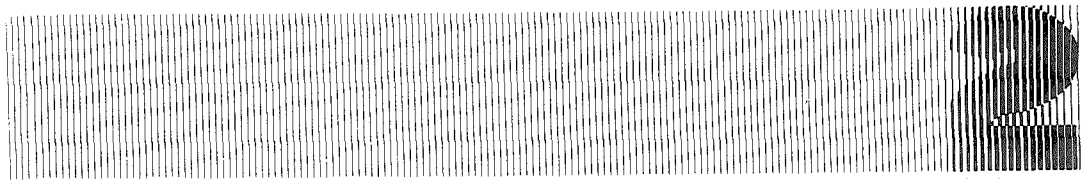
1.5 - De voorlopige planning

Bij de planning werd uitgegaan van de zeven fasen uit de levenscyclus van software zoals deze in 1.4 zijn genoemd. De onderstaande planning werd gemaakt voordat met de afstudeeropdracht werd begonnen en bestrijkt een periode van zes maanden, namelijk van 1 januari tot en met 28 juni 1985. Hoewel de planning vrij globaal was, kon deze bij de evaluatie (zie 6.4) toch worden gebruikt om een vergelijking met de werkelijke tijdsbesteding te maken. In tabel 1.1 is de voorlopige planning samengevat.

Fase	datum begin	datum einde	aantal weken
2 Systeemdefinitie	1 januari 1985	3 februari 1985	5
3 Systeemontwerp	4 februari 1985	17 maart 1985	6
4 Implementatie en 5 Test	18 maart 1985	26 mei 1985	10
Afstudeerverslag	27 mei 1985	28 juni 1985	5

Tabel 1.1 - De voorlopige planning

Opmerking : Later is er wel binnen één fase een uitwerking van de planning gemaakt; zie bijvoorbeeld het systeemontwerp in hoofdstuk 3 en in ref. 37.



Systeemdefinitie

Nadat in de volgende paragraaf eerst het doel van de systeemdefinitie (2.1) duidelijk gemaakt zal worden, zal in de paragrafen 2.2 en 2.3 de algemene probleemstelling behandeld worden. Dit gebeurt door in 2.2 het aerodynamisch ontwerpproces te beschrijven en dan in 2.3 aan te geven waaruit de ontwikkelingsomgeving bestond. Daarna zullen in 2.4 de functionele eisen voor het systeem weergegeven worden. De randvoorwaarden, waaraan het systeem moest voldoen, staan vermeld in 2.5. Tenslotte zullen in 2.6 de belangrijkste beslissingen die tijdens de systeemdefinitie genomen zijn, besproken worden.

2.1 - Het doel van de systeemdefinitie

In deze fase is de gebruikersspecificatie aangevuld tot een samenhangend programma van eisen voor het te ontwikkelen systeem. Verder moesten in deze fase alle onduidelijkheden omtrent het beschikbaar zijn van de diverse hulpmiddelen opgelost worden, zodat beslist kon worden welke middelen wel en welke niet gebruikt zouden gaan worden.

Werden in de gebruikersspecificatie nog een groot aantal gewenste functies genoemd, in de systeemdefinitie werden alleen die functies opgenomen, waarvan het de bedoeling was dat deze in het kader van de afstudeeropdracht uitgevoerd zouden worden. In ref. 29 was al enigszins aangegeven welke functies dat zouden zijn, maar in 2.4 van dit afstudeerverslag zijn alle functies, die uitgevoerd zouden moeten worden, aangegeven.

Tenslotte moesten tijdens de systeemdefinitie de randvoorwaarden opgesteld worden. Dit betreft onder andere zaken als overdraagbaarheid, snelheid, gebruikersvriendelijkheid, uitbreidbaarheid (onderhoudbaarheid), etc.

Vanuit het programma van eisen, dat wil zeggen de functionele systeemeisen aangevuld met de randvoorwaarden, moest in de volgende fase (het systeem ontwerp) verder gewerkt worden, zodat een geschikte oplossing werd gecreëerd. Het resultaat van de systeemdefinitie werd vastgelegd in een document; ref. 30.

2.2 - De beschrijving van het aerodynamische ontwerpproces

In deze paragraaf zal het numerieke aerodynamische ontwerpproces, voor zover het van belang was voor VISUAL, beschreven worden. Hierna zal de "ruggegraat" van dit proces, namelijk de wijze waarop de geometrie gerepresenteerd wordt, beschreven worden.

Het ontwerpen van vliegtuigen is een multi-disciplinaire activiteit, waarbij de volgende disciplines worden geïntegreerd: configuratieontwerp, aerodynamica, prestaties en economie, voortstuwing, constructie, stabiliteit en besturing, boordsystemen, fabricage- en onderhouds- aspecten (ref. 26). Bij de aerodynamica afdeling worden naast onder andere windtunnelproeven ook een aantal programma's gebruikt voor drie-dimensionale stromingsberekeningen. Deze twee methoden om stromingsgegevens te verkrijgen moeten (voorlopig) complementair gezien worden.

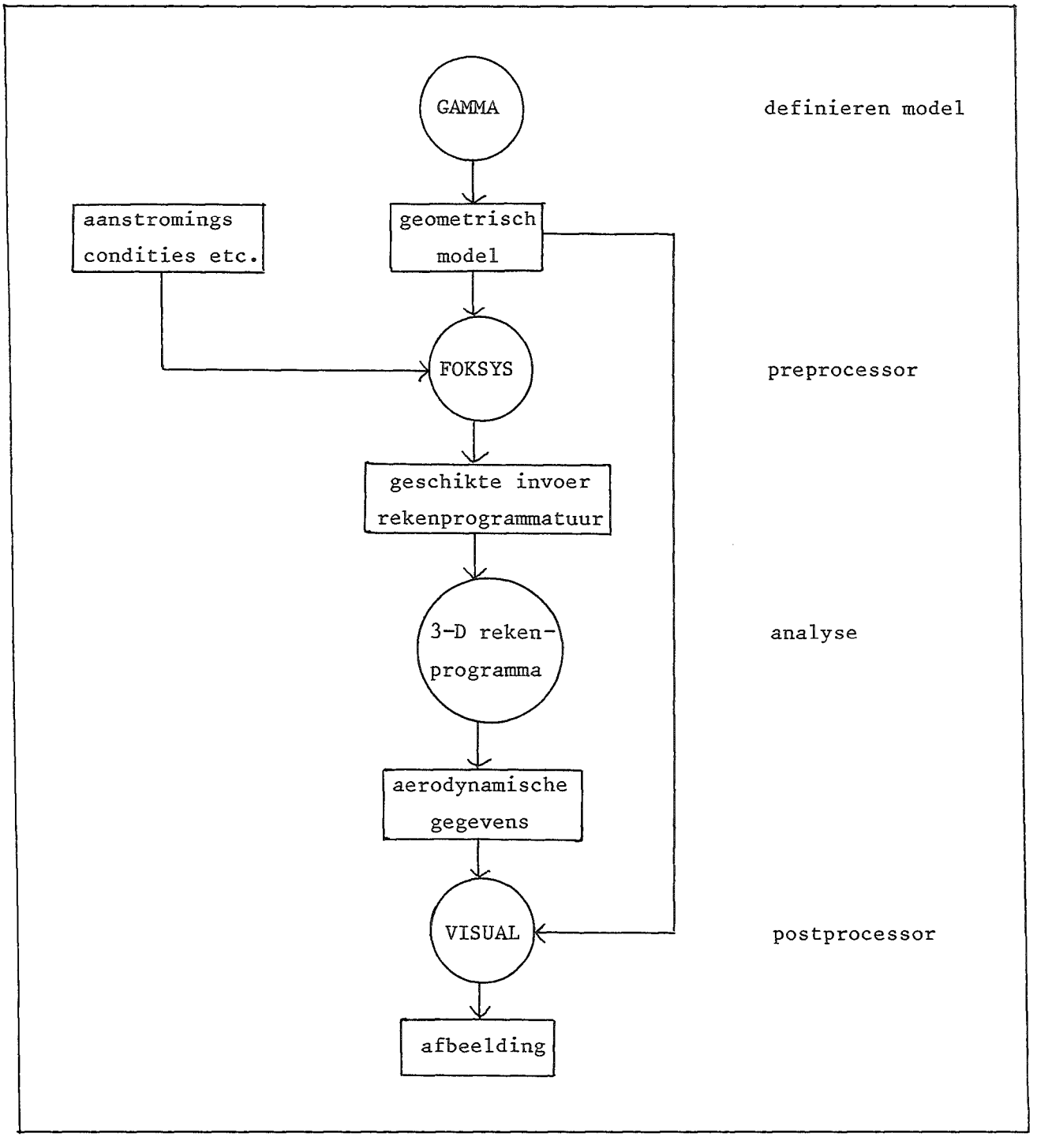
Het aerodynamische ontwerpproces begint bij het definiëren van een model. Meestal is dit een variant op een al bestaand model, maar soms wordt er ook een geheel nieuw model gedefiniëerd. Dit gebeurt met GAMMA. Het is echter ook mogelijk dat dit (gedeeltelijk) met een ander pakket zoals CATIA of CADAM gebeurt. Wel moet in zo'n geval via een interface met GAMMA het model op eenzelfde wijze gerepresenteerd worden.

Nadat het model gereed is, moet het met enkele globale gegevens gecombineerd worden tot een geschikte invoer voor de 3D stromingsberekenings programma's. Deze gegevens bestaan onder andere uit aanstromingscondities welke gekenmerkt worden door: het machgetal, de invalshoek alfa, de sliphoek beta, etc. Dit alles gebeurt met het programma FOKSYS. Hierna moeten de 3D stromingsberekeningen worden uitgevoerd, waarvoor verschillende programma's beschikbaar zijn. De belangrijkste zijn:

- NLR panelenmethode, subsone compressibele panelenmethode;
- XFLO22, transone finite difference methode;
- NLRAERO, subsone vortex lattice methode.

Deze programma's worden in "batch" gedraaid omdat ze vooral voor een meer complexe geometrie veel rekentijd nodig hebben. De uitvoer van deze stromingsberekeningen moet met een speciaal programma bewerkt worden voordat deze wordt opgeslagen. Voor de NLR panelenmethode is dit het programma 0215DB.

Tenslotte worden de berekende en opgeslagen aerodynamische resultaten bekeken. De aerodynamische gegevens bestaan onder andere uit een drukverdeling en een snelheidsverdeling van de luchtstroming. Om conclusies te trekken omtrent de aerodynamische eigenschappen van het model is er een speciaal programma (de postprocessor VISUAL) ontwikkeld, waarmee de resultaten gevisualiseerd kunnen worden. In figuur 2.1 is het aerodynamische ontwerpproces met behulp van 3D rekenprogrammatuur schematisch weergegeven.



Figuur 2.1 - Het aerodynamische ontwerpproces

De opbouw van de geometrie

Hoe het model met GAMMA wordt opgebouwd en welke relaties er tussen de verschillende elementen aanwezig zijn, zal nu beschreven worden. Voor VISUAL is dit van belang omdat dit de invoer betreft, evenals de aerodynamische gegevens die hieraan gerelateerd zijn.

De beschrijving van een object, bijvoorbeeld een vliegtuig of een onderdeel hiervan, zoals dit nu gebeurt lijkt enigszins op een methode van solid modelling die in de literatuur grensbeschrijving (ref. 33) wordt genoemd. Bij een grensbeschrijving wordt het object gerepresenteerd door (vaak platte) vlakken (faces) die elkaar alleen in gemeenschappelijke zijden (edges) en hoekpunten (vertices) snijden. Een zijvlak wordt gerepresenteerd door de vergelijking van het oppervlak waarin het ligt en de begrenzende zijden. Een zijde wordt ge- representeerd door de vergelijking van de kromme waarop ze ligt en de begrenzende hoekpunten. Een hoekpunt wordt gerepresenteerd door de (X,Y,Z)-coördinaten. We kunnen de volgende drie typen van gegevens onderscheiden:

- Topologische gegevens: dit zijn kwalitatieve gegevens die bestaan uit de relaties tussen zijvlakken, zijden en hoekpunten (deze worden weergegeven als verwijzingen van de elementen naar elkaar).

- Geometrische gegevens: dit zijn kwantitatieve gegevens, hierbij moet men denken aan vergelijkingen van oppervlakken en krommen en de coördinaten van hoekpunten.

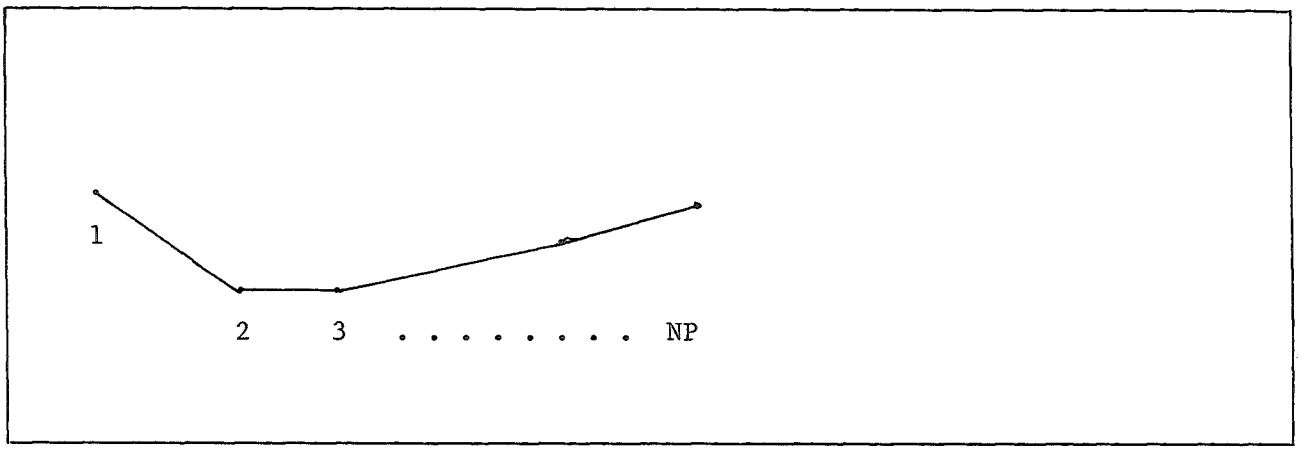
- Overige gegevens: dit zijn (kwantitatieve) gegevens die niet (direct) de vorm van een bepaald object beschrijven maar die meestal wel aan de vorm zijn gerelateerd; bijvoorbeeld aerodynamische gegevens.

De methode van objectbeschrijving in GAMMA is echter geen solid modelling, hierbij is nl. altijd sprake van een complete beschrijving van massieve drie-dimensionale objecten en GAMMA vertoont op dit gebied enige tekortkomingen; bijvoorbeeld gaten in het oppervlak van het model. Voor VISUAL is het feit dat de modellen doorgerekend zijn met behulp van een programma voor drie-dimensionale stromingsberekeningen (NLR panelenmethode, XFL022, NLRAERO) een redelijke garantie dat de tekortkomingen niet te groot zijn.

De manier van objectbeschrijving in GAMMA is het uitgangspunt voor de postprocessor VISUAL doordat de aerodynamische gegevens hieraan gekoppeld

zijn. Dit wil niet zeggen dat VISUAL precies dezelfde manier van objectbeschrijving als GAMMA moet gebruiken. Wel moet er een conversie mogelijk zijn van "GAMMA naar VISUAL". VISUAL is een postprocessor, dus in dit geval is het produceren van een afbeelding van een object en overige gegevens, die hieraan gerelateerd zijn, belangrijk. Het zal duidelijk zijn dat bepaalde manieren van representatie (objectbeschrijving) hiervoor beter geschikt zijn dan anderen. Hier zal in de ontwerpfase rekening mee moeten worden gehouden.

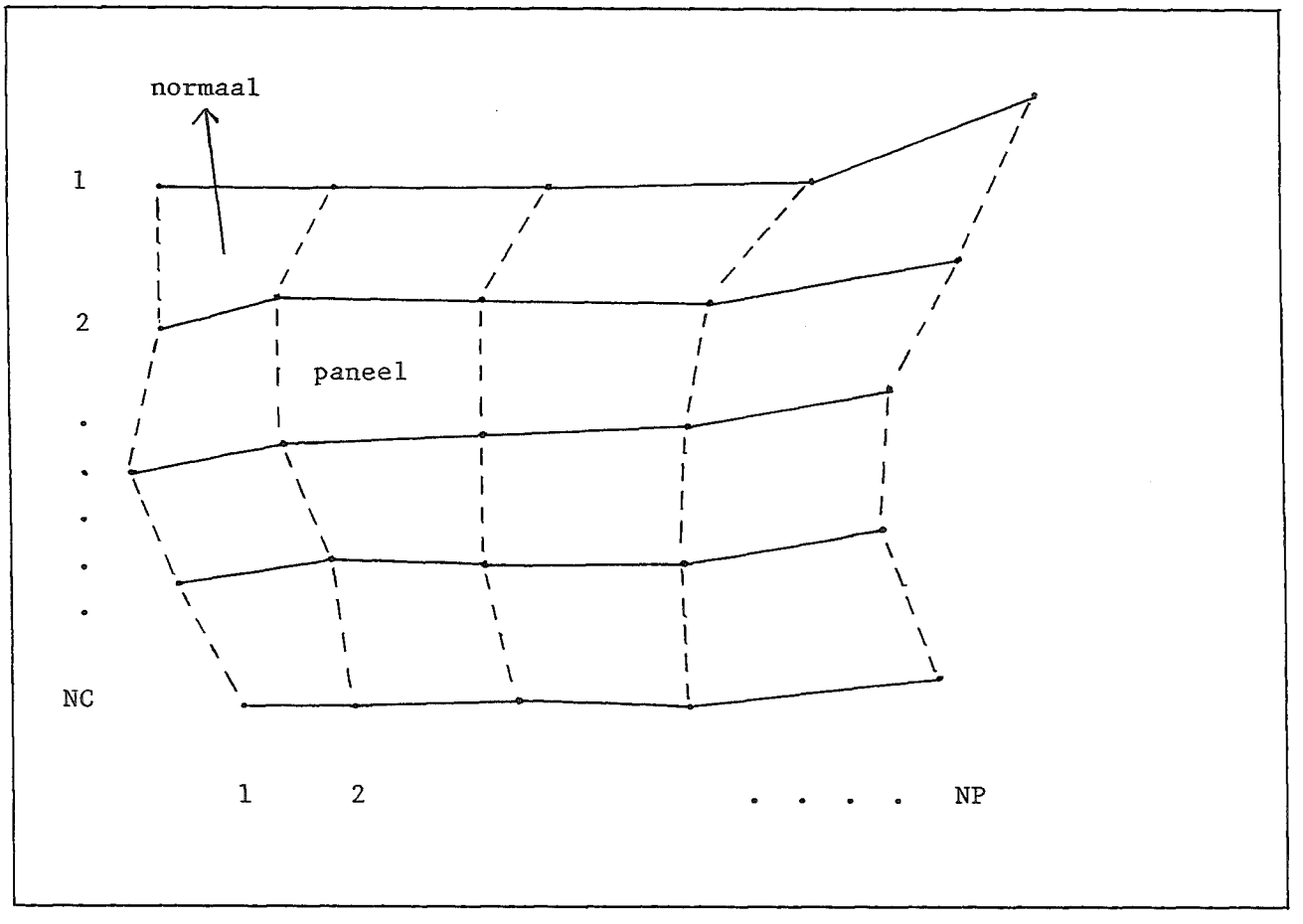
Het basiselement van de geometrie is een punt, deze bestaat uit drie coördinaten. Punten worden door lijnstukken verbonden en vormen op deze wijze een contour. Deze contour wordt gedefinieerd door NP (geordende) punten; zie figuur 2.2. Binnen een contour heeft elk punt een index (die ligt tussen 1 en NP met de grenzen inbegrepen).



Figuur 2.2 - Een contour gedefinieerd door NP punten

Een aantal contouren (die elk door NP punten worden gedefinieerd) vormen samen één segment door de punten met een gelijke index te verbinden; zie figuur 2.3. Het segment in de figuur wordt gedefinieerd door NC contouren en dus door NP * NC punten. De oorspronkelijke contouren worden ook wel primaire of hoofdcontouren genoemd. De dwarsverbindingen (in de figuur gestippeld aangegeven) tussen de oorspronkelijke contouren vormen de secundaire of dwarscontouren. Punten en contouren zullen in de database per segment opgeslagen worden en moeten dan ook in één keer voor het hele segment worden opgehaald uit de database. In één segment bevinden zich (NC-1) * (NP-1) panelen die worden gedefinieerd door de vier hoekpunten. Aangezien vier punten niet in één vlak behoeven te liggen, is één paneel niet per definitie vlak. Er

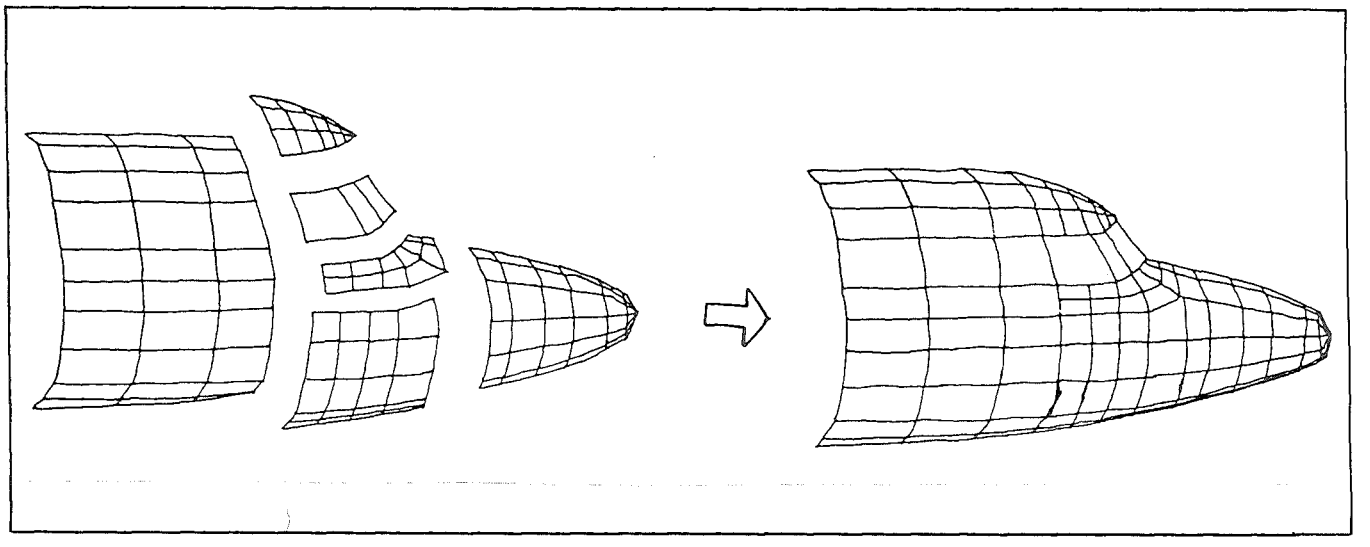
wordt aan het paneel een "normaalvector" toegevoegd, welke berekend wordt via twee vlakke driehoeken die het paneel benaderen.



Figuur 2.3 - Een segment gedefinieerd door NC * NP punten

Door een aantal segmenten, die logisch gezien bij elkaar horen, samen te nemen ontstaat een part. Voorbeelden van parts zijn: een cockpit (zie figuur 2.4), een vleugel, een motorgondel, etc.

Een group is een verzameling parts (en dus ook een verzameling segmenten), die als één geheel worden doorgerekend voor een bepaalde stromingsconditie. Een group kan bijvoorbeeld een kompleet vliegtuig zijn. Een configuratie tenslotte is een verzameling van groups, parts en segmenten die bij elkaar horen.



Figuur 2.4 - Segmenten worden samengevoegd tot één part

De aerodynamische gegevens zijn in twee categorieën te verdelen, t.w.:

- de snelheidsverdelingen: één vector per paneel;
- de drukverdelingen : één waarde per paneel.

De overige aerodynamische gegevens, zoals lift-, weerstands,- en momenten-verdeling, waarbij mogelijk één waarde per contour, segment, part of configuratie aanwezig is, worden alleen in die functies gebruikt die voorlopig nog niet geïmplementeerd worden.

Alle aerodynamische gegevens die bij één berekening (of meting) behoren, worden één datapoint genoemd. Bij het selecteren van aerodynamische gegevens, wordt alles wat bij één datapoint behoort in één keer geselecteerd.

2.3 - De ontwikkelingsomgeving

De ontwikkelingsomgeving van VISUAL valt op te splitsen in een hardware en een software omgeving.

De hardware omgeving omvat de volgende drie categoriën apparatuur:

- computersystemen: DEC-10/20, VAX, IBM-3081. De VAX en de DEC-10/20 worden gebruikt voor interactief werk en voor de kleinere batch jobs; bijvoorbeeld minder dan één uur CPU-tijd op de IBM-3081. De IBM-3081 wordt gebruikt voor grote batch jobs; bijvoorbeeld rekentijd van ongeveer één uur CPU-tijd of meer. VISUAL moest ontwikkeld worden op een VAX onder het operating system VAX/VMS.

- terminals: VT132, Datamedia DT80/1 (alfanumerieke beeldscherm terminals, VT100-achtig), Decwriter III, Decwriter IV (alfanumerieke papier terminals), Tektronix 4014-1 (line drawing display, storage tube), Tektronix 4114B (line drawing display, storage tube met refresh mogelijkheden), Tektronix 4115B (kleuren raster display).

- Hardcopiers/plotters: Tektronix 4611 (aangesloten op de Tektronix 4014-1 en de Tektronix 4114B), Tektronix 4691 (aangesloten op de Tektronix 4115B), Benson 1302 plotter, zowel met smalle als brede drum), Kongsberg (tekenmachine, net als de Benson een pen plotter).

De software omgeving

Nu volgt een overzicht van de ondersteunende software op de VAX.

GINO-F (Mark 2.7 a): grafisch pakket van procedures (routines), die het schrijven van (interactieve) grafische toepassings programma's vergemakkelijkt. Voor de volgende zaken zijn procedures aanwezig:

- Tekenen van rechte lijnen, cirkelbogen en karakters
- Initialisatie en apparaat besturing
- Windowing, Viewing
- Transformaties (translatie, rotatie, schalen en shearing)
- Picture segmenten

- Kleur definitie
- Polygon definitie en area filling

ORACLE: Relationeel database management systeem (dat gebruikt zal gaan worden voor de opslag van o.a. geometrische en aerodynamische gegevens).
De volgende functies zijn aanwezig (Structured Query Language):

- Select
- Delete
- Update
- Insert

De postprocessor VISUAL zal niet rechtstreeks met de database communiceren maar via de routines uit DIADBF.

DIADBF: Routines voor interactie met de database. (Een schil om ORACLE: bevat routines van een "hoger" niveau, gericht op het toepassingsgebied.)
Dit pakket bevat de volgende routines:

- Voor elke tabel (relatie) uit de database:
  * Select
  * Delete
  * Update
  * Insert
- Directory functies voor het produceren van overzichten

DIALIB: Routines voor de alfanumerieke communicatie met de gebruiker en voor het wegschrijven van alfanumerieke informatie naar files. (Verder zullen er diverse algemene routines in dit pakket komen.)
De volgende routines zijn aanwezig:

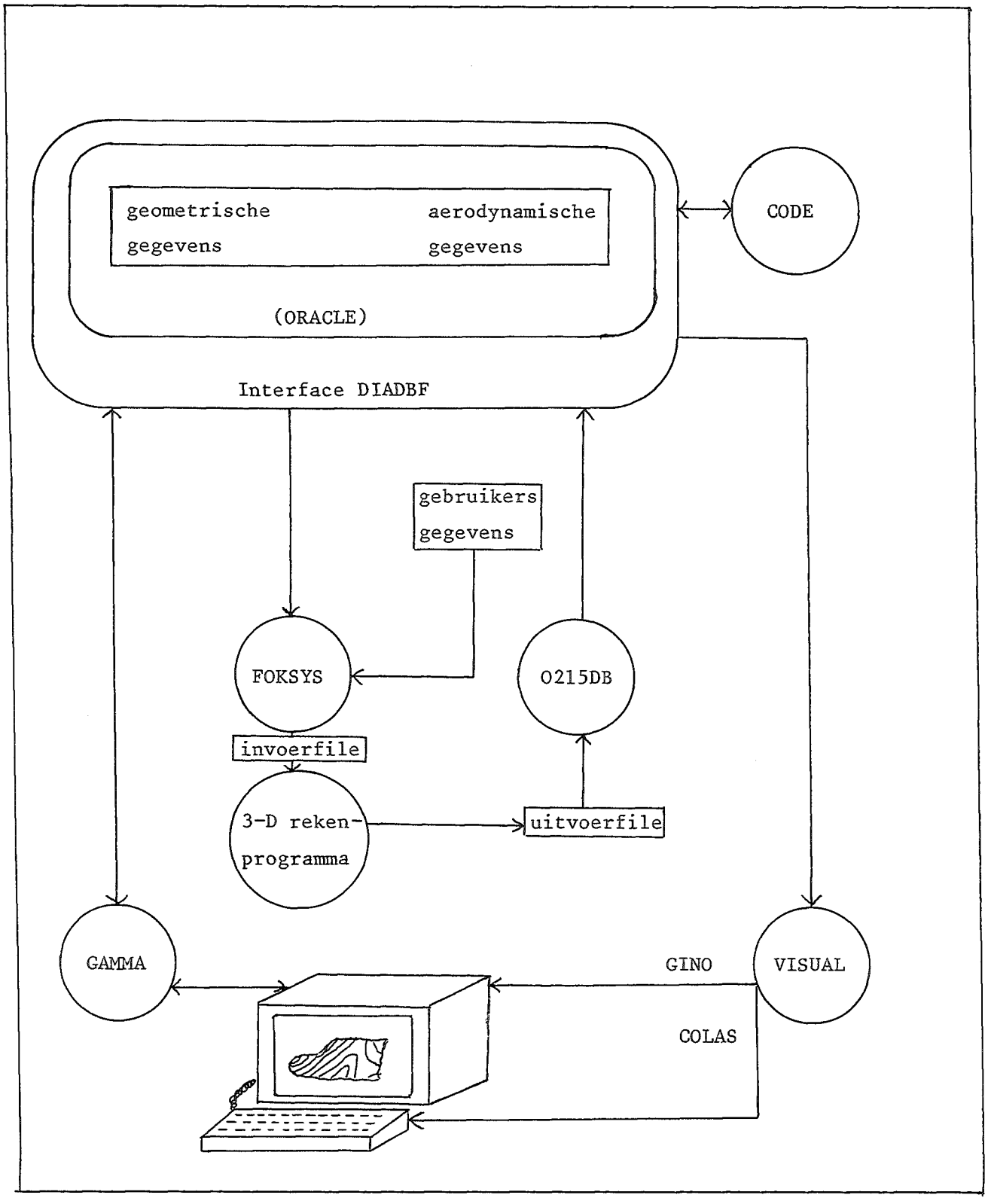
- Schrijven van een vraag naar en lezen van het antwoord van de terminal
- Herhalen van een vraag (en geven van het vorige antwoord)
- Lezen van één of meer regels van de terminal
- Bijhouden van zogenaamde logfile
- Etc.

COLAS (COmmand LAnguage System): COLAS is een interface, bestaande uit een aantal routines, tussen een gebruiker en een applicatie-programma; ref. 31.

Het systeem is gebaseerd op commandosets, die door aanroep van een procedure geactiveerd worden, door het applicatie programma. Een gebruiker mag afkortingen van een commando geven, zolang deze binnen de actieve commandoset uniek is. COLAS zorgt dat er meldingen worden gegeven en vraagt om een parameterwaarde als deze niet opgegeven is en als er geen default waarde bestaat. Een uitgebreidere brschrijving van COLAS is te vinden in 3.4.

FIPS (Flexible Image Processing System): Een systeem (uit de omgeving van de beeldverwerking; ref. 32 en 33) waarin diverse applicatie programma's draaien. Een belangrijk onderdeel is het user interface package dat onder andere zorgt voor multitasking en dat de high level command language FCL (FIPS Command Language) ondersteunt. FCL lijkt veel op een programmeertaal als Pascal: control statements, compound statements, functies en procedures, declaratie van variabelen, assignment statement en commentaar. Alle alfanumerieke interactie tussen het applicatieprogramma en de interactieve gebruiker gaat via FCL door middel van GATEWAY. Dit programma is een mix van compiler en interpretator voor FCL, een zogenaamde "interactive compiler". Bij Fokker werd overwogen om dit systeem in maart/april 1985 aan te schaffen. Dit was te laat om FIPS nog in VISUAL te kunnen gebruiken. FIPS is inmiddels ook aangeschaft.

Hoe de diverse programma's rond de database "gesitueerd" zijn wordt aangegeven in figuur 2.5. In deze figuur is ook het programma CODE opgenomen. Hiermee kunnen interactief 2D-vleugelprofielen ontworpen worden. Deze profielen kunnen later in GAMMA weer gebruikt worden om de geometrie van de vleugels te beschrijven.



Figuur 2.5 - De diverse programma's rond de database

2.4 - De functionele systeemeisen

De functionele systeemeisen geven aan over welke functies VISUAL moet beschikken. Er wordt niet aangegeven hoe deze functies gebruikt moeten kunnen worden of hoe deze functies in elkaar zitten. Deze zaken zijn nu nog niet van belang en komen in één van de volgende fasen aan de orde.

Voordat de gebruiker één van de visualisatie functies kan gebruiken moet hij eerst gegevens selecteren. Dit selecteren kan in de regel pas gebeuren nadat de gebruiker weet waaruit hij kan selecteren. Hiervoor moeten er in VISUAL de overzichts of DIRECTORY functies komen, te weten:

- CONFIGURATION: het geven van een overzicht van de beschikbare configuraties.
- SEGMENT/PART/GROUP: het geven van een overzicht van de segmenten, parts of groups van een geselecteerde configuratie.
- AERODYNAMIC: het geven van een overzicht van de beschikbare aerodynamica gegevens behorende bij een segment van een configuratie.

Nadat de gebruiker een DIRECTORY functie heeft gebruikt, wil hij gegevens gaan selecteren. Dit gebeurt met een van de volgende SELECT functies:

- CONFIGURATION: het selecteren van een bepaalde configuratie.
- SEGMENT/PART/GROUP: het selecteren van segmenten, parts of een group binnen een bepaalde configuratie.
- AERODYNAMIC: het selecteren van aerodynamische gegevens bij de segmenten van een bepaalde configuratie.

Nadat er segmenten geselecteerd zijn, moet het mogelijk zijn om deze zichtbaar te maken. Voor het visualiseren van het model moeten er een aantal basis-functies komen, namelijk:

- WIRE: het tekenen van het draadmodel. Het moet mogelijk zijn om de primaire, secundaire of beide soorten contouren te tekenen.

- OUTLINE: het tekenen van het draadmodel, waarbij alleen de contouren die nog net zichtbaar zijn (dat wil zeggen de component van de normaalvector, die loodrecht staat op het vlak van het beeldscherm, van het paneel aan de linker- heeft een tegenovergesteld teken in vergelijking met die aan

de rechterzijde van de contour) worden getekend.

Er moeten drie functies komen welke voor de transformaties gebruikt kunnen worden, genaamd ROTATE, SHIFT en ZOOM. De laatste twee functies zal de gebruiker slechts zelden nodig hebben; VISUAL zorgt ervoor dat de afbeelding juist geschaald en getransleerd wordt. Er zijn twee mogelijke manieren van ZOOMen: door twee punten aan te wijzen die dan een box vormen welke moet worden uitvergroot, of door één punt aan te wijzen en één schalingsfactor te specificeren waarmee rond het punt geschaald wordt.

Het moet altijd mogelijk zijn gebruik te maken van een aantal standaard aanzichten. Ook moet de gebruiker zelf een aantal aanzichten kunnen bewaren, zodat hij later niet opnieuw dezelfde serie rotaties hoeft uit te voeren voordat hij hetzelfde aanzicht terug heeft. Er moet een overzicht van de standaard en van de door de gebruiker gespecificeerde aanzichten gegeven kunnen worden. De functie die dit alles verzorgt, krijgt de naam MATRIX.

Er moet ook een functie komen waarmee het scherm schoongemaakt kan worden, genaamd CLEAR. Met dit commando moet het mogelijk zijn een keuze te maken: Of de huidige matrix moet bewaard blijven, of de eenheidsmatrix moet gebruikt worden. De matrix wordt gebruikt om het model op het scherm af te beelden en bepaalt onder andere welk aanzicht de gebruiker heeft.

De volgende functies moeten op de oppervlak van het model kunnen worden weergegeven:

- SHADE: Inkleuren van de panelen afhankelijk van de stand en afstand tot de lichtbron.
- ISO-LINES: tekenen van de isobaren (lijnen van gelijke druk) met of zonder aanduiding van drukniveau. Dit moet zodanig gebeuren dat een isobaar zo weinig mogelijk onderbrekingen bevat.
- ISO-BAND: inkleuren van isobarenbanden, dat wil zeggen het gebied tussen twee opeenvolgende isobaren met een van de druk afhankelijke kleur. De bij een kleur behorende drukwaarde (schaling) moet worden aangegeven.
- VELOCITY: tekenen van de snelheidsvectoren, één op elk paneel. De lengte van een snelheidsvector is afhankelijk van de grootte van de snelheid.
- TUFT: tekenen van richtingsvectoren (alle vectoren zijn even lang); één per paneel.
- COLOUR-PRESSURE: de panelen worden afhankelijk van de waarde van het drukniveau ingekleurd. De bij een kleur behorende drukwaarde (schaling)

moet worden aangegeven.

- NUMBER-PRESSURE: een getal waarde in elk paneel schrijven welke het drukniveau aangeeft.

- CONTOUR: tekenen van de drukverdeling op de contouren (eigenlijk op een rij panelen tussen twee contouren) van een segment in 3D-aanzicht. Vergelijking met een andere drukverdeling (bijvoorbeeld van andere een aanstromings conditie) moet gemaakt kunnen worden. Het selecteren van de contouren waarop de drukverdeling getekend moet worden gebeurt door deze op het beeldscherm met kruisdraden aan te wijzen.

De functie CARPET of heuvellandschap wordt niet op het oppervlak van het model weergegeven. Het niveau van drukverdeling wordt aangegeven door deze te gebruiken als "offset" van een referentievlak. Het referentievlak moet door de gebruiker worden opgegeven. Het heeft geen zin om deze functie te combineren met één van de hierboven genoemde functies, omdat er op een geheel andere wijze gevisualiseerd moet worden.

2.5 - De randvoorwaarden

Naast de functionele eisen zijn er nog een aantal andere eisen, namelijk de randvoorwaarden. Deze randvoorwaarden betreffen zoal de prestaties, de uitbreidbaarheid, de overdraagbaarheid en de gebruikersvriendelijkheid.

Prestaties

Omdat de postprocessor VISUAL een interactief programma is, is het belangrijk dat de wachttijden zo kort mogelijk zijn. Dit kan echter nogal wat problemen opleveren omdat het vaak het tekenen van zeer complexe afbeeldingen betreft hetgeen veel tijd kost. Toch moet worden getracht de wachttijden zo veel mogelijk te beperken, vooral tijdens de sessie wanneer een gebruiker de geselecteerde gegevens wil visualiseren. Het is beter dat er één keer (bijvoorbeeld tijdens het selecteren van de gegevens) wat langer gewacht moet worden en dat de rest van de sessie sneller verloopt, dan dat er steeds weer tijdens het tekenen veel tijd verloren gaat. Door een goede opzet van de datastructuur (waarin het model moet worden opgeslagen), kan de verwerkingssnelheid van VISUAL gunstig worden beïnvloed. Verder zijn de wachttijden natuurlijk afhankelijk van de bezettingsgraad van het time-sharing systeem.

