Evolutionary computing in an integrated multi-objective optimisation environment in Matlab for aeronautic design

Mette Nolte

Master thesis
April 2006

Evolutionary computing in an integrated multi-objective optimisation environment in Matlab for aeronautic design

Mette Nolte

Master thesis

April 2006

Nationaal Lucht- en Ruimtevaartlaboratorium

Divisie Aerospace Vehicles

Afdeling Aerospace Vehicles Collaborative Engineering Systems

Anthony Fokkerweg 2

1059 CM Amsterdam

Vrije Universiteit

Faculteit der Exacte Wetenschappen

Divisie Wiskunde en Informatica

Studierichting Bedrijfswiskunde en Informatica

De Boelelaan 1081a

1081 HV Amsterdam

Voorwoord

Aan het einde van de studie BedrijfsWiskunde en Informatica dient iedere student een stage te vervullen. Mijn stage heeft plaatsgevonden op het Nationaal Lucht- en Ruimtevaartlaboratorium bij de afdeling Aerospace Vehicles Collaborative Engineering Systems (AVCE) te Amsterdam. Tijdens deze stage heb ik onderzoek gedaan naar multi-objective optimalisatie en tevens een multi-objective optimalisatie tool ontwikkeld, welke in gebruik genomen is voor een winglet ontwerp optimalisatie studie.

Dit was een zeer leerzame stage voor mij, maar alleen zou ik het niet gered hebben. Daarom wil ik graag een aantal personen bedanken. Jos Vankan voor zijn begeleiding op het NLR, hij heeft altijd veel tijd uitgetrokken om de vorderingen tijdens mijn stage te bespreken en mij geholpen wanneer ik zijn hulp nodig had. Elena Marchiori als eerste begeleidster van de VU, zij heeft ertoe bijgedragen dat mijn stageonderzoek ook daadwerkelijk naar het doel leidde dat in eerste instantie voor ogen was. André Ran als tweede begeleider van de VU, voor het doornemen van de vele documenten die deze stage heeft opgeleverd. Iedereen van de afdeling AVCE voor het altijd bereid zijn om te helpen en voor de gezellige gesprekken tijdens de lunch. Verder wil ik iedereen bedanken die hierboven niet genoemd zijn, maar er wel aan hebben bijgedragen dat ik met goed gevoel deze stage heb doorlopen.

Mette Nolte

April 2006

Samenvatting

Bij een ontwerp van complexe producten, zoals vliegtuigen of vliegtuigonderdelen, worden meestal meerdere doelstellingen en beperkingen tegelijkertijd beschouwd. Dergelijke ontwerpvraagstukken kunnen geformuleerd worden als optimalisatieprobleem met meerdere doelfuncties en bijvoorwaarden, ofwel objectives en constraints. Voor dergelijke multi-objective optimalisatieproblemen bestaat er niet één enkel optimum, maar een set van zogenaamde Pareto optimale punten (Pareto front). Voor dit type optimalisatieproblemen zijn tal van specifieke oplosalgoritmen ontwikkeld, waarbij met name de evolutionary computing algoritmen goed blijken te werken.

Het Nondominated Sorting Genetic Algorithm 2 (NSGA2) is een uitermate efficiënt evolutionary computing algoritme. NSGA2 kenmerkt zich vooral, ten opzichte van andere multi-objective evolutionary computing algoritmen, door Pareto optimale punten te vinden waarvan, behalve de kwaliteit (convergentie), ook de spreiding van die punten over het Pareto front goed is.

Vanwege de goede resultaten die NSGA2 behaalt, is dit algoritme geïmplementeerd in een Matlab programma. Dit programma is toegepast in een winglet ontwerpstudie waarin, met negen vrije ontwerpparameters, drie ontwerpdoelstellingen zijn geoptimaliseerd. De gevonden Pareto front punten leveren een zinvolle selectie van de meest interessante ontwerppunten binnen deze hoog-dimensionale ontwerpruimte. Ten opzichte van de beginsituatie zijn verbeteringen in de ontwerpdoelstellingen bereikt van globaal ongeveer 5%.

Inhoud

1. Inleiding

1.1 Optimalisatie problemen

In veel vakgebieden bestaan er vraagstukken die als optimalisatieproblemen geformuleerd kunnen worden, deze problemen kunnen dan met behulp van optimalisatietechnieken opgelost worden. De beslissing die gemaakt zal worden met betrekking tot een dergelijk vraagstuk zal worden gebaseerd op de drie voornaamste variabelen in het optimalisatieprobleem: de beslissingsvariabelen, de doelfuncties en de bijvoorwaarden [22]. Deze variabelen voor een optimalisatieprobleem kunnen zowel scalair (één variabele) als vectorwaardig (meer dan één variabele) zijn. De beslissingsvariabelen en de bijvoorwaarden zijn in praktische optimalisatieproblemen vrijwel altijd vectorwaardig. De doelstelling blijkt daarentegen, meestal doordat de formulering van het optimalisatieprobleem intuïtief daartoe leidt, vaak scalair uitgedrukt te worden, dat wil zeggen dat de meeste vraagstukken gebruik maken van een single-objective optimalisatie. Echter kan het bij de formulering van het optimalisatieprobleem uiterst zinvol zijn de doelfunctie vectorwaardig te definiëren, een dergelijke formulering staat ook bekend als een multi-objective optimalisatie probleem. Multi-objective optimalisatie vereist een speciale behandeling waarbij specifieke multi-objective optimalisatietechnieken gebruikt kunnen worden [2].