Uitbreidbaarheid

Omdat de 3D stromingsberekenings programma's nog geruime tijd gebruikt zullen worden zal ook VISUAL gedurende vele jaren het gereedschap moeten vormen voor de visualisatie van de berekeningsresultaten. Uitbreiding van VISUAL, na de afstudeeropdracht, wordt zeer waarschijnlijk geacht. Doordat de gebruiker, na enige tijd met het systeem gewerkt te hebben, nieuwe ideëen zal opdoen en doordat de omstandigheden (en toepassingen van VISUAL) kunnen veranderen, zullen er nieuwe aanvullende gebruikerseisen komen. In hoofdstuk 5 worden een aantal mogelijke uitbreidingen/verbeteringen opgenoemd.

Wil het systeem voor uitbreiding vatbaar zijn, dan moet daar reeds bij de globale systeemopzet rekening mee worden gehouden. Dit vereist een degelijke en duidelijke algemene systeemopzet welke goed onderhoudbaar is. Het begint met voldoende standaard systeemdocumentatie en vereist vervolgens het gebruik van programmeertechnieken volgens vastgestelde richtlijnen; ref. 32.

Overdraagbaarheid

Er moet gestreefd worden naar een zo hoog mogelijke graad van overdraagbaarheid naar andere computersystemen. Dit is echter een ingewikkeld probleem, waarbij onderscheid moet worden gemaakt tussen machine-onafhankelijkheid en apparaat-onafhankelijkheid (zie 1.3). Het laatste kan men trachten te bereiken door gebruik te maken van een standaard grafisch pakket. Zo moet het eenvoudig mogelijk zijn verschillende soorten terminals (bijvoorbeeld VT100, T4014, T4114 en T4115) te gebruiken. Wel zal duidelijk zijn dat bepaalde functies niet op alle terminals te gebruiken zijn.

Verder wordt er naar gestreefd zoveel mogelijk een standaard programmeertaal te gebruiken. Ten slotte wordt opgemerkt dat de overdraagbaarheid van VISUAL staat of valt met de overdraagbaarheid van de ondersteunende software zoals, GINO, ORACLE, DIADBF of COLAS (zie 2.3).

Gebruikersvriendelijkheid

De gebruikers van VISUAL zijn in hoofdzaak medewerkers van de afdeling TOAP bij Fokker. Van de gebruikers mag niet worden verwacht dat ze computerexperts zijn. Ook niet regelmatige gebruikers moeten snel met het systeem overweg kunnen. Een aantal voorwaarden, waaraan moet worden voldaan om het systeem gebruikersvriendelijk te maken, zijn:

- Duidelijke vragen en foutmeldingen (in eenvoudig Engels).

- Een mogelijkheid om het tekenen en andere langdurige bewerkingen te onderbreken zonder dat het programma zelf afgebroken wordt en dus gewoon weer verder gaat met de alfanumerieke communicatie met de gebruiker (bijvoorbeeld door middel van een speciale toets).

- Het kiezen van geschikte, duidelijke prettige kleuren en achtergrond (b.v. zwart). Ook moet het mogelijk zijn dat de gebruiker zelf op een eenvoudige wijze kleuren kan definiëren. Dit geldt natuurlijk alleen voor de Tektronix 4115B (de kleuren raster display).

- Het aantal commando's waarmee een gebruiker een bepaald resultaat kan verkrijgen moet zo klein mogelijk zijn. Dit kan getracht worden te realiseren door onder andere gebruik te maken van default waarden.

- De interface met de gebruiker zal voor een groot deel bestaan uit een commandotaal welke veel weg heeft van de wijze waarop dit gebeurt bij het VAX/VMS operating system. Dit houdt onder andere in dat:

  * Men kan volstaan met het intypen van de eerste paar letters (b.v. HE i.p.v. HELP), mits deze uniek zijn.

  * Commando's die uit meerdere delen bestaan kunnen op twee manieren gegeven worden, n.l.: in één keer (b.v. SELECT CONFIGURATION) of in meerdere stappen (b.v. SELECT en dan komt de commandotaal interpretator terug met: "What do you want to select?", waarna door de gebruiker geantwoord wordt met CONFIGURATION).

  * Er moet een help functie (met de nodige on-line documentatie) aanwezig zijn.

- Om het systeem goed te kunnen gebruiken is het kunnen beschikken over een duidelijke en beknopte gebruikershandleiding (met veel voorbeelden) een vereiste. Een speciale opleiding van de gebruikers is niet noodzakelijk. De gebruikershandleiding is als bijlage aan dit afstudeerverslag toegevoegd.

2.6 - De belangrijkste beslissingen

Tijdens de systeemdefinitie zijn er reeds een aantal belangrijke beslissingen genomen over de te gebruiken programmeertaal en software pakketten. In deze paragraaf worden deze beslissingen uitvoeriger bekeken, zodat ook de motivatie van bepaalde keuzen duidelijk zal worden. Nu zullen achtereenvolgens aan de orde komen: de programmeertaal, de database, het grafische pakket en de besturing van het systeem door de gebruiker.

Programmeertaal

De keuze van de programmeertaal was zeer eenvoudig. De enige op de VAX beschikbare programmeertaal op het moment dat VISUAL geïmplementeerd moest worden was Fortran 77. Bovendien was deze taal de standaard binnen Fokker. Bij een vrije keuze zouden er zeker andere programmeertalen in overweging genomen zijn; bijvoorbeeld Pascal.

Een nadeel van Fortran is dat men altijd met een vaste hoeveelheid gegevens werkt door middel van array-declaraties met vaste grenzen en het ontbreken van types als records en pointers. Dit terwijl men niet weet hoeveel gegevens men binnen krijgt omdat dit onder andere afhankelijk is van de grootte en gedetailleerdheid van een bepaalde configuratie (geometrische/topologische gegevens) en de hoeveelheid daarmee samenhangende aerodynamische gegevens.

Voordelen van Fortran zijn de snelheid van de gecompileerde en gelinkte code en het eenvoudige gebruik van modulen uit libraries. Een voordeel van het Operating System (VAX/VMS) is dat dit over een virtueel geheugen mechanisme beschikt. Deze zorgt ervoor dat als er een grote array (b.v. 30.000 reals) wordt gedeclareerd, deze niet direct in zijn geheel wordt toegewezen omdat mogelijk niet alle elementen nodig zijn. Als men alleen de eerste 5.000 elementen gebruikt dan wordt ook alleen aan deze 5.000 elementen geheugenplaats toegewezen.

Database

Beschikbaar was het relationele database pakket ORACLE dat via de routines uit DIADBF in een Fortran 77 programma, vrij eenvoudig te gebruiken was. Verder waren er geen database systemen aanwezig, dus het enige alternatief was

om files te gebruiken. Het voordeel van files is dat het zeer efficiënt werkt, hetgeen voor VISUAL zeker van belang is in verband met de wachttijden. Hier tegenover staat dat het werken met files het gebruik van VISUAL snel zal beperken tot enkele specifieke toepassingen, omdat standaardisatie vrij moeilijk is. Dit is met een database systeem gemakkelijker realiseerbaar.

Het zal duidelijk zijn dat het werken met een relationeel database systeem als ORACLE vele voordelen biedt; zie hiervoor ref. 18. De 3D stromingsberekenings programma's zullen hun invoer halen uit en hun uitvoer schrijven naar de ORACLE database net zoals GAMMA, het pakket waarmee de geometrie gedefinieerd wordt. Het lag dus voor de hand dat er voor gekozen werd ORACLE (samen met DIADBF) in VISUAL te gebruiken.

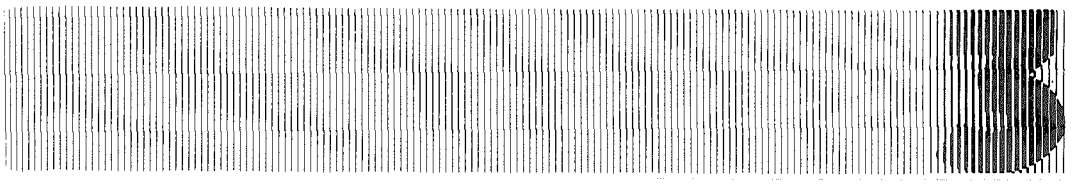
Grafisch pakket

Als grafisch pakket was alleen GINO-F beschikbaar. Het was natuurlijk mogelijk om routines te maken welke de tekenopdrachten naar de terminal sturen door middel van "escape sequences". Op deze wijze maakt men een eigen grafisch pakket, doch het zal duidelijk zijn dat dit zeer veel werk zou betekenen. Bovendien zouden er dan ook aspecten van de apparaat-onafhankelijkheid een rol spelen; zie in 2.5 het punt "overdraagbaarheid". Het zal daarom niemand verbazen dat aan GINO-F de voorkeur werd gegeven.

Besturing van het systeem

Er moest worden gekozen of VISUAL bestuurd zou worden door middel van een commandotaal of een menusysteem. Zowel een commandotaal als een menusysteem hebben voor- en nadelen. Voordelen van een (grafisch) menusysteem zijn onder andere: eenvoudig gebruik, prettige presentatie en daarom geschikt voor incidentele systeem gebruikers. Voordelen van een commandotaal zijn onder andere: onafhankelijkheid van het type terminal (elke terminal heeft wel 80 characters per regel en kan scrollen) en indien de gebruiker met de commando's bekend is, snelle interactie met het applicatieprogramma (mogelijkheden als defaults en macro's).

Vooral vanwege het feit dat een commandotaal apparaatonafhankelijker is dan een menusysteem, is er gekozen voor een commandotaal. Daar FIPS te laat beschikbaar en DIALIB te eenvoudig was, werd er voor COLAS gekozen.



Systeemontwerp

Tijdens het systeemontwerp werd naar een oplossing gezocht voor het probleem dat in de systeemdefinitie gespecificeerd is. In 3.1 wordt nog eens precies duidelijk gemaakt welke de doelstellingen van dit systeemontwerp zijn. Het is verstandig, alvorens er zelf allerlei oplossingen te creëren, eerst de literatuur er op na te slaan welke oplossingen daar worden aangedragen voor soortgelijke problemen. Dit betreft dan voornamelijk de datastructuur, doch ook zullen bekende algoritmen uit de literatuur onderzocht worden op hun bruikbaarheid. Dit alles wordt beschreven in 3.2. In 3.3 is het ontwerp van de datastructuur uitgewerkt.

Nadat de commandotaal COLAS in 3.4 wordt besproken, zal in 3.5 het ontwerp van de globale systeemopzet beschreven worden. Tijdens de probleem- en de systeemdefinitie is reeds enkele keren benadrukt dat de prestaties op het gebied van wachttijden van groot belang zijn. Voordat wordt besloten hoe dit specifieke snelheidsprobleem moet worden aangepakt, zal eerst duidelijk moeten zijn waar zich in het gehele afbeeldingsproces de "bottle neck" bevindt. Hiervoor kan men dan passende maatregelen treffen zodat het gewenste resultaat bereikt wordt. De belangrijkste resultaten van het onderzoek naar de snelheidsaspecten van het afbeeldingsproces zijn te vinden in 3.6.

3.1 - Het doel van het systeemontwerp

Het systeemontwerp moet leiden tot een systeemstructuur welke een decompositie van het systeem in modules met een kleinere complexiteit mogelijk maakt. Het systeem wordt opgesplitst in deelsystemen, die op hun beurt weer worden opgesplitst in modules.

Bij het ontwerp moesten een aantal aspecten extra aandacht krijgen; zie 1.3. Dit waren o.a. de globale systeemopzet, iets dat heel dicht bij het ontwerp staat, en visualisatie aspecten (het opzetten van een geometrisch model dat wordt vastgelegd in een bepaalde datastructuur). Het detailontwerp vond pas tijdens de implementatie plaats; zie hoofdstuk 4.

Centraal bij het opzetten van de systeemstructuur stond de datastructuur. Juist in het geval van visualisatie van aerodynamische en geometrische gegevens moest dit zeer goed gedaan moest worden, vanwege de gevolgen die dit voor het systeem met zich mee zou brengen. Het ontwerp van de datastructuur gebeurde niet in Fortran, omdat deze taal hiervoor te beperkt is. Voor de beschrijving van de datastructuur is een Pascal-achtige notatie gebruikt. Nadat de datastructuur werd gedefinieerd, moest worden aangegeven welke modules welke gegevens nodig hadden.

Het resultaat van het systeemontwerp is vastgelegd in een document; ref. 36. Doordat hierin later nog enkele wijzigingen zijn aangebracht is er een tweede versie van dit document verschenen; ref. 37.

3.2 - De geraadpleegde literatuur voor mogelijke oplossingen

Het is zinvol een objectbeschrijving te gebruiken, resulterend in een bepaalde datastructuur, die optimaal geschikt is voor het visualiseren van de gegevens. Voordat getracht werd zelf zo'n datastructuur op te zetten, werden eerst enkele in de literatuur beschreven methodes onderzocht. Ook al waren deze methodes niet direct bruikbaar, ze dienden zeker toch als inspiratiebron.

In deze paragraaf zal in het kort enige literatuur worden besproken, waarin datastructuren om geometrische modellen op te slaan worden gepresenteerd. Verder wordt er enige literatuur aangegeven welke algoritmen bevatten die bij de ontwikkeling van VISUAL gebruikt zouden kunnen worden.

- "Collegediktaat a215 CAD/CAM I, Hoofdstuk 3 Vormbeschrijving"; ref. 34. In dit hoofdstuk worden de basisprincipes van de vormbeschrijving duidelijk gemaakt. Vooral solid modelling, het beschrijven van 3D-objecten, komt vrij uitvoerig aan de orde. Hiervoor worden er een aantal representatieschema's beschreven, namelijk cellulaire decompositie, grensbeschrijving, constructieve geometrie en sweeping.

- "A polyhedron representation for computer vision"; ref. 38. In dit artikel wordt de "winged edge" representatie beschreven. Dit is een methode van grensbeschrijving waarbij een vlak wordt begrensd door een willekeurig aantal zijden en in een hoekpunt een willekeurig aantal zijden bij elkaar kunnen komen. Door opslag van redundante topologische gegevens in de vorm van pointers is een snelle toegang tot alle zijvlakken, zijden en hoekpunten op allerlei manieren mogelijk. De datastructuur wordt in Pascal-achtige termen beschreven.

- "Graph based models and their data structures"; ref. 39. In dit artikel worden de basisbegrippen van de grensbeschrijving, zoals hoeken, zijden, zijvlakken, geometrische gegevens, topologische gegevens, etc. behandeld. Grensbeschrijving wordt gerelateerd aan de graaf theorie, met begrippen als: knoop, tak en duaal. Ten slotte worden er ook een aantal varianten van de reeds eerder genoemde "winged edge" representatie gegeven.

- "GLIDE: A language for design information systems"; ref. 40. GLIDE is een Pascal-achtige taal met typen (integer, real, boolean, text en

vectoren), controle structuren (IF, FOR, en WHILE) en door de gebruiker gedefinieerde structuren (FORM, ziet eruit als een record in Pascal). De taal wordt gebruikt voor het modelleren van objecten in een DBMS omgeving. Basis elementen zijn veelvlakken, die door de gebruiker gedefinieerd worden, welke dan op allerlei wijzen gecombineerd kunnen worden.

- "Edge-based data structures for solid modelling in curved-surface environments"; ref. 41. Ook in dit artikel worden weer een aantal datastructuren beschreven voor een grensbeschrijving en er worden weer begrippen als topologische gegevens en graaf theorie gebruikt. Eén van de vier datastructuren is weer de "winged edge", de overige drie zijn varianten hierop die in Pascal- achtige termen beschreven worden. Dit artikel is vrij theoretisch van aard.

- "Principles of interactive computer graphics"; ref. 42. In dit boek worden de basisbeginselen van computer graphics uiteengezet. Het bevat veel algoritmen die bij de ontwikkeling van VISUAL gebruikt zouden kunnen worden, zoals diverse transformaties (perspectief, roteren om een willekeurige as), "painters algorithm", shading modellen (Phong en Gouraud), etc.

- "Fortran 77, paragraaf 15.6 Following and drawing contours"; ref. 43. In dit boek wordt als voorbeeld een algoritme beschreven waarmee contouren (bijvoorbeeld isobaren) worden berekend, uitgaande van data opgeslagen in een twee-dimensionale array.

- "Lighting controls for synthetic images"; ref. 44. In dit artikel wordt een shading model uitgewerkt dat gebruikt wordt in het programma GMR AUTOCOLOUR van General Motors. Dit shading model is een verfijning van het Phong model; ref. 38. Het artikel bevat enkele fraaie voorbeelden geïllustreerd aan de hand van een Chevrolet Camaro.

Het blijkt dat vooral de "winged edge" representatie en enkele varianten hierop zeer populaire methodes van grensbeschrijving zijn. Gezien de vorm waarin de (geometrische en aerodynamische) gegevens VISUAL binnen komen lijkt het echter weinig zinvol deze in een geheel andere representatie om te zetten. Bovendien blijkt deze vorm, dus een configuratie die is opgebouwd uit segmenten, welke op hun beurt weer bestaan uit een in een twee-dimensionale array opgeslagen hoeveelheid punten die de panelen beschrijven, te voldoen.

Door de eenvoudige structuur is altijd bekend dat panelen vier hoekpunten bevatten (en dus ook vier zijden) en dat interne hoekpunten in verbinding staan met vier zijden. Door deze structuur, waarin impliciet de topologische gegevens verwerkt zijn, is een snelle toegang tot alle panelen, zijden en hoekpunten binnen één segment, op allerlei manieren mogelijk, via de indexen waarmee deze worden aangeduid.

Alleen bij de grens tussen twee aansluitende segmenten treden er problemen op. Hier zullen de topologische gegevens expliciet opgeslagen moeten worden, door referenties van een punt uit het ene segment naar een punt in een ander segment. In de volgende paragraaf zal het ontwerp van de datastructuur beschreven worden.

3.3 - Het ontwerp van de datastructuur

De (invoer) gegevens zijn afkomstig uit een database, die deel uitmaakt van het database systeem ORACLE. ORACLE is een relationeel database management systeem waarmee met de query language SQL gegevens opgevraagd kunnen worden. Voor het gebruik van de database vanuit een (Fortran) applicatie programma zijn een aantal routines geschreven, die de query's verzorgen. Deze routines vormen gezamelijk een programma interface tussen ORACLE en een applicatie programma. Hoe deze programma interface gebruikt moet worden is te vinden in ref. 45.

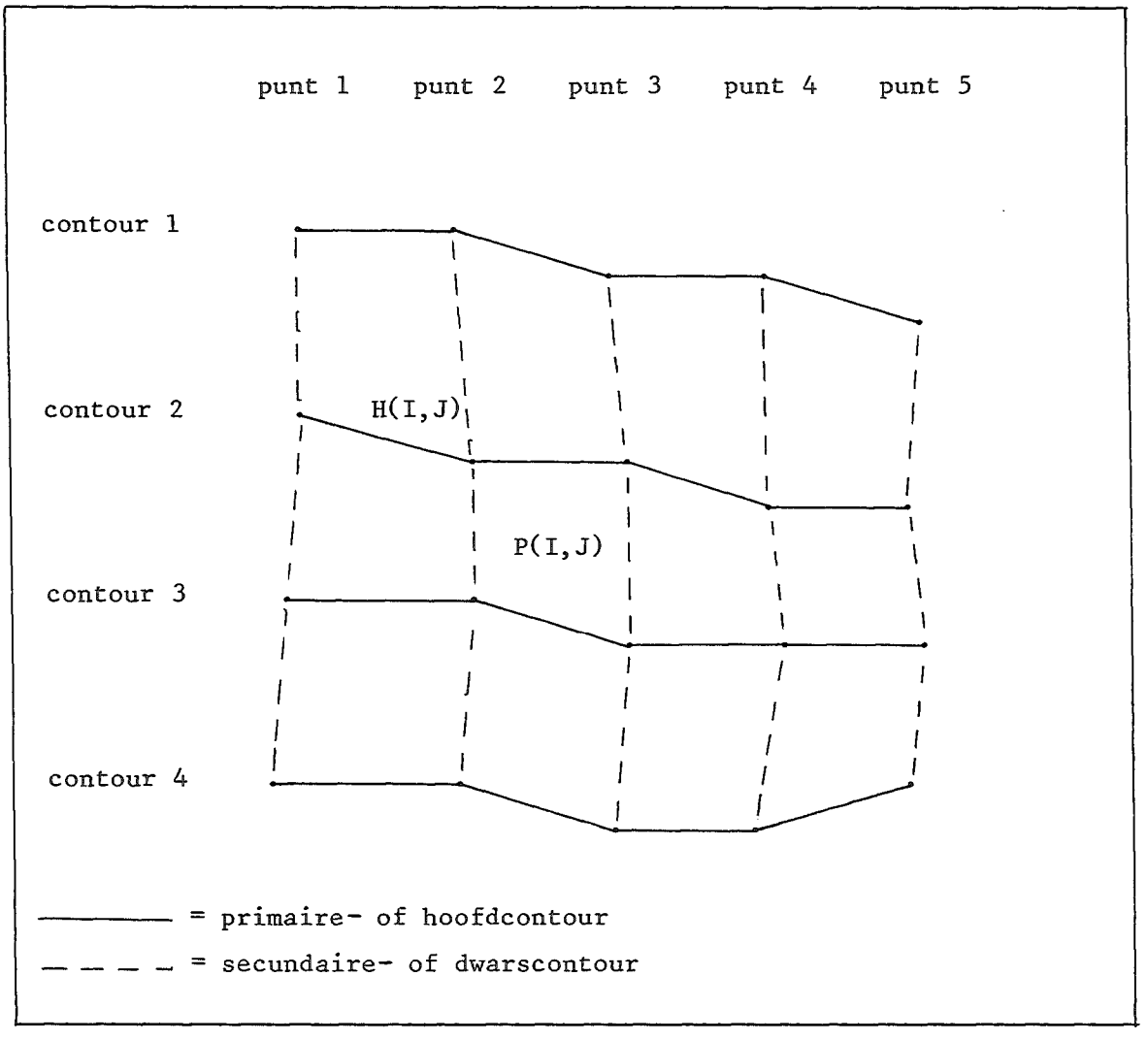
Door de mogelijkheden van het Virtual Mapping System kan een zodanige datastructuur opgezet worden, dat er efficiënt met gegevens gemanipuleerd kan worden vanuit het applicatie programma VISUAL. Nadat de gegevens uit de database zijn opgehaald, worden ze na een voorbewerking in de datastructuur opgeslagen.

Gegevens die men uit de datastructuur op een bepaalde wijze (in verband met de functie die op dat ogenblik gevisualiseerd wordt, zie 2.4) wil ophalen zijn:

A. Binnen segment: Gegevens welke opgehaald worden via "relaties" binnen één segment. Een segment (met configuratie nummer CNFNR en segment nummer SEGNR) bevat NC contouren, die elk uit NP punten bestaan en dus (NC-1)*(NP-1) panelen definieren. In figuur 3.1 is een eenvoudig segment getekend (NC=4 en NP=5).

   Elk paneel kan worden aangeduid (binnen dat segment) door middel van twee indices n.l. I,J (met $1 \leq I \leq NC-1$, $1 \leq J \leq NP-1$). Als een index buiten de range valt dan bestaat het bijbehorende paneel niet (b.v. P (2,3) bestaat en P (0,3) bestaat niet. De P staat voor paneel).

   Ook elk hoekpunt kan met twee indexen (I,J) worden aangeduid
   (H (I,J) met $1 \leq I \leq NC$ en $1 \leq J \leq NP$). Een hoofd of primaire contour kan met één index worden aangeduid (I met $1 \leq I \leq NC$) en een dwars of secundaire contour ook (J met $1 \leq J \leq NP$).
   Vanuit een paneel (P(I,J)) zou men een aantal gegevens willen opvragen. De letters duiden de belangrijkheid aan: N = Noodzakelijk, Z = Zeer belangrijk, B = Belangrijk en M = Minder belangrijk.



Figuur 3.1 - Een segment met vier contouren en vijf punten per contour

De beoordeling is afhankelijk van:

- de frequentie waarmee een bepaalde vraag gesteld wordt;
- hoeveel rekenwerk het zou kosten, indien het antwoord van de vraag niet expliciet wordt opgeslagen;
- de duurzaamheid van de gegevens (treden er vaak wijzigingen op ?);
- de hoeveelheid ruimte die nodig is om de gegevens op te slaan.

Het gaat hierbij om de volgende gegevens:

N) vier hoekpunten (n.l. H (I,J), H(I,J+1), H(I+1,J) en H(I+1,J+1));
Z) één normaalvector (NX(I,J),NY(I,J) en NZ(I,J));

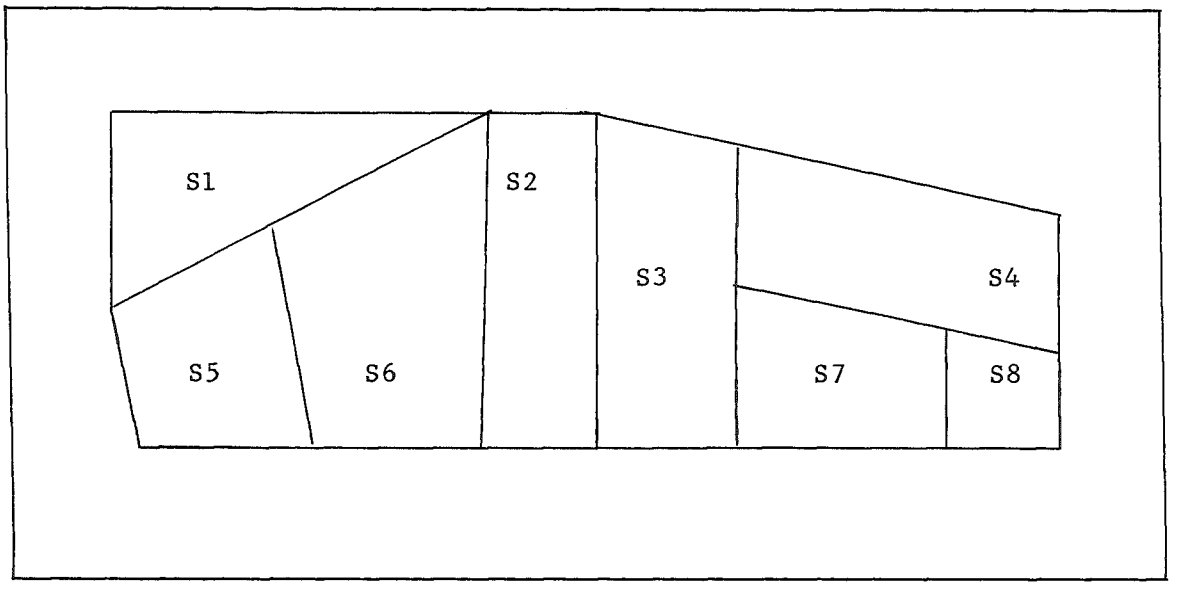
M) één oppervlak, zodat later het omspoeld oppervlak berekend kan worden (Opmerking: het oppervlak van een paneel kan ook worden bepaald via de normaalvector);
B) één paneel middelpunt (PMX(I,J),PMY(I,J) en PMZ(I,J));
B) één variabele die aangeeft of hoekpunten van het paneel samenvallen;
B) vier direct aangrenzende panelen (n.l. P (I-1,J), P(I,J+1), P(I+1,J) en P(I, J-1));
M) vier diagonaal aangrenzende panelen (n.l. P(I-1, J-1), P(I-1,J+1), P(I+1,J-1) en P (I+1, J+1));
B) twee primaire contouren (PC (I), PC (I+1) en twee secundaire contouren (SC (J), SC (J+1));
N) één drukwaarde CP1, (n.l. CP1 (I,J));
W) één drukwaarde CP2, dit is de drukwaarde waarmee vergeleken moet kunnen worden (n.l. (CP2 (I,J));
N) één snelheidsvector (n.l. VX (I,J), VY (I,J), VZ (I,J)).

Vanuit een (hoek)punt (H (I,J)) zou men de volgende gegevens willen ophalen:

N) vier panelen (n.l. P (I-1, J-1), P (I-1,J), P (I,J), P (I,J-1));
B) één primaire contour (n.l. PC (I));
B) één secundaire contour (n.l. SC (J)).

Opmerking: Er zijn twee soorten segmenten, n.l. segmenten waar de drukwaarden en de snelheidsvectoren per paneel gegeven worden (dit geldt voor de "normale geometrie") en segmenten waar deze per hoekpunt gegeven worden. In het laatste geval (dit zijn de zogenaamde "veld- punten", d.w.z. punten die niet het oppervlak van het vliegtuig zelf beschrijven, maar de ruimte erom heen) wil men de aerodynamische gegevens per hoekpunt ophalen in plaats van per paneel.

B. Op segmentniveau: Gegevens welke op segmentniveau nodig zijn en gegevens die opgehaald moeten worden via relaties tussen de segmenten. Een segment kan worden aangeduid met een segment nummer (dat uniek is binnen een bepaalde configuratie). In figuur 3.2 is schematisch een achttal segmenten weergegeven.



Figuur 3.2 - Een achttal segmenten

De volgende gegevens zou men via segment (SEGNR) willen opvragen:

N) aantal contouren en aantal punten per contour (NC, NP);
Z) minimale waarde X,Y,Z en maximale waarde X,Y,Z van de hoekpunten binnen dit segment. Dit is nuttig voor het automatisch schalen van het scherm en voor het bepalen van de tekenvolgorde van de segmenten;
Z) minimale en maximale drukwaarde (van CP1) binnen dit segment. Dit is nuttig voor de zogenaamde "colour-scaling";
N) lijntype (b.v. een code die de kleur aangeeft);
M) de oppervlakte van dit segment;
Z) de aangrenzende segmenten (evt. toegankelijk via de randpunten van dit segment, dat zijn er 2*NC + 2*NP -4).

Nu de wensen vanuit het applicatie programma VISUAL bekend zijn, kan een datastructuur worden ontworpen die hieraan voldoet. De gegevens zullen voor het grootste gedeelte rechtstreeks uit ORACLE worden opgehaald. De overige gegevens zullen hieruit worden afgeleid; bijvoorbeeld de normalen worden berekend uit de hoekpunten.

Een configuratie wordt geïdentificeerd met een nummer (CNFNR). Een group heeft binnen deze configuratie een volgnummer (GRPREL) en wordt dus geïdentificeerd met het paar CNFNR,GRPREL. Hetzelfde geldt voor parts (CNFNR, PRTREL) en voor segmenten (CNFNR,SEGREL).

Naast deze volgnummers worden groups, parts en segmenten binnen de database aangeduid met een uniek nummer (resp. GRPNR, PRTNR en SEGNR). Deze nummers worden alleen intern in VISUAL gebruikt; de gebruiker werkt steeds met de volgnummers, die hij verkrijgt via de DIRECTORY functie.

Nu volgt een beschrijving van de datastructuur. Hierbij zal het volgende type op een aantal plaatsen gebruikt worden:

```
type vector = array (1..3) of real;
```

Vervolgens de beschrijving van de datastructuur zelf:

```
type configuratie = record
                        CNFNR: integer;
                        beschrijving: string of char;
                    end

type group =        record
                        CNFNR, GRPREL, GRPNR: integer;
                        beschrijving: string of char;
                    end

type part =         record
                        CNFNR, PRTREL, PARTNR: integer;
                        SEGMENTS: set of integer; (bevat alle segmenten
                        die in dit part zitten)
                        SELECTED-SEGMENTS: set of integer; (bevat de
                        geselecteerde segmenten, dit is een
                        deelverzameling van SEGMENTS)
                    end
```

```
type segment =      record
                        CNFNR, SEGREL, SEGNR: integer;
                        NP, NC, LIJNTYPE: integer;
                        hoekpunten: array (1..NC, 1..NP) of vector;
                        normalen:   array (1..NC-1, 1..NP-1) of vector;
                        case (segment-type) of
                             geometrie : CP1: array (1..NC-1, 1..NP-1)
                                              of real;
                                         V:  array (1..NC-1, 1..NP-1)
                                              of vector;
                             veldpunten: CP1: array (1..NC, 1..NP)
                                              of real;
                                         V:  array (1..NC, 1..NP)
                                              of vector;
                        end;
                        buren-van-randpunten: array (1..2*NC+2*NP-4) of
                                                 record
                                                   SEGNR, I, J: integer
                                                 end;
                        (deze buren worden opgeslagen, zodat er
                        verbindingen met andere segmenten gemaakt kunnen
                        worden, dit is van belang bij het uittekenen van
                        isobaren)
                        min-hoekpunt: vector;
                        max-hoekpunt: vector;
                        min-CP, max-CP: real;
                    end

var  CONF: configuratie;
var  GROUP: group;
var  PARTS: set of part;
var  SEGMENTS: set of segment;
```

Opmerking 1: GROUP wordt alleen gebruikt als de selectie mode "group" is en PART wordt alleen gebruikt als de selectie mode "group" of "part" is. Dit zal worden aangegeven door de variabele SELECTION-MODE, welke van het type string is.

Opmerking 2: Het record element in het type segment genaamd "buren-van-randpunten", wordt tijdens de detailimplementatie verder uitgewerkt aan de hand van een aantal eisen die hieraan gesteld worden. Dit wordt uitvoerig beschreven in 4.4.

De opslag van de isobaren

Tenslotte moeten er nog datastructuren ontworpen worden voor de opslag van de isobaren en isobaren banden. Isobaren (en isobaren banden) kunnen onafhankelijk of gerelateerd aan de rest van het model worden opgeslagen. Het voordeel van het onafhankelijk opslaan is dat deze methode zeer eenvoudig is (lijkt); er behoeven alleen een aantal coördinaten te worden opgeslagen. Het nadeel hiervan is dat men niet weet of de isobaar zichtbaar is, want het is niet bekend bij welke panelen zo'n isobaar behoort. Daarom wordt er gekozen voor het opslaan van isobaren gerelateerd aan de rest van het model.

Nu zijn er ook weer een aantal mogelijke datastructuren te bedenken. Twee daarvan zullen nu worden uitgewerkt. Het verschil tussen de twee datastructuren is dat de ene uitgaat van een paneelgewijze verwerking en dat de andere uitgaat van een isobaargewijze verwerking (per segment). Bij het berekenen van de isobaren wordt er van uit gegaan dat de drukwaarden op de hoekpunten gegeven zijn. Dit is het geval bij veldpunten. Bij segmenten die het oppervlak van het model beschrijven is dit niet het geval; hier zal eerst een omzetting van drukwaarden per paneel naar drukwaarden per hoekpunt nodig zijn. Indien de isobaren ook over de segmentgrenzen heen zoveel mogelijk continu dienen te lopen, dan zal voor de randpunten van een segment informatie uit een aangrenzend segment meegenomen moeten worden.

A. Per paneel: vastgelegd wordt waar op een contourlijnstuk, welke isobaar dit lijnstuk snijdt. Het zal duidelijk zijn dat één lijnstuk 0,1 of meer keer door (verschillende) isobaren doorsneden kan worden. Waar een isobaar met de waarde "isobar-value" een lijnstuk snijdt wordt aangegeven met een getal "fraction", dat tussen 0 en 1 ligt. Wat voor een contourlijnstuk (primair) geldt, geldt ook voor een dwarscontourlijnstuk (secundair). Een contourlijnstuk kan door twee indexen aangeduid worden, (namelijk CONTLIJNSTUK(I,J) met $1 \leq I \leq NC$ en $1 \leq J \leq NP-1$). Hetzelfde geldt voor een dwarscontourlijnstuk (DWARS-LIJNSTUK(I,J) met $1 \leq I \leq NC-1$ en $1 \leq J \leq NP$), alleen de range van de indeces is iets anders. Bij het record type segment moeten de volgende

elementen worden opgenomen:

```
CONTLIJNSTUK = array (1..NC, 1..NP-1) of set of record
                                        isobar-value: real;
                                        fraction: real (tussen 0 en 1)
                                                end

DWARSLIJNSTUK = array (1..NC-1, 1..NP) of set of record
                                        isobar-value: real;
                                        fraction: real (tussen 0 en 1)
                                                end
```

Bij het tekenen van de isobaren wordt nu steeds één paneel beschouwd. Bij dit paneel behoren vier begrenzende lijnstukken (twee CONTLIJNSTUK en twee DWARSLIJNSTUK), die allen nul of meer snijpunten met isobaren hebben. Nu worden er steeds twee snijpunten, die bij een zelfde "isobar-value" behoren, verbonden met een rechte lijn.

B. Per isobaar: vastgelegd wordt voor elke isobaar (binnen een segment) hoeveel keer deze, lijnstukken (contourlijnstukken en/of dwars-contourlijnstukken) snijdt en de serie snijpunten. Een snijpunt bestaat uit een boolean (prim) die aangeeft of het een snijpunt met een primair contour lijnstuk of met een secundair contourlijnstuk (dwars) is. Vervolgens twee indexen, die op eenzelfde wijze werken als in A en tenslotte een fractie. Bij het record type segment moet dan dus het volgende element worden opgenomen:

```
ISOBAR = array (isobarvalues) of record
                                        aantal_pnt: integer;
                                        pnt: array (1..aantal_pnt) of
                                                record
                                                     prim:boolean;
                                                     I,J: integer;
                                                     fraction:real
                                                end
                                end
```

Bij het tekenen van een isobaar worden alle punten nagelopen (dit zijn er "aantal_pnt") en verbonden door lijnstukken. Ligt een

lijnstuk echter op een paneel (I,J) dat niet zichtbaar is, dan moet dit lijnstuk niet getekend worden.

Op het eerste gezicht lijkt de eerste methode (A) beter aan te sluiten bij de paneelgewijze verwerking, die ook in de rest van het programma veel zal voorkomen. De tweede methode (B) is logischer, omdat men nu met een isobaar "als object" kan werken. Dit heeft het voordeel dat men met isobaren kan manipuleren, bijvoorbeeld door een isobaar toe te voegen of te verwijderen. Daarom is voor de implementatie methode B gekozen.

Als men de isobarenbanden wil inkleuren en men loopt isobaar voor isobaar het hele model door, dan worden de panelen die niet door een isobaar doorsneden worden nog niet ingekleurd. Er zal een speciale "verwerkingsslag" nodig zijn om de nog niet gekleurde panelen in te kleuren.

De datastructuur is dynamisch van aard en kan zoals hierboven beschreven niet rechtstreeks in Fortran geïmplementeerd worden. Hiervoor zullen speciale routines geschreven moeten worden.

3.4 - De commandotaal COLAS

Voordat het ontwerp wordt uitgewerkt zullen eerst de belangrijkste eigenschappen van de commandotaal COLAS (ref. 42) worden besproken, voor zover dit voor het systeemontwerp van belang is. Opgemerkt wordt dat dit niet uitsluitend voor VISUAL geldt, maar voor alle programmatuur waarin COLAS wordt toegepast.

COLAS werkt met commandosets; verzamelingen van commando's welke gezamenlijk één set vormen. Het applicatieprogramma, bijvoorbeeld VISUAL, kan ervoor zorgen dat een bepaalde commandoset actief wordt. Er kan maximaal één commandoset gelijktijdig actief zijn.

Binnen één commandoset zijn de commando's onafhankelijk van elkaar; er is dus geen volgorde of hiërarchie. Bij elke commandoset behoort één bepaalde prompt waarmee de gebruiker telkens wordt gevraagd om een commando te geven. Voordat de eerste keer met een prompt om een commando wordt gevraagd verschijnen er ter introductie een aantal regels tekst (niet verplicht).