1.2 Omschrijving van de stageopdracht

De laatste jaren is er veel onderzoek gedaan naar multi-objective optimalisatie algoritmen, met name naar de algoritmen die tot de evolutionary computing behoren. Deze algoritmen zouden kunnen bijdragen aan het optimaliseren van de productieontwerpen van vliegtuigen (of onderdelen daarvan) waar het NLR zich onder andere mee bezig houdt.
Om te onderzoeken of er inderdaad een tot de evolutionary computing behorend algoritme bestaat dat kan bijdragen aan de optimalisatie ontwerp problemen van het NLR, is er op basis van het Nondominated Sorting Genetic Algorithm 2 (NSGA2) [15] een matlab omgeving gebouwd welke toegepast zal worden op een multi-objective ontwerp probleem.

1.3 Indeling

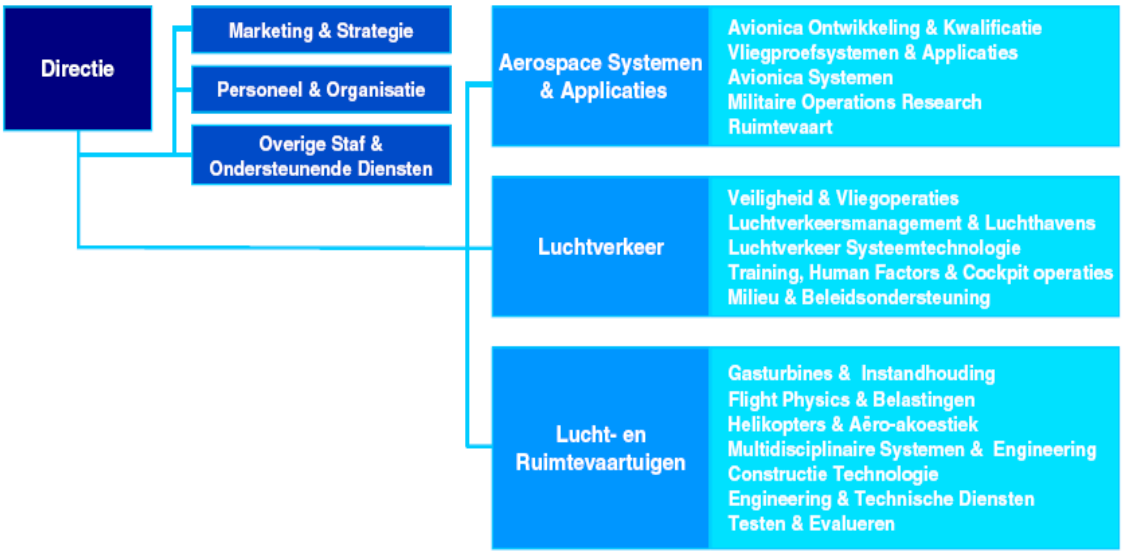
Dit afstudeerverslag is verder opgebouwd uit vijf hoofdstukken, waarvan het volgende hoofdstuk een beschrijving van het NLR bevat met speciale aandacht voor de divisie AV en afdeling AVCE waar de stage plaats gevonden heeft.
In hoofdstuk 3 wordt dieper ingegaan op multi-objective optimalisatie en op de evolutionary computing, waarbij uitgebreid stil zal worden gestaan bij NSGA2.
Behalve verschillende algoritmen die speciaal voor multi-objective optimalisatie zijn ontwikkeld, bestaan er ook al enige tools voor dit type optimalisatieproblemen, twee daarvan zullen in hoofdstuk 4 besproken worden op basis van een testfunctie.
De implementatie van NSGA2 (MNSGA) wordt in hoofdstuk 5 besproken, waarin ook een aantal testresultaten van MNSGA gegeven wordt.
Hoofdstuk 6 bevat de beschrijving en (de door MNSGA) gevonden oplossingen van een winglet ontwerp optimalisatie probleem.
Dit verslag zal tot slot worden afgesloten met een conclusie die gebaseerd zal zijn op gevonden resultaten zoals deze in de loop van het verslag gegeven zullen worden.

2. Het Nationaal lucht- en ruimtevaartlaboratorium

Het Nationaal Lucht- en Ruimtevaartlaboratorium (NLR) is opgericht in 1919 en is sinds 1937 een zelfstandige non-profit organisatie [1]. Het NLR verricht toegepast onderzoek voor de lucht- en ruimtevaartsector, zowel nationaal als internationaal, waarbij het niet alleen om wetenschappelijk onderzoek draait, maar ook om de toepassing in de industrie en bij de overheid. Tegenwoordig telt het NLR circa 700 medewerkers die verspreid zijn over de twee vestigingen; Amsterdam en de Noordoostpolder.