Een commando bestaat uit een commandonaam (evt. afgekort) gevolgd door nul of meer parameters, afhankelijk van het type commando. Parameters kunnen al of niet verplicht zijn (mandatory of optional), in free position (bijvoorbeeld COPY, TO = FILE 2, FROM = FILE 1) of in fixed position (bijvoorbeeld COPY, FILE 1, FILE 2) worden gegeven. Indien parameters in free position worden opgegeven moet elke parameter voorafgegaan worden door de parameternaam en een = teken; zie het bovenstaande voorbeeld. Er zijn 3 typen parameters, n.l. reële getallen (real), gehele getallen (integer) en string type.

Helpinformatie kan altijd met COLAS worden verkregen. De helpinformatie kan door de gebruiker worden opgevraagd door "?" in te typen, waarna hij een overzicht krijgt van alle beschikbare commando's op dat ogenblik. Typt de gebruiker "?,commandonaam" in, dan krijgt hij specifieke informatie over het desbetreffende commando. Door "?,commandonaam,parameternaam" in te typen krijgt de gebruiker helpinformatie over de parameter "parameternaam" uit het commando "commandonaam".

Wil de gebruiker de actuele waarden van de parameters van het commando "commandonaam" weten, dan moet hij "LIST,commandonaam" intypen. Naast de commando's LIST en ? zijn er nog enkele commando's die altijd in elke

commandoset (uit COLAS) gebruikt kunnen worden, dit zijn:

1) STOP: hiermee wordt de gebruikers sessie beëindigd;
2) QUIT: hiermee wordt de huidige commandoset verlaten.

De interface met het programma bestaat uit een aantal routines, welke vanuit het programma worden aangeroepen. Met COLAS kunnen (op verzoek van het applicatieprogramma) berichten naar de terminal gestuurd worden. Deze berichten (of messages zoals ze in COLAS heten) worden evenals de commando-sets, commando's en de parameters beschreven in een speciale file. Deze file moet eerst "vertaald" worden met het programma CDM (Command Definition Module), voordat COLAS deze kan gebruiken. Ook de helpinformatie staat in de COLAS file beschreven.

3.5 - Het ontwerp van de globale systeemstructuur

In deze paragraaf zullen de volgende onderwerpen aan de orde komen: de globale systeemstructuur, de PROGRAM MODES, de SET-UP, het selecteren, het tekenen en ten slotte de deelsystemen 1D- en 2D-FUNCTIE.

De globale systeemstructuur

In figuur 3.3 is de globale structuur van VISUAL weergegeven met behulp van pseudo-code. VISUAL wordt opgesplitst in een vijftal deelsystemen, welke verderop in deze paragraaf zullen worden besproken.

```
INITIALISATIE
WELKOM GEBRUIKER
LEES COMMANDO IN
WHILE COMMANDO ≠ STOP DO
            CASE COMMANDO OF
                        DIRECTORY : DIRSEL(COMMANDO)
                        SELECT    : DIRSEL(COMMANDO)
                        DRAW      : 3D(COMMANDO)
                        3D-FUNCTIE: 3D(COMMANDO)
                        2D-FUNCTIE: 2D(COMMANDO)
                        1D-FUNCTIE: 1D(COMMANDO)
                        SET-UP    : SET-UP(COMMANDO)
            END
            IF COMMANDO ∉ {DIRECTORY, SELECT, DRAW, 3D-FUNCTIE,
                           2D-FUNCTIE, 1D-FUNCTIE, SET-UP, STOP}
            THEN LEES COMMANDO IN
END
```

Figuur 3.3 - De structuur

Binnen elk deelsysteem kunnen een aantal commando's gegeven worden. De parameter COMMANDO wordt gebruikt om aan te geven hoe een deelsysteem is verlaten. Dit kan op drie manieren:

1) Met het commando STOP; dit houdt in dat de sessie met VISUAL beëindigd wordt. Dit commando kan overigens ook op het hoofdniveau gegeven worden.
2) Met het commando QUIT; hiermee komt de gebruiker weer op het hoofdniveau terug.
3) Door één van de zeven hiervoor genoemde commando's; de gebruiker springt rechtstreeks van het ene deelsysteem naar het andere.

Bij de initialisatie moeten de volgende zaken afgehandeld worden:

- bepalen van het type terminal;
- initialisatie grafisch pakket (GINO);
- initialisatie commandotaal pakket (COLAS);
- initialisatie programma interface met ORACLE;
- declareren van buffers (b.v. voor GINO);
- geven van systeem default waarden (of voor een bepaalde gebruiker karakteristieke waarden, die b.v. in een file staan opgeslagen).

Het welkom heten van de gebruiker bestaat uit een "titel pagina" welke aangeeft dat de gebruiker met VISUAL werkt. Vervolgens wordt er op het hoofdniveau om een commando gevraagd. Zolang dit commando ongelijk is aan het stopcommando, wordt het betreffende deelsysteem uitgevoerd en wordt er opnieuw een commando op het hoofdniveau bepaald.

Elk deelsysteem heeft z'n eigen commandoset en prompt, zodat de gebruiker snel kan zien in welk deelsysteem hij zich bevindt. In appendix C is per deelsysteem een overzicht gegeven van de beschikbare commando's. Normaal wordt een deelsysteem verlaten met het commando QUIT, waarna men op het hoofdniveau terecht komt en opnieuw een commando moet geven om een ander deelsysteem te activeren. Omdat dit nogal omslachtig is kan men in VISUAL ook direct van het ene naar het andere deelsysteem springen. Dit is gerealiseerd door aan elke commandoset alle commando's uit het hoofdniveau toe te voegen.

De deelsystemen hebben een soortgelijke structuur als het globale systeem, namelijk één grote "case" statement. Dit komt door het feit dat er gebruik wordt gemaakt van een commandotaal en het gebruik van een case statement ligt

dan erg voor de hand. Dit geldt zowel op het hoofdniveau als op het niveau van de verschillende deelsystemen. Pas op het niveau van de verschillende functies worden de routines echt anders. Voor zover van belang, zullen deze functies tijdens het detailontwerp worden besproken; zie hoofdstuk 4.

De PROGRAM MODES

Om het systeem gebruikersvriendelijk te maken zal, naast een aantal andere maatregelen, het systeem worden voorzien van een aantal PROGRAM MODES. Het idee hierachter is dat de gebruiker met een minimum aantal commando's een bepaald resultaat moet kunnen verkrijgen. De PROGRAM MODES bepalen voor een groot deel de werking van het systeem. Zo bepalen de PROGRAM MODES onder andere of er al dan niet automatisch geschaald moet worden tijdens het tekenen en of er bij het berekenen van de isobaren van een segment wel of niet informatie moet worden meegenomen uit aangrenzende segmenten, etc.

Als het systeem VISUAL wordt gestart, worden eerst de variabelen (die de PROGRAM MODES karakteriseren) van een waarde voorzien. Deze initialisatie waarden kunnen per gebruiker verschillen en worden uit een file van de gebruiker gehaald. Op deze wijze werkt iedere gebruiker met die "versie" van het systeem, welke hem het beste bevalt. Indien een dergelijke file niet aanwezig is, dan worden systeem default waarden gebruikt voor de initialisatie. In appendix E is een overzicht gegeven van de PROGRAM MODES met hun default waarden.

De SET-UP

Er zal een aantal functies moeten komen, welke speciaal zijn bedoeld om met de PROGRAM MODES te manipuleren. Deze functies zullen tesamen één deelsysteem, genaamd SET-UP, vormen. Met behulp van het commando MODE kan de gebruiker een overzicht opvragen van alle variabelen, welke de PROGRAM MODE karakteriseren. Het commando MODE geeft dan gelijktijdig voor elke variabele de actuele waarde aan. Deze waarde kan met behulp van het commando CHANGE-VAR veranderd worden. Met het commando INFO-VAR krijgt de gebruiker de waarde en specifieke informatie over deze variabele.

Het commando USER-DEF zorgt ervoor dat de file met PROGRAM MODES wordt ingelezen. Voor elke variabele is een waarde gespecificeerd, welke aan de

betreffende variabele wordt toegewezen. Deze functie wordt ook automatisch aangeroepen bij het starten van VISUAL. Het commando SYSTEM-DEF geeft alle variabelen, welke de PROGRAM MODE karakteriseren, de systeem default waarde.

Met het commando STORE worden de huidige waarden van de PROGRAM MODE variabelen naar (een nieuwe versie van) de file met PROGRAM MODES geschreven. Hiermee kan de gebruiker dus een karakterisering van z'n eigen versie van VISUAL bewaren.

Ten slotte zal het deelsysteem SET-UP een aantal functies bevatten waarmee de kleurdefinities veranderd kunnen worden. Deze functies zijn uiteraard alleen zinvol indien er met een kleurenscherm gewerkt wordt. De volgende functies zijn hiertoe gewenst: HUE-COLOUR, LIGHTNESS, SATURATION en COLOUR-DEF.

Het selecteren

Zoals reeds in 2.4 is vermeld zal er ook een aantal functies moeten komen voor het geven van overzichten en het selecteren van gegevens. Dit doen we met de functies DIRECTORY en SELECT, welke, aangevuld met nog enkele andere functies, het deelsysteem DIRSEL zullen vormen. Tot op welk niveau (dat wil zeggen het niveau van configuratie, group, parts, segmenten of aero- dynamische gegevens) de gegevens geselecteerd zijn wordt aangegeven met een globale variabele, genaamd FASE. Meer hierover in 4.3.

De gebruiker kan een geselecteerd object (zoals group, part of segment) laten vervallen met behulp van de functie DROP. Tevens vervalt dan alles wat er in de hiërarchische structuur onder hangt. Met de functie ADD kan de gebruiker op het niveau van segmenten iets toevoegen. Met de functie SELECT kan men op een bepaald niveau iets selecteren. Het gevolg is wel dat eerst alles wat op dit niveau (en lager) al geselecteerd was, wordt weggegooid.

Vervolgens hebben we de functie ACTIVE waarmee de gebruiker kan opvragen in welke selectiemode hij werkt en wat hij tot nu toe heeft geselecteerd. Dit is een nuttige functie, omdat er van een commandotaal gebruik wordt gemaakt (om het systeem te besturen) en er dus geen vragen worden gesteld als "Geef nu aan welke segmenten U wilt selecteren", waarmee de gebruiker door het programma geleid wordt.

Tenslotte is er de functie LISTING, waarmee de geselecteerde gegevens alfa-numeriek kunnen worden bekeken. Er moet kunnen worden aangegeven of de listing al of niet geheel op het scherm moet komen. Is dit niet het geval, dan wordt er slechts een samenvatting van de belangrijkste gegevens verstrekt.

Bij veel van de genoemde functies moet worden aangegeven om welk type object het gaat (b.v. SELECT,GROUP). In sommige gevallen behoort er ook nog een lijst met nummers bij welke aangeven om welke objecten het gaat. Een range van nummers moet kunnen worden aangegeven door de onder- en bovengrens te specificeren met daartussen een dubbele punt (:). Ook kan er natuurlijk één nummer worden opgegeven, hetwelk dan een range is met lengte één. Een scheiding tussen twee ranges wordt aangegeven door een slash (/).
Bij voorbeeld: SELECT,SEGMENT 1:5/9/12:15
Dit betekent: selecteer de segmenten 1 tot en met 5, segment 9 en de segmenten 12 tot en met 15.

Het tekenen

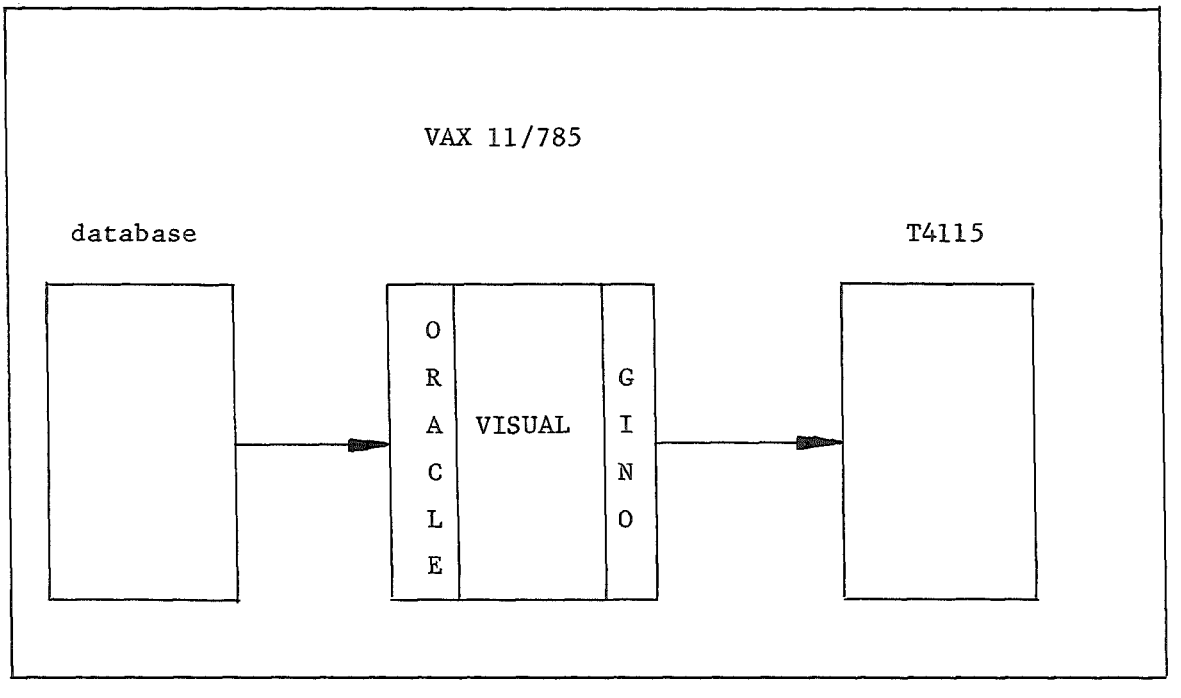
Het derde deelsysteem dat geïmplementeerd zal worden, genaamd 3D-FUNCTIE, bevat diverse functies waarmee de geometrische en de aerodynamische gegevens gevisualiseerd kunnen worden (WIRE, OUTLINE, HIDDEN, SHADE, VELOCITY, TUFT, COLOUR-CP, NUMBER-CP, ISO-LINES, ISO-BAND, CONTOUR en CARPET). Verder bevat het deelsysteem functies waarmee transformaties kunnen worden uitgevoerd, zoals ROTATE, SHIFT, ZOOM en MATRIX. Over dit deelsysteem valt pas iets interessants te vertellen op het niveau van de individuele functies; dit zal gebeuren in 4.4.

Nog eens twee deelsystemen

Naast de hiervoor genoemde deelsystemen zullen er in de toekomst nog twee deelsystemen in VISUAL komen. Deze deelsystemen zijn echter al wel in het ontwerp opgenomen, waardoor ze later eenvoudig kunnen worden ingepast. Met het deelsystemen 2D-FUNCTIE kunnen de aerodynamische gegevens op een 2D-wijze worden gevisualiseerd. Dit kan bijvoorbeeld gebeuren met behulp van een grafiek. In het deelsysteem 1D-FUNCTIE kunnen gegevens in de vorm van één of enkele waarden worden opgevraagd. Voorbeelden hiervan zijn: totale lift of weerstand van een configuratie, cp-waarden of snelheidsvectoren op één bepaald paneel, het oppervlak van een configuratie, etc.

3.5 - De snelheidsaspecten

Voor een interactief systeem, als de postprocessor VISUAL, zijn de wachttijden een belangrijke factor in betrekking tot de gebruikersvriendelijkheid van dat systeem; zie 2.5. Als het resultaat van een gebruikerscommando het maken van een (vrij ingewikkelde) afbeelding is, dan kan de wachttijd nogal eens te wensen overlaten; bijvoorbeeld in GAMMA tot $\pm$ 10 minuten. In een time sharing systeem heeft men de wachttijden nooit helemaal in eigen hand (deze is namelijk afhankelijk van de belasting van het systeem), maar door het nemen van bepaalde maatregelen kan men proberen de wachttijden toch zoveel mogelijk te beperken. In de onderstaande figuur is het afbeeldingsproces (sterk vereenvoudigd voor de situatie van VISUAL) weergegeven voor raster graphics.
Opmerking: deze paragraaf gaat voornamelijk over raster graphics, omdat dit het voornaamste probleemgebied aangaande de wachttijden is.



Figuur 3.4 - Het afbeeldingsproces

Nu rijst de vraag: "Welke maatregelen zullen zinvol zijn om de wachttijden te beperken ?" Dit zijn die maatregelen, die de "bottle neck" van het afbeeldingsproces verruimen. Globaal gezien (zie Figuur 3.4) bestaat het

afbeeldingsproces uit 3 fasen:

I het ophalen van de gegevens (dat zijn in dit geval dus geometrische en aerodynamische gegevens) uit de database;

II het uitvoeren van diverse bewerkingen op die gegevens (zoals selecteren, ordenen, transformeren, etc.);

III het sturen van de beeldgegevens naar en het tekenen op het beeldscherm (T4115) van de afbeelding.

De maatregelen die eventueel genomen zullen worden moeten niet worden gezocht in bijvoorbeeld het in assembler taal schrijven van grafische routines, die in GINO nogal wat overhead veroorzaken. Wel kunnen de maatregelen bestaan uit een ander gebruik van GINO (en ORACLE) en een andere architectuur van het systeem.

Voorbeeld

In GAMMA wordt een raster graphics afbeelding (van een vliegtuig) uit vele panelen, dat wil zeggen polygonen met vier hoekpunten, opgebouwd. Aan GINO worden de coördinaten van de hoekpunten en een code voor de kleur doorgegeven. Dit polygoon wordt op het scherm ingekleurd. Stel nu dat fase III de "bottle neck" is. Dan zou men het afbeeldingsproces kunnen versnellen door aan elkaar grenzende panelen met een zelfde kleur samen in één polygoon te zetten. Er hoeven nu minder coördinaten overgezonden te worden. Sommige worden minder vaak en andere worden zelfs helemaal niet meer overgezonden, indien het geval zich voordoet dat een paneel helemaal omsloten wordt door panelen met dezelfde kleur.

Fase III zal dus sneller verlopen, maar doordat eerst zal moeten worden uitgezocht welke panelen samengenomen kunnen worden, zal fase II nu langer duren. Omdat fase III de "bottle neck" was zal het afbeeldingsproces nu sneller verlopen, mits fase II niet te snel toeneemt in rekentijd. Als fase II de "bottle neck" was zou het afbeeldingsproces niet sneller maar juist nog langzamer verlopen.

Uit het bovenstaande blijkt dat het nodig is enige kwantitatieve inzichten te hebben in de diverse fasen van het afbeeldingsproces. Voor fase I, het ophalen van de gegevens uit de database, zijn een aantal snelheidsmetingen verricht tijdens het testen van ORACLE. Dit in het kader van de aanschaf van een relationele database. Hieruit kon worden afgeleid dat het inlezen van een

totale configuratie (25 segmenten) ongeveer 10 sec. wachttijd kost. Later (dus gedurende de implementatie) bleek echter, dat door een andere opzet van de database dan welke tijdens het testen was gebruikt, dit in de praktijk de onaanvaardbare tijd van ± 5 min. zou worden.

De communicatiesnelheid van de fysieke verbindingslijn tussen de VAX en de terminal (T4115) is 9600 baud, uitgaande van twee signaalniveaus betekent dit 9600 bps. De grafische en alfanumerieke informatie wordt in buffers (packets) van 200 bytes verzonden van de VAX naar de terminal. Deze beginnen en eindigen met een aantal bytes (± 2%) voor initialisatie e.d. Bijvoorbeeld: na elke tekenopdracht wordt de grafische terminal weer in alfanumerieke mode gezet. Voor elke tekenopdracht moet de terminal dus weer in grafische mode gezet worden.

Om een beter inzicht te verkrijgen zijn er een aantal snelheidstests uitgevoerd. In deze snelheidstests worden geen gegevens uit de database opgehaald en vindt er nagenoeg geen rekenwerk (uit fase II) plaats. In totaal zijn er 200 snelheidsmetingen verricht verdeeld over acht snelheidstests. Voor het tekenen en het inkleuren van polygonen (bijvoorbeeld: driehoeken, panelen of sterren) kunnen twee methoden gebruikt worden, namelijk:

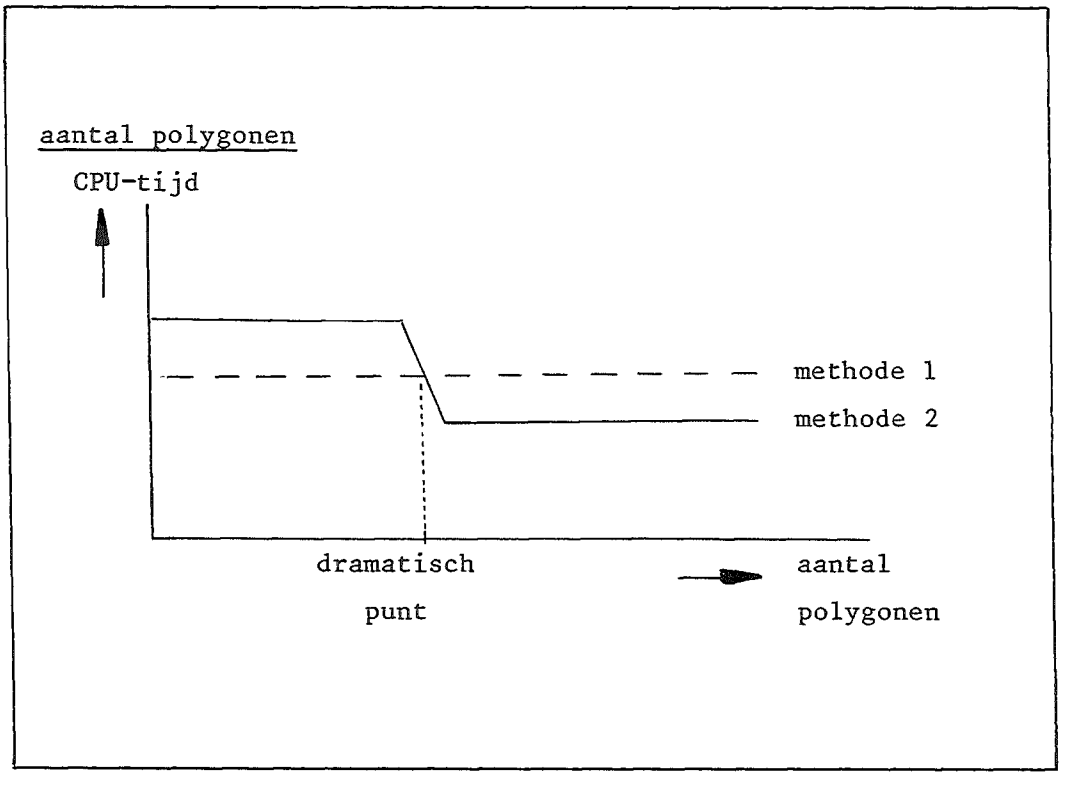
1) Eén polygoon beschrijven (opslaan van hoekpunten in een GINO buffer) en bepalen van de kleur. Vervolgens het tekenen (en inkleuren) van deze ene polygoon op het scherm en de GINO buffer (bevindt zich in het "blank common" blok van Fortran) weer leegmaken. Dit proces herhalen voor elke polygoon die getekend moet worden (dit is de methode die ook in GAMMA gebruikt wordt).

2) Definiëren van alle polygonen (inclusief een code voor de kleur) en deze allemaal in een grote GINO buffer stoppen. Vervolgens kleur voor kleur (dat zijn er op dit ogenblik 16) de bijbehorende polygonen selecteren en tekenen.

De polygonen kunnen met of zonder randen getekend worden. Uit de snelheidstesten (ref. 46) kunnen de volgende globale conclusies getrokken worden:

1) Bij methode 1 (het polygoon voor polygoon tekenen en inkleuren en deze niet opslaan), is de totale tijd die een afbeelding kost bij benadering recht evenredig met het aantal te tekenen polygonen.

2) Bij methode 2 (het kleur voor kleur tekenen en alle polygonen opslaan) is er een dramatisch punt bij ongeveer 2000 panelen, waarna de snelheid (gerelateerd aan het aantal te tekenen polygonen) sterk daalt; zie de doorgetrokken lijn in figuur 3.7. Methode 1 is in deze figuur door middel van een gestreepte lijn weergegeven.

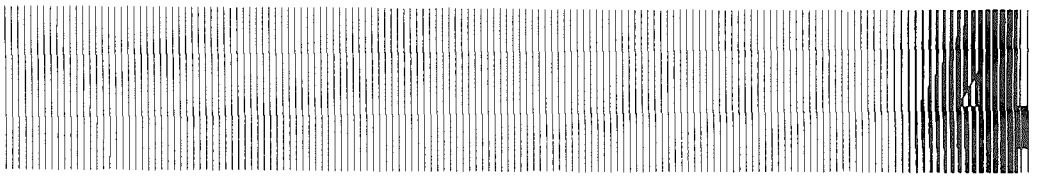


Figuur 3.7 - Vergelijk van methode 1 en 2

Het blijkt dat methode 2 voor het dramatische punt ca. 25 % sneller is dan methode 1. Na het dramatische punt geldt precies het omgekeerde. Het zal dus verstandig zijn om methode 2 per segment toe te passen, omdat in de meeste gevallen het aantal panelen per segment nog voor het dramatische punt ligt.

3) Bij het tekenen van zeer grote hoeveelheden panelen met behulp van methode 2 blijkt dat ongeveer halverwege de wachttijd alle gegevens overgezonden zijn. Dit betekent dat de T4115 alle pixels langzamer berekent, dan dat deze de informatie die hiervoor nodig is binnen krijgt. Hieruit blijkt dat in de normale situatie de host-computer, de terminal en de verbinding tussen deze twee, redelijk goed gebalanceerd zijn.

4) Het tekenen van randen heeft een snelheidsvermindering met een factor 1,7 à 1,9 tot gevolg. Als de randen echter worden weggelaten ontstaat er tussen de verschillende polygonen (die wel aansluiten, omdat ze gemeenschappelijke hoekpunten hebben) kieren met de achtergrond kleur. Met behulp van de PROGRAM MODE EXTENDED-PANEL (zie 3.5) kan de gebruiker kiezen of hij de randen wel of niet wil tekenen.



Implementatie

In dit hoofdstuk zullen de fasen 4 en 5, dat zijn respectievelijk "detail-ontwerp en implementatie" en "integratie en test", aan de orde komen. Om dit verslag leesbaar te houden zal met de term "implementatie" zowel fase 4 als fase 5 worden bedoeld. Omdat deze fasen zijn samengenomen wegens het relatief kleine project, zullen ze ook in dit hoofdstuk gezamelijk behandeld worden. Wat de doelen van deze fasen zijn is te vinden in 4.1. In 4.2 staat vermeld welke richtlijnen en standaarden er tijdens de implementatie zijn aangehouden. Hoe de datastructuur (zie 3.3) en het deelsysteem DIRSEL geïmplementeerd zijn staat in 4.3. De meest interessante zaken tijdens de implementatie van het deelsysteem 3D-FUNCTIE, zoals bijvoorbeeld isobaren die continu moeten verlopen, staan vermeld in 4.4. Het deelsysteem SET-UP wordt behandeld in 4.5 en in 4.6 wordt ten slotte de opzet van de gebruikershandleiding uitgewerkt.

4.1 - Het doel van de implementatie

Zoals in de introductie van dit hoofdstuk reeds is vermeld zullen nu aan de orde komen: detailontwerp, implementatie, integratie en test. De modules die tijdens het ontwerp zijn aangeduid moesten eerst in detail worden uitgewerkt. Daarna konden ze worden geïmplementeerd en ten slotte moesten ze getest worden.

Door de gevolgde aanpak tijdens deze fasen is de integratie en het testen eenvoudig verlopen. De aanpak was als volgt: eerst de globale systeemopzet (het raamwerk) uit 3.5 implementeren en testen (hiervoor was het ontwerp reeds in voldoende detail beschikbaar), daarna de deelsystemen SET-UP en DIRSEL toevoegen, door steeds met een aantal routines uit te breiden. Als de globale systeemstructuur gereed is kunnen de visualisatie routines één voor één uitgewerkt worden, dat wil zeggen detailontwerp, implementatie en aan het systeem toevoegen. Door de aard van het systeem, het visualiseren van gegevens, is al snel duidelijk of de routine enigszins correct is.

Om de continuïteit te bevorderen werden de routines in een zo vroeg mogelijk stadium getest. Dit is niet alleen gedaan door degene die het systeem moest ontwikkelen, maar ook door de toekomstige gebruikers, welke reeds in een zeer vroeg stadium met het systeem konden werken. Door deze werkwijze konden de gebruikers nog aanvullende wensen kenbaar maken en verliep de integratie op een zeer natuurlijke wijze.

Toen een groot deel van het systeem gereed was, werd er een gebruikers-handleiding samengesteld. Door het verspreiden van deze handleiding werd de toegankelijkheid van het systeem vergroot en kon het aantal gebruikers van het systeem toenemen. Het gevolg was dat nog meer mensen het systeem konden "testen".

4.2 - De gevolgde richtlijnen en standaarden

Om het te ontwikkelen systeem overzichtelijk en goed onderhoudbaar te maken moesten er diverse richtlijnen en standaarden in acht genomen worden. Om te beginnen ten aanzien van de programmeertaal. Er is uitgegaan van de beschrijving van Fortran 77 zoals dat gebeurd is in het boek van M. Merchant; ref. 47. Er is bewust gekozen om geen gebruik te maken van de VAX systeem documentatie over Fortran 77. Zo zijn er op de VAX fraaie constructies voor DO en WHILE-loops zonder labels. Hier is echter geen gebruik van gemaakt in verband met de overdraagbaarheid naar andere computersystemen.

Op twee uitzonderingen na is er alleen gebruik gemaakt van standaard Fortran 77. Deze twee uitzonderingen zijn:

1) Het "IMPLICIT NONE" statement. Dit statement komt altijd direct na de heading van elke programma eenheid. Het zorgt ervoor dat er tijdens het compileren een foutmelding wordt gegeven als er een variabele aanwezig is die niet is gedeclareerd. Hiermee wordt voorkomen dat door bijvoorbeeld typefouten ongewenste variabelen geïntroduceerd worden. Als het systeem eenmaal gereed is kunnen indien gewenst deze "IMPLICIT NONE" statements weer verwijderd worden, zodat men toch weer standaard Fortran 77 krijgt.

2) Het "INCLUDE" statement. Dit statement wordt gebruikt om common blokken, die in meerdere programma eenheden voorkomen, te beschrijven. De common blokken worden beschreven in speciale files, welke altijd beginnen met de letters "CM" gevolgd door vier letters, die de naam van de common blokken aanduiden en de filenaam eindigt met ".INC". Het include statement doet niets anders dan de letterlijke tekst, van de file die "geïncluded" moet worden, overnemen op de plaats van het statement. Zo wordt gegarandeerd dat alle common blokken op dezelfde wijze worden beschreven. De include statements staan altijd boven in de programma eenheden. Als het systeem eenmaal gereed is kan men met een editor de include statements vervangen door de bijbehorende files, zodat men toch weer standaard Fortran 77 krijgt.

Er is tijdens de implementatie getracht gestructureerd te programmeren, iets dat met Fortran 77 enige dicipline vergt. Dit is gedaan door tijdens de implementatie uit te gaan van structuurdiagrammen, die tijdens het detail-

ontwerp gemaakt zijn. Op belangrijke plaatsen in het programma is commentaar toegevoegd door een "C" in kolom 1 te zetten. Commentaar staat vaak in de kop van een programma eenheid en kan onder andere zaken bevatten als het doel, de vooronderstellingen, de beperkingen, de invoer, de uitvoer, de gebruikte methode en indien van toepassing de datum en reden van wijziging. Verder is er gebruik gemaakt van een overzichtelijke lay-out in geval van loops, IF-statements, etc. Er is geen gebruik gemaakt van GOTO statements, behalve in de volgende twee situaties:

1) Om een WHILE-loop structuur te maken. Aangezien Fortran 77 niet beschikt over een WHILE-loop moet deze gecreëerd worden met behulp van een IF en een GOTO statement.

2) In geval van zogenaamde "exceptions". Treedt er een uitzonderings-situatie op, zoals bijvoorbeeld verkeerde data en heeft het geen zin om de programma eenheid verder uit te voeren dan kan naar het einde van de programma eenheid worden gesprongen, waar eventueel passende maatregelen getroffen kunnen worden.

Verder worden de richtlijnen gevolgd voor het opstellen van computer programma's welke gelden binnen de afdeling TOAP van Fokker; ref. 32. De gehele source code van VISUAL is verdeeld over een groot aantal files, welke eenvoudig met de editor aangepast en vervolgens weer gecompileerd kunnen worden. Deze files bevatten allemaal routines, welke de volgende vaste structuur hebben:

- Heading;
- "IMPLICIT NONE" statement;
- Aantal regels commentaar waarin de functie en de werking van de routine worden beschreven. Ook wordt de betekenis van de diverse parameters uit de heading duidelijk gemaakt;
- Declaratie van de formele parameters uit de heading;
- Declaratie van de common blokken, die door deze routine gebruikt worden, door middel van het "INCLUDE" statement;
- Declaratie van de lokale variabelen;
- De body van de routine in Fortran 77 voorzien van commentaar;
- De statements "RETURN" en "END" welke de routine afsluiten. Er komt slechts één "RETURN" statement voor per routine.

In figuur 4.1 is een denkbeeldige routine volgens de hiervoor beschreven structuur uitgewerkt.

```
      SUBROUTINE DEMO(A,B,C)
      IMPLICIT NONE
C     DEZE ROUTINE HEEFT EEN INVOERPARAMETER, NL. A. DE UITVOER
C     PARAMETERS B EN C KRIJGEN RESPECTIEVELIJK ALS WAARDEN:
C     A*FASE EN ABS(A+B).
C     HIERIN IS "FASE" EEN VARIABELE UIT EEN COMMON BLOK.
      REAL A,B,C

C     COMMON BLOK
      INCLUDE 'CMDIVS.INC'

C     LOKALE VARIABELE
      REAL TEMP

C     DE BODY VAN DE ROUTINE
      TEMP=A
      B=TEMP*FASE
      C=ABS(A+B)

      RETURN
      END
```

Figuur 4.1 - Een routine met de vaste structuur

De gecompileerde routines worden in de library "VISUAL.OLB" opgeslagen. Deze wordt dan samen met het hoofdprogramma en de ondersteunende software, zoals COALS, GINO-F en ORACLE (zie 2.6) gelinkt tot een executable code welke in de file "VISUAL.EXE" wordt gezet. Een overzicht van alle filenamen welke voor VISUAL van belang zijn is te vinden in appendix F.

4.3 - Het selecteren in het deelsysteem DIRSEL

Voordat wordt besproken hoe het selecteren (en andere routines) in het deelsysteem DIRSEL is geïmplementeerd zal eerst worden beschouwd hoe de datastructuur, die in 3.3. beschreven is, zal worden geïmplementeerd. Dit is namelijk de basis waarvan bij het selecteren wordt uitgegaan omdat hierin de gegevens moeten worden opgeslagen. We hebben gezien dat de datastructuur dynamisch van aard is, omdat de hoeveelheid gegegevens variëren.

Fortran 77 beschikt echter niet over dynamische geheugenallocatie en daarom zal er zelf een systeem moeten worden ontworpen waarmee dat wel mogelijk is. Dit gebeurt door één zeer grote array (de "POOL") te declareren en binnen deze array de geheugen administratie zelf bij te houden. In deze array kan zowel REAL, INTEGER als CHARACTER data worden opgeslagen. Dit kan doordat er gebruik is gemaakt van het EQUIVALENCE statement. De gebruiker van dit mechanisme, de programmeur dus, kan via routines, waarin onder andere meestal een buffer als parameter wordt meegegeven, gebruik maken van het dynamische geheugen.

Dit hierboven beschreven mechamisme is overgenomen van het "GKS DYNAMIC MEMORY SYSTEM"; ref. 45. De routine-namen beginnen allemaal met de vier letters "POOL", gevolgt door twee letters die de functie van de routine aanduiden. Het systeem bevat onder andere routines voor:

- het openen en sluiten van het systeem (POOLOP en POOLCL);
- het aanvragen en vrijgeven van een blok (POOLNW en POOLRL);
- het opbergen van data in het systeem (POOLPT, POOLPR, POOLPI en POOLPC);
- het ophalen van data uit het systeem (POOLGT, POOLGR, POOLGI en POOLGC);
- het copiëren van data binnen het systeem (POOLCP en POOLCF).

Met behulp van pointers, welke door het dynamische geheugensysteem verzorgd worden, wordt er verwezen naar blokken,die de data bevatten, die van variabele omvang is. Variabelen welke pointers bevatten zijn van het type integer en eindigen met de twee letters "PT". Bij de implementatie van de datastructuur uit 3.3 zijn drie constanten van belang, namelijk het maximum aantal parts, segmenten en isobaren dat moet kunnen worden opgeslagen. Alle overige "afmetingen" zijn variabel. De constanten MAXPA, MAXSE en MAXISO behoeven slechts op één plaats in het programma veranderd te worden bij een wijziging. De uiteindelijke implementatie van de datastructuur, gebruik makende van het dynamische geheugensysteem, is te vinden in appendix G.

Het selecteren van gegevens vindt plaats in een aantal fasen. In de eerste fase is er nog niets geselecteerd. In de tweede fase is er één configuratie geselecteerd en in de volgende fasen worden er steeds meer soorten objecten geselecteerd. Dit is dan onder meer afhankelijk van de selectie mode; zie 3.5. In figuur 4.2 is het verband tussen de fase, selectie mode en de geselecteerde objecten duidelijk gemaakt.

MODE / FASE	SEGMENT MODE	PART MODE	GROUP MODE
1	-	-	-
2	C	C	C
3	C S	C P	C G
4	C S A	C P S	C G P
5	n.v.t.	C P S A	C G P S
6	n.v.t.	n.v.t.	C G P S A

betekenis

- = niets geselecteerd
C = configuratie geselecteerd
G = group geselecteerd
P = één of meer parts geselecteerd
S = één of meer segmenten geselecteerd
A = aerodynamische gegevens geselecteerd
n.v.t. = niet van toepassing; situatie kan nooit voorkomen

Figuur 4.2 - Het verband tussen fase, selectie mode en geselecteerde gegevens

Voordat de SELECT routine wordt aangeroepen wordt eerst getest of de FASE correct is. Als bijvoorbeeld geldt dat FASE = 2 en SELETION MODE = "GROUP" en de gebruiker wil segmenten selecteren dan is dat niet mogelijk, omdat er eerst nog één group en een aantal parts geselecteerd moeten worden. Hiervan moet een

melding worden gegeven en de selectie mag niet worden uitgevoerd. Een andere situatie treedt bijvoorbeeld op als FASE = 4 en SELECTION MODE = "GROUP" terwijl de gebruiker een (andere) configuratie wil selecteren. Dit is wel mogelijk, doch dan moeten de reeds eerder geselecteerde objecten verwijderd worden en moet er een waarschuwing gegeven worden voordat de selectie wordt uitgevoerd. Na een geslaagde selectie opdracht wordt de FASE met één opgehoogd.

De meldingen, zoals foutmeldingen en waarschuwingen, worden door COLAS verzorgd. Elke melding heeft een eigen identificatie nummer en wordt hiermee vanuit VISUAL aangeroepen. De routine (genaamd FASEOK) die test of de FASE correct is, heeft als basis een drietal twee-dimensionale tabellen (array's), welke met het DATA statement gevuld zijn. Voor elke selectie mode is er zo'n twee-dimensionale tabel. De eerste index van de tabel geeft de FASE aan en de tweede index geeft het type object aan, dat wil zeggen configuratie, group, part, etc. De tabel bevat een positief nummer als er een foutmelding gegeven moet worden, een negatief nummer als er een waarschuwing gegeven moet worden en een nul als de fase precies goed is. De absolute waarde van het nummer geeft aan welke COLAS melding er gegeven moet worden. Via een uitvoerparameter geeft de routine FASEOK aan of de FASE correct is, dus als het nummer kleiner of gelijk is aan nul. De "kennis" van deze routine is dus opgeslagen in de vorm van een aantal gevulde tabellen.

Ook de DIRECTORY routine wordt voorafgegaan door een routine die test of de fase correct is. Dit gebeurt op eenzelfde wijze als hierboven beschreven. Het is mogelijk dat de gebruiker geen type object opgeeft bij het selecteren of bij een directory. In dit geval wordt dan een default type genomen afhankelijk van de selectie mode en fase. Als bijvoorbeeld geldt dat FASE = 3 en SELECTIE MODE = "PART" dan is het default type object een "SEGMENT".

De routines ADD en DROP werken op eenzelfde wijze als de routines DIRECTORY en SELECT en moeten op een correcte wijze de FASE bijwerken. In geval van DROP betekent dit dat de FASE verminderd wordt. De body van de hier genoemde routines DIRECTORY, SELECT en ADD bevatten aanroepen van routines uit DIADBF (zie 2.3), welke de interface met de ORACLE database vormen. De overige routines uit het deelsysteem DIRSEL en het deelsysteem zelf zijn minder interessant en de implementatie hiervan zal hier niet worden besproken. In appendix C is een overzicht van de commando's opgenomen welke zich in het deelsysteem DIRSEL bevinden.

4.4 - Het deelsysteem 3D-FUNCTIE

De functies in dit deelsysteem vormen het hart van VISUAL. De overige deelsystemen, DIRSEL en SET-UP, hebben alleen als doel het deelsysteem 3D-FUNCTIE goed te laten functioneren. Welke functies dit deelsysteem moet bevatten is reeds beschreven in 2.4. De structuur van 3D-FUNCTIE is, evenals de globale systeemopzet en de andere deelsystemen, één groot CASE statement (zie 3.5). In dit CASE statement worden alle mogelijke commando's opgenomen, welke dan via routine aanroepen afgehandeld worden. In appendix C is ook een overzicht opgenomen van alle commando's, welke zich in het deelsysteem 3D-FUNCTIE bevinden.

De functies in het deelsysteem 3D-FUNCTIE zijn in twee groepen te verdelen, te weten, de functies die geometrische transformaties etc. verzorgen en de functies die het echte tekenwerk verrichten. Tot de eerste groep behoren onder andere CLEAR, ZOOM, ROTATE, SHIFT, VIEW en MATRIX. Deze functies verrichten allemaal manipulaties met de transformatie matrix. Daar dit voornamelijk via GINO routines gebeurt is het minder interessant om het detailontwerp en de implementatie hier nog verder te bespreken.

De tweede groep van functies in het deelsysteem bestaat onder andere uit WIRE, OUTLINE, HIDDEN, SHADE, VELOCITY, TUFT, COLOUR-CP en NUMBER-CP. Ten slotte zijn er de functies ISO-LINES, ISO-BAND, CONTOUR en CARPET. Opgemerkt wordt dat de twee laatste functies wegens tijdgebrek helaas niet geïmplementeerd zijn. De structuur van de functies ISO-LINES en ISO-BAND is enigszins afwijkend van de structuur van de overige routines en zal daarom later in deze paragraaf behandeld worden.

De 3D-tekenfuncties

De functies WIRE tot en met NUMBER-CP hebben allen een soortgelijke structuur. Daarom zal de opbouw van deze routines gezamenlijk worden beschreven. De transformatie matrix is via GINO reeds van een juiste inhoud voorzien voordat deze routines worden aangeroepen. Dit kan zijn gebeurd doordat de gebruiker zelf een transformatie heeft uitgevoerd of doordat VISUAL het model automatisch heeft geschaald. Dit laatste gebeurt door de extreme waarden van de coördinaten van het model op een zo goed mogelijke wijze te passen op de afmetingen van het beeldscherm waar de gebruiker mee werkt.

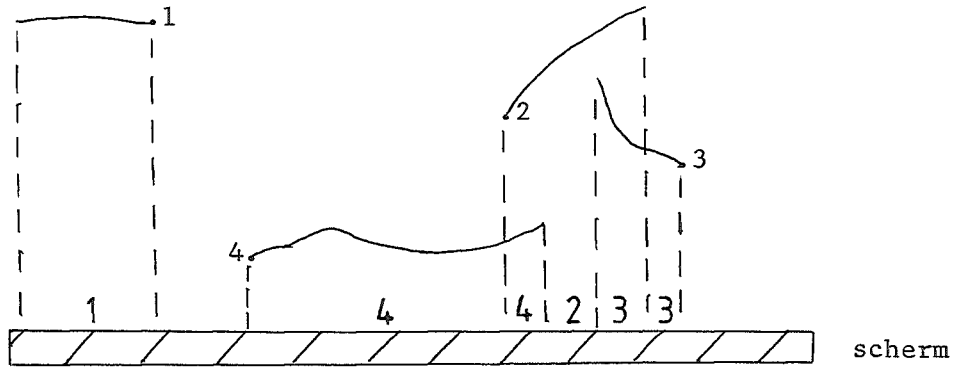
Om ervoor te zorgen dat er een snelle "hidden surface removal" plaatsvindt zijn er een tweetal maatregelen genomen. Opgemerkt wordt dat dit enigszins primitieve, doch zeer snelle maatregelen zijn waarmee niet in alle gevallen een perfect plaatje wordt gegenereerd. In de praktijk is echter gebleken dat deze maatregelen in de meeste gevallen toch voldoen. Dit komt doordat de geometrie dusdanig is gedefinieerd dat deze goed te verwerken is. Het zijn dus geen ingewikkelde gekromde segmenten waarmee gewerkt wordt. De twee maatregelen zijn:

1) Sorteer de segmenten zodanig dat het segment met de grootste minimale afstand tot het scherm als eerste en het segment met de kleinste minimale afstand als laatste wordt getekend. In figuur 4.3 zijn twee bovenaanzichten gegeven van situaties waarin zich een viertal segmenten bevinden. In situatie I zal het model goed worden afgebeeld en in situatie II zal het afbeelden fout gaan.

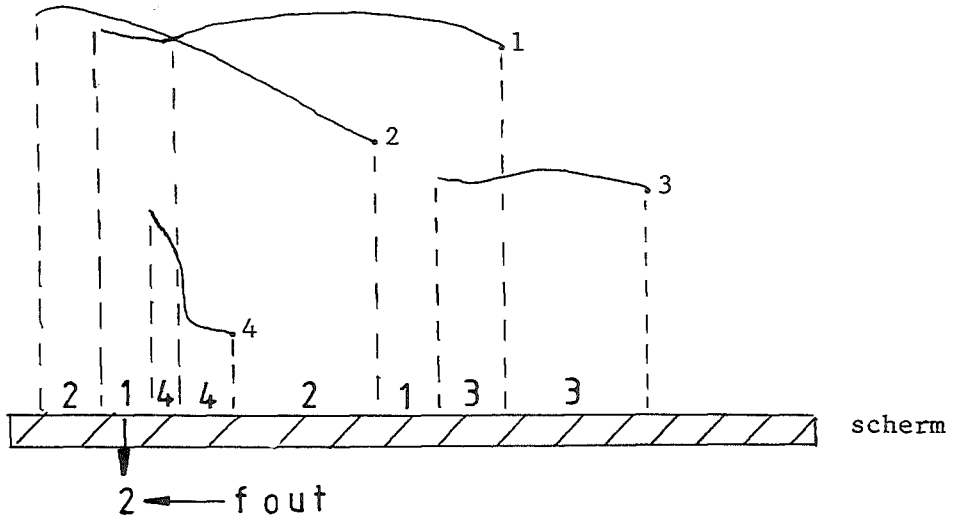
2) Voordat een functie op een paneel wordt uitgevoerd, wordt eerst gecontroleerd of de normaalvector van dit paneel een positieve z-component heeft. Dat wil zeggen, dat de normaal uit het scherm richting de gebruiker wijst en dus zichtbaar is. Is dit niet het geval dan wordt de functie niet uitgevoerd.

In figuur 4.4 is het detailontwerp van de 3D-tekenfuncties met behulp van pseudo code weergegeven. De twee hierboven genoemde maatregelen voor de "hidden surface removal" zijn hierin opgenomen.

I) In deze situatie gaat het goed



II) In deze situatie gaat het fout



betekenis: ~ = een segment

. = punt van segment met minimale afstand tot het scherm

3 = nummer dat de volgorde aangeeft

Figuur 4.3 - Het sorteren van de segmenten (bovenaanzicht)

```
IF FASE AND TERMINAL O.K. THEN
BEGIN
      HAAL PARAMETERS DIE BIJ DEZE FUNCTIE HOREN OP
      SORTEER DE SEGMENTEN VOLGENS MAATREGEL 1
      FOR ALLE GESELECTEERDE SEGMENTEN DO
      BEGIN
           HAAL DE BENODIGDE DATA OP UIT DE GEHEUGEN POOL
           (BIJ VOORBEELD: COORDINATEN, NORMAALVECTOREN, ETC.)
           IF ALLE DATA AANWEZIG THEN
           BEGIN
                FOR ALLE PANELEN IN SEGMENT DO
                BEGIN
                     VOER DE FUNCTIE OP PANEEL UIT
                END
           END
      END
END
ELSE
BEGIN
      GEEF MELDING FASE OF TERMINAL NIET O.K.
END
```

Figuur 4.4 - Het detailontwerp van de 3D-tekenfuncties

Het resultaat van de implementatie is te vinden in appendices A en B. De functies SHADE (de lichtintensiteit waarmee een paneel wordt ingekleurd is afhankelijk van de hoek tussen de normaalvector en de "richtingsvector" van het licht) en COLOUR-CP staan in respectievelijk A1 en A2. De functies WIRE, OUTLINE, HIDDEN, VELOCITY, TUFT en NUMBER-CP zijn te vinden in respectievelijk B1, B2, B3, B4, B5 en B6.

De functies ISO-LINES en ISO-BAND

Bij deze functies wordt niet de paneelgewijze verwerking gekozen, zoals dat wel gebeurt bij de meeste andere tekenfuncties, maar er wordt uitgegaan van een isobaargewijze verwerking binnen een segment. Het tekenen van deze functies verloopt in een aantal fasen, namelijk:

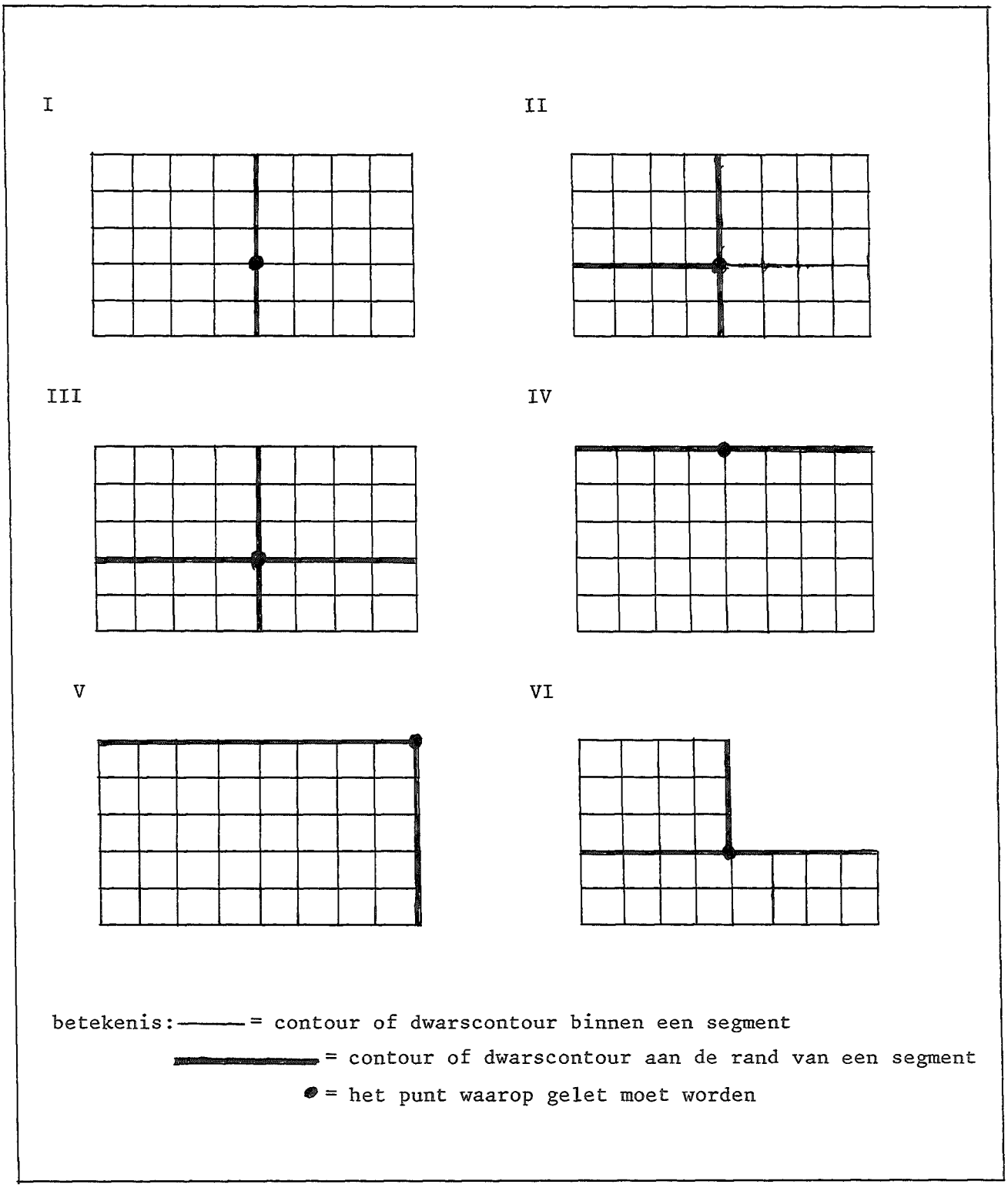
1) Het uitrekenen van isobaren en het opslaan van deze isobaren in de datastructuur; zie 3.3 en 4.3. Deze fase behoeft voor de geselecteerde aerodynamische drukverdeling maar één keer te worden uitgevoerd. Na bijvoorbeeld een rotatie behoeven deze isobaren niet opnieuw uitgerekend te worden.

2) Het tekenen van de isobaren. Voor de functie ISO-LINES ligt het voor de hand hoe dit moet gebeuren en dit zal dan ook niet verder worden beschreven. Wel moet er rekening gehouden worden met het feit dat het drukniveau eenmaal per segment en per isobaar moet kunnen worden aangegeven. Bij de functie ISO-BAND worden de panelen welke door isobaren worden doorsneden op een juiste wijze met meerdere kleuren gevuld. Het is nog wel noodzakelijk dat ook de panelen, die niet door een isobaar doorsneden worden, alsnog worden ingekleurd. Dit gebeurt in een speciale paneelgewijze verwerkingsslag.

Het uitrekenen van de isobaren binnen één segment kan nu nog op twee manieren plaatsvinden, namelijk door wel of geen rekening te houden met aangrenzende segmenten welke het verloop van de isobaren in het segment in kwestie kunnen beïnvloeden. Indien er geen rekening wordt gehouden met aangrenzende (buur)segmenten dan is het rekenwerk wat eenvoudiger en sneller, maar de isobaren zullen niet continu verlopen. Wordt er wel rekening met buursegmenten gehouden dan zullen de isobaren meer continu verlopen. Het is echter nog steeds mogelijk dat de isobaren niet geheel continu verlopen. Dit komt doordat het model dat met behulp van GAMMA is gedefinieerd, op bepaalde punten nogal wat gebreken vertoont; bijvoorbeeld een gat of kier tussen de aansluiting van twee segmenten. Indien segmenten wel goed aansluiten dan moeten de isobaren ook continu verlopen over segmentgrenzen.

Bij het berekenen van de isobaren, rekening houdende met "grenscorrecties", is vooral de datastructuur, waarin de relaties tussen de aangrenzende (of buur) segmenten wordt opgeslagen van belang; zie 3.3. Voordat dit gedeelte van de datastructuur en het berekenen van de isobaren wordt geïmplementeerd, zal

eerst worden aangegeven in welke situaties er isobaren berekend moeten worden die continu verlopen. Dit is gebeurd door middel van figuur 4.5 waarin de zes belangrijkste situaties zijn weergegeven.



Figuur 5.5 - Situaties waarin isobaren continu moeten verlopen

Om de situaties uit figuur 4.5 goed te kunnen behandelen is besloten om per randpunt van een segment (dat wil zeggen punten die op de eerste of laatste primaire of secundaire contour liggen) één verwijzing naar een buurpunt op te nemen. Om voor hoekpunten van een segment (dat wil zeggen punten die zowel op de eerste of laatste primaire contour als op de eerste of laatste secundaire contour liggen) drie verwijzingen naar buurpunten op te nemen. Dit om op eenvoudige wijze de situaties II en III uit figuur 4.5 correct te verwerken.

Een verwijzing naar een buurpunt bestaat uit een combinatie van drie gehele getallen, waarvan het eerste het segmentnummer van het buurpunt aanduidt en het tweede en derde getal respectievelijk het nummer van de primaire en secundaire contour aanduiden. Heeft een punt geen buur dan krijgt het getal dat het buursegment aanduidt de waarde -1. Bij het bepalen van een buur wordt gezocht naar randpunten uit segmenten, welke niet te ver weg liggen zodat ze mogelijk een buur zouden kunnen bevatten. Het randpunt dat de kortste afstand heeft tot het punt in kwestie wordt het buurpunt, mits de afstand kleiner is dan een bepaalde "maximum afstand". Deze "drempel afstand" is onder andere afhankelijk van de grofheid van het segment waarin het punt zich bevindt en is experimenteel zodanig bepaald dat het een goed resultaat oplevert.

Omdat Fortran over een zeer beperkte hoeveelheid datatypen beschikt, zijn er een aantal routines geschreven, waarmee de datastructuur waarin de buren zijn opgeslagen (natuurlijk weer als een array geïmplementeerd) toegankelijk gemaakt is. Er zijn onder andere routines voor het opbergen en ophalen van buren (BUURPUT en BUURGET) en voor het opbergen, ophalen en toevoegen van buren van hoekpunten (BUURHPUT, BUURHGET en BUURHADD). Dergelijke routines zijn overigens ook ontwikkeld voor het ophalen van geometrische en aerodynamische gegevens vanuit buffers die gevuld zijn met delen van de geheugen POOL

Het verschil tussen de isobaren, die wel of niet met grenscorrecties zijn berekend, is goed zichtbaar aan de hand van de functie ISO-BAND in appendix A. In A4 zijn beide gevallen (voor de cockpit van de Fokker 100) vlak boven elkaar weergegeven. De functie ISO-BAND met grenscorrectie is voor een complete configuratie in A3 getekend. In appendix B is de functie ISO-LINES met een groot aantal isobaren weergegeven in B7. Het zal duidelijk zijn dat hierop de grenscorrectie is toegepast.

4.5 - De SET-UP van VISUAL

In 3.5 is de noodzaak van de PROGRAM MODES en het deelsysteem SET-UP beschreven. Nu de beschrijving van de implementatie van de deelsystemen DIRSEL en 3D-FUNCTIE al heeft plaatsgevonden is deze op enkele uitzonderingen na voor het deelsysteem SET-UP minder interessant. Het deelsysteem heeft weer dezelfde "CASE" structuur als de overige deelsystemen.

Het detailontwerp, de implementatie, het testen en de integratie van de routines MODE, STORE, SYSTEM-DEF, USER-DEF, INFO-VAR en CHANGE-VAR heeft weinig problemen opgeleverd. Wel moest er worden opgepast dat een waardeverandering van bepaalde PROGRAM MODES door een van de functies SYSTEM-DEF, USER-DEF of CHANGE-VAR soms ingrijpende gevolgen kan hebben. Zo vervallen bij een waardeverandering van SELECTION-MODE alle reeds eerder geselecteerde objecten, behalve de configuratie. Dit wordt dan op een soortgelijke wijze als bij het commando DROP in het deelsysteem DIRSEL (administratief correct) verwerkt. Ook krijgt de gebruiker weer een melding van wat er heeft plaatsgevonden. Een waardeverandering van de PROGRAM MODE BOUNDARY-CORRECTION heeft verstrekkende gevolgen. Als deze namelijk de waarde "true" krijgt dan moeten de relaties tussen de buursegmenten worden uitgerekend en in de datastructuur opgeslagen worden.

Met de overige commando's in het deelsysteem SET-UP kan de gebruiker één of meer kleurdefinities veranderen. Een bepaalde kleurdefinitie wordt aangeduid met een identificatie nummer. Op het moment dat VISUAL ontwikkeld werd waren er 16 kleuren tegelijk mogelijk; dit kan tot 256 uitgebreid worden. De identificatie nummers lopen van 0 tot en met 15 en in de toekomst wellicht tot 255. De functies waarmee de kleurdefinities veranderd kunnen worden zijn alleen zinvol wanneer de gebruiker met een kleurenscherm werkt.

De routines die hiervoor ontwikkeld zijn maken gebruik van de GINO-F routines om kleuren samen te stellen. Bij kleuren zijn begrippen als "hue", "lightness" en "saturation" van belang. De betekenis van deze begrippen is te vinden in de verklarende woordenlijst in appendix D. De volgende vier functies zijn aanwezig:

1) HUE-COLOUR. Met deze functie worden de kleuren met identificatie nummers van COL-FROM naar COL-TO opnieuw gedefinieerd met een hue variërend van HUE-FROM naar HUE-TO. De lightness en saturation

liggen vast. COL-FROM, COL-TO, HUE-FROM en HUE-TO zijn parameters die bij het commando HUE-COLOUR behoren.

2) LIGHTNESS. Met deze functie worden de kleurdefinities met identificatie nummers van COL-FROM naar COL-TO opnieuw gedefinieerd met lightness waarden die variëren van 0 (donker) naar 1 (licht). De hue waarde wordt bepaald door de parameter HUE en de saturation ligt vast. COL-FROM, COL-TO en HUE zijn de parameters die bij het commando LIGHTNESS behoren.

3) SATURATION. Met deze functie worden de kleurdefinities met identificatie nummers van COL-FROM naar COL-TO opnieuw gedefinieerd, zodanig dat de saturation van alle kleuren de waarde van de parameter SATURATION krijgt. De hue en lightness van de kleuren is gelijk aan de waarden welke ze hadden voordat de functie SATURATION gebruikt werd. COL-FROM, COL-TO en SATURATION zijn de parameters die bij het commando SATURATION behoren.

4) COLOUR-DEF. Met deze functie wordt één kleurdefinitie aangepast. Dat is de kleur met identificatie nummer COLOUR-NR, deze krijgt als hue, lightness en saturation de waarden HUE, LIGHTNESS en SATURATION. COLOUR-NR, HUE, LIGHTNESS en SATURATION zijn de parameters die behoren bij het commando COLOUR-DEF.

Tot zover de implementatie van het deelsysteem SET-UP. In appendix C is een overzicht van alle commando's in dit deelsysteem te vinden.

4.6 - De opzet van de gebruikershandleiding

VISUAL vormt het gereedschap waarmee de aerodynamische ontwerper op een efficiënte wijze de gegevens welke in de ORACLE database zijn opgeslagen kan analyseren. Van de ontwerper mag niet worden verwacht dat hij een computer expert is. Daarom moeten het programma en de gebruikershandleiding zodanig worden gemaakt dat deze zeer toegankelijk zijn voor de ontwerpers.

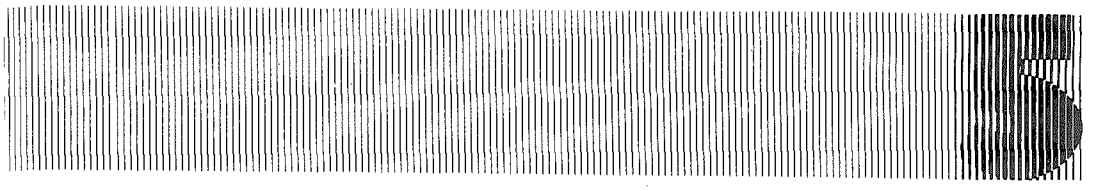
De handleiding is bedoeld om beginnende gebruikers enigszins wegwijs te maken in VISUAL. Dit wordt gedaan door eerst wat over algemene zaken, zoals de omgeving en de globale opzet van het systeem, te vertellen. Vervolgens wordt aan de hand van enkele eenvoudige voorbeelden duidelijk gemaakt hoe de diverse functies van het systeem gebruikt moeten worden.

In de handleiding worden ook enkele zaken behandeld, welke niet direct op de programmatuur van VISUAL betrekking hebben. Zoals bijvoorbeeld een beknopte uitleg over de computer, de terminals, de database en het gebruik van de commandotaal COLAS. Het nut en gebruik van de deelsystemen DIRSEL en SET-UP voor het deelsysteem 3D-FUNCTIE worden eveneens duidelijk gemaakt.

Verder is er één uitgebreid voorbeeld van 15 bladzijden, welke integraal in de handleiding is opgenomen. Dit is een sessie met VISUAL (achter een Tektronix 4114), waarin een aantal eenvoudige functies gebruikt worden. Elk commando in dit voorbeeld is genummerd, zodat er vanuit de tekst door middel van een nummer naar een commando verwezen kan worden.

Er is getracht zo weinig mogelijk "Informatica vakjargon" te gebruiken. Op sommige plaatsen in de handleiding viel er echter niet aan te ontkomen en daarom is achter in de handleiding een verklarende woordenlijst met Informatica begrippen opgenomen.

De handleiding is geen komplete opsomming van alle aanwezige functies met de bijbehorende parameters. Indien de gebruiker eenmaal wat bekend is met VISUAL kan hij al deze informatie "on-line" opvragen. Daarom is dit niet allemaal in de gebruikershandleiding, welke meer is bedoeld als een eerste introductie van het systeem, opgenomen. Het is waarschijnlijk dat VISUAL nog verder ontwikkeld zal worden; zie hoofdstuk 5. Het gevolg is dan ook dat de gebruikershandleiding aangepast zal moeten worden. Hiertoe is deze op de WANG tekstverwerker (bij Fokker) gezet.



Aanbevelingen

Tijdens de ontwikkeling van VISUAL is een flinke lijst ontstaan met aanbevelingen voor verdere ontwikkeling. Deze lijst bestaat uit tips voor verbetering, nieuwe functies en andere uitbreidingen. Het is de vraag, of als er nog één vol jaar aan VISUAL gewerkt zou worden, deze dan zijn definitieve vorm bereikt zou hebben. Het zal duidelijk zijn dat dit echter buiten het kader van het afstudeerwerk valt. Ook in de toekomst zal de lijst met aanbevelingen nog verder groeien als steeds meer gebruikers ervaring met het systeem hebben opgedaan. Om dit hoofdstuk overzichtelijk te houden is de lijst in een aantal delen opgesplitst. In 5.1 worden de functies CONTOUR en CARPET, welke wegens tijdgebrek niet gedurende de afstudeeropdracht geïmplementeerd zijn, uitgebreid besproken. Ook de deelsystemen 1D- en 2D- FUNCTIE (zie 5.2) zijn niet geïmplementeerd, doch dit was reeds zo afgesproken voordat met de afstudeeropdracht werd begonnen. Een aantal verbeteringen en nieuwe functies worden in respectievelijk 5.3 en 5.4 voorgesteld. In 5.5 is een andere (en mogelijk snellere) opzet van de database te vinden. Enkele algemene aanbevelingen en opmerkingen in 5.6 besluiten dit hoofdstuk.

5.1 - De functies CONTOUR en CARPET in het deelsysteem 3D-FUNCTIE

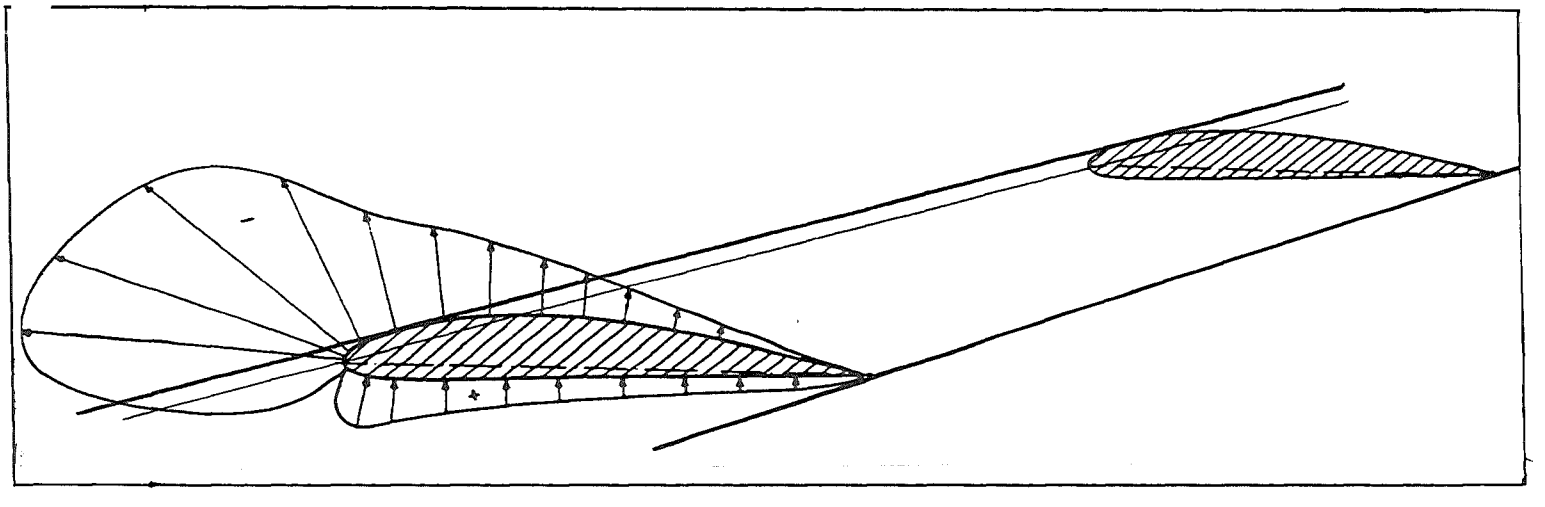
De functie CONTOUR

De functie CONTOUR houdt in het tekenen van de drukverdeling op een rij panelen van een segment in 3D-aanzicht; zie 2.4. De term CONTOUR zoals deze hier gebruikt wordt heeft geen enkele binding met de term contour zoals die gebruikt wordt bij de opbouw van de geometrie in 2.2. Men moet hier bij de term CONTOUR eerder denken aan het begrip profiel zoals bij een vleugel. Er zijn twee mogelijke manieren te bedenken om de drukverdeling op een rij panelen weer te geven.

De eerste is dat er een serie punten in de 3D-ruimte worden bepaald, welke vervolgens met elkaar verbonden worden. Zo'n punt wordt per paneel als volgt uitgerekend:

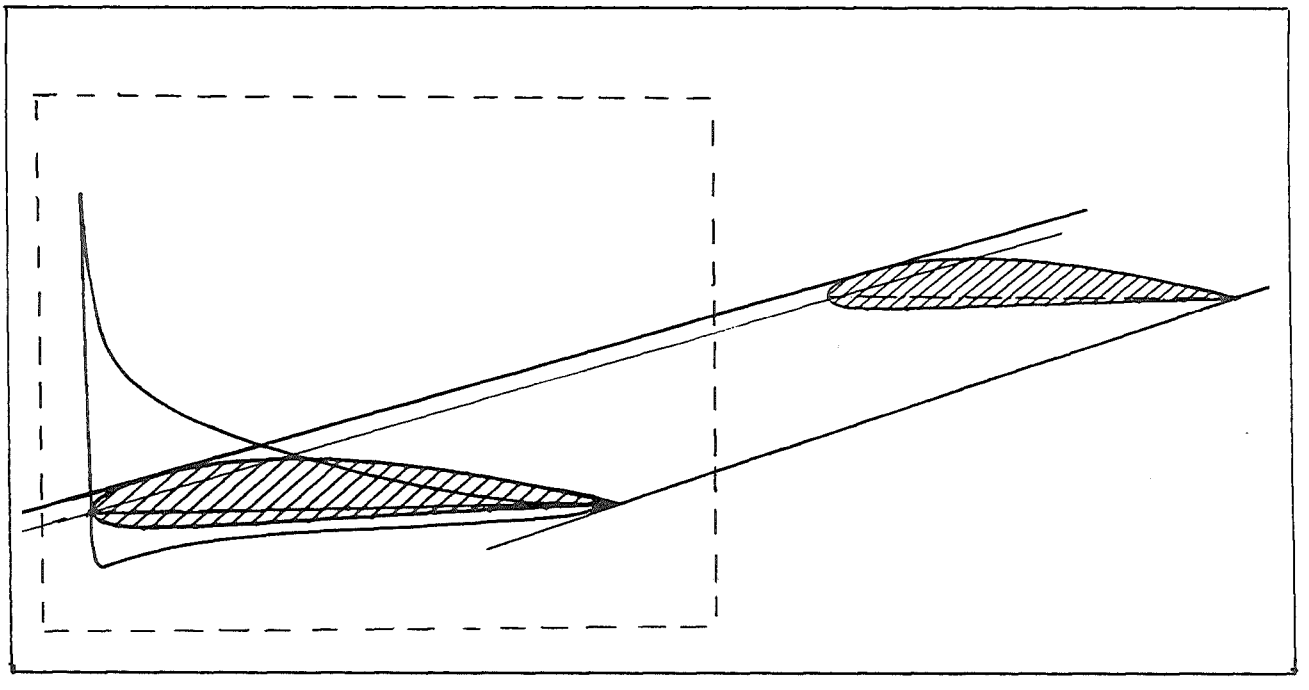
- ga uit van het middelpunt van een paneel;
- volg de normaalvector van dat paneel over een lengte die gelijk is aan de absolute waarde van de cp-waarde die bij dat paneel behoort.

Het resultaat van deze methode is weergegeven voor de drukverdeling over het profiel NACA 4412 in figuur 5.1. Eventueel kan het teken van de drukwaarde worden aangegeven door middel van een kleurverschil (- = rood en + = blauw) en ook kunnen pijlen de duidelijkheid van de figuur vergroten.



Figuur 5.1 - De eerste methode voor de functie CONTOUR

De tweede methode om de functie CONTOUR te implementeren is een twee-dimensionale grafiek, met de drukverdeling, schalen en projecteren op bij-voorbeeld het profiel in de drie-dimensionale afbeelding. In figuur 5.2 is hiervan een voorbeeld gegeven. Het vlak van de twee-dimensionale grafiek is hierin gestippeld aangegeven.



Figuur 5.2 - De tweede methode voor de functie CONTOUR

Om de tweede methode te implementeren moet eerst het vlak, waarin de grafiek moet komen, bepaald worden. Vervolgens moet het voorste en achterste punt bepaald worden, zodat er geschaald kan worden. Bij het tekenen is het raadzaam om de grafiek zelf een andere kleur te geven.

De gebruiker moet kunnen aangeven op welke serie panelen hij de functie getekend wil hebben, door een paneel met behulp van kruisdraden aan te wijzen en door aan te geven in welke richting de panelen moeten lopen ; langs de primaire of langs de secundaire contouren. Het moet mogelijk zijn om de functie CONTOUR op meerdere plaatsen op de geometrie tegelijk zichtbaar te maken. Ook moet het mogelijk zijn een vergelijking met een willekeurige andere drukverdeling te maken; bijvoorbeeld voor een andere aanstromingsconditie.

De afstudeerder geeft vanwege esthetische aspecten de voorkeur aan de eerste methode. Waarschijnlijk is deze ook eenvoudiger (en eenduidiger) te implementeren in wat algemenere gevallen.

De functie CARPET

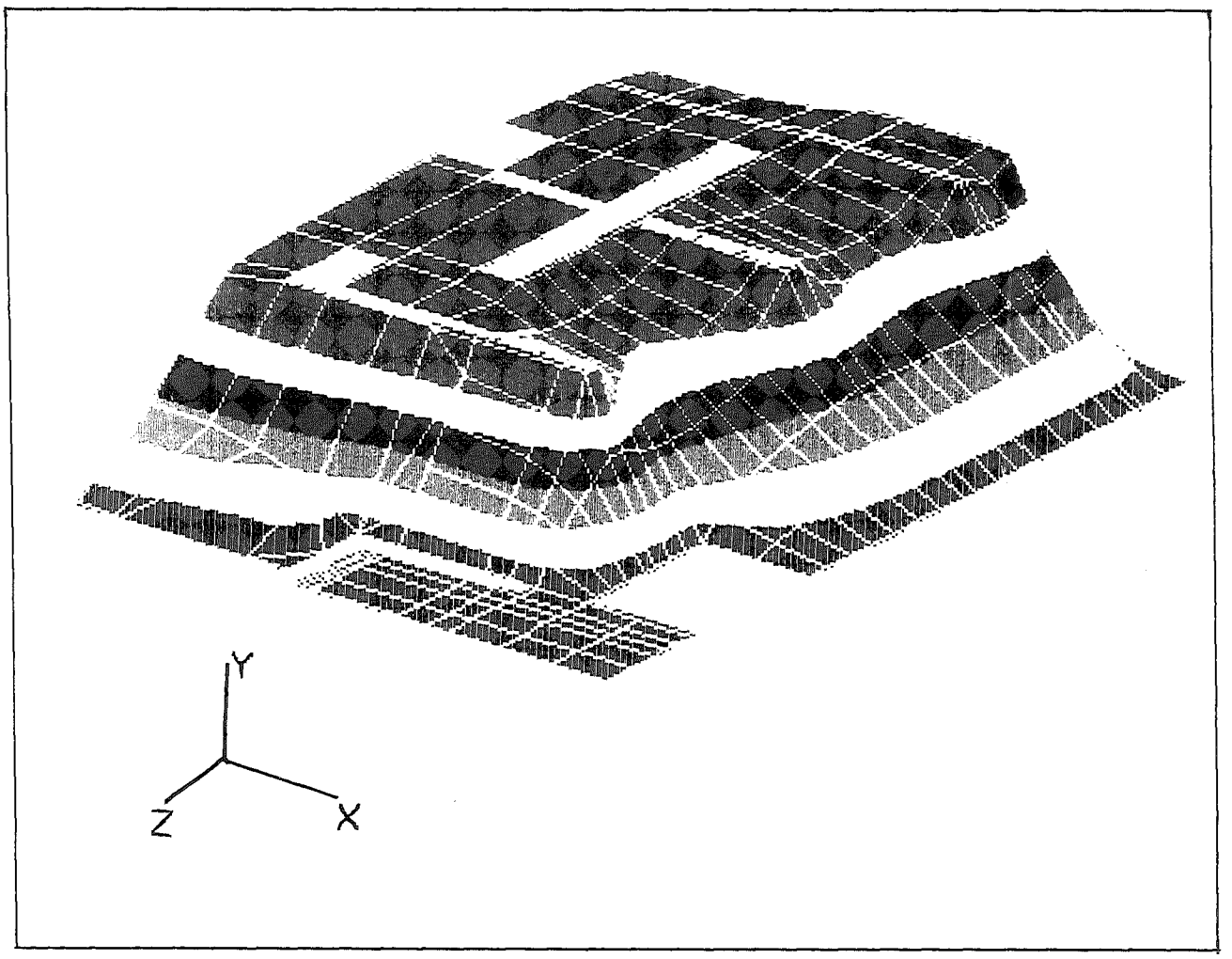
De functie CARPET houdt in: het niveau van de drukverdeling wordt aangegeven door deze te gebruiken als "offset" van een referentievlak, welke door de gebruiker moet worden gespecificeerd; zie 2.4. Op deze wijze onstaat een heuvellandschap waarin de toppen overeenkomen met de hoge cp-waarden en de dalen overeenkomen met de lage cp-waarden.