2.1 Aerospace Vehicles Division

Het NLR is onderverdeeld in drie divisies zoals in figuur 2.1 afgebeeld is.



Figuur 2.1: organisatie structuur NLR

De derde genoemde divisie in figuur 2.1 is de Lucht- en Ruimtevaartuigen (Aerospace Vehicles Division (AV)). Bij de divisie AV worden er onderzoeken verricht voor het verbeteren van de huidige lucht- en ruimtevaarttuigen. Deze divisie valt weer onder te verdelen in verschillende afdelingen waaronder de afdeling Aerospace Vehicles Collaborative Engineering Systems(AVCE).

2.2 Aerospace Vehicles Collaborative Engineering Systems

De afdeling AVCE houdt zich bezig met het leveren van diensten, het uitvoeren van onderzoek en het ontwikkelen van producten en totaaloplossingen op het gebied van collaborative engineering, multidisciplinair ontwerpen, ‘structural mechanics’ en simulatietechnologie ten behoeve van de Nederlandse overheid, het Nederlandse bedrijfsleven, Nederlandse Overheidsinstellingen en andere NLR afdelingen vanuit een lucht- en ruimtevaartexpertise. De opdrachten die worden uitgevoerd door deze afdeling vinden niet alleen op nationale bodem plaats, maar ook in het buitenland onder andere door een bestaande internationale samenwerking.

3. Multi-objective optimalisatie en evolutionary computing

3.1 Multi-objective optimalisatie

Multi-objective optimalisatie is een belangrijk gebied van onderzoek omdat veel problemen een multi-objective achtergrond bezitten en omdat er nog veel open vragen op dit gebied zijn. Om een duidelijk beeld van een multi-objective optimalisatie probleem te kunnen schetsen, zal er in deze paragraaf eerst gekeken worden naar beslissingsvariabelen en bijvoorwaarden zoals deze kunnen voorkomen in zowel single-objective optimalisatie als multi-objective optimalisatie. Vervolgens zal het principe van multi-objective optimalisatie en het Pareto front, een veel voorkomend begrip binnen de multi-objective optimalisatie technieken, uitgelegd worden.

3.1.1 Beslissingsvariabelen en bijvoorwaarden

Wanneer een optimalisatieprobleem opgelost dient te worden, zullen die beslissingsvariabelen gezocht moeten worden waarmee de optimale oplossing kan worden bereikt voor het probleem. De beslissingsvariabelen mogen alleen waarden aannemen die in een voor hen specifiek interval gedefinieerd zijn, deze intervallen mogen wel per beslissingsvariabele verschillend zijn. Beslissingsvariabelen worden genoteerd als x_i, met i = 1,2,...,n en n het aantal beslissingsvariabelen.

Beperkingen behorend bij een optimalisatieprobleem worden in veel vakgebieden veelal voortgebracht door de bijzondere eigenschappen van de omgeving of door de beperkte hoeveelheid aan materiaal en/of bronnen. Aan deze beperkingen, ook wel de bijvoorwaarden genoemd, moet worden voldaan voordat een oplossing accepteerbaar genoemd kan worden. De bijvoorwaarden zijn m ongelijkheidsbeperkingen en/of p gelijkheidsbeperkingen die als volgt geformuleerd kunnen worden [9]:

$$g_j \geq 0 \quad \text{voor j} = 1,...,m,$$
$$h_l = 0 \quad \text{voor l} = 1,...,p.$$

Hierbij zal p wel kleiner moeten zijn dan n, het aantal beslissingsvariabelen, omdat er anders geen vrijheid meer overblijft om te optimaliseren.

3.1.2 Multi-objective optimalisatie

Veel optimalisatieproblemen bevatten meerdere mogelijk tegenstrijdige en niet-lineaire doelen die geoptimaliseerd dienen te worden en kunnen daarbij gebruik maken van multi-objective optimalisatietechnieken. Multi-objective optimalisatie kan als volgt gedefinieerd worden [10]:

Multi-objective optimalisatie is het probleem van het vinden van een vector bestaande uit de beslissingsvariabelen die aan de bijvoorwaarden moeten voldoen en een vector van functies, waarvan de elementen de doelfuncties voorstellen, moeten optimaliseren.

Oftewel, er dient een vector x^* gevonden te worden die aan de m ongelijkheidsbijvoorwaarden en aan de p gelijkheidsbijvoorwaarden voldoet en die de vector functies f(x) optimaliseert.

Aangezien de doelfuncties vaak tegenstrijdig kunnen zijn, zullen de beslissingsvariabelen in de meeste gevallen niet een enkele waarde kunnen aannemen waarvoor alle doelfuncties optimaal zijn. Vandaar dat de term optimaliseren bij multi-objective optimalisatie inhoudt dat er een oplossing gevonden dient te worden waarbij de waarden van alle doelfuncties zo goed mogelijk zijn.