In principe is de functie CARPET niets anders dan het weergeven van een functie z = f(x,y) , waarvoor bestaande algoritmen en routines beschikbaar zouden moeten zijn. Zo zou er gebruik kunnen worden gemaakt van hidden line/ surface en zou het hoogte verschil ook met een kleur kunnen worden duidelijk gemaakt; bijvoorbeeld toppen blauw, dalen rood en daartussen een geleidelijk verloop via groen.

Een belangrijke beslissing welke bij de implementatie van CARPET genomen moet worden is van welk grid wordt er uitgegaan bij het tekenen van de functie z = f(x,y). Hiervoor zijn onder andere de volgende twee oplossingen:

- Ga uit van een regelmatig fijn grid, onafhankelijk van de geometrie van de configuratie. Men zou hiervoor een rechthoekig grid kunnen nemen en voor elk roosterpunt (x,y), via een projectie, kunnen bepalen wat de waarde van z = f(x,y) hierbij zou moeten zijn.

- Ga uit van een grid dat gerelateerd is aan de geometrie. Deze geometrie wordt dan op een plat vlak geprojecteerd met de bijbehorende cp-waarden voor z. In figuur 5.3 is een voorbeeld uit de wereld van de sterkte berekening gegeven van de functie CARPET. Deze figuur is met het programma PATRAN gemaakt.



Figuur 5.3 - Een voorbeeld voor de functie CARPET

De functie CARPET kan zeer spectaculaire resultaten opleveren, mits zij goed geïmplementeerd wordt. Deze implementatie zal waarschijnlijk niet geheel triviaal zijn, ondanks de hierboven genoemde richtlijnen.

De vraag rijst nu of de resultaten van de functie CARPET ook zo nuttig zijn dat deze opwegen tegen de inspanningen die ervoor verricht moeten worden. Voor de functie CONTOUR zal dit eerder het geval zijn. Voor de functie CARPET kan er waarschijnlijk beter gebruik worden gemaakt van een bestaand softwarepakket.

5.2 - De ontwikkeling van de deelsystemenen 1D- en 2D-FUNCTIE

De functies in de deelsystemen 1D- en 2D-FUNCTIE moeten nog in hun geheel worden geïmplementeerd. Wel is er tijdens het ontwerp met deze functies rekening gehouden en zijn de deelsystemen zelf al wel geïmplementeerd. Dit weer in de vorm van de bekende CASE opdracht. Ook zijn er al commandosets (zie 3.5) voor deze deelsystemen gemaakt.

Nu volgt een beschrijving van beide deelsystemen; eerst zal 2D-FUNCTIE behandeld worden en daarna komt 1D-FUNCTIE aan bod. Bij de beschrijving zal worden uitgegaan van de gebruikersspecificatie (ref. 5), het verslag van de oriëntatiefase (ref. 28) en van de ervaringen die door de afstudeerder bij Fokker zijn opgedaan gedurende het afstudeerwerk.

Het deelsysteem 2D-FUNCTIE

In dit deelsysteem moeten op 2D-wijze de aerodynamische gegevens gevisualiseerd worden. Er zullen nu een tweetal functies met verschillende opties besproken worden.

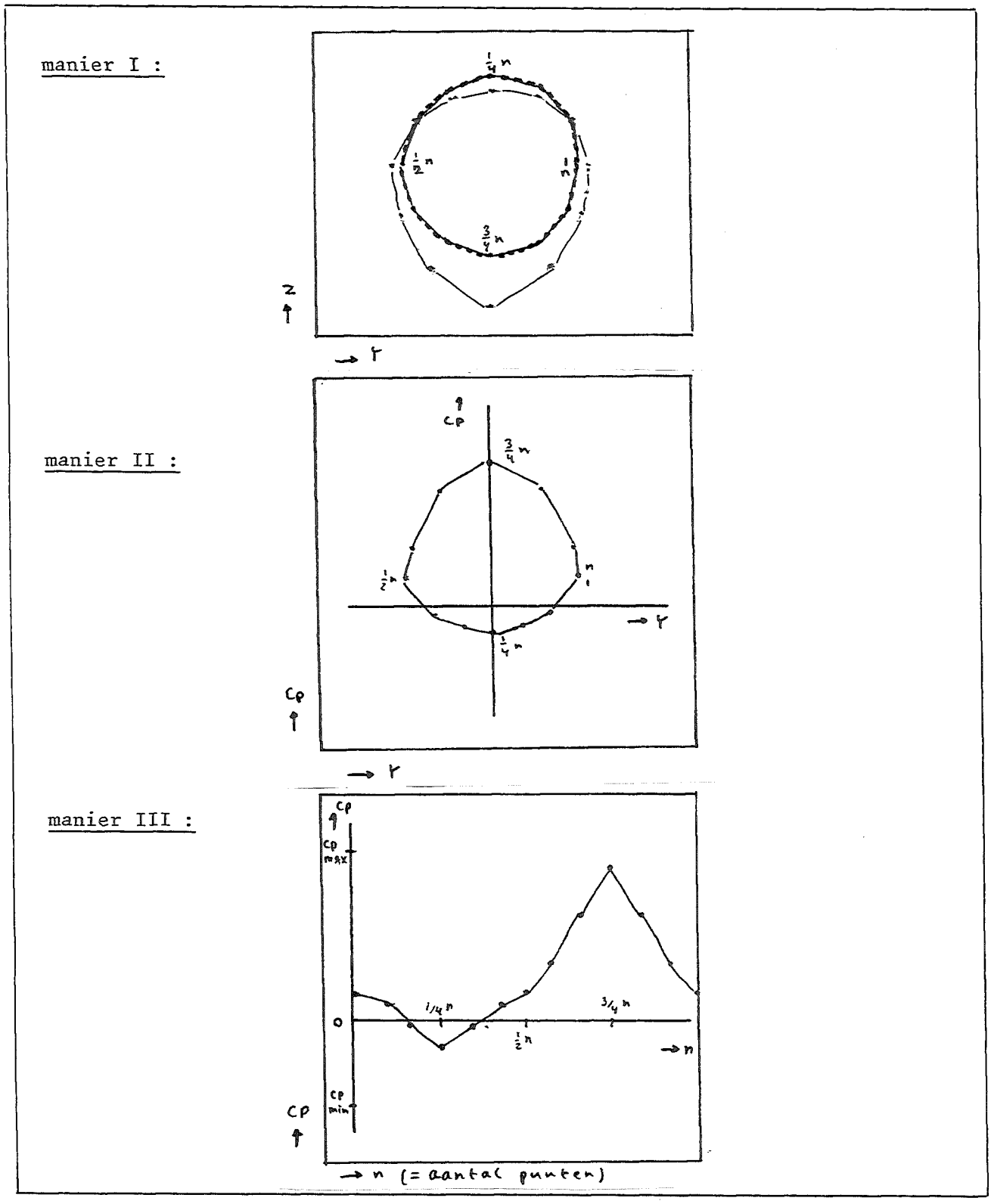
- PRESSURE. Het tekenen van de drukverdeling van een contour in een bepaald aanzicht xy, xz of yz waarbij de schaling van de assen kan worden veranderd. Een vergelijking moet gemaakt kunnen worden met een willekeurige andere drukverdeling. Het supersone gebied moet kunnen worden aangegeven. Een raster moet over de tekening kunnen worden aangebracht.

  Dit zou dezelfde functie als CONTOUR in het deelsysteem 3D-FUNCTIE kunnen zijn, alleen behoeft de grafiek niet in de 3D-ruimte te worden geprojecteerd. In figuur 5.4 worden een aantal manieren getoond om deze functie uit te voeren. Dit gebeurt aan de hand van een yz-aanzicht van de Fokker 100. De romp is bij manier I gestippeld aangegeven.

- LIFT/DRAG/MOMENT. Het tekenen van een liftverdeling/weerstandsverdeling/momentenverdeling (Cl, Cd en Cm waarden) van een aantal segmenten in xy, xz of yz-aanzicht. Een vergelijking moet gemaakt kunnen worden met een willekeurige andere Cl, Cd of Cm verdeling. Een

raster moet over de tekening kunnen worden aangebracht.

Voor een toelichting bij deze functie wordt verwezen naar de toelichting bij de functie PRESSURE uit hetzelfde deelsysteem.



Figuur 5.4 - Drie manieren voor de functie PRESSURE

Het is mogelijk dat de datastructuur, zie 3.3 en 4.3, en de selectie functie in het deelsysteem DIRSEL uitgebreid zullen moeten worden in verband met de lift-, weerstands- en momenten-verdeling. De implementatie van het deelsysteem 2D-FUNCTIE zal zeker enige inspanning vergen. Wellicht dat dit met routines uit het programma CODE (zie 1.2 en 2.3) verminderd kan worden.

Het deelsysteem 1D-FUNCTIE

In tegenstelling tot de deelsystemen 3D- en 2D-FUNCTIE is het deelsysteem 1D-FUNCTIE geen grafisch deelsysteem. Met dit deelsysteem moeten gegevens in de vorm van één of enkele waarden kunnen worden opgevraagd. Een aantal voorbeelden hiervan zijn de volgende functies:

- FORCE. Het bepalen van de totale kracht die werkt op het geselecteerde model. Dit komt neer op het sommeren over alle panelen, rekening houdend met de grootte en richting van elk paneel.

- AREA. Het bepalen van het omspoeld oppervlak van het geselecteerde model. Hiervoor hoeft men alleen de lengte van de normaalvectoren, welke in de geheugen POOL zijn opgeslagen, te sommeren en door twee te delen; zie 3.3.

- MOMENT. Het bepalen van de momentencoëfficiënten van het geselecteerde model ten opzichte van een willekeurig te kiezen referentiepunt. Ook dit moet weer worden geïmplementeerd met behulp van een sommatieformule.

- DRAG. Het bepalen van de visceuse weerstandscoëfficiënt van de geselecteerde configuratie/group/part/segment met behulp van een eenvoudige weerstandsformule.

- INDUCED-DRAG. Het bepalen van de geïnduceerde weerstand van de geselecteerde configuratie/group/part/segment op een analoge wijze als bij de functie DRAG.

- PANEL-INFO. Weergeven van de drukcoëfficiënt (cp-waarde) en snelheids-componenten van een paneel dat met kruisdraden wordt aangewezen. De drukcoëfficiënt en de snelheids-componenten kunnen uit de geheugen

POOL worden opgehaald, nadat bepaald is welk paneel er is aangewezen. Er moet een functie komen waarmee bepaald wordt welk paneel er met de kruisdraden is aangewezen. Dit kan met dezelfde functie als die in CONTOUR gebruikt wordt. Eerst moet er een inverse transformatie plaatsvinden en daarna moet in de geheugen POOL worden gezocht welk paneel dit is. Dit zoeken kan worden beperkt door gebruik te maken van de extreme waarden die per segment worden opgeslagen, zodat het grootste deel van de segmenten (en dus panelen) geëlimineerd kan worden.

Er wordt verwacht dat de implementatie van het deelsysteem 1D-FUNCTIE zeer weinig problemen zal opleveren en dat dit ook niet veel tijd zal kosten.

5.3 - Mogelijke verbeteringen voor een aantal functies

Nu volgt een opsomming van een aantal verbeteringen, die per onderwerp verder zullen worden uitgewerkt:

- Hidden-surface removal. In 4.4 zijn hiervoor een tweetal maatregelen genoemd. Er is daar ook opgemerkt dat deze maatregelen niet in alle gevallen voldoende zullen zijn voor een perfecte afbeelding. Dit kan soms storend zijn en daarom kan een van de volgende maatregelen getroffen worden:

  * Splits een segment in 4 of 16 delen op indien dit mogelijk is. Ga nu sorteren voordat er getekend wordt. De meeste fouten zullen nu wel uit de bestaande hidden-surface removal verwijderd zijn, maar er is nog steeds geen garantie dat er in alle gevallen een perfecte afbeelding komt.

  * Ga niet per segment of per gedeelte van segment sorteren, maar sorteer per paneel. Dit is eigenlijk een voortzetting van de vorige maatregel, maar deze zal nog meer sorteertijd vragen. Deze sorteertijd kan weer worden beperkt door eerst per segment te sorteren en dan te gaan "mergen"; ref. 49 en 50 bevatten een aantal sorteer algoritmen. In dit geval zal er wel een perfecte afbeelding worden geleverd.

- Hidden-line algoritme. De functie HIDDEN in het deelsysteem 3D-FUNCTIE maakt gebruik van een hidden-surface algoritme dat alleen geschikt is voor raster displays. Om deze functie ook geschikt te maken voor de andere typen displays moet er ook een hidden-line algoritme worden toegevoegd. Bij Fokker is reeds zo'n algoritme aanwezig (voor de geometrie welke uit panelen is opgebouwd) en het moet vrij eenvoudig mogelijk zijn om deze routine ook in VISUAL te gebruiken. Het nadeel is dat hidden-line algoritmen nogal traag zijn en daarom kan op een raster display beter gebruik gemaakt worden van een hidden-surface algoritme.

- Afbreek functie bij database routines. In verband met de gebruikersvriendelijkheid is in 2.5 gesteld dat langdurige bewerkingen afgebroken

moeten kunnen worden. Dit is reeds geïmplementeerd voor de routines uit het deelsysteem 3D-FUNCTIE, waar de gebruiker door "#" (SHIFT 3) in te typen het tekenen kan afbreken. Voor de database routines DIRECTORY, SELECT en ADD uit het deelsysteem DIRSEL is dit nog niet gebeurd. Hiervoor moet namelijk de interface met de database (DIALIB, zie 2.3) worden aangepast door Dhr. F. Tromp.

Dubbel selecteren. In de meeste gevallen wordt slechts de helft van een vliegtuig in de database opgeslagen en door stromingsberekenings programma's geanalyseerd. Dit omdat de andere helft gespiegeld in het xz-vlak precies hetzelfde is. Wil men in VISUAL een aardige afbeelding maken welke een compleet vliegtuig bevat, dan kan VISUAL het beste als volgt worden aangepast:

* Voeg een PROGRAM MODE "DOUBLE-SELECTION" toe, welke aangeeft of er wel of niet dubbel geselecteerd moet worden.

* Pas de selectie routines zodanig aan dat als er één segment uit de database wordt opgehaald er twee in VISUAL worden opgeslagen; één met een oneven volgnummer (het oorspronkelijke segment) en één met een even volgnummer (het segment dat ontstaat door het vorige in het xz-vlak te spiegelen). De aerodynamische gegevens behoeven niet dubbel te worden opgeslagen; er wordt namelijk via een pointer tweemaal naar hetzelfde gebied in de geheugen POOL verwezen.

* De routines uit het deelsysteem 3D-FUNCTIE behoeven niet gewijzigd te worden.

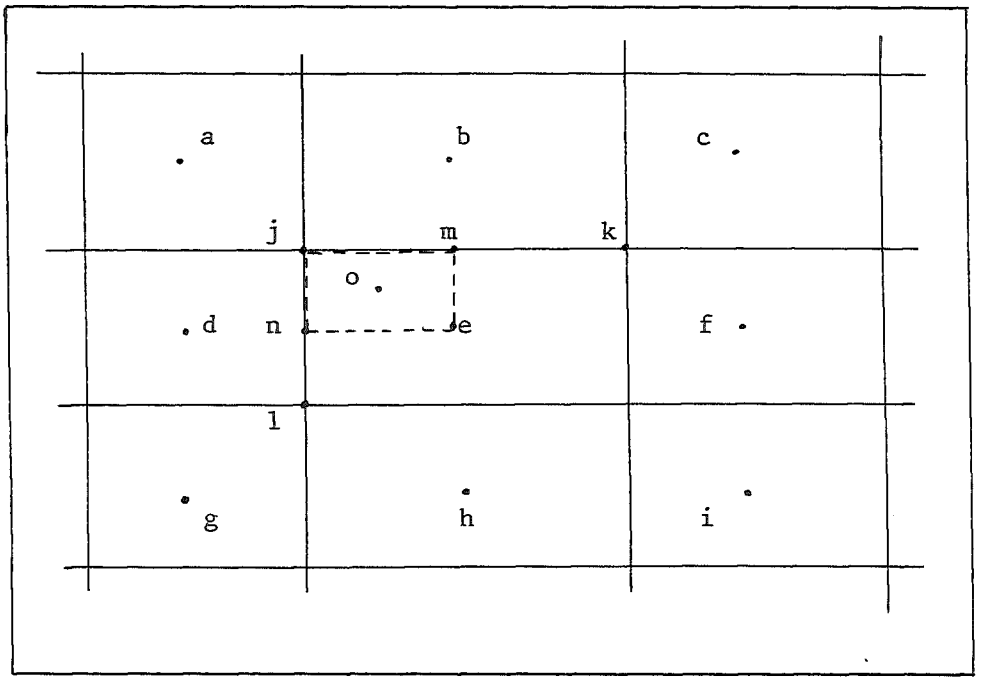
Opmerking: Er zijn ook nog andere oplossingen mogelijk welke voorkomen, dat de geometrie dubbel wordt opgeslagen. Deze zullen hier niet verder worden uitgewerkt.

- Uitbreiden van de functie MATRIX. Deze functie in het deelsysteem 3D-FUNCTIE beschikt over de opties: SAVE, LOAD en DIRECTORY. Het ligt voor de hand om ook een DELETE optie toe te voegen, waarmee één van de door de gebruiker gedefinieerde aanzichten verwijderd kan worden. Dit vindt overigens al automatisch plaats wanneer er een SAVE wordt toegepast op een reeds gedefinieerd aanzicht.

- Verbeteren van de functie SHADE. Er zullen nu een aantal verbeteringen van het commando SHADE uit het deelsysteem 3D-FUNCTIE worden besproken. Deze verbeteringen behoeven elkaar niet altijd uit te sluiten.

  * Maak gebruik van zogenaamde "iso-lights", dit zijn lijnen van gelijke belichtingsintensiteit, net zoals isobaren lijnen van gelijke druk beschrijven. Hiermee is eenzelfde verbetering te bereiken als met de functie ISO-BAND t.a.v. de functie COLOUR-CP. In verband met "scherpe knikken" in de geometrie zou per segmentgrens moeten kunnen worden aangegeven of er wel of niet BOUNDARY-CORRECTION moet plaatsvinden.

  * Opsplitsen van panelen in kleinere panelen: bijvoorbeeld vier subpanelen. Per subpaneel moet er een normaal worden uitgerekend. Met deze normaal moet er een belichtings-intensiteit worden bepaald. In figuur 5.5 is een subpaneel gestippeld getekend waarmee is getracht een en ander duidelijk te maken.



Figuur 5.5 - Opsplitsen van een paneel in subpanelen

Oorspronkelijk zijn de normalen bekeken in de middelpunten van de panelen a tot en met i. Deze normalen zullen worden

aangeduid met respectievelijk Na tot en met Ni. We willen de normaal bepalen van subpaneel o (dat onderdeel is van paneel e), de normaal zal worden aangeduid met No. Door lineair te interpoleren krijgen we de volgende formule voor No:

$$
\begin{aligned}
No &= \tfrac{1}{4}\,(Nj + Nm + Nn + Ne) \\
&= \tfrac{1}{4}\,(Nj + \tfrac{1}{2}\,(Nj + Nk) + \tfrac{1}{2}\,(Nj + Nl) + Ne) \\
&= \tfrac{1}{4}\,(2*Nj + \tfrac{1}{2}*Nk + \tfrac{1}{2}*Nl + Ne) \\
&= \tfrac{1}{4}\,(2*\tfrac{1}{4}\,(Na + Nb + Nd + Ne) + \tfrac{1}{2}*\tfrac{1}{4}\,(Nb + Nc + Ne + Nf) \\
&\qquad + \tfrac{1}{2}*\tfrac{1}{4}\,(Nd + Ne + Ng + Nh) + Ne) \\
&= 4/32\ Na + 5/32\ Nb + 1/32\ Nc + 5/32\ Nd + 14/32\ Nd \\
&\qquad + 1/32\ Nf + 1/32\ Ng + 1/32\ Nn.
\end{aligned}
$$

Nu hebben we de normaalvector No uitgedrukt in bekende termen. Het is ook mogelijk om No te berekenen door gebruik te maken van de afstand tot de bekende paneel middelpunten. Dit heeft als nadeel dat de factoren van weging niet meer constant zijn en steeds opnieuw moeten worden uitgerekend.

* Het gebruik van een geavanceerder belichtingsmodel om de lichtintensiteit in een bepaald punt (met normaal) te berekenen. Hier kan men bijvoorbeeld denken aan het Phong model (ref. 34) of een verfijning hiervan (ref. 44). Het resultaat hiervan wordt pas echt mooi zichtbaar als er meerdere lichtintensiteitswaarden per paneel gebruikt zouden worden (4, 9, 16 of nog meer). Het bepalen van de lichtintensiteit kan gebeuren door lineair te interpoleren, hetzij voor het uitrekenen van de lichtintensiteit (zie vorig punt; door de normalen te interpoleren) of na het uitrekenen van de lichtintensiteit.

* Uitbreiding van de hardware van de kleuren raster display van 16 naar 256 kleuren die tegelijk op het scherm zichtbaar gemaakt kunnen worden. Hiervoor behoeft in VISUAL alleen de constante MAXCOL aangepast te worden.

5.4 - Een aantal nieuwe functies

Gedurende de ontwikkeling van VISUAL zijn er een aantal ideeën ontstaan voor nieuwe functies. Ook zitten er in PATRAN (zie 1.2) een aantal zaken die bruikbaar zouden kunnen zijn.

- Bewaren van kleurdefinities. VISUAL kent een aantal default kleurendefinities, zoals een schaal met variërende lightness voor de functie SHADE en een schaal met variërende hue voor de functies COLOUR-CP en ISOBAND. Maar zoals in 4.5 beschreven kan de gebruiker ook zelf kleuren definiëren met de commando's HUE-COLOUR, LIGHTNESS, SATURATION en COLOUR-DEF in het deelsysteem SET-UP. Wanneer de sessie wordt beëindigd gaan deze definities echter weer verloren. Daarom zou er een functie moeten komen waarmee kleurdefinities kunnen worden opgeslagen. Dit zou net zo kunnen gebeuren als er aanzichten kunnen worden bewaard met het commando MATRIX. Er zouden dus de volgende opties moeten komen: LOAD, SAVE, DIRECTORY en DELETE.

- Lokale functies voor Tektronix 4115. Sinds enige tijd zijn op de T4115 een harde schijf en een floppy disk geïnstalleerd. Hierop kunnen afbeeldingen bewaard worden onder een bepaalde naam. Er kan een overzicht (DIRECTORY) van afbeeldingen worden gevraagd en er kunnen natuurlijk ook weer afbeeldingen worden opgehaald. Er zou een functie in VISUAL moeten komen in het deelsysteem SET-UP waarmee deze taken verricht zouden kunnen worden. Een naam voor deze functie zou kunnen zijn LOCAL en als opties zouden er kunnen komen LOAD, SAVE, DIRECTORY en DELETE.

- Stroomlijnen. Dit is een echte nieuwe visualisatie routine in het deelsysteem 3D-FUNCTIE. De gebruiker mag met kruisdraad aangeven waar een stroomlijn op de geometrie moet beginnen en VISUAL rekent deze dan uit en tekent deze vervolgens op het scherm. Het moet dan ook mogelijk zijn dat het systeem zelf automatisch begin punten genereert. Stroomlijnen lopen net als isobaren over de geometrie van paneel tot paneel. Stroomlijnen zouden als volgt kunnen worden bepaald:

    * Laat vanaf het punt dat een stroomlijn een paneel "binnenkomt" deze zijn weg vervolgen over dit paneel in de richting van de snelheidsvector (dat bij dit paneel hoort)

net zolang totdat er weer een grens van dit paneel wordt bereikt.

* Bepaal in welk ander paneel, dat grenst aan het paneel dat door de stroomlijn wordt verlaten, de stroomlijn nu terecht komt en herhaal de vorige stap. In het geval dat het paneel aan de rand van een segment ligt dan kan het zijn dat de stroomlijn z'n weg vervolgt in een ander segment. Om dit te bepalen kan er gebruik worden gemaakt van de reeds bestaande datastructuren waarin de buren zijn opgeslagen

- Veranderen van teken van de normaalvectoren. Het is mogelijk dat er segmenten zijn, waarvan de coördinaten in een verkeerde volgorde in de ORACLE database zijn opgeslagen, zodat de normaalvectoren op het oppervlak naar binnen in plaats van naar buiten wijzen. Daarom zou het nuttig zijn als er in het deelsysteem DIRSEL de functie REVERSE-NORMAL zou komen, waarmee dit eenvoudig gerealiseerd zou kunnen worden.

- Bepalen van een aanzicht met een eenvoudig voorwerp. Dit is een idee dat afkomstig is uit PATRAN. Het bepalen van een fraai aanzicht kan nogal veel tijd vergen. Want na elke rotatie die de gebruiker geeft, moet de configuratie getekend worden; wat vooral voor complexere configuraties veel tijd kost, omdat er enorm veel lijnen getekend moeten worden. Daarom is het vaak veel handiger wanneer het bepalen van een aanzicht met een eenvoudig voorwerp kan gebeuren. Misschien is deze functie in VISUAL overbodig omdat de gebuiker met het commando OUTLINE een zeer snelle schets van de configuratie kan krijgen. Bovendien beschikt VISUAL over het commando MATRIX waarmee aanzichten bewaard kunnen worden.

- Aangeven van de "toestand" van VISUAL met een kleur. Ook dit idee is afkomstig uit PATRAN. Men zou de "toestand" van VISUAL kunnen aangeven door een blokje (of de tekst van de dialog-area) van een bepaalde kleur code te voorzien. Een voorbeeld van een kleurcode is:

  * GROEN: VISUAL wacht op invoer van de gebruiker;
  * GEEL : VISUAL is bezig met een database opdracht;
  * ROOD : VISUAL is aan het rekenen;
  * BLAUW: VISUAL is aan het tekenen.

- Maken van een "SAVDRA" file. Dit zou een optie kunnen zijn bij het commando DRAW in het deelsysteem 3D-FUNCTIE. Een "SAVDRA" file kan door een ander programma worden gebruikt om op een plotter te tekenen; bijvoorbeeld op de Benson 1302 plotter of Kongsberg tekenmachine, zie 2.3. Het maken van een "SAVDRA" file kan met GINO gebeuren. Dit heeft als nadeel dat het huidige device eerst moet worden vrijgegeven. Na het maken van een "SAVDRA" file wil de gebruiker gewoon weer de sessie vervolgen met hetzelfde device. Daarom moet de terminal-toestand van dat ogenblik (kleurdefinities, lijntype, etc.) bewaard worden voordat het device wordt vrijgegeven, zodat als de "SAVDRA" file gemaakt is en hetzelfde device weer gebruikt moet worden, de toestand weer hersteld kan worden.

5.5 - Een andere opzet van de database

In appendix H is het ontwerp van de database weergegeven, zoals deze op het ogenblik gebruikt wordt. Dit ontwerp voldoet redelijk goed, maar er zitten twee nadelen aan vast, namelijk:

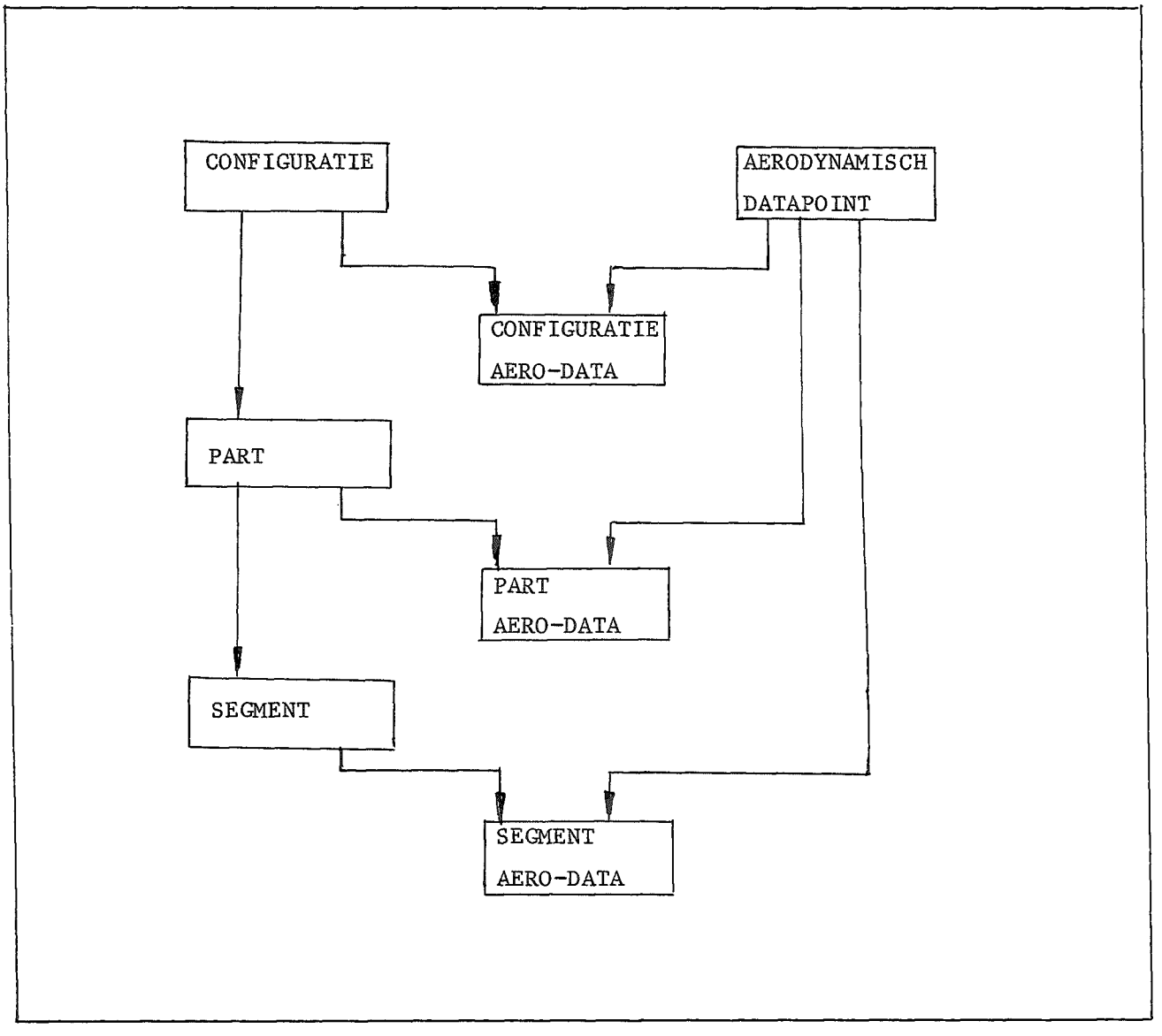
1) Bij het toevoegen van een configuratie (met groups, parts, segmenten, etc.) moeten ook de zogenaamde relatie tabellen worden gevuld; zie H3 in appendix H. Dit is iets dat vrij lastig is en dat gauw kan worden vergeten, zodat de database niet meer consistent is met de over de geometrie in 2.2 gemaakte afspraken. Bij voorbeeld: we gaan een configuratie C1 toevoegen met parts P1 (waarin segmenten S1, S2 en S3 zitten) en P2 (waarin S3, S4, S5 en S6 zitten) en segmenten S1, S2, S3, S4 en S5. Nu ontstaat de vreemde situatie dat S6 wel in P2 zit (welke onderdeel van C1 is), maar niet in C1.

2) De database werkt te langzaam. Het ophalen van de coördinaten van een configuratie bestaande uit 25 segmenten kan onder normale omstandigheden 5 tot 10 minuten wachttijd kosten. Het zal duidelijk zijn dat dit veel te lang is.

Beide problemen zijn waarschijnlijk het gevolg van het trachten een te algemene database op te zetten, waarin ook nog eens zeer grote hoeveelheden geometrische en aerodynamische gegevens zijn opgeslagen. Met een zeer algemene opzet wordt bedoeld dat er van elkaar losstaande tabellen van objecten zijn (configuratie, group, part en segment), welke door de gebruiker (van de database) op elke willekeurige wijze aan elkaar gekoppeld worden. Dit komt doordat bij het ontwerpen van de database er onder andere de voorwaarde gesteld is: "een segment moet ook tot meerdere configuraties kunnen behoren". Naast het normale feit dat een configuratie uit meerdere segmenten bestaat, zie 2.2. En ditzelfde geldt voor de andere typen objecten.

Het zou het beter zijn geweest wanneer de betekenis van de diverse begrippen opnieuw gedefiniëerd zouden worden, zodanig dat er een eenvoudig te hanteren en consistent begrippenstelsel ontstaat. Het begrip group ("een verzameling parts, die als één geheel worden doorgerekend voor een bepaalde stromingsconditie") zou moeten vervallen, omdat het de zaak onnodig ingewikkeld maakt. In de praktijk blijkt dat alles toch per configuratie wordt doorgerekend (of zou kunnen worden doorgerekend).

Segmenten worden geordend in parts. Elk segment moet precies één keer in één van de parts voorkomen; dus niet meer en niet minder. Men kan eventueel een "nul-part" toevoegen, waarin alle segmenten, die nog niet in een ander part zitten, worden gestopt. Alle parts tesamen vormen de configuratie. Een part/segment kan slechts tot één configuratie behoren. Wil men deze toch tot meerdere configuraties laten behoren, dan moeten hiervan meerdere copieën gemaakt worden. In figuur 5.6 is een schema van het ontwerp van de database gegeven volgens de bekende notatie voor relationele database systemen (ref. 18). De eenvoud van dit ontwerp is tevens een van de sterkste punten ervan.



Figuur 5.6 - Ontwerp van de database

Het ontwerp uit figuur 5.6 ziet er in tabelvorm als volgt uit:

CONFIGURATIE (CONFNR, BESCHRIJVING, ...)
PART (CONFNR, PARTNR, BESCHRIJVING, ...)
SEGMENT (CONFNR, PARTNR, SEGMNR, BESCHRIJVING, GEOMETRIE, ...)

AERODYNAMISCH-DATAPOINT (AERONR, BESCHRIJVING, ...)
CONFIGURATIE-AERO-DATA (AERONR, CONFNR, DATA1, DATA2, ...)
PART-AERO-DATA (AERONR, CONFNR, PARTNR, DATA1, DATA2, ...)
SEGMENT-AERO-DATA (AERONR, CONFNR, PARTNR, SEGMNR, DATA1, DATA2, ...)

Met "BESCHRIJVING, ..." worden soortgelijke attributen bedoeld als in de tabellen van het oorspronkelijke ontwerp in appendix H. En met "DATA1, DATA2, ..." worden attributen bedoeld waarin de aerodynamische gegevens kunnen worden opgeslagen. De attributen die onderbroken onderstreept (_ _ _ _ _) zijn, bevatten zeer grote hoeveelheden gegevens. Het is mogelijk dat door deze zeer omvangrijke attributen de snelheidsprestaties van de database nadelig worden beïnvloed. Dit is afhankelijk van de manier waarop dit in ORACLE is geïmplementeerd. Als de prestaties inderdaad nadelig worden beïnvloed dan kunnen deze attributen worden vervangen door verwijzingen naar files buiten het database systeem. De fysieke omvang van de database kan hierdoor met een faktor 100 (schatting) kleiner worden.

Aan de hand van een eenvoudig voorbeeld zal worden aangetoond dat dit nieuwe database ontwerp veel efficiënter is. Stel we hebben 10 configuraties, die elk 3 parts bezitten, die op hun beurt weer bestaan uit 10 segmenten. In totaal bestaat de database dus uit: 10 configuraties, 30 parts en 300 segmenten. Er wordt gevraagd: "Zoek de geometrie die hoort bij de segmenten van het tweede part in configuratie nummer 7".

Oude situatie: Er vindt een JOIN plaats tussen de tabellen CONFPART en PARTSEGM. Dit levert 30 * 300 = 9000 combinaties waaruit de segment nummers bepaald moeten worden. Hierna moet in de tabel SEGMGEOM uit 300 tuples degene met de juiste segment nummers worden gezocht. In totaal moeten er dus 9000 + 300 = 9300 combinaties worden nagegaan.

Nieuwe situatie: Er kan direct naar de tuples in de tabel SEGMENT worden gezocht. In totaal moeten er nu dus 300 combinaties worden nagegaan.

Hieruit blijkt dat de nieuwe situatie veel sneller is, want er worden 31 keer zo weinig combinaties nagegaan. Er zijn zo nog veel meer voorbeelden waaruit blijkt dat het nieuwe database ontwerp veel sneller is.

Als dit nieuwe ontwerp geïmplementeerd zou worden, dan moeten er voornamelijk aanpassingen plaatsvinden in de programma interface (DIALIB) met ORACLE; zie 2.3. Dit zal een vrij ingrijpende aanpassing zijn omdat alle SQL statements veranderd zullen moeten worden. Indien, zoals voorgesteld, het begrip "group" wordt verwijderd, dan zullen ook in VISUAL enige maatregelen getroffen moeten worden. Dit kan heel simpel zijn door gewoon de SELECTION-MODE "GROUP" te verbieden, zonder dat de source code zelf verder behoeft te worden aangepast. Wil men echter alles wat met GROUPs te maken heeft uit VISUAL verwijderen, dan zal meer inspanning moeten worden verricht.

5.6 - Algemene aanbevelingen en opmerkingen

Tot slot van dit hoofdstuk volgen nog enkele algemene aanbevelingen, welke niet direct tot één van in de vorige twee paragrafen genoemde categorieën onderwerpen behoort. Het betreft hier voornamelijk de interface met de gebruiker en ook komt het geheugen POOL mechanisme ter sprake. Er volgt nu weer een puntsgewijze opsomming:

- De geheugen POOL routines. VISUAL maakt door middel van een aantal routines gebruik van een dynamisch geheugen POOL systeem; zie 4.3. Via deze routines worden er regelmatig met zeer grote hoeveelheden gegevens (tot maximaal 16.000 woorden) gemanipuleerd. De routines zijn in standaard Fortran 77 geïmplementeerd, maar met behulp van macro's zou er een versnelling te bereiken zijn. De vraag is echter of dit voordeel een verandering van de routines waard is. Want in de praktijk is gebleken dat de POOL routines snel genoeg zijn en als men macro's gaat gebruiken dan wordt afgeweken van standaard Fortran 77.

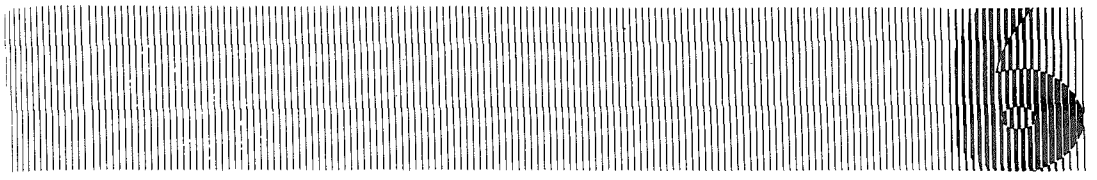
- Andere indeling van de commando's. De commando's in VISUAL zijn over een aantal deelsystemen verdeeld. In de praktijk van de gebruiker kan echter blijken dat een andere indeling voordeliger zou kunnen zijn in verband met de gebruikersvriendelijkheid. Zo is reeds op verzoek van de gebruikers het commando MATRIX (zie 2.4) verplaatst van het deelsysteem SET-UP naar 3D-FUNCTIE.

- Andere namen voor commando's. De reden dat commandonamen veranderd zouden moeten worden, zou kunnen zijn omdat de nieuwe namen duidelijker zijn voor de gebruikers of omdat de commando's eenvoudiger te gebruiken zijn doordat binnen elk deelsysteem de belangrijkste functies tot één letter uniek kunnen worden afgekort. Een tweetal voorbeelden van de eerste reden zijn: het commando VIEW krijgt de naam PERSPECTIVE en het commando MATRIX krijgt de naam VIEW. Ook zouden er synoniemen voor een functie gebruikt kunnen worden.

- Gebruik van PF-toetsen. Voor meer ervaren gebruikers van het systeem zou het efficiënt kunnen werken als er ook door middel van PF-toetsen (Programmeerbare Functie) commando's gegeven zouden kunnen worden. Bij voorbeeld een PF-toets voor het commando DRAW en een aantal PF-toetsen om naar andere deelsystemen te springen. In verband met het gebruik van

COLAS zal de implementatie hiervan niet eenvoudig zijn. Een ander nadeel is dat niet alle terminals over PF-toetsen beschikken (bijvoorbeeld de T4014).

- Gebruik van menu's. Voor onervaren gebruikers zou het prettig zijn als VISUAL ook met behulp van menu's bestuurd zou kunnen worden. In 2.6 is al een opsomming gegeven van de voor- en nadelen van menu's in vergelijking met een commandotaal. De meest elegante oplossing is dat de gebruiker zelf mag kiezen of hij met menu's of met de commandotaal wil werken via de PROGRAM MODE "COMMAND-DRIVEN", zie 3.5. Doordat er gebruik wordt gemaakt van verschillende soorten grafische terminals zal het veel inspanning kosten om (grafische) menu's te implementeren. Als er menu's geïmplementeerd worden, dan moet dit vanuit COLAS gebeuren. Dit heeft als voordeel dat er in VISUAL weinig zal behoeven te veranderen.



Evaluatie

Ter afronding van dit afstudeerverslag volgt nu een evaluatie van het afstudeerwerk. In 6.1 zullen de gebruikte hulpmiddelen geëvalueerd worden. De kwaliteit van VISUAL (dus in hoeverre voldoet het aan de systeemdefinitie? Is het gebruikersvriendelijk en voldoet de handleiding?) zal in 6.2 worden beschouwd. De driehoeksrelatie "THD-Fokker-afstudeerder" zal worden besproken in 6.3. Enige aspecten betreffende de planning en enkele kwan- titatieve aspecten komen aan de orde in respectievelijk 6.4 en 6.5. De eind- conclusie over het afstudeerwerk is te vinden in 6.6.

6.1 - Evaluatie van de gebruikte hulpmiddelen

Gedurende de ontwikkeling van VISUAL is gebruik gemaakt van een aantal hulpmiddelen. In 2.6 is aangegeven welke hulpmiddelen dat zijn en waarom juist deze gekozen zijn. In deze paragraaf zal worden nagegaan in hoeverre de keuzen terecht zijn geweest. Achtereenvolgens zullen aan de orde komen: de programmeertaal, de database, het grafische pakket en de besturing van het systeem door de gebruiker.

Programmeertaal

Het was verplicht de programmeertaal Fortran 77 te gebruiken. Er is een aantal malen zonder problemen overgeschakeld op een nieuwere versie van de Fortran compiler. Het impliciet definiëren van variabelen is, na enige negatieve ervaringen in de beginfase, onderdrukt door het IMPLICIT NONE statement; zie 4.2. Het is voorgekomen dat twee verschillende programma-eenheden dezelfde naam kregen, hetgeen dat verwarrende resultaten oplevert. Er is weinig gebruik gemaakt van ondersteunende faciliteiten, zoals de debugger en andere (speciale) opties in de compiler, omdat het gebruik hiervan bij de afstudeerder onbekend was. Toch was het waarschijnlijk beter geweest wanneer er bijvoorbeeld één week zou zijn uitgetrokken om het systeem beter te leren kennen. De tijd die hierdoor verloren gegaan zou zijn, zou later wel weer terug gewonnen zijn.

Bij gebruik van een programmeertaal als Pascal zou de tijd benodigd voor implementatie zo'n 30 à 40 % korter geweest zou zijn. Dit komt voornamelijk doordat deze taal meer data typen kent (die dus niet nagebootst behoeven te worden) en doordat er tijdens het compileren strengere tests uitgevoerd worden. Het is voor Fokker zeker nuttig om het gebruik van een andere programmeertaal te overwegen. Bij voorbeeld Pascal, dat reeds zeer wijd verspreid is, of in de toekomst een geavanceerde taal als Ada, de mogelijke nieuwe standaard taal.

Database

Het gebruik van het relationele database pakket ORACLE is over het geheel genomen goed bevallen. Het is een fraaie manier om gegevens gestructureerd en

overzichtelijk op te slaan. Duidelijk was, dat voor Fokker het gebruik van dit pakket nieuw was, zodat er sprake was van enige onwennigheid. Maar inmiddels zijn ervaringen opgedaan die in de toekomst gebruikt kunnen worden.

Bij enige overschakelingen (zoals van de VAX 11/780 naar de VAX 11/785 en het in gebruik nemen van een nieuwe versie van het operating system VAX/VMS) waren er enkele kleine moeilijkheden die overwonnen moesten worden. Er vindt regelmatig back-up van ORACLE plaats, wat betekent dat het tijdelijk niet beschikbaar is. De twee grootste nadelen van de database komen mogelijk voort uit het ontwerp hiervan, namelijk de database is te traag en het bijhouden van de relatie tabellen is lastig; zie 5.5. Met de in 5.5 gedane voorstellen moeten deze nadelen opgelost kunnen worden.

Grafisch pakket

Het gebruik van GINO-F is zeer goed bevallen. Omdat GINO veel overeenkomsten vertoont met het (voor de afstudeerder bekende) grafische pakket GPGS en omdat GINO goed gedocumenteerd is, kon dit pakket moeiteloos worden gebruikt. Bovendien bleek GINO in de praktijk ook zeer betrouwbaar. Wel zou het prettig zijn wanneer er leesbare foutmeldingen worden gegeven in plaats van nummers waarvan de betekenis in de appendix van de handleiding (ref. 19) moet worden opgezocht. Ook ontbraken er enkele bruikbare functies, zoals het overgaan bij tekenen op een Tektronix 4114 (zie 2.3) van de refresh mode naar de storage mode en een functie waarmee op de Tektronix 4115 snel het scherm schoongemaakt kan worden. Daarom zijn voor deze functies zelf routines geschreven die gebruik maken van "escape sequences". Ondanks de positieve ervaringen is het voor Fokker zinvol om na te gaan in hoeverre de standaard GKS gebruikt kan worden voor nieuwe grafische toepassingen.

Besturing van het systeem

Bij gebruik van het commandotaal pakket COLAS, is gebleken dat de betrouwbaarheid, documentatie en de algemene indruk toch duidelijk minder zijn dan die van pakketten van professionele software houses, zoals GINO-F en ORACLE. Het eindresultaat in VISUAL, zoals dat door de aerodynamische ontwerper gebruikt moet worden, is toch zeker wel acceptabel ondanks enkele nadelen. Enkele nadelen voor de programmeur zijn onder andere (volgende blz.):

- Er kan geen willekeurig scheidingsteken worden gebruikt bij een commando met één of meer parameters. De meeste gebruikers geven de voorkeur aan een spatie als scheidingsteken in plaats van een komma, zoals dat in COLAS gebeurt.

- De file, waarin de commando's en helpinformatie zijn beschreven, moet met een gewone editor (bijvoorbeeld EDT) gemaakt worden en dit is heel erg lastig. Hiervoor zou een speciale editor moeten komen. Ook het vertalen van deze file met het programma CDM duurt erg lang en er wordt bij fouten slecht aangegeven wat er mis is.

- Gebrekkige mogelijkheden voor parameter typen. Zo bestaan er geen logicals (of booleans) en ook het enumeratie type wordt node gemist. Verder kunnen er geen ranges van getallen worden opgegeven. Hiervoor is in VISUAL speciaal een parser geschreven.

- Het ontbreken van de mogelijkheid voor de (eind)gebruiker om macro's te definiëren. Een macro is een reeks commando's die vaak achter elkaar gebruikt worden en daarom door de gebruiker via één commandonaam aangeroepen moet kunnen worden.

- Er ontbreken globale commando's in COLAS. In veel toepassingsprogrammatuur zijn er commando's die in elk deelsysteem gebruikt moeten kunnen worden. In VISUAL zijn dit bijvoorbeeld de commando's waarmee naar een ander deelsysteem gesprongen kan worden. Dit moet nu geïmplementeerd worden door bepaalde commando's zes maal te beschrijven, namelijk in elke commandoset éénmaal.

- Onduidelijke foutmeldingen wanneer er iets mis gaat. Hiervoor zijn twee mogelijkheden: of het programma gaat "plat" (de uitvoering wordt gestopt) in een van de COLAS routines of er volgt een foutmelding "WARN YOUR SYSTEM ANALIST", waarna soms ook de executie wordt gestaakt.

- Bij het gebruik van bepaalde versies van de file waarin de commando's en meldingen worden beschreven, die wel goed worden vertaald met CDM, gaat het programma na verloop van tijd (op een schijnbaar willekeurig tijdstip) "plat" in de routine INTACT.

- Namen van een aantal standaard commando's in COLAS kunnen niet veranderd worden. Bij voorbeeld EXIT in plaats van QUIT en HELP in

plaats van ?. Ook zou er helpinformatie gegeven kunnen worden wanneer de gebruiker (tweemaal) RETURN intypt en zou bij de helpinformatie over een bepaald commando ook gelijk de helpinformatie gegeven moeten worden van de parameters die bij dat commando horen.

- In de file waar de commando's en de meldingen beschreven staan moeten de meldingen in een oplopende volgorde staan. Vooral bij het toevoegen van meldingen is het lastig, omdat vaak een aantal meldingen bij elkaar horen (en die moeten dan ook bij elkaar staan).

- Bij het linken wordt altijd de melding gegeven: "UNDEFINED GLOBAL SYMBOL ABTCOL"; er ontbreekt kennelijk iets in COLAS.

- Als bij het beschrijven van een parameter wordt gesteld dat het maximum kleiner is dan het minimum (per vergissing) dan geeft het programma CDM hiervan geen melding.

- Als er geen aanhalingstekens worden gebruikt bij het beschrijven van messages dan is het niet toegestaan de tekens "(", ")" en "=" te gebruiken.

- Bij het gebruik van COLAS ontstaan veel (overbodige) files; bijvoorbeeld: CIN.DAT, COUT.DAT, FOR000.DAT, FOR002.DAT, FOR080.DAT, FOR081.DAT, FOR082.DAT, FOR088.DAT, LOGCOL.DAT, etc.

6.2 - De kwaliteit van VISUAL

Dit is natuurlijk een moeilijk punt om objectief over te oordelen, desalniettemin zal er toch een poging worden gewaagd. Het systeem voldoet zonder problemen aan de in 2.4 gestelde functionele systeemeisen, met uitzondering van de functies CONTOUR en CARPET die wegens tijdgebrek niet geïmplementeerd zijn; zie 5.1. Naast de functies uit de systeemspecificatie zijn er een aantal functies toegevoegd, waarvan in de praktijk bleek dat deze wenselijk waren. Bij voorbeeld de functies voor kleurdefinitie. Op enkele uitzonderingen na zijn er door de gebruikers geen fouten ontdekt nadat het systeem door de afstudeerder grondig was getest.

Ook is aan de randvoorwaarden voldaan, met dien verstande dat het selecteren van gegevens te traag verloopt, maar de oorzaken hiervan liggen buiten VISUAL; zie 5.5. Door de gevolgde aanpak kan met grote zekerheid worden gesteld dat zowel de uitbreidbaarheid als de overdraagbaarheid van het systeem goed zal zijn; dit zal later in de praktijk blijken. Hoewel het moeilijk is om objectieve maatstaven te hanteren, kan toch worden gezegd dat VISUAL een gebruikersvriendelijk systeem is. Voor de gebruikers van VISUAL was er enige gewenning aan de commandotaal COLAS nodig. Dit kwam doordat deze een andere vorm van interactie gewend waren (in GAMMA). Gebruikers die nog niet zo aan GAMMA gewend waren hadden deze problemen niet. Pas nadat de diverse hulpmiddelen uit COLAS (opvragen van commando's, helpinformatie, parameterwaarden, etc.) gebruikt kunnen worden, kan VISUAL ten volle worden benut.

De gebruikershandleiding van VISUAL blijkt in de praktijk goed te voldoen. Deze handleiding is een eerste introductie tot het systeem, en bevat daarom geen volledige beschrijving van alle aanwezige functies. Bij een tweede versie van deze handleiding is het toch gewenst dat alle functies uitgebreid worden beschreven (ondanks het feit dat deze informatie ook "on-line" aanwezig is). En verder kunnen er meer tips worden gegeven voor een handig gebruik van VISUAL, zoals positioneren met de functie OUTLINE en het gebruik van de PROGRAM MODES bij het maken van afbeeldingen die in meerdere delen zijn opgesplitst; bijvoorbeeld 2, 3, 4 of meer delen.

6.3 - De driehoeksrelatie "THD-Fokker-afstudeerder"

Gedurende het afstudeerwerk zijn er steeds twee partijen geweest, die beiden hun eigen eisen stelden aan de afstudeerder. De twee partijen waren de Technische Hogeschool Delft en Fokker B.V. Schiphol. Vanuit de THD was er de eis dat de afstudeeropdracht voldoende wetenschappelijke inhoud moest hebben. Aan de andere kant wilde Fokker natuurlijk graag zoveel mogelijk praktisch bruikbare resultaten. Overigens had Fokker ook baat bij een opdracht zoals deze door de THD werd gewenst, want dit leverde een kwalitatief goed systeem op; zie 6.2. Door middel van gezamelijk overleg was al in een vroeg stadium een overeenkomst gesloten over de eisen die aan het afstudeerwerk gesteld zouden moeten worden.

In totaal zijn er zes bijeenkomsten geweest (waarvan drie bij de THD en drie bij Fokker), waarbij begeleiders van beide instellingen aanwezig waren. Verder zij er een tiental documenten geproduceerd, welke voortvloeiden uit de gevolgde software engineering methodiek en welke tevens als communicatie middel dienden. Ten slotte werd er ook via de student, tijdens formeel en informeel overleg met begeleiders van de THD of Fokker, gecommuniceerd. Achteraf kan worden vastgesteld dat de relatie "THD-Fokker" zowel effectief als prettig was.

Vanaf de oriëntatiefase (1 december 1984) tot en met de uitslag vergadering (29 november 1985) is de afstudeerder gedurende zes maanden in dienst geweest bij Fokker. Het zal duidelijk zijn dat dit verplichtingen betreffende het aanwezig zijn met zich meebracht. Opgemerkt wordt dat het nooit een probleem was om vrij te krijgen voor overleg met de begeleiders van de TH.

Het zou overigens zeer nuttig zijn geweest wanneer de afstudeerder gedurende het afstudeerwerk de beschikking zou hebben gehad over een Personal Computer (PC). Dit in verband met de vele documentatie die is geproduceerd. De toch al druk bezette WANG tekstverwerker zou zo ontlast kunnen worden en bovendien zou de documentatie in alle rust thuis geschreven kunnen worden. Verder zou ook een groot gedeelte van de programma ontwikkeling op de PC hebben kunnen plaatsvinden. Helaas heeft de afstudeerder hier niet tijdig aan gedacht en heeft hij ook nooit bij Fokker geïnformeerd naar de mogelijkheden die hiervoor bestaan. Het is misschien toch een goede tip voor toekomstige afstudeerders.

6.4 - Enige aspecten betreffende de planning

Oorspronkelijk was het de bedoeling dat eind juni 1985 het afstudeerwerk afgerond zou zijn; zie 1.5. Aangezien de uitslagvergadering, waarin het afstudeerwerk beoordeeld zal worden, plaatsvindt op 29 november 1985, is het duidelijk dat er sprake is van een grote vertraging. De reden hiervan is niet dat er door de afstudeerder langzaam is gewerkt, maar dat het afstudeerwerk veel te omvangrijk was voor een periode van zes maanden, welke officieel voor het afstuderen staat; ref. 1, 2 en 3. Toch is bij de planning in 1.5 getracht, tegen beter weten in, het werk over zes maanden te verdelen.

Andere oorzaken, welke minder belangrijk dan bovengenoemde oorzaak zijn, zijn onder andere het op de (schijnbaar) meest ongelegen momenten "down" gaan van de computer, het (soms veel te) lang zoeken naar kleine fouten in het programma en het ontbreken van een plannings methode (bijvoorbeeld netwerk planning) in de software engineering methodiek (ref. 6). En ten slotte heeft het produceren van documenten, en dit afstudeerverslag niet in de laatste plaats, ook veel tijd gekost.

Fase	aantal weken	
	werkelijk	gepland
2 Systeemdefinitie	6	(5)
3 Systeemdefinitie	8	(6)
4 en 5 Implementatie en test	16	(10)
Vakantie	5	(0)
Afstudeerverslag *)	8	(5)
Afstudeervoordracht *)	4	(0)
TOTAAL	47	(26)

Tabel 6.1 - De werkelijke tijdsbesteding

*) Tijdens deze fasen zijn ook nog een aantal laatste wensen van de gebruiker geïmplementeerd en nog niet eerder ontdekte fouten verwijderd uit het programma.

In tabel 6.1 wordt een overzicht gegeven van de werkelijke tijdsbesteding vanaf fase 2, de systeemspecificatie. Het betreft hier een periode van ongeveer 11 maanden ofwel 47 weken, wat 80 % meer is dan de oorspronkelijke 26 weken. Tussen haakjes wordt aangegeven hoeveel weken er voor een bepaalde fase gepland was.

Uit tabel 6.1 blijkt dat fasen 2 en 3 redelijk volgens de planning verlopen zijn. De fasen 4 en 5 (waarin nog veel detailontwerp moest worden verricht) hebben echter aanzienlijk meer tijd gekost. Deze fasen omvatten echter ook het schrijven van een vijftig pagina's tellende handleiding. Omdat duidelijk werd dat het afstudeerwerk niet voor eind juni gereed zou zijn is er een vakantie periode ingelast van 5 weken.

Ook het produceren van het afstudeerverslag heeft meer tijd gekost dan de bedoeling was. Het produceren van het verslag omvatte niet alleen het schrijven, maar ook het typen, verzorgen van de figuren, vermenigvuldigen en het inbinden van het geheel. Het houden van een voordracht kost tijd, want er moet materiaal en tekst worden verzameld. De vier weken die hiervoor in tabel 6.1 zijn vermeld hebben niet alleen betrekking op de afstudeervoordracht maar ook op de voordracht die op 25 april 1985 is gehouden in het kader van een projectbijeenkomst Technische Toepassingen van de Informatica.

6.5 - Enkele kwantitatieve aspecten rondom VISUAL

Er zal nu een lijst volgen van kwantitatieve aspecten welke een indicatie van de omvang van het werk geven. Er zal hierop geen commentaar worden gegeven.

- documenten	10
- bladzijden van de documenten	450 blz.
- files met source code van VISUAL	26
- routines in VISUAL	226
- source code van VISUAL	12.662 regels
- idem	283 Kbyte
- compileertijd (CPU)	4 min 19 sec
- linktijd (CPU)	37 sec
- executable code van VISUAL	347 Kbyte
- deelsystemen in VISUAL	5
- commando's	81
- parameters bij deze commando's	71
- meldingen en waarschuwingen in VISUAL	132
- file met beschrijving van commando's en param.	1.395 regels
- door CDM vertaalde file	126 Kbytes
- ontwikkeling van VISUAL	11 maanden

Opmerking 1: De bovenstaande getallen/hoeveelheden betreffen alleen VISUAL zelf en niet de gebruikte ondersteunende pakketten zoals GINO-F, COLAS en ORACLE.

Opmerking 2: Veel van de hierboven genoemde getallen/hoeveelheden zijn (vrij nauwkeurig) geschat en het is ook mogelijk dat deze in de loop der tijd iets veranderen.

Opmerking 3: Bij de afmetingen van de source code van VISUAL is elk common-blok maar één keer meegeteld, ook al komt het meerdere keren voor.

6.6 - De conclusie

De gebruikte software engineering methodiek heeft ertoe geleid dat bij de implementatie nauwelijks afgeweken behoefde te worden van het ontwerp, dat was gecreëerd uitgaande van de systeemdefinitie. De methodiek leverde veel documentatie op maar had als nadeel dat zij niet voorzag in een ondersteuning voor de planning.

De hardware en de ondersteunende software bleken uitstekend te voldoen, met uitzondering van COLAS. Het resultaat van de samenwerking "THD-Fokker-afstudeerder" is een stuk gereedschap voor de aerodynamische ontwerper waar de komende jaren mee gewerkt kan worden, eventueel in een uitgebreidere en aangepaste vorm.

Tot slot nog één voorbeeld waarin het bovenstaande nog eens duidelijk wordt onderstreept. In de loop der tijd heeft het weergeven van de drukverdeling op een geometrie de volgende ontwikkeling ondergaan:

1) De drukverdeling wordt weergegeven door middel van één numerieke waarde per paneel. Dit eventueel weergegeven op de geometrie. Dit gebeurt met de functie NUMBER-CP; zie appendix B5.

2) De drukverdeling wordt weergegeven door de panelen van de geometrie in te kleuren met een van de drukwaarde afhankelijke kleur. Dit gebeurt met de functie COLOUR-CP; zie appendix A2.

3) De drukverdeling wordt weergegeven in de vorm van isobaren-banden, welke nog niet continu verlopen over de segmentgrenzen. Dit gebeurt met de functie ISO-BAND; zie appendix A4 bovenaan.

4) De drukverdeling wordt weergegeven in de vorm van isobaren-banden, welke continu verlopen over de segmentgrenzen. Dit gebeurt met de functie ISO-BAND; zie appendix A4 onderaan.

De eerste twee manieren van weergeven van de drukverdeling waren ook al in GAMMA mogelijk. Het zal verder duidelijk zijn dat het visualiseren van de drukverdeling op de geometrie in de loop der tijd aanzienlijke verbeteringen heeft ondergaan.

Literatuur

Tijdens het afstuderen en het schrijven van dit afstudeerverslag is er gebruik gemaakt van een aantal documenten. Vanuit de tekst van dit verslag zal hiernaar verwezen worden door middel van de letters "ref.", welke gevolgd worden door één of meer nummers. De documentatie bestaat onder meer uit voorlichtingsmateriaal en collegedictaten van de THD, rapporten van Fokker, boeken, (systeem)documentatie, wetenschappelijke artikelen en documenten welke door de afstudeerder geschreven zijn.

1. Voorlichtingscommissie van de Onderafdeling der Wiskunde en Informatica.
Informatica studeren in Delft.
Augustus 1983.

2. Vakgroep Technische Toepassingen van de Informatica.
Inlichtingen voor studenten.
Mei 1985.

3. Voorlichtingscommissie van de Onderafdeling der Wiskunde en Informatica.
Afstuderen als Informatica Ingenieur.
Augustus 1984.

4. H. Wittenberg.
Dictaat Elementaire Vliegtuigbouwkunde.
April 1983.

5. D. Laan.
Gebruikersspecificatie.
December 1984.

6. J. Groesdijk en P. Schouten.
Software Engineering - Procedures en Standaarden.
November 1984.

7. Onderafdeling der Wiskunde en Informatica, TH-Delft.
Handleiding voor het Practicum (In 3) Informatica (omvat PRODOSTA).
Augustus 1983.

8. J. van der Kolk en H. Sytsum.
Program Panel (Version 1.8) User Guide.
November 1983.

9. Digital Equipment Corporation, Maynard, Massachusetts.
VAX/VMS Primer.
May 1982.

10. Digital Equipment Corporation, Maynard, Massachusetts.
VAX/VMS User Introduction Student Guide.
1982.

11. Digital Equipment Corporation, Maynard, Massachusetts.
Introduction to EDT editor.
April 1982.

12. C. Hall.
SED Editor.
August 1983.

13. Oracle Corporation, Menlo Park, California.
Oracle User Manual, Volume 1 & 2.
May 1983.

14. Oracle Corporation, Menlo Park, California.
Oracle training course, student workbook for module 1 & 2.
May 1983.

15. J. Langendoen.
Dictaat Databases (Hoofdstuk 7 - SQL)
1983.

16. J. ter Bekke.
Database Ontwerp.
Juni 1983.

17. J. Ullman.
Principles of Database Systems.
1982.

18. C. Data.
An Introduction Database Systems.
March 1981.

19. CAD Centre, Cambridge.
GINO-F (2.7A) reference guide.
October 1983.

20. CAD Centre, Cambridge.
GINO-F (2.7B) reference guide.
October 1983.

21. D. Doornink en C. Dalrymple.
The Architectural Evaluation of a High-Performance Graphics Terminal.
April 1984.

22. Tektronix, Beaverton, Oregon.
4110 series command references.
December 1983.

23. Tektronix, Beaverton, Oregon.
4110 series host programmers.
January 1984.

24. Tektronix, Beaverton, Oregon.
4114B/4116B computer display terminals.
May 1983.

25. N. Voogt.
Computer gesteund aerodynamisch ontwerp bij Fokker.
Mei 1983.

26. R. van der Dam en T. Mandemaker.
Een CAE-systeem voor multi-disciplinair ontwerp.
December 1983.

27. T. Mandemaker.
GAMMA User Manual (version 1.6).
April 1985.

28. P. van Oosterom.
Verloop van de oriëntatiefase bij Fokker B.V., Amsterdam.
November 1985.

29. P. van Oosterom.
Aantekeningen voor de bespreking op 16 januari 1985.
Januari 1985.

30. P. van Oosterom.
Systeemdefinitie VISUAL.
Februari 1985.

31. G. Nijhuis.
User's Manual for ORLAS "Commando Language System".
March 1984.

32. T. Westrup, W. Kegel en J. Gras.
User Interaction with an environment for image processing and graphics.
1984.

33. W. Kegel.
FIPS User's Manual.
February 1984.

34. F. Post, F. Pauwels, W. Bronsvoort, H. Stienen en D. McConalogue.
Collegedictaat a215 CAD/CAM I, Deel I en II.
Februari/maart 1984.

35. J. Rigter.
Richtlijnen voor opstellen van computerprogamma's.
Augustus 1984.

36. P. van Oosterom.
Systeemontwerp VISUAL.
Maart 1985.

37. P. van Oosterom.
Systeemontwerp VISUAL (2e versie).
April 1985.

38. B. Baumgart.
A polyhedron representation for computer vision.
1975.

39. M. Pratt.
Graph based models and their data structures.
July 1981.

40. C. Eastman en M. Henrion.
GLIDE: A language for design information systems.
1977.

41. K. Weiler.
Edge-Based Data Structures for Solid Modeling in Curved-Surfase Environments.
January 1985.

42. W. Newman en R. Sproull.
Principles of interactive computer graphics.
November 1978.

43. D. Monro.
Fortran 77.
1983.

44. D. Warn.
Lighting controls for synthetic images.
October 1983.

45. P. van Oosterom.
Gebruik van de programma interface tussen oracle en een applicatie programma.
Maart 1984.

46. P. van Oosterom.
Enige snelheidsaspecten betreffende het maken van afbeeldingen.
Februari 1985.

47. M. Merchant.
Fortran 77 Language and Style.
1981.

48. D. Groot.
GKS Dynamic Memory application.
December 1983.

49. J. van Amstel, J. Bomhoff en G.J. Schoenmakers.
Inleiding tot het programmeren, deel I.
1980.

50. J. van Amstel, J. Bomhoff en G.J. Schoenmakers.
Inleiding tot het programmeren, deel II.
1981.

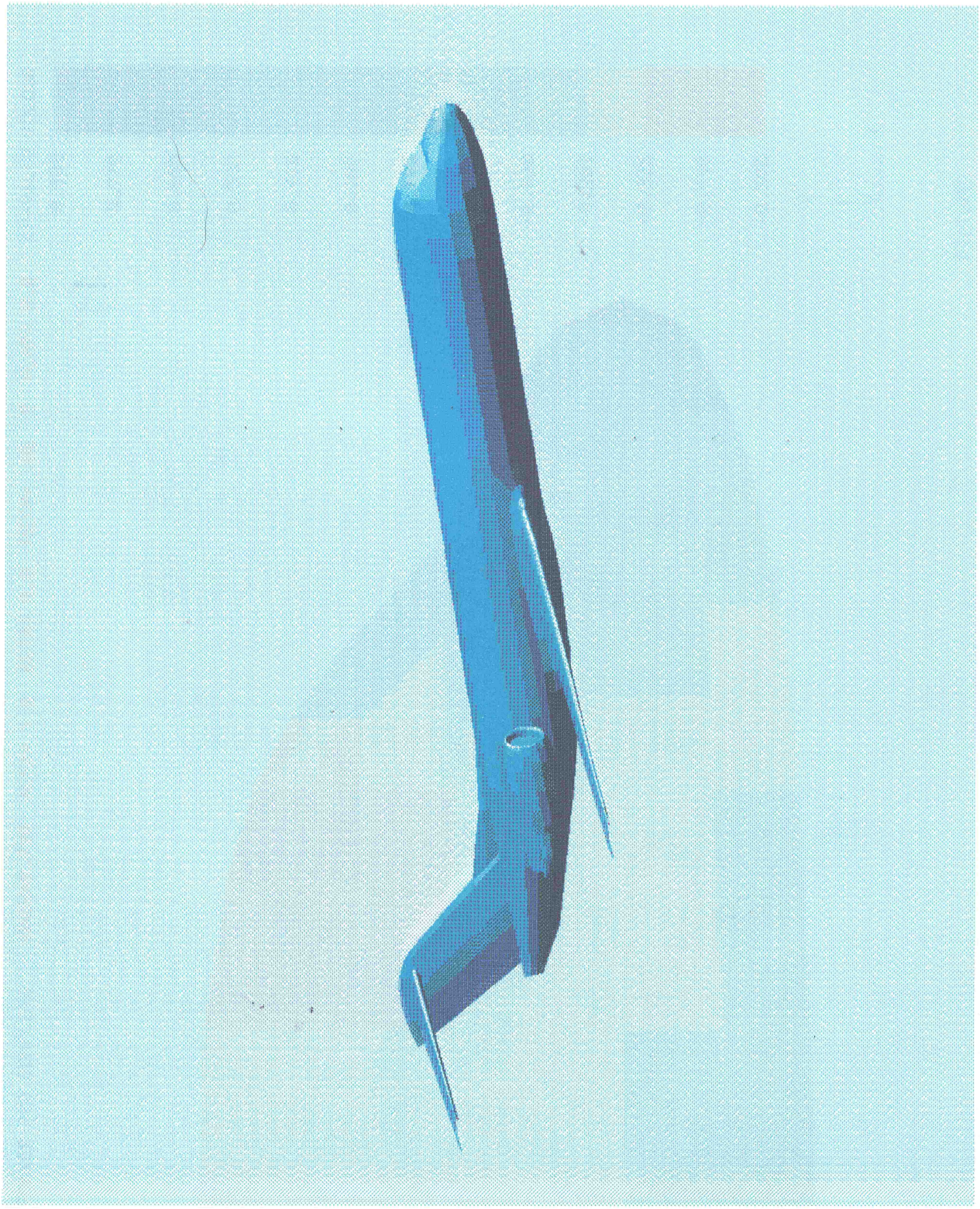
Appendix A

Kleurenappendix

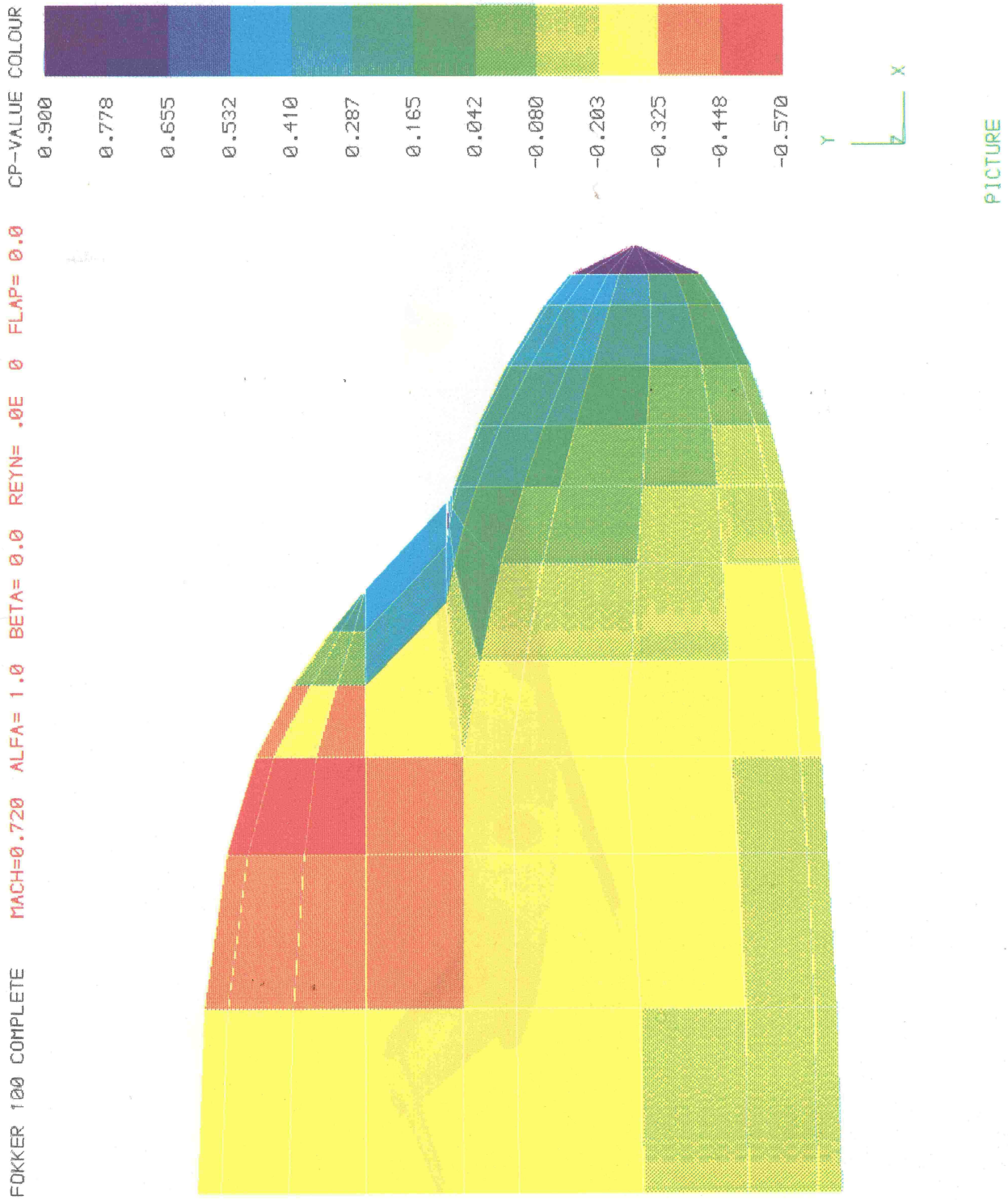
INHOUD

A1. De functie SHADE 116
A2. De functie COLOUR-CP 117
A3. De functie ISO-BAND 118
A4. De functie ISO-BAND met en zonder "BOUNDARY-CORRECTION" 119