Voor het afbeelden van de beslissingsvariabelen en de waarden van de doelfuncties, wordt de set van de n beslissingsvariabelen en de set van de k doelfuncties genoteerd als respectievelijk R^n, de n-ruimte of de zoekruimte, en R^k, de k-ruimte of de oplossingsruimte [9]. Bij de n-ruimte komt iedere as overeen met een component van de vector x en bij de k-ruimte komt iedere as overeen met een component van de vector f(x). Ieder punt dat zich in de n-ruimte bevindt representeert een oplossing en geeft dus een punt in de k-ruimte; $f : \Re^n \rightarrow \Re^k$.

3.1.3 Het Pareto front

Door het ontbreken van een ideaal optimum voor een multi-objective optimalisatie probleem in de meeste gevallen, is het zeer efficiënt om de gebruiker van een set van oplossingen te voorzien. Hierdoor kan de gebruiker vervolgens bepalen welk punt in de oplossingsruimte het meest optimale punt is voor een bepaald probleem. Deze set van oplossingen staat bekend als de Pareto set.
De Pareto set is onder te verdelen in meerdere groepen. Alle oplossingen binnen een groep kunnen allemaal even goed zijn, maar ze zijn minder goed of beter dan de oplossingen in een andere groep. Door deze verdeling kan er nu aan iedere groep een rang gegeven worden en de groep met rang één is de groep die de Pareto optimale oplossingen bevat. De punten die Pareto optimaal zijn bij een minimalisatie probleem, zijn die punten met de beslissingsvariabelen x^* waarvoor geldt dat

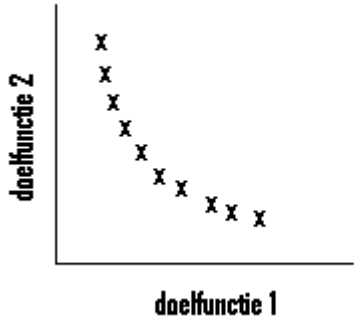
$$f_i(x) \geq f_i(x^*),$$

en er is tenminste één i waarbij

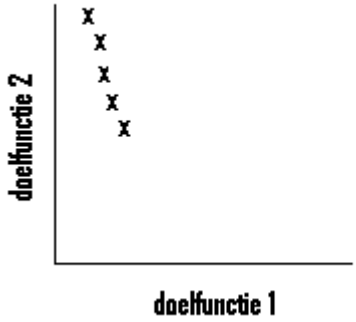
$$f_i(x) > f_i(x^*).$$

Ofwel, x^* is Pareto optimaal als er geen andere toegankelijke vectoren x bestaan die voor een daling zorgt in minimaal één doelfunctie zonder een andere doelfunctie te doen laten stijgen [10]. Deze punten worden ook wel de niet-gedomineerde oplossingen genoemd en samen vormen deze oplossingen het Pareto front. Met andere woorden, het Pareto front vormt de set van oplossingen die zoveel mogelijk naar de daadwerkelijke optimale oplossing convergeert.
Het Pareto front kan getoond worden als een lijn, vlak of hypervlak in de oplossingsruimte, dat wil zeggen dat de ruimte (twee- of meerdimensionaal) opgespannen wordt door de verschillende doelfuncties. Het idee is nu om zoveel mogelijk verschillende Pareto optimale oplossingen te vinden die samen dit Pareto front vormen, zodat de beslissing voor een multi-objective optimalisatie probleem op meerdere optimale oplossingen gebaseerd kan worden.

Voorbeeld
In de figuren 3.1 en 3.2 is een Pareto front aangegeven voor twee doelfuncties, aangenomen wordt hier dat de assen in de beide figuren hetzelfde bereik hebben en dat beide doelfuncties geminimaliseerd moeten worden.



Figuur 3.1: Pareto front



Figuur 3.2: Pareto front zonder veel spreiding

De gebruiker dient nu een keuze te maken welk punt van het Pareto front voor hem het meest optimaal is. Zijn keuze zal onder meer afhangen van zijn mening over welke doelfunctie het belangrijkste is om te minimaliseren, maar hij kan er uiteraard ook voor kiezen beide doelfuncties ongeveer dezelfde waarden aan te laten nemen. In ieder geval kan hij een punt van het Pareto front kiezen die naar zijn mening de beste oplossingen bevatten voor de doelfuncties.
In figuur 3.2 is een Pareto front afgebeeld, waarbij de spreiding van de oplossingen veel minder is dan bij figuur 3.1. Door deze kleine spreiding heeft de gebruiker veel minder keuze waardoor hij er eigenlijk niet omheen kan om de tweede doelfunctie een veel hogere waarde te laten aannemen dan de eerste doelfunctie als hij gebruik maakt van een punt van dit Pareto front.

De twee hoofddoelen van multi-objective optimalisatie zijn nu als volgt te definiëren:

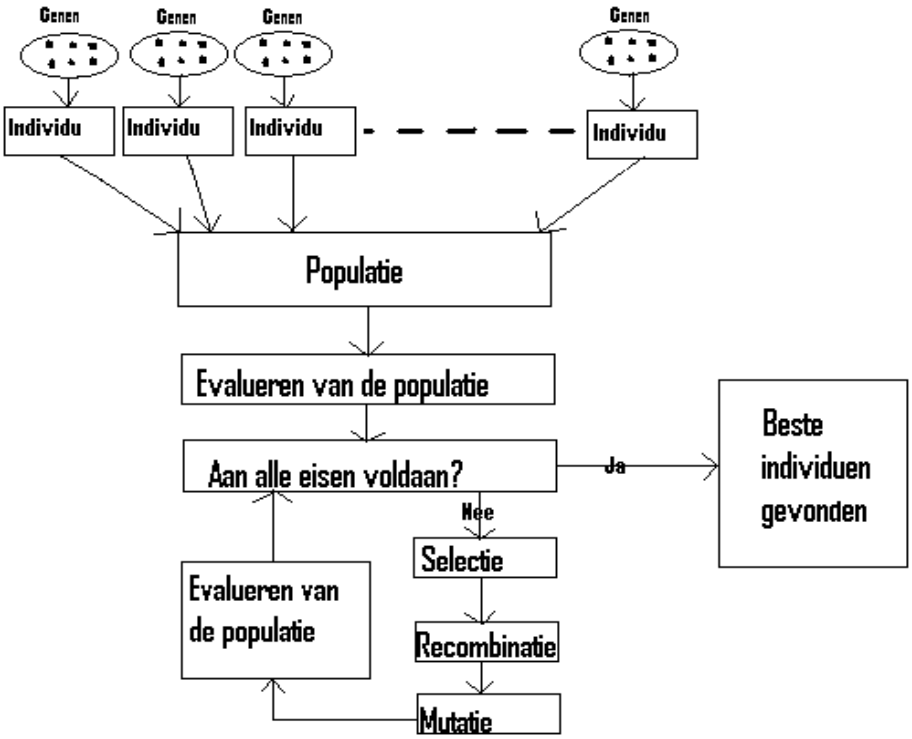
1. Convergentie naar de optimale Pareto set
2. Zoveel mogelijk verschillende oplossingen hebben in het Pareto front

De categorie algoritmen Evolutionary Computing lijkt zeer effectief met deze twee hoofddoelen van multi-objective optimalisatie om te kunnen gaan.

3.2 Evolutionaire en genetische algoritmen

In de jaren 60 is het idee van evolutionaire algoritmen geïntroduceerd door Ingo Rechenberg [21] en sindsdien is hier door andere auteurs op voortgebouwd. Evolutionaire algoritmen zijn stochastische optimalisatietechnieken die gebaseerd zijn op het nabootsen van de biologische evolutie [20]. Hiervoor wordt er gebruik gemaakt van een populatie die uit meerdere individuen bestaat. Deze individuen hebben één of meerdere genen die de beslissingsvariabelen representeren en daarmee de eigenschappen van deze individuen bepalen; de waarden van de doelfuncties. Op de genen van de individuen in een populatie kunnen evolutionaire mechanismen zoals selectie, crossover en mutatie[1] worden toegepast om een nieuwe en meer passende populatie te creëren. Zo ontstaat er net als in de natuur een evolutie van populaties van individuen die beter passen bij de omgeving dan de individuen waaruit ze gecreëerd zijn [20]. De beste individuen, dat wil zeggen de individuen die de genen bezitten die de meeste fitness behalen bij de doelfuncties, zullen nu steeds samen de gezochte optimale oplossingen vormen, in het geval van multi-objective optimalisatie, zullen deze individuen nu op het Pareto front liggen.
Anders gezegd, evolutionaire algoritmen passen het principe van overleven toe op de meest fitte individuen om zo een steeds betere set van oplossingen te kunnen vinden voor een probleem.

[1] Deze processen zullen in de paragraaf evolutionaire mechanismen verder toegelicht worden



Figuur 3.3: Evolutionair algoritme [5]

Een speciaal type van de evolutionaire algoritmen zijn de genetische algoritmen, die ook gebaseerd zijn op de natuurlijke evolutie. De belangrijkste acties in een genetisch algoritme voor een multi-objective optimalisatie probleem zijn: de selectiemethode, de crossover methode, de mutatie operator[1] en de plaatsingsprocedure [17].

Wanneer een multi-objective optimalisatieprobleem veel variabelen bevat, wat vaak het geval is, zal de oplossing in een hoog dimensionale zoekruimte gezocht moeten worden. Uit onderzoek [19] is gebleken dat de genetische en evolutionaire algoritmen hier het meest geschikt voor zijn. Dit komt onder meer doordat [7], [20], [22]:

- Per iteratie er een Pareto front gevormd kan worden met verschillende oplossingen
- Het toepassen van evolutionaire mechanismen eraan bijdraagt dat de zoekkracht waarmee genetische en evolutionaire algoritmen in een zoekruimte kunnen werken groot is
- Genetische en evolutionaire algoritmen goed overweg kunnen met bijvoorwaarden
- Genetische en evolutionaire algoritmen geen informatie van de afgeleiden van de doelfuncties eisen, maar alleen de waarde van de doelfunctie zelf
- Genetische en evolutionaire algoritmen over het algemeen simpel zijn toe te passen, omdat er weinig eisen (bijvoorbeeld van continuïteit en differentieerbaarheid) gesteld worden aan de doelfuncties
- Genetische en evolutionaire algoritmen gebruik maken van een probabilistische aanpak in plaats van een deterministische. Hierdoor is de kans dat het globale optimum gevonden wordt, in plaats van een lokaal optimum waarin veel deterministische methoden vastlopen, een stuk groter
- Per individu worden alle doelfuncties gecheckt zodat deze verwerkt kunnen worden tot een totale beoordeling, oftewel de totale fitness van een individu

Het laatste voordeel dat hierboven wordt genoemd, dat alle individuen op alle doelfuncties gecontroleerd worden, brengt ook het nadeel voort voor genetische en evolutionaire algoritmen, namelijk dat door deze beoordeling de algoritmen nogal rekenintensief kunnen zijn.

3.3 Evolutionaire mechanismen

Evolutionaire en genetische algoritmen zijn, zoals eerder beschreven, gebaseerd op evolutionaire mechanismen. Het idee achter deze mechanismen zal hieronder beschreven worden.

3.3.1 Selectie

Bij de selectieprocedure wordt er bepaald welke individuen worden gebruikt voor de crossover methode en de mutatie methode. Om ervoor te zorgen dat een selectiemethode in staat is om de beste individuen er uit te pikken, zal er allereerst een fitness bepaald worden van alle individuen of zal er aan alle individuen een rang toebedeeld worden. In de loop der tijd zijn er verschillende soorten selectiemethoden ontwikkeld, waarvan een aantal in Appendix A wordt beschreven.

3.3.2 Crossover

In het proces van crossover worden er nieuwe individuen, ofwel kinderen, gecreëerd die bepaalde eigenschappen van hun ouders kunnen overnemen, dat wil zeggen dat de waarden voor een aantal van de genen (beslissingsvariabelen) gelijk kan blijven bij een nieuwe generatie van oplossingen, terwijl de andere genen een nieuwe waarde aan zullen nemen. In Appendix A worden enkele voorbeelden gegeven hoe de genen, behorende bij individuen van een volgende generatie, veranderen ten opzichte van de genen van de huidige generatie door middel van een crossover methode.

3.3.3 Mutatie

Zodra de crossover geschiedt is, vindt de mutatie plaats (een uitgebreidere beschrijving van dit proces is te vinden in Appendix A) . De genen waaruit een individu bestaat, kunnen meerdere waarden aannemen, welke bij het proces van mutatie willekeurig worden veranderd. De reden dat mutatie wordt toegepast is om te proberen om de suboptimale oplossingen te verbeteren en om de verscheidenheid binnen de populatie te behouden door het herintroduceren van waarden die verloren zijn gegaan bij het herhaald toepassen van een crossover methode.

3.4 Algoritmen

In de afgelopen decennia zijn er vele verschillende algoritmen ontwikkeld die tot de evolutionary computing categorie behoren. Enkele voorbeelden hiervan zijn de algoritmen Vector Evaluated Genetic Algorithm (VEGA), Constrained Optimisation by Multi-Objective Genetic Algorithms (COMOGA), Niched Pareto Genetic Algorithm (NPGA), Pareto Achieved Evolutionary Strategy (PAES) en Strength Pareto Evolutionary Algorithm 2 (SPEA2), welke hieronder nader besproken zullen worden om zo een beeld te schetsen in welke mate de algoritmen die tot de evolutionary computing behoren kunnen verschillen.

3.4.1 VEGA

Het Vector Evaluated Genetic Algorithm werd al midden jaren ‘80 ontwikkeld [8]. Dit algoritme maakt gebruik van subpopulaties die allemaal apart een andere doelfunctie optimaliseren, waardoor het aantal subpopulaties ook gelijk is aan het aantal doelfuncties. Per subpopulatie wordt nu de fitness berekend die dus alleen op deze enkele doelfunctie betrekking heeft, waarbij het dus zo kan zijn dat de individuen van een zekere subpopulatie geen goede fitness hebben voor een doelfunctie die toegewezen is aan een andere subpopulatie, maar wel voor de doelfunctie waar hun fitness op gebaseerd is.
Als er een aannemelijke oplossing is gevonden dan wordt er als het ware een hypercube om de oplossing in de oplossingsruimte getrokken om te vermijden dat dit aannemelijke oplossingsgebied verlaten kan worden.

VEGA heeft het voordeel dat het gemakkelijk te implementeren is, maar er wordt hier geen gebruik gemaakt van de Pareto dominantie [3] en meestal heeft VEGA de neiging om naar een enkele oplossing te convergeren wanneer er van veel generaties gebruik gemaakt wordt [9].