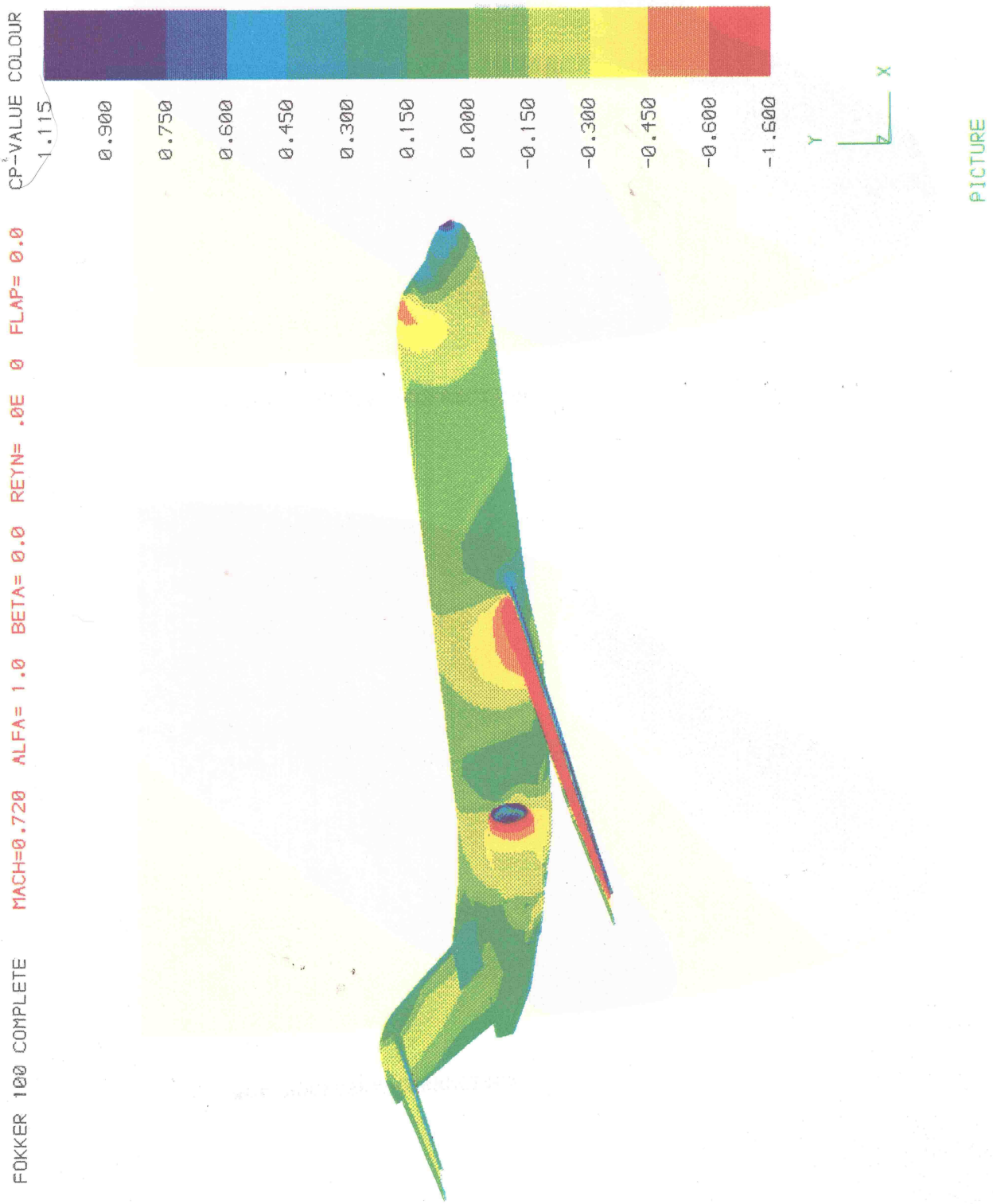
A1. De functie SHADE



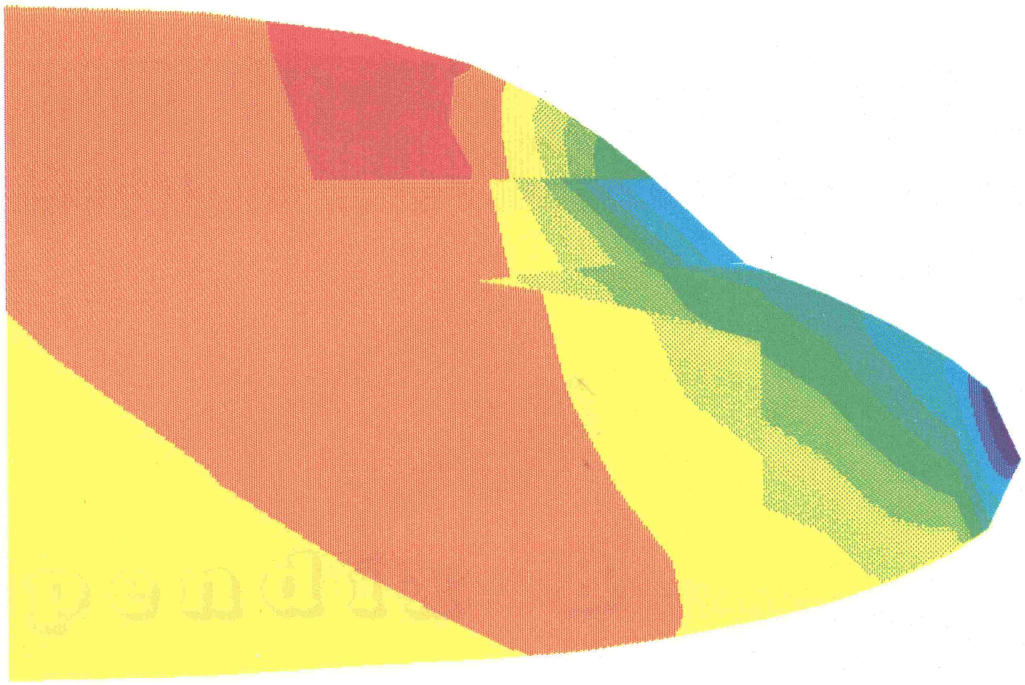
A2. De functie COLOUR-CP



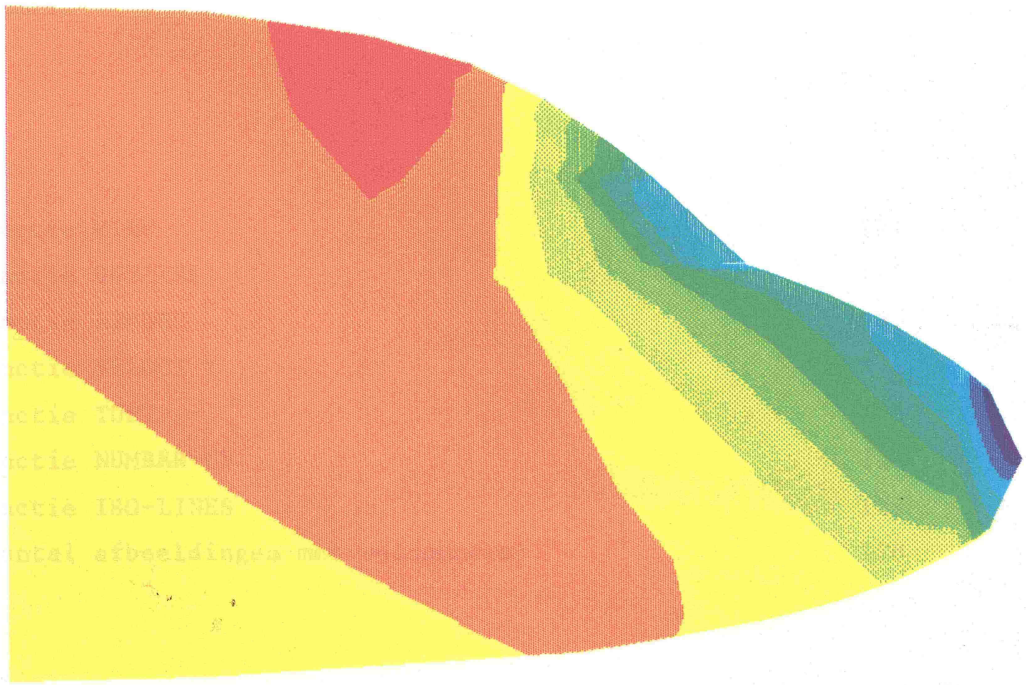
A3. De functie ISO-BAND



A4. De functie ISO-BAND met en zonder "BOUNDARY-CORRECTION"



zonder BOUNDARY-CORRECTION



met BOUNDARY-CORRECTION

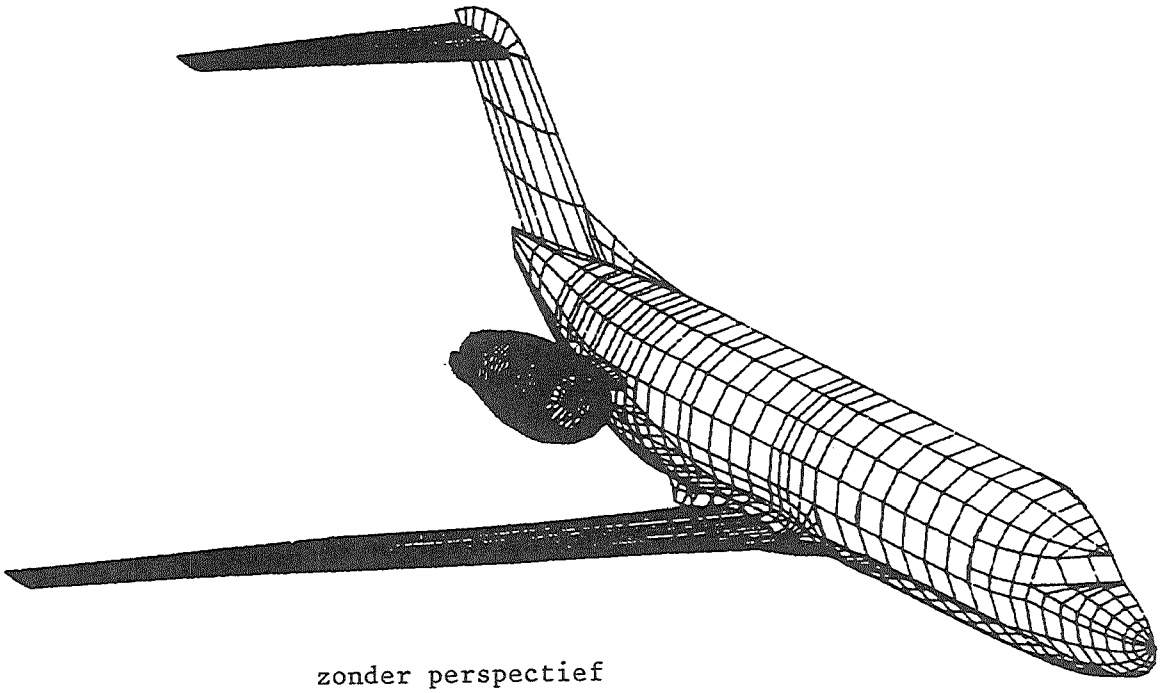
Appendix B

Tekeningen Z/W

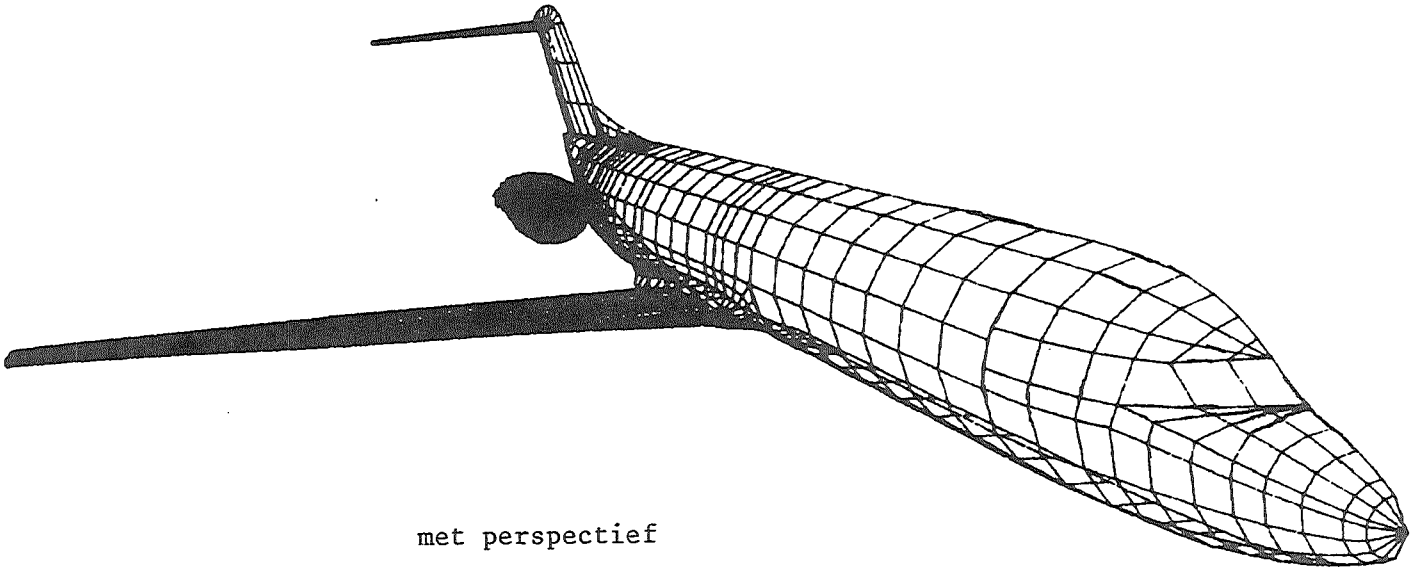
INHOUD

B1. De functie WIRE 121
B2. De functie OUTLINE 122
B3. De functie HIDDEN 123
B4. De functie VELOCITY 124
B5. De functie TUFT 125
B6. De functie NUMBER-CP 126
B7. De functie ISO-LINES 127
B8. Een aantal afbeeldingen met veldpunten 128

B1. De functie WIRE

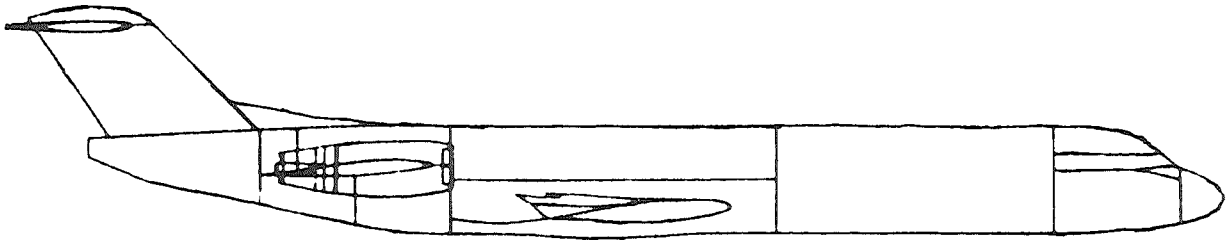


zonder perspectief

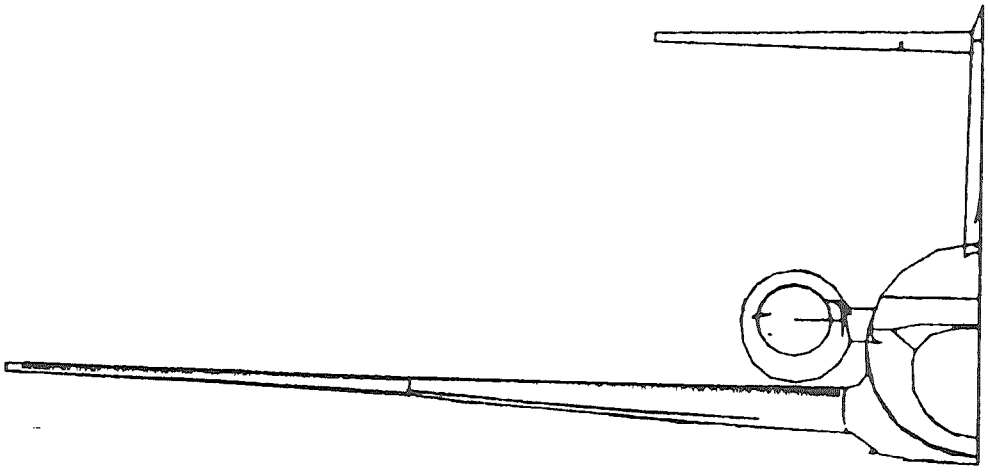


met perspectief

B2. De functie OUTLINE

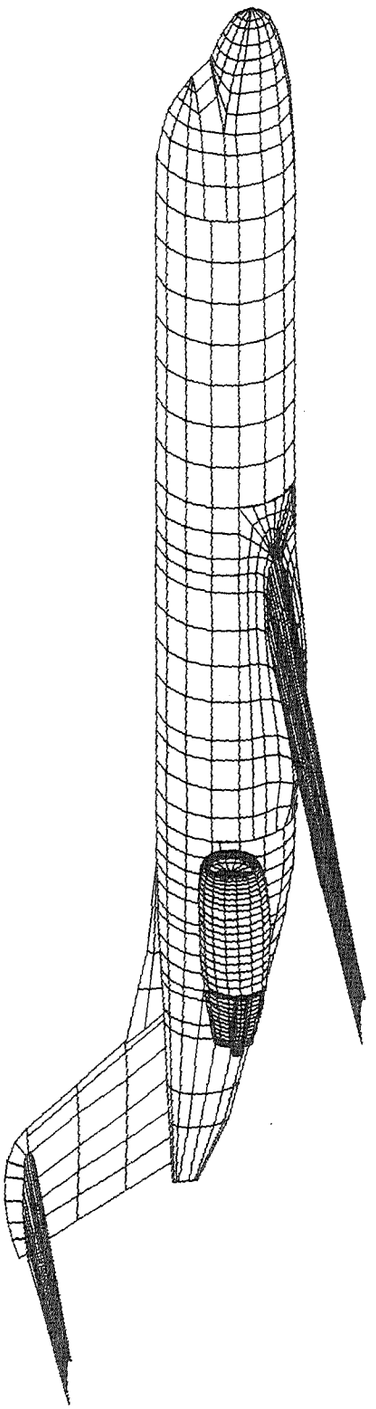


zijaanzicht

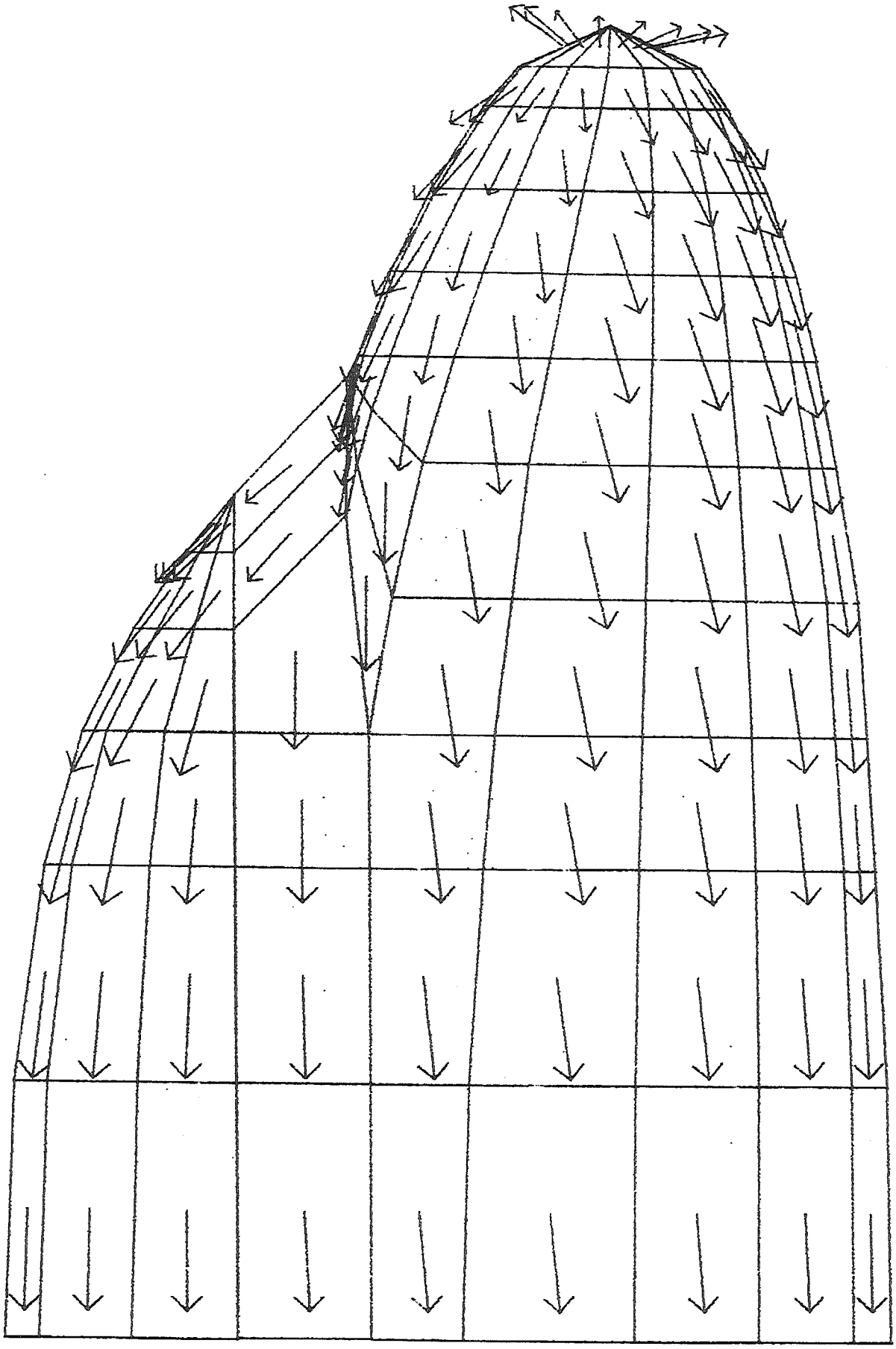


vooraanzicht

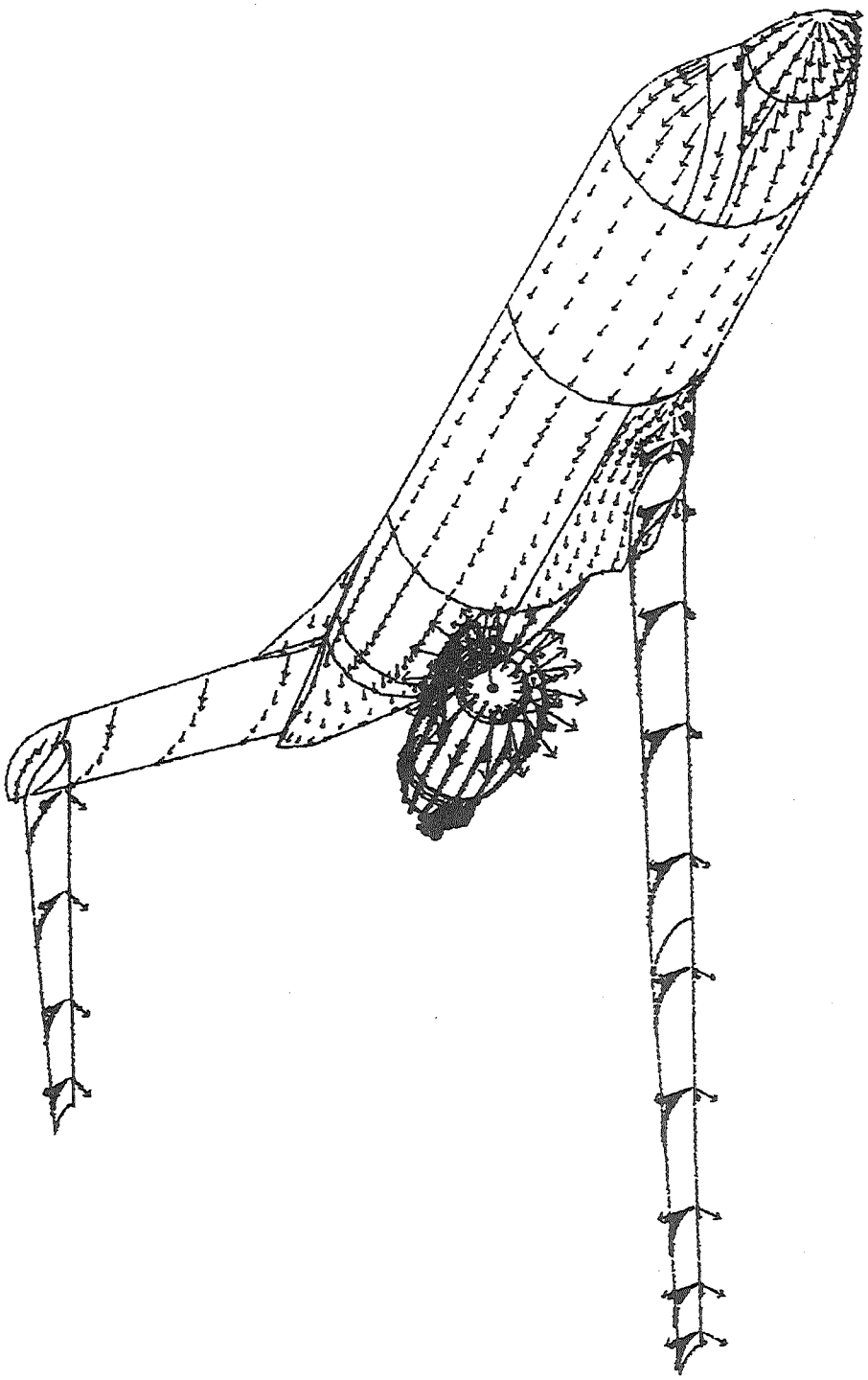
B3. De functie HIDDEN



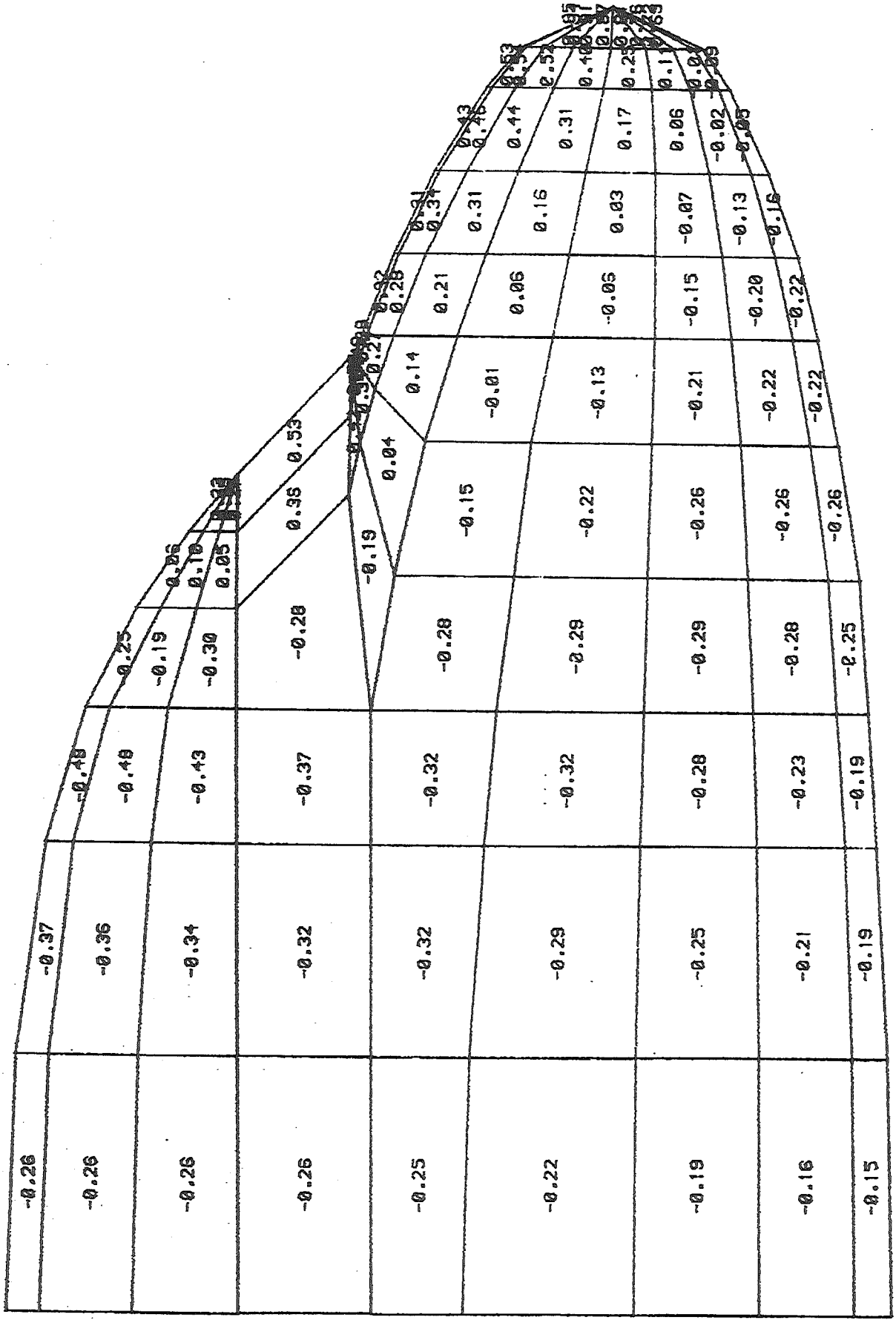
B4. De functie VELOCITY



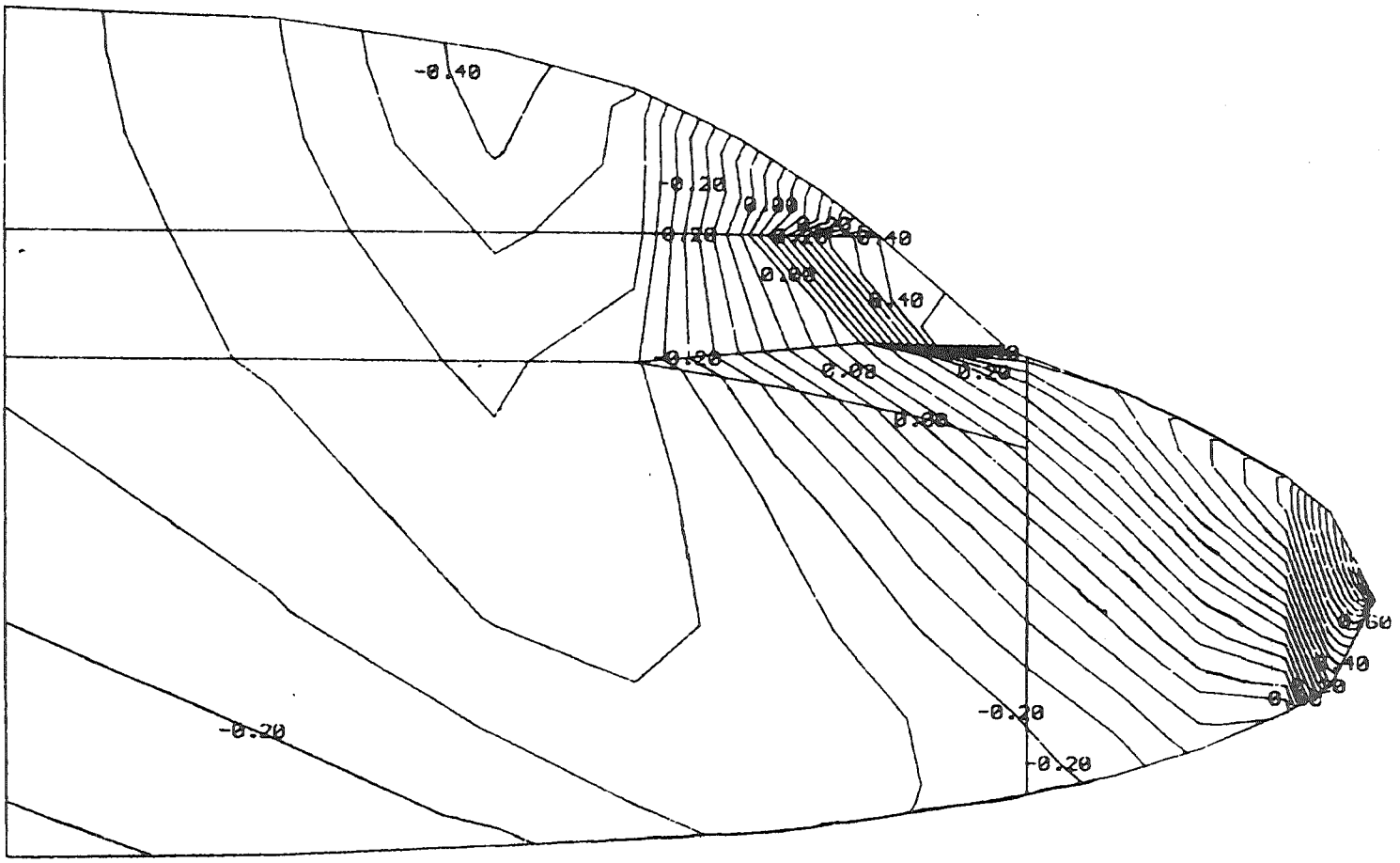
B5. De functie TUFT



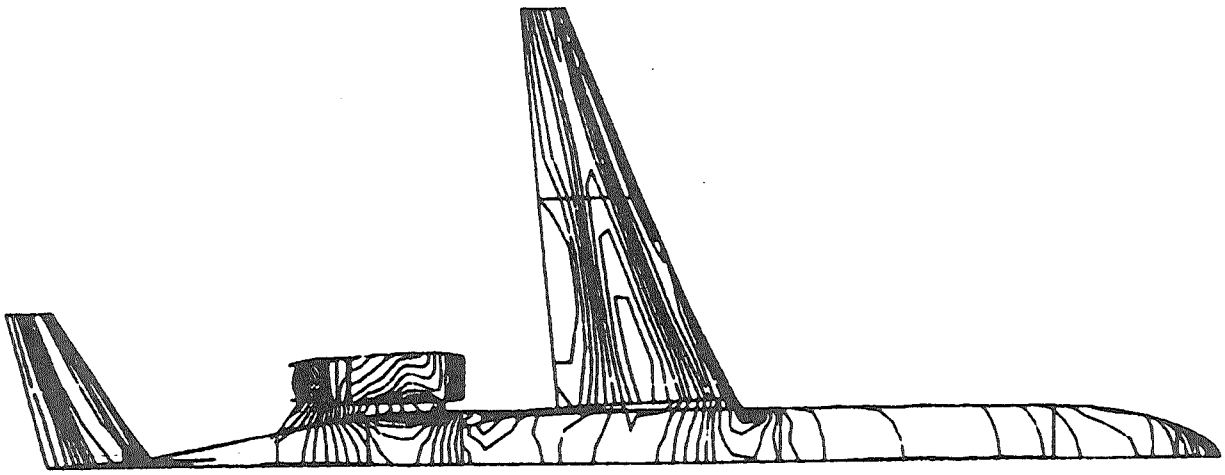
B6. De functie NUMBER-CP



B7. De functie ISO-LINES

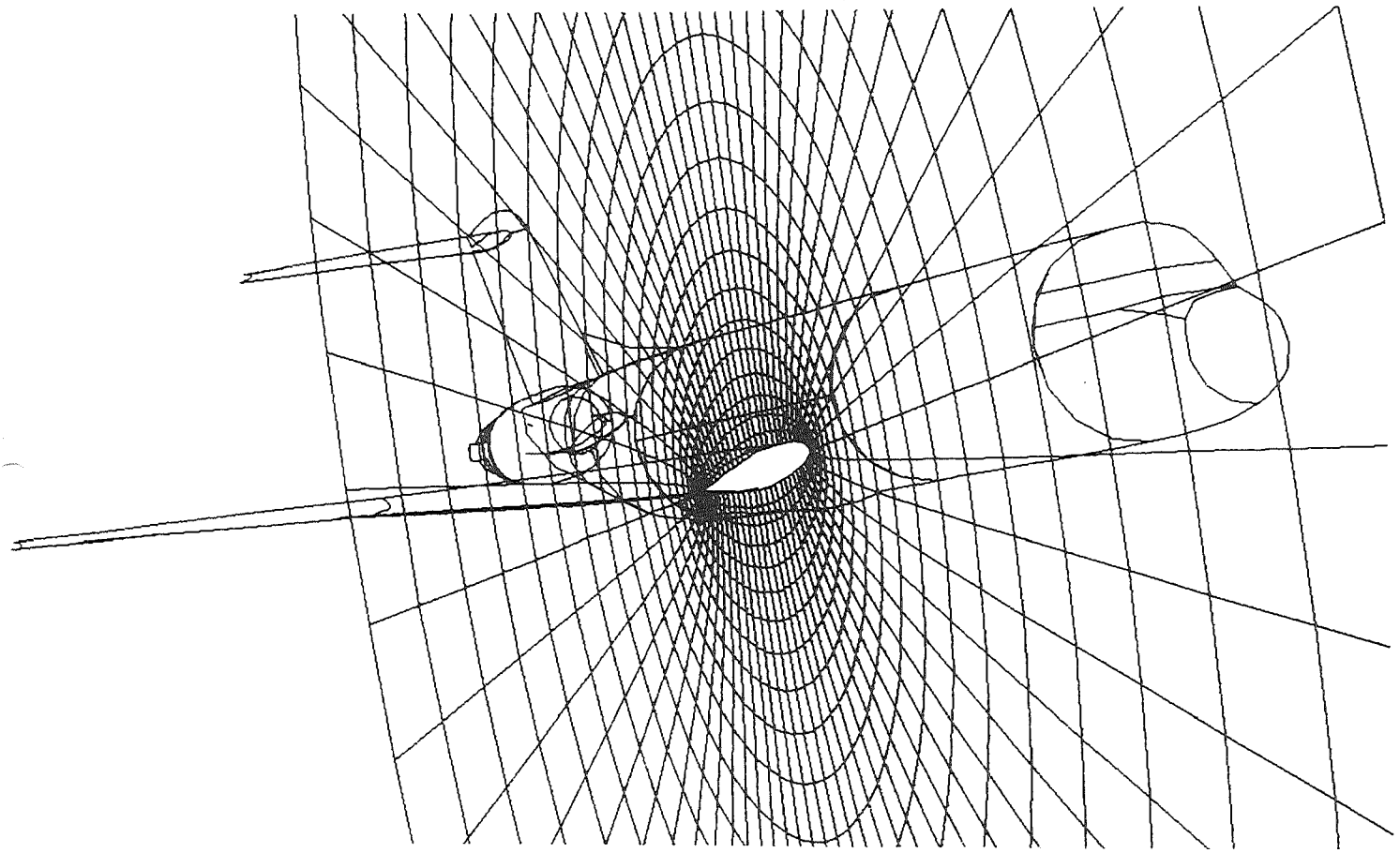


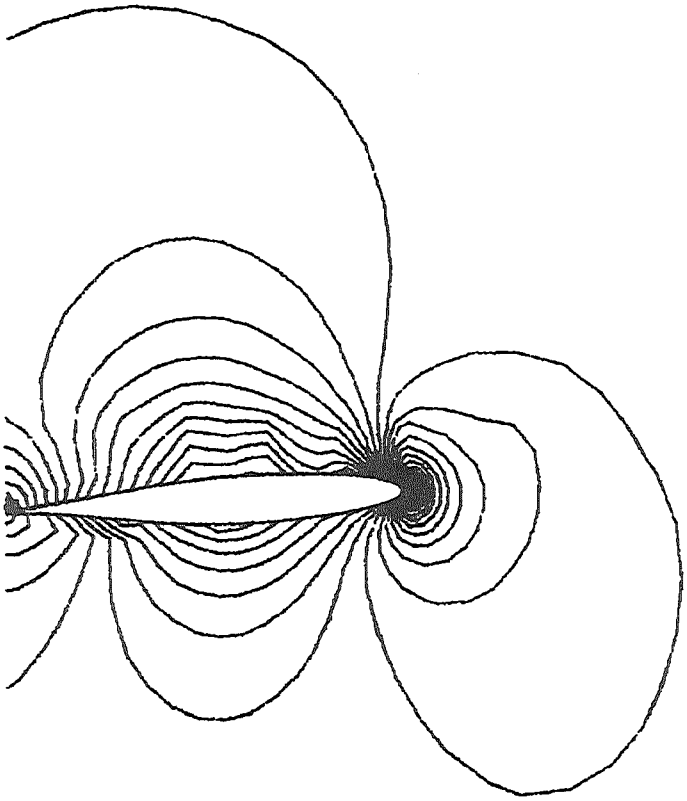
cockpit

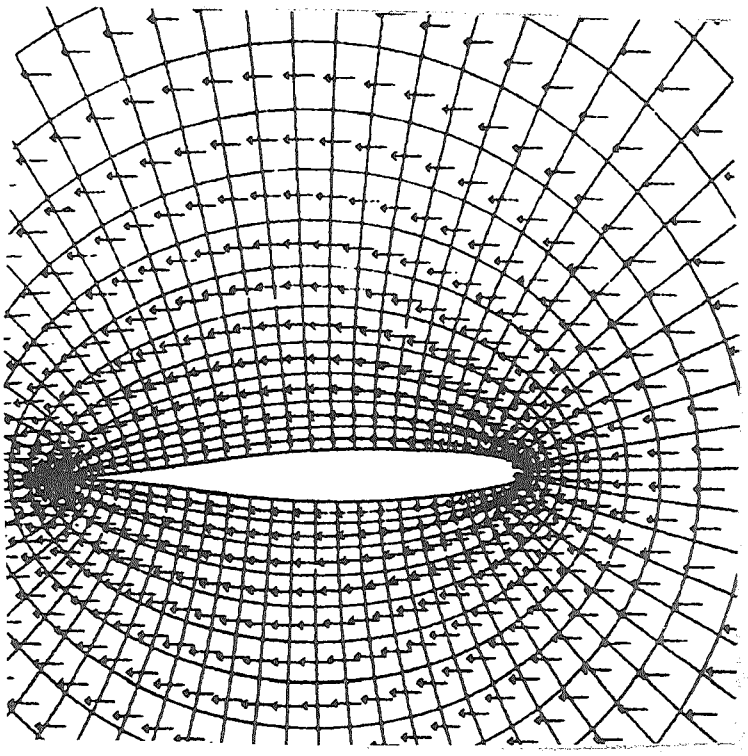


onderaanzicht

B8. Een aantal afbeeldingen met veldpunten







Appendix C

Commando overzicht VISUAL

INHOUD

C1. Commando's op het hoofdniveau 130
C2. Commando's in het deelsysteem DIRSEL 130
C3. Commando's in het deelsysteem 3D-FUNCTIE 131
C4. Commando's in het deelsysteem SET-UP 132

C1. Commando's op het hoofdniveau

	COMMANDONAAM	PARAMETERS
1	DIRECTORY	
2	SELECT	
3	DRAW	
4	3D-FUNCTIE	
5	2D-FUNCTIE	
6	1D-FUNCTIE	
7	SET-UP	
8	MODE	

C2. Commando's in het deelsysteem DIRECTORY-SELECT

	COMMANDONAAM	PARAMETERS	
1	DIRECTORY	TYPE	CAPTAIN
2	SELECT	TYPE	VERSION
3	DRAW		
4	3D-FUNCTIE		
5	2D-FUNCTIE		
6	1D-FUNCTIE		
7	SET-UP		
8	MODE		
9	ACTIVE		
10	ADD	TYPE	VERSION
11	DROP	TYPE	VERSION
12	LISTING	TYPE	FULL/SHORT

C3. Commando's in het deelsysteem 3D-FUNCTIE

	COMMANDONAAM	PARAMETERS				
1	DIRECTORY					
2	SELECT					
3	DRAW					
4	3D-FUNCTIE					
5	2D-FUNCTIE					
6	1D-FUNCTIE					
7	SET-UP					
8	MODE					
9	WIRE	ON/OFF	WIRETYPE			
10	OUTLINE	ON/OFF				
11	SHADE	ON/OFF	X-LIGHT	Y-LIGHT	Z-LIGHT	
12	HIDDEN	ON/OFF				
13	ISO-LINE	ON/OFF	ISOBAR	START-CP	STEP	NUMBER
14	ISO-BAND	ON/OFF	ISOBAND	START-CP	STEP	
15	VELOCITY	ON/OFF	LENGTH			
16	TUFT	ON/OFF	LENGTH			
17	COLOUR-CP	ON/OFF				
18	NUMBER-CP	ON/OFF	X-SIZE	Y-SIZE		
19	CONTOUR	ON/OFF				
20	CARPET	ON/OFF				
21	ZOOM	ZOOMTYPE	FACTOR			
22	ROTATE	AX	ANGLE	MODE		
23	SHIFT	X	Y	Z	MODE	
24	VIEW	FACTOR				
25	MATRIX	OPERATION	NUMBER	NAME		
26	CLEAR	SAV-MATRIX	DEF-COLOUR			
27	LOGO	ON/OFF	RADIUS			

C4. Commando's in het deelsysteem SET-UP

	COMMANDONAAM	PARAMETERS			
1	DIRECTORY				
2	SELECT				
3	DRAW				
4	3D-FUNCTIE				
5	2D-FUNCTIE				
6	1D-FUNCTIE				
7	SET-UP				
8	MODE				
9	STORE				
10	SYSTEM-DEF				
11	USER-DEF				
12	INFO-VAR	VARIABLE			
13	CHANGE-VAR	VARIABLE	VALUE		
14	HUE-COLOUR	HUE-FROM	HUE-TO	COL-FROM	COL-TO
15	LIGHTNESS	HUE	COL-FROM	COL-TO	
16	SATURATE	SATURATE	COL-FROM	COL-TO	
17	COLOUR-DEF	COLOUR-NR	HUE	LIGHTNESS	SATURATION

Appendix D

Verklarende woordenlijst

INHOUD

Baud - Cl 134
Cm - directory 135
EDT - grensbeschrijving 136
group - parameter 137
part - VAX 138
VAX/VMS - winged edge 139

Baud : Dit is een maat voor de snelheid van een datalijn. Tussen Baud en bps bestaat het volgende verband : 1 Baud = log V bps. Hierin is V het aantal gebruikte signaalniveau's bij de transmissie.

bit : BInary digiT. Dit is de kleinste eenheid van informatie en kan de waarde 0 of 1 hebben.

boolean : zie logical.

bps : bits per seconde; Een maat voor de snelheid waarmee signalen over datalijn worden verzonden.

byte : In een byte staan een aantal bits (meestal 8) gegroepeerd tot een nieuwe eenheid, hiermee kan bijvoorbeeld één karakter gerepresenteerd worden.

CAD : Computer Aided Design. Het ontwerpen met behulp van de computer. Meestal door het interactief opbouwen en analyseren van 3D-modellen.

CAE : Computer Aided Engineering. Dit is een verzamelnaam voor alle technische toepassingen van de informatica. Het heeft een meer uitgebreide betekenis dan CAD/CAM.

CAM : Computer Aided Manufacturing. Het automatiseren in het productie proces, zowel de voorbereiding als de uitvoering.

CDM : Command Definition Module. Dit programma behoort bij het commando taal pakket COLAS. Het "vertaalt" de file waarin de commando's en helpinformatie staan beschreven (bijvoorbeeld VISCOL.DAT) naar de file BCLIF.DAT, die door de COLAS routines gebruikt kan worden.

Cd : Weerstandscoëfficiënt.

Cl : Liftcoëfficiënt.

Cm : Momentencoëfficiënt.

COLAS : COmmand LAnguage System; commandotaal ontwikkeld door het NLR. In 3.4 van dit afstudeerverslag wordt COLAS beschreven.

commando : Een opdracht of verzoek aan het systeem om een bepaalde actie uit te voeren. Een commando bestaat uit één commandonaam, die eventueel gevolgd wordt door één of meer parameters.

configuratie: Een verzameling van groups, parts en segmenten die bij elkaar horen en als zodanig in de database worden opgeslagen. In 2.2 wordt het begrip configuratie beschreven.

contour : Een serie punten welke door rechte lijnstukken worden verbonden (zie 2.2).

Cp : Drukcoëfficiënt.

database : Een verzameling gegevens waarop, via een speciaal software systeem (het DBMS), operaties kunnen worden uitgevoerd als : opslaan, wijzigen, tussenvoeggen, verwijderen en raadplegen van grote hoeveelheden gegevens.

datapoint : Alle aerodynamische gegevens van één configuratie welke bij één berekening (of meting) horen. Zie ook 2.2 van dit afstudeerverslag.

DBMS : DataBase Management System; Een software systeem waarmee op een database operaties uitgevoerd kunnen worden als : opslaan, wijzigen, tussenvoegen, verwijderen en raadplegen van de gegevens. Een voorbeeld van een DBMS is ORACLE.

DEC : Digital Equipment Corporation. Grote fabrikant van computers, onder andere van de VAX; de computer waarop VISUAL ontwikkeld is.

directory : Een overzicht van aanwezige gegevens op een bepaald gebied. Dit begrip heeft zowel betekenis binnen VISUAL als op het niveau van het operating system VAX/VMS.

EDT : EDiTor. Een software systeem waarmee op de computer (bijvoorbeeld de VAX) teksten en programma's geschreven en gewijzigd kunnen worden.

file : Verzameling van gegevens die als één geheel behandeld worden; meestal om naar gegevens die op schijf of op magneetband staan te verwijzen.

FIPS : Flexible Image Processing System. Een systeem waarin diverse applicatie programma's draaien. Een belangrijk onderdeel is het user interface package dat onder andere zorgt voor multitasking en dat de high level command language FCL (FIPS Command Language) ondersteund. Voor een uitgebreidere beschrijving van FIPS zie 2.3 van dit afstudeerverslag.

Fortran 77 : Formula Translator. Programmeertaal die voornamelijk wordt gebruikt voor het oplossen van wiskundige en wetenschappelijke problemen. Fotran is in vergelijking met andere programmeertalen zeer efficiënt ten aanzien van de rekentijd. De eerste versie van Fortran is in 1950 ontwikkeld door IBM. VISUAL is in Fortran 77 geïmplementeerd, zie 4.2.

GAMMA : Geometry Analysis & Manipulation for Multiple Application; interactief grafisch pakket waarmee geometrie gecreëerd en gewijzigd kan worden. Bovendien beschikt GAMMA ook over een aantal visualisatie functies.

geometrische gegevens : Dit zijn kwantitatieve gegevens, hierbij moet men denken aan vergelijkingen van oppervlakken en krommen en de coördinaten van hoekpunten; zie 2.2.

GINO-F : Grafisch pakket van procedures (routines), die het schrijven van (interactieve) grafische toepassings programma's vergemakelijkt; zie 2.3.

grens-beschrijving: Methode van solid modelling waarin een object gerepresenteerd wordt door zijn zijvlakken (faces) die elkaar alleen in

gemeenschappelijke zijden (edges) en hoekpunten (vertices) snijden; zie 2.2.

group : Een verzameling parts, die als één geheel worden doorgerekend voor een bepaalde stromingsconditie. In 2.2 wordt dit begrip beschreven.

hue : De hue geeft aan waar een kleur zich bevindt in het kleurenspectrum; bijvoorbeeld 0, 120 en 240 graden betekenen resp.de kleuren rood, groen en blauw.

IBM : International Business Machine. De grootste computer fabrikant ter wereld.

interactief : Manier van communiceren met de computer waarbij de gebruiker een opdracht (commando) geeft en het systeem vervolgens reageert door de opdracht uit te voeren.

lightness : De lightness geeft de lichtintensiteit van een kleur aan.

logical : Een type dat slechts twee waarden kan aannemen, n.l. "true" en "false" (resp. waar en onwaar).

on-line : Hierbij kan de gebruiker direct via zijn terminal met de computer communiceren om bij voorbeeld helpinformatie op te vragen.

operating system : Besturingssysteem van de computer. Dit is systeem software die de computer operaties bestuurd. VAX/VMS is één voorbeeld van een operating systeem.

ORACLE : Een relationeel database systeem. In deze database zijn de gegevens opgeslagen die VISUAL gebruikt (zie 2.3).

paneel : Een vierhoek die als basiselement wordt gebruikt om de geometrie te beschrijven. Elk paneel bezit een normaalvector (zie 2.2).

parameter : Een deel van een commando, waarmee een bepaalde waarde wordt gespecificeerd die bij het betreffende commando van belang is;

bij voorbeeld de parameter "X" in het commando "SHIFT,X=10".

part : Een verzameling van segmenten, die logisch gezien bij elkaar horen (zie 2.2).

PRODOSTA : PROject kontrole en DOcumentatie STAndaarden. Dit is een software engineering methodiek, zie ref. 7.

prompt : Eén of meer woorden waarmee het systeem aangeeft dat het op invoer van de gebruiker wacht.

saturation : De saturation geeft de "felheid" (of "kleurigheid") van een kleur aan, dat wil zeggen is de kleur heel fel of is deze juist grijs-achtig.

segment : Een aantal contouren, die elk uit een even groot aantal punten bestaan, vormen tesamen één segment (zie 2.2). Segmenten worden gebruikt om 'lappen' van een oppervalk te beschrijven.

solid modelling : Het maken van complete representaties van massieve drie-dimensionale objecten, zodat automatisch divese goed-gedefinieerde geometrische bewerkingen kunnen worden uitgevoerd.

SQL : Structured Query Language. SQL is de taal waarin de communicatie opdrachten met het relationele database systeem (ORACLE) worden geschreven.

string : Een serie aaneengesloten karakters; bij voorbeeld : "Dit is een string" of "configuratie".

topologische gegevens : Dit zijn kwalitatieve gegevens die bestaan uit de relaties tussen zijvlakken, zijden en hoekpunten (deze worden weergegeven m.b.v. pointers), zie 2.2.

UFI : User Friendly Interface; dit is een hulpmiddel voor de gebruiker, om rechtstreeks met de ORACLE database te communiceren.

VAX : Virtual Address eXtension; computernaam waarin gelijk de

"specialiteit" van de computer wordt aangegeven, n.l. de virtuele adressering. De naam wordt vaak gevolgd door een type aanduiding; bij voorbeeld VAX 11/780.

VAX/VMS : VAX Virtual Memory System; de naam van een operating system voor de VAX.

veldpunten : Dit zijn punten die niet het oppervlak van het object (bijvoorbeeld een vliegtuig) maar de ruimte eromheen beschrijven; zie 3.3 en B8 in appendix B.

winged edge : Dit is een methode van grensbeschrijving waarbij een zijvlak wordt begrensd door een aantal zijden en in een hoekpunt een aantal zijden bij elkaar kunnen komen. Door opslag van redundante topologische gegevens in de vorm van pointers is een snelle toegang tot alle zijvlakken, zijden en hoekpunten op allerlei manieren mogelijk.

Appendix E

De PROGRAM MODES van VISUAL

INHOUD

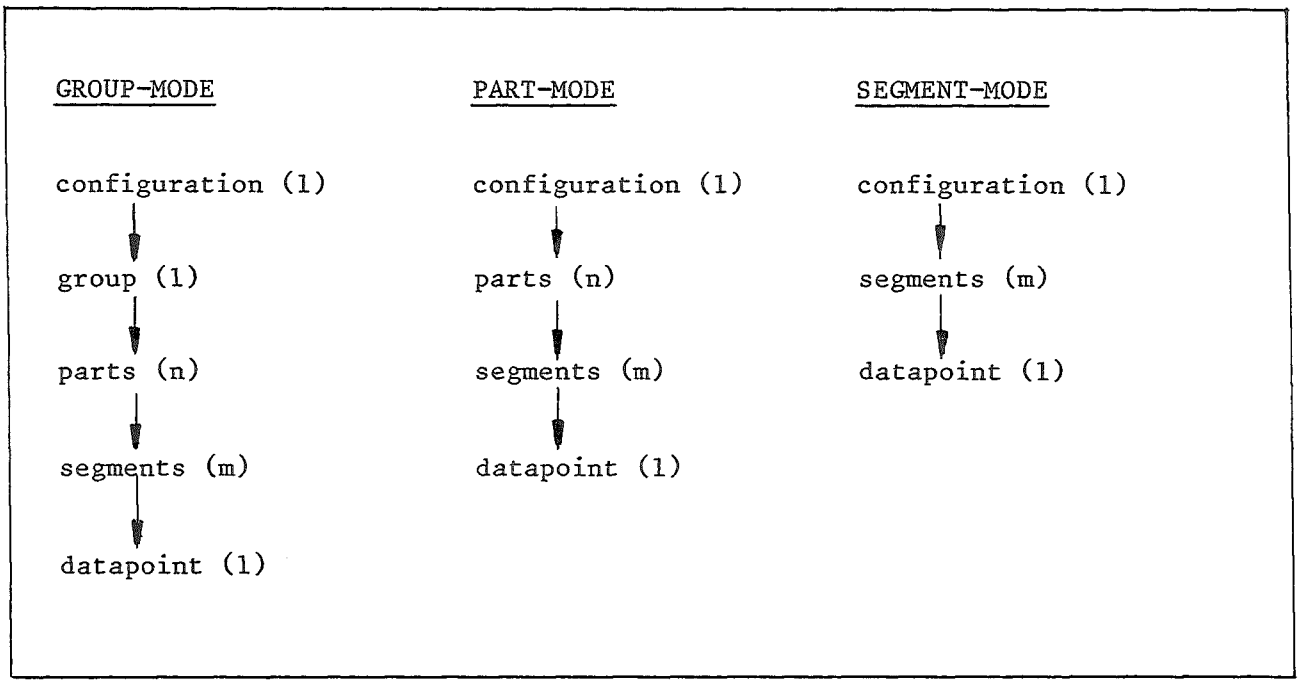
E1. Opsomming van de PROGRAM MODES 141
E2. Een overzicht van de default waarden van de PROGRAM MODES 144

E1. Opsomming van de PROGRAM MODES

De volgende variabelen die de PROGRAM MODE karakteriseren zijn aanwezig; tussen haakjes is het type van de variabele aangegeven :

- TERMINAL-TYPE (string): Geeft het type van de terminal aan waar de gebruiker op het ogenblik mee werkt of zegt dat hij mee werkt. De mogelijke waarden zijn : T4014, T4114, T4115 en VT100.

- SELECTION-MODE (string): Geeft aan op welke manier de gegevens geselecteerd moeten worden. Mogelijke waarden zijn : group, part en segment.
  In figuur E.1 zijn de diverse selectie modes schematisch weergegeven. Tussen haakjes staat vermeld hoeveel objecten van dit type er geselecteerd moeten worden. Hierbij staan n en m voor een willekeurig aantal.



Figuur E.1 - Overzicht selection modes

- TRANSFORMATION-MODE (string): Geeft aan op welke wijze de transformaties (roteren en transleren) worden uitgevoerd. De mogelijke waarden zijn : picture, space en configuration.

- NUMBER-OF-COLOURS (geheel getal): Geeft het aantal kleuren aan dat voor een bepaalde functie (zoals SHADE, ISO-BAND, CARPET, etc.) wordt gebruikt. Op dit ogenblik zijn het er maximaal 16; in de nabije toekomst zullen dit er 256 worden.

- DEFAULT-MATRIX (geheel getal): Geeft aan welke matrix (behorende bij een bepaald aanzicht) wordt gebruikt, nadat de gebruiker een CLEAR opdracht heeft gegeven, waarbij de oude matrix niet meer bewaard moet blijven.

- SCREEN-SCALING (logical): Als deze variabele de waarde "true" heeft, dan wordt de afbeelding op het scherm bij een tekenopdracht automatisch geschaald. Als deze variabele de waarde "false" heeft, dan zal (indien nodig) de gebruiker zelf moeten schalen.

- COMMAND-DRIVEN (logical): Als deze variabele de waarde "true" heeft, dan wordt het systeem met behulp van de commandotaal COLAS bestuurd (zie 2.6). Heeft deze variabele de waarde "false", dan wordt het systeem door middel van menu's bestuurd (dit wordt voorlopig nog niet geïmplementeerd).

- AUTO-DRAW (logical): Na een transformatie (d.w.z. rotatie, translatie of schaling) wordt automatisch het beeld opnieuw op het scherm getekend indien deze variabele de waarde "true" heeft. Als AUTO-DRAW "false" is, dan moet de gebruiker zelf het commando DRAW geven na een transformatie om het resultaat te zien.

- BOUNDARY-CORRECTION (logical): Bij de functies ISO-LINE en ISO-BAND (zie 2.4) wordt geen informatie uit aangrenzende segmenten gebruikt als deze variabele "false" is. Dit heeft tot gevolg dat de isobaren sneller uitgerekend kunnen worden, maar dat deze niet continu verlopen over segmentranden. Indien er wel informatie (drukwaarden) uit aangrenzende segmenten wordt gebruikt, zullen de isobaren meer continu over de segmentranden verlopen.

- GRAPHICAL-SELECTION (logical): Indien "true", dan wordt bij het selecteren van parts, segmenten (etc.) zoveel mogelijk gebruik gemaakt van de kruisdraden. Overigens kan niet alles met de kruisdraden geselecteerd worden (b.v. een configuratie).
  Als de gebruiker bekend is met een bepaalde configuratie, dan kan efficiënter werken door niet met de cross-wire segment voor segment te selecteren, maar door een range van segmentnummers op te geven (na deze

eerst met een directory functie te hebben opgevraagd). In dat geval moet deze variabele de waarde "false" hebben. GRAPHICAL-SELECTION is nog niet geïmplementeerd.

- EXTENDED-PANEL (logical): Indien deze de waarde "true" heeft wordt een paneel ingekleurd, zodanig dat gelijktijdig ook de rand van het paneel gekleurd wordt. Heeft deze variabele de waarde "false" dan wordt alleen het "inwendige" van een paneel ingekleurd.

- COUPLED-FUNCTIONS (logical): Indien deze de waarde "true" heeft worden paren van functies (zoals ISO-LINE en ISO-BAND) in één keer berekend. Anders worden deze functies één voor één berekend.

- INTRODUCTION (logical): Indien deze de waarde "true" heeft dan wordt, nadat de gebruiker heeft opgegeven met welk type terminal hij werkt, een introductiepagina getoond. Anders gebeurt dit niet.

- CLEAR-AUTOMATIC (logical): Na een transformatie (d.w.z. rotatie, translatie of schaling) wordt, indien deze variabele de waarde "true" heeft, automatisch het beeld schoongemaakt voordat er opnieuw op het scherm getekend wordt. Als CLEAR-AUTOMATIC "false" is, dan moet de gebruiker zelf het commando CLEAR geven om ervoor te zorgen dat het scherm wordt schoongemaakt.

E2. Een overzicht van de default waarden van de PROGRAM MODES

PROGRAM MODE	Default waarde
SELECTION-MODE	GROUP
TRANSFORMATION-MODE	CONFIGURATION
NUMBER-OF-COLOURS	12
DEFAULT-MATRIX	1
SCREEN-SCALING	true
COMMAND-DRIVEN	true
AUTO-DRAW	true
BOUNDARY-CORRECTION	false
GRAPHICAL-SELECTION	false
EXTENDED-PANEL	false
COUPLED-FUNCTIONS	false
INTRODUCTION	true
CLEAR-AUTOMATIC	false

Appendix F

Overzicht van files voor VISUAL

INHOUD

F1. De files met source code 146
F2. De files voor en van COLAS 149
F3. De overige files 150

F1. De files met source code

ALGMAT.FOR : Bevat diverse algemene mathematische routines, bijvoorbeeld improdukt, uitprodukt, matrixvermenigvuldiging, etc.

ALGTEK.FOR : Bevat diverse algemene grafische routines die in GINO-F ontbreken; bijvoorbeeld een routine om de T4114 om te schakelen van refresh naar storage mode.

BAND.FOR : Bevat de routines voor de functie ISO-BAND in het deelsysteem 3D-FUNCTIE.

BUFFER.FOR : Bevat routines voor de communicatie met het dynamische geheugen POOL systeem (via omvangrijke buffers).

BUREN.FOR : Bevat de routines die de buren uitrekent en opslaat in de datastructuur indien de PROGRAM MODE BOUNDARY-CORRECTION "true" is.

DIRSEL.FOR : Bevat de routines uit het deelsysteem DIRSEL; bijvoorbeeld DIRECTORY, SELECT, ADD, DROP, etc.

HULP3D.FOR : Bevat diverse hulproutines voor het deelsysteem 3D-FUNCTIE; bijvoorbeeld voor het tekenen van het assenstelsel.

INTRO.FOR : Bevat de routines die de introductie van de gebruiker verzorgen; deze routines zorgen ervoor dat op het beeldscherm de tekst "You are welcome to Vano's VISUAL" komt te staan.

ISOBAR.FOR : Bevat de routines welke nodig zijn om de isobaren uit te rekenen en te tekenen (de functie ISO-LINE).

LIST.FOR : Bevat de routines voor de functie LISTING uit het deelsysteem DIRSEL.

MAINRO.FOR : Bevat routines die niet tot één van de deelsystemen behoren; bijvoorbeeld routines voor de initialisatie van VISUAL.

ONED.FOR : Bevat de routines voor het deelsysteem 1D-FUNCTIE. Deze file is nauwelijks gevuld omdat het deelsysteem 1D-FUNCTIE nog niet geïmplementeerd is.

PARSER.FOR : Bevat aanvullende routines (voor COLAS) welke een invoerstring kunnen analyseren (parsen).

POOL.FOR : Bevat de routines van het dynamische geheugen POOL systeem.

SETUP.FOR : Bevat de routines van het deelsysteem SETUP; bijvoorbeeld CHANGE-VAR, INFO-VAR, MODE, STORE, etc.

THREED.FOR : Bevat de routines van het deelsysteem 3D-FUNCTIE; bijvoorbeeld WIRE, HIDDEN, OUTLINE, VELOCITY, etc.

TRANSF.FOR : Bevat routines die de transformaties (dat wil zeggen de bewerkingen op de transformatie matrix) uitvoeren.

TWOD.FOR : Bevat de routines voor het deelsysteem 2D-FUNCTIE. Deze file is nauwelijks gevuld omdat het deelsysteem 2D-FUNCTIE nog niet geïmplementeerd is.

VISUAL.FOR : Bevat het hoofdprogramma van VISUAL.

CMDATA.INC : Bevat de common-blokken, welke de datastructuur beschrijven; zie appendix G.

CMDEFL.INC : Bevat de common-blokken, waarin de PROGRAM MODES worden opgeslagen; zie appendix E.

CMDIVS.INC : Bevat de common-blokken, waarin diverse globale variabelen (bijvoorbeeld de FASE) zijn opgeslagen.

CMDRAW.INC : Bevat de common-blokken, waarin diverse gegevens voor het tekenen zijn opgeslagen; bijvoorbeeld de afmetingen van de beeldschermen.

CMGINO.INC : Bevat de common-blokken, waarin de buffers voor GINO-F staan.

CMORAC.INC : Bevat de common-blokken, welke nodig zijn voor de programma interface met ORACLE.

CMPOOL.INC : Bevat de common-blokken met de geheugen POOL en andere variabelen welke nodig zijn voor het dynamische geheugen POOL systeem.

F2. De files voor en van COLAS

BCLIF.DAT : Dit is een file die gemaakt wordt door het programma CDM (zie appendix D) en die nodig is voor de diverse COLAS routines.

COLAS.COM : Commando file welke ervoor zorgt dat eerst de juiste logische namen aan een aantal files (onder andere VISCOL.DAT) wordt gegeven alvorens het programma CDM wordt geëxecuteerd.

FORO??.DAT : Files die om onbekende redenen door COLAS geproduceerd worden. Op de plaats van ?? kan staan 00, 02, 80, 81, 82 of 88.

LOGCOL.DAT : Een logfile waarin een verslag staat van de executie van het programma CDM.

VISCOL.DAT : Deze file bevat de beschrijving van commando's en helpinformatie voor VISUAL en dient als invoer voor het programma CDM.

VISUAL.LOG : File welke een verslag van de laatste sessie met VISUAL bevat.

F3. De overige files

Voor elke ".FOR"-file (zie F1) is er een ".OBJ"-file welke de gecompileerde versie van de betreffende file bevat.

T4014.COM : Commando-file waarmee een sessie met VISUAL op een T4014 kan worden nagespeeld.

T4114.COM : Commando-file waarmee een sessie met VISUAL op een T4114 kan worden nagespeeld.

T4115.COM : Commando-file waarmee een sessie met VISUAL op een T4115 kan worden nagespeeld.

VT100.COM : Commando-file waarmee een sessie met VISUAL op een VT100 kan worden nagespeeld.

VISUAL.EXE : Bevat de executable code van VISUAL.

VISUAL.OLB : Library waarin alle routines van VISUAL zitten.

VISUAL.COM : Commando file welke gebruikt kan worden bij het opnieuw linken van VISUAL (type : "LINK@VISUAL").

PROGMD.DAT : File met specifieke waarden van de PROGRAM MODES en standaardaanzichten van een bepaalde gebruiker.

Appendix G

Implementatie van de datastructuur

INHOUD

G1. De Fortran code van de declaratie van variabelen 152
G2. Het common-blok CONF 153
G3. Het common-blok GROUP 153
G4. Het common-blok PART 153
G5. Het common-blok SEGM 154
G6. Het common-blok AERO 154
G7. Het common-blok CDATA 155
G8. Het common-blok GDATA 155
G9. Het common-blok PDATA 156
G10 Het common-blok SDATA 157

G1. De Fortran code van de declaratie van variabelen

```
INTEGER MAXPA,MAXSE,MAXISO
PARAMETER(MAXPA=50,MAXSE=100,MAXISO=50)

INTEGER CNFNR,CNFVN,GRPNR,GRPVN,GRPREL,PRTNR,PRTVN,PRTREL
INTEGER SEGNR,SEGVN,SEGREL,AANTPA,AANTSE
LOGICAL CNFOPG,GRPOPG,PRTOPG,AANTSE
CHARACTER CNFID*20,CNFDS*240,CNFCR*20,CNFDT*20,CNFCRU*20
CHARACTER CNFDTU*20
REAL CNFAX,CNFMA,CNFMIN,CNFMAX
INTEGER CNFPR
CHARACTER GRPID*20,GRPDS*240,GRPCR*20,GRPDT*20,GRPCRU*20
CHARACTER GRPDTU*20
REAL GRPAX,GRPMA
INTEGER GRPPR
CHARACTER PRTID*20,PRTDS*240,PRTCR*20,PRTDT*20,PRTCRU*20
CHARACTER PRTDTU*20
REAL PRTAX,PRTMA
INTEGER PRTPR,PRTSMX,SNRPT,SVNPT,SAANWPT
INTEGER NC,NP,LM,PUNTPT,NORMPT,CPPT,VPT,BUURPT,SORT
LOGICAL VELDPNT,CPAANW,VAANW
REAL MINPNT,MAXPNT,MINCP,MAXCP,MINV,MAXV
INTEGER DAPNR,DAPVN
REAL MACH,ALFA,BETA,REYN,FLAP,CPMIN,CPMAX,VMIN,VMAX
INTEGER ISOAANT,ISOPNT,ISOLPT,ISOSPT,ISOFPT,ISOSORT
INTEGER ACTAANT,ACTBAANT
REAL ISOCP,ACTISO,ACTBND
```

G2. Het common-blok CONF

```
COMMON/CONF/CNFNR,CNFVN,CNFOPG
```

CNFNR : CoNFiguratie NummeR
CNFVN : CoNFiguratie Versie Nummer
CNFOPG : CoNFiguratie OPGehaald

G3. Het common-blok GROUP

```
COMMON/GROUP/GRPNR,GRPVN,GRPREL,GRPOPG
```

GRPNR : GRouP NummeR
GRPVN : GRouP Versie Nummer
GRPREL : RELatief GRouP nummer (d.w.z. nummer binnen configuratie)
GRPOPG : GRouP OPGehaald

G4. Het common-blok PART

```
COMMON/PART/AANTPA,PRTNR(MAXPA),PRTVN(MAXPA),
1       PRTREL(MAXPA),PRTOPG(MAXPA)
```

AANTPA : AANTal PArts (dit is kleiner of gelijk aan MAXPA)
PRTNR : PaRT NummeR
PRTVN : PaRT Versie Nummer
PRTREL : RELatief PaRT Nummer (d.w.z. nummer binnen configuratie)
PRTOPG : PaRT OPGehaald

G5. Het common-blok SEGM

```
COMMON/SEGM/AANTSE,SEGNR(MAXSE),SEGVN(MAXSE),
1       SEGREL(MAXSE),SEGOPG(MAXSE)
```

AANTSE : AANTal SEgmenten (dit is kleiner of gelijk aan MAXSE)
SEGNR : SEGment NummeR
SEGVN : SEGment Versie Nummer
SEGREL : RELatief SEGment nummer (d.w.z. nummer binnen configuratie)
SEGOPG : SEGment OPGehaald

G6. Het common-blok AERO

```
COMMON/AERO/DAPNR,DAPVN,MACH,ALFA,BETA,REYN,
1       FLAP,CPMIN,CPMAX,VMIN(3),VMAX(3)
```

DAPNR : DAtaPoint NummeR
DAPVN : DAtaPoint Versie Nummer
MACH : MACH getal
ALFA : invalshoek ALFA
BETA : sliphoek BETA
REYN : REYNolds getal
FLAP : klepstand FLAP
CPMIN : CP MINimaal (over alle segmenten)
CPMAX : CP MAXimaal (over alle segmenten)
VMIN : Velocity MINimaal (over alle segmenten)
VMAX : Velocity MAXimaal (over alle segmenten)

G7. Het common-blok CDATA

```
COMMON/CDATA/CNFID,CNFDS,CNFAX(4,3),CNFMA(4,4),CNFCR,
1       CNFDT,CNFCRU,CNFDTU,CNFPR,CNFMIN(3),CNFMAX(3)
```

```
CNFID :       CoNFiguratie IDentifier
CNFDS :       CoNFiguratie DeScription
CNFAX :       CoNFiguratie AXes
CNFMA :       CoNFiguratie MAtrix
CNFCR :       CoNFiguratie CReator
CNFDT :       CoNFiguratie creation DaTe
CNFCRU :      CoNFiguratie CReator Update
CNFDTU :      CoNFiguratie UpDaTe
CNFPR :       CoNFiguratie PRotectie
CNFMIN :      CoNFiguratie MINimum punt
CNFMAX :      CoNFiguratie MAXimum punt
```

G8. Het common-blok GDATA

```
COMMON/GDATA/GRPID,GRPDS,GRPAX(4,3),GRPMA(4,4),GRPCR,
1       GRPDT,GRPCRU,GRPDTU,GRPPR
```

```
GRPID :       GRouP IDentifier
GRPDS :       GRouP DeScription
GRPAX :       GRouP AXes
GRPMA :       GRouP MAtrix
GRPCR :       GRouP CReator
GRPDT :       GRouP creation DaTe
GRPCRU :      GRouP CReator Update
GRPDTU :      GRouP UpDaTe
GRPPR :       GRouP PRotectie
```

G9. Het common-blok PDATA

```
COMMON/PDATA/PRTID(MAXPA),PRTDS(MAXPA),PRTAX(MAXPA,4,3),
1       PRTMA(MAXPA,4,4),PRTCR(MAXPA),PRTDT(MAXPA),
1       PRTCRU(MAXPA),PRTDTU(MAXPA),PRTPR(MAXPA),
1       PRTSMX(MAXPA),SNRPT(MAXPA),SVNPT(MAXPA),SAANWPT(MAXPA)
```

PRTID :	PaRT IDentifier
PRTDS :	PaRT DeScription
PRTAX :	PaRT AXes
PRTMA :	PaRT MAtrix
PRTCR :	PaRT CReator
PRTDT :	PaRT creation DaTe
PRTCRU :	PaRT CReator Update
PRTDTU :	PaRT UpDaTe
PRTPR :	PaRT PRotectie
PRTSMX :	MaXimum aantal Segmenten in dit PaRT
SNRPT :	Segment NummeRs PoinTer
SVNPT :	Segment Versie Nummers PoinTer
SAANWPT :	Segment AANWezig PoinTer

G10 Het common-blok SDATA

```
COMMON/SDATA/NC(MAXSE),NP(MAXSE),LM(MAXSE),PUNTPT(MAXSE),
1       NORMPT(MAXSE),VELDPNT(MAXSE),
1       BUURPT(MAXSE),MINPNT(MAXSE),MAXPNT(MAXSE),SORT(MAXSE),
1       CPPT(MAXSE),CPAANWPT(MAXSE),MINCP(MAXSE),MAXCP(MAXSE),
1       VPT(MAXSE),VAANWPT(MAXSE),VMIN(MAXSE,3),VMAX(MAXSE,3),
1       ISOAANT,ISOCP(MAXISO),ISOSORT(MAXISO),ISOPNT(MAXSE,MAXISO),
1       ISOLPT(MAXSE,MAXISO),ISOSPT(MAXSE,MAXISO),ISOFPT(MAXSE,MAXISO),
1       ACTAANT,ACTISO(MAXISO),ACTBAANT,ACTBND(MAXISO)
```

NC :	Number of Contours
NP :	Number of Points (per contour)
LM :	Line Mode
PUNTPT :	PUNTen (d.w.z. de coördinaten) PoinTer
NORMPT :	NORMaal-vectoren PoinTer
VELDPNT :	VELDPuNTen segment
BUURPT :	BUUR punten PoinTer
MINPNT :	MINimum PuNT (in segment)
MAXPNT :	MAXimum PuNT (in segment)
SORT :	geSORTeerde segmenten array
CPPT :	CP-waarden PoinTer
CPAANW :	CP-waarden AANWezig
MINCP :	MINimum CP-waarde in segment
MAXCP :	MAXimum CP-waarde in segment
VPT :	Velocity-waarden PoinTer
VAANW :	Velocity-waarden AANWezig
MINV :	MINimum Velocity-waarde in segment
MAXV :	MAXimum Velocity-waarde in segment
ISOAANT :	AANTal berekende ISObaren (dit is kleiner of gelijk aan MAXISO)
ISOCP :	CP-waarden van berekende ISObaren
ISOSORT :	geSORTeerde ISObaren array
ISOPNT :	aantal PuNTen in deze ISObaar
ISOLPT :	PoinTer naar Lijnnummers van deze ISObaar
ISOSPT :	PoinTer naar Stuknummers van deze ISObaar
ISOFPT :	Pointer naar Fracties van deze ISObaar
ACTAANT :	ACTuele AANTal isobaren in de functie "iso-line"
ACTISO :	cp-waarden van de ACTuele ISObaren
ACTBAANT :	ACTuele AANTal isobaren in de functie "iso-Band"
ACTBND :	cp-waarden van de ACTuele isobaren voor functie "iso-BaND"

Appendix H

Ontwerp van de database

INHOUD

H1. Object tabellen 159
H2. Relatie tabellen 159
H3. Quantity tabellen 159
H4. Verklaring van gebruikte afkortingen 160

H1. Object tabellen

CONFIG(CONFIG-NR, CONFIG-ID, CONFIG-VN, CONFIG-DS, CONFIG-AX, CONFIG-MA, CONFIG-CR,
CONFIG-DT, CONFIG-CRU, CONFIG-DTU, CONFIG-PR)
CONFIG(GROEP-NR, GROEP-ID, GROEP-VN, GROEP-DS, GROEP-AX, GROEP-MA, GROEP-CR,
GROEP-DT, GROEP-CRU, GROEP-DTU, GROEP-PR)
CONFIG(PART-NR, PART-ID, PART-VN, PART-DS, PART-AX, PART-MA, PART-CR,
PART-DT, PART-CRU, PART-DTU, PART-PR)
CONFIG(SEGMENT-NR, SEGMENT-ID, SEGMENT-VN, SEGMENT-DS, SEGMENT-AX, SEGMENT-MA,
SEGMENT-CR, SEGMENT-DT, SEGMENT-CRU, SEGMENT-DTU, SEGMENT-PR)

H2. Relatie tabellen

CONFGROUP(CONFIG-NR, CONFIG-VN, GROEP-NR, GROEP-VN, CONFGROUP-SN, CONFGROUP-CR,
CONFGROUP-DT, CONFGROUP-CRU, CONFGROUP-DTU)
CONFPART(CONFIG-NR, CONFIG-VN, PART-NR, PART-VN, CONFPART-SN, CONFPART-CR,
CONFPART-DT, CONFPART-CRU, CONFPART-DTU)
CONFSEGM(CONFIG-NR, CONFIG-VN, SEGMENT-NR, SEGMENT-VN, CONFSEGM-SN, CONFSEGM-CR,
CONFSEGM-DT, CONFSEGM-CRU, CONFSEGM-DTU)
GROUPPART(GROUP-NR, GROUP-VN, PART-NR, PART-VN, GROUPPART-SN, GROUPPART-CR,
GROUPPART-DT, GROUPPART-CRU, GROUPPART-DTU)
GROUPSEGM(GROUP-NR, GROUP-VN, SEGMENT-NR, SEGMENT-VN, GROUPSEGM-SN, GROUPSEGM-CR,
GROUPSEGM-DT, GROUPSEGM-CRU, GROUPSEGM-DTU)
PARTSEGM(PART-NR, PART-VN, SEGMENT-NR, SEGMENT-VN, PARTSEGM-SN, PARTSEGM-CR,
PARTSEGM-DT, PARTSEGM-CRU, PARTSEGM-DTU)

H3. Quantity tabellen

SEGMGEOM(SEGMENT-NR, SEGMENT-VN, SEGMGEOM-CR, SEGMGEOM-DT, SEGMGEOM-CRU,
SEGMGEOM-DTU, SEGMGEOM-VA, SEGMGEOM-NC, SEGMGEOM-NP, SEGMGEOM-LM,
SEGMGEOM-EX, SEGMGEOM-XYZ)
DATAPOINT(DATAPOINT-NR, DATAPOINT-VN, DATAPOINT-DESCR, DATAPOINT-PROGRAM,
DATAPOINT-DT, DATAPOINT-MACH, DATAPOINT-ALFA, DATAPOINT-BETA,
DATAPOINT-REYN, DATAPOINT-FLAP, DATAPOINT-VAL1, DATAPOINT-VAL2,
DATAPOINT-VAL3, DATAPOINT-VAL4, DATAPOINT-VAL5, DATAPOINT-VAL6,
DATAPOINT-VAL7)
AERODATA(DATAPOINT-NR, DATAPOINT-VN, AERODATA-OBJECT, OBJECT-NR, OBJECT-VN,
AERODATA-TYPE, AERODATA-NVAL, AERODATA-EXTR, AERODATA-VALUES)

H4. Verklaring van gebruikte afkortingen

CONFIG :	CONFIGuratie
CONF :	CONFiguratie
SEGM :	SEGMent
GEOM :	GEOMetrie
AERO :	AERODynamisch

NR :	NummeR
ID :	IDentificatie
VN :	Versie Nummer
DS :	DeScription
DESCR :	DESCRiption
AX :	AXis
MA :	MAtrix
CR :	CReator
DT :	DaTe
CRU :	CReator Update
DTU :	UpDaTe
PR :	PRotectie
SN :	Serie Nummer
VA :	VAlue
VAL :	VALue
NC :	Number of Contours
NP :	Number of Points
LM :	Line Mode
EX :	EXtreme waarden
EXTR :	EXTReme waarden
NVAL :	Number of VALues