3.4.2 COMOGA

In de Constrained Optimisation by Multi-Objective Genetic Algorithms wordt een combinatie gebruikt van de VEGA methode en van de Pareto rangschikking [19]. De individuen worden hier gerangschikt volgens hun aantal overtredingen van de bijvoorwaarden, waarbij de bijvoorwaarden als aparte doelfuncties worden gepresenteerd. De fitnessevaluaties worden vervolgens gebaseerd op de doelfunctie waaraan een subpopulatie is toegewezen én het aantal overtredingen van de bijvoorwaarden. Bij het gebruik van COMOGA zal er door de gebruiker wel extra parameters ingevuld moeten worden.

3.4.3 NPGA

Het Niched Pareto Genetic Algorithm maakt gebruik van de tournament selectie, door steeds twee willekeurige individuen te vergelijken met een subset van de populatie. Het blijkt echter dat het bepalen van de optimale grootte van deze subset een moeilijk onderdeel is [8]. Als nu één van de twee individuen de subset domineert, dan zal dit individu gekozen worden om toegevoegd te worden aan het Pareto front. Als beiden individuen de subset domineren of als de twee individuen worden gedomineerd door één of meerdere individuen uit de subset, dan zal nog steeds één van de twee individuen geselecteerd worden om te worden toegevoegd aan het Pareto front, namelijk het individu dat de kleinste niche count bezit in de oplossingsruimte, met andere woorden het individu dat het kleinst aantal andere individuen in zijn buurt zal krijgen in de oplossingsruimte.
Bij dit algoritme worden dus niet alle individuen met elkaar vergeleken, maar wordt er een steekproef van de populatie gebruikt. Een voordeel hiervan is dat de methode reken-efficiënt is, maar het algoritme heeft tegelijkertijd weer het nadeel dat de prestatie ervan omlaag gaat, naarmate het aantal beslissingsvariabelen stijgt [3].

3.4.4 PAES

De Pareto Achieved Evolutionary Strategy is een lokaal zoekalgoritme. Dit algoritme begint met het willekeurig kiezen van een individu uit de populatie. Met behulp van een mutatie operator zal er uit dit individu een kind worden gecreëerd, waarna er gekeken zal worden of het kind de ouder domineert of dat de ouder het kind domineert. Indien de ouder het kind domineert, zal het kind genegeerd worden en zal er een nieuw kind uit deze ouder geproduceerd worden. Echter, als het kind de ouder domineert, zal het kind aan de populatie worden toegevoegd, waar het kind vergeleken zal worden met een set van niet-gedomineerde individuen. Deze vergelijking vindt plaats om spreiding tussen de individuen op het Pareto front te handhaven. Als het kind één of meer individuen van deze set domineert, zelf niet gedomineerd wordt en bovendien op een punt in het Pareto front zal komen te liggen waar minder andere individuen liggen dan bij de ouder van het kind, zal het worden toegevoegd, maar als het kind zelf wordt gedomineerd door één van de individuen uit deze set, dan zal het uiteindelijk toch niet worden toegevoegd. Wanneer geen van beiden bovengenoemde gevallen waar zijn, zal er worden gekeken waar het kind zal komen te liggen op het Pareto front. Blijkt dit dan op een punt te zijn waar minder individuen liggen dan op de positie waar de ouder lag, dan zal het kind alsnog worden toegevoegd [15].

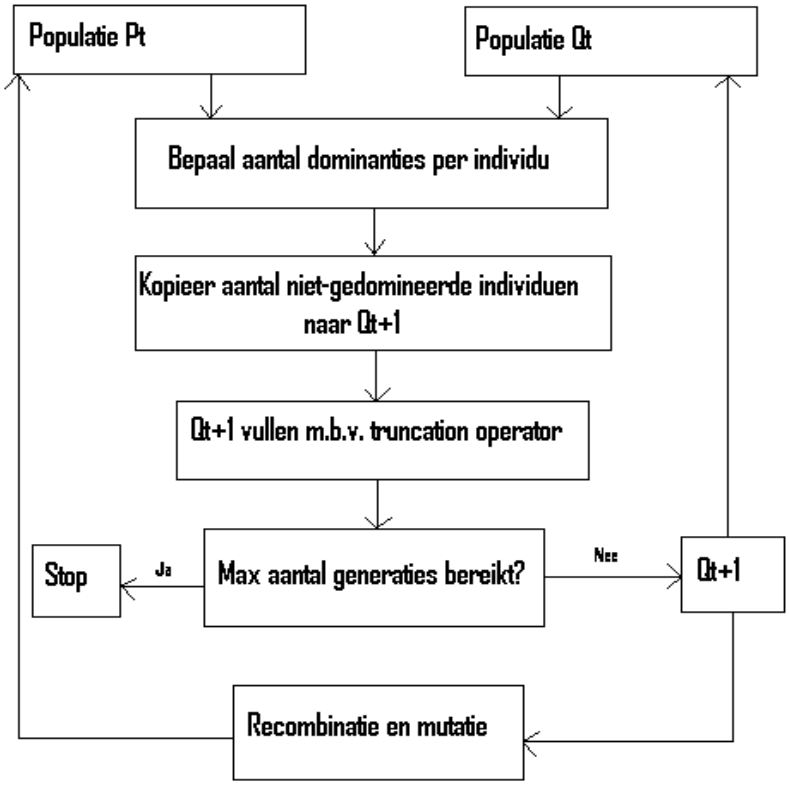
3.4.5 SPEA2

Strength Pareto Evolutionary Algorithm 2 is een verbeterde versie van Strength Pareto Evolutionary Algorithm (SPEA) [25]. De verbeterde punten in SPEA2 ten opzichte van SPEA zijn; een verbeterde toekenning van fitness, waarbij wordt bijgehouden hoeveel individuen een individu domineert en door hoeveel individuen een individu wordt gedomineerd. Verder bezit SPEA2 een techniek voor het

benaderen van hoeveel individuen er vlakbij een bepaald individu liggen en bevat SPEA2 een nieuwe benadering van de truncation methode die het behoud van de optimale oplossingen garandeert.

Bij SPEA2 wordt er met twee sets van N individuen gewerkt: P_t en Q_t. Op de individuen die in Q_t zitten zal er steeds crossover en mutatie worden toegepast en de kinderen die hieruit voortkomen worden vervolgens in P_{t+1} geplaatst. Alle individuen die in de set P_{t+1} en Q_t niet gedomineerd worden, worden overgebracht naar Q_{t+1}. Indien het aantal individuen in Q_{t+1} groter is dan N, dan zullen er volgens de truncation operator individuen uit Q_{t+1} worden gehaald. Indien het aantal individuen in Q_{t+1} juist kleiner blijkt te zijn dan N, dan zullen er extra individuen uit P_{t+1} worden gehaald, met behulp van hun fitnesswaarde en de truncation operator, om de set Q_{t+1} tot N op te vullen. Iedere keer als Q_{t+1} gevuld is tot en met N individuen, zal er opnieuw crossover en mutatie plaatsvinden om weer een nieuwe generatie P_{t+1} te creëren. Dit proces gaat net zolang door tot het van te voren opgegeven aantal generaties bereikt is.

De fitnesswaarde van een individu wordt bij dit algoritme bepaald door het aantal individuen op te tellen die het eerste individu domineren vanuit zowel P als Q. Om onderscheid te maken tussen individuen die dezelfde fitnesswaarde hebben, wordt er voor de totale fitness ook een populatiedichtheid toegevoegd die de truncation operator gebruikt. Dit wordt gedaan door voor ieder individu de afstand tot de overige individuen te berekenen (in de oplossingsruimte), deze te sorteren en daarvan het k-de individu te nemen, σ_i^k. De populatiedichtheid per individu wordt nu berekend door: $D(i) = 1/(\sigma_i^k+2)$. Indien de truncation operator nodig is om individuen te selecteren, dan zal hij die individuen voor Q_{t+1} selecteren die de kleinste populatiedichtheid bezitten.



Figuur 3.4: SPEA2

3.5 NSGA2

Tot de evolutionary computing behoort ook het Nondominated Sorting Genetic Algorithm 2. Dit algoritme lijkt, ten opzichte van alle andere algoritmen, dusdanig veel goede eigenschappen te bezitten, dat er hier een complete paragraaf aan dit algoritme besteedt zal worden.

3.5.1 Van NSGA naar NSGA2

In de beginjaren bij het onderzoek naar genetische algoritmen werd het algoritme NSGA (Nondominated Sorting Genetic Algorithm) ontwikkeld door Deb en Srinivas [4]. Dit algoritme werd in de loop der jaren zwaar bekritiseerd [15] onder meer vanwege de rekenkundige complexiteit, deze kon namelijk $O(MN^3)$ bedragen met M het aantal doelfuncties en N de grootte van de populatie. NSGA was rekeninefficiënt doordat er gebruik werd gemaakt van een plaatsingsprocedure die na het definiëren van het eerste niet-gedomineerde front de rest van de populatie opnieuw ging doorzoeken om het volgende niet-gedomineerde front te vinden, waarbij dus alle individuen weer met elkaar vergeleken moesten worden. Verder konden goede resultaten verloren gaan doordat in een nieuwe generatie alle ouders vervangen waren door de kinderen die zij geproduceerd hadden. Bovendien was het noodzakelijk dat de gebruiker steeds een sharing parameter, die de spreiding van de verschillende oplossingen diende te bepalen, moest specificeren. Het Nondominated Sorting Genetic Algorithm 2 vermindert de drie bovenstaande moeilijkheden [15].

3.5.2 Ranking van de individuen

Bij NSGA2 wordt de populatie van iedere generatie gesorteerd op basis van niet-gedomineerde individuen. Hierbij worden er voor ieder individu p twee zaken bijgehouden: de eerste is een domination count, n_p, die het aantal individuen bijhoudt die het individu p domineren. De tweede is een verzameling van individuen, S_p, die het individu p zelf domineert.
Alle individuen die door geen enkel ander individu worden gedomineerd, dus waarvoor $n_p = 0$, zullen in het eerste niet-gedomineerde front terecht komen en krijgen daar rang 1. Alle overige individuen worden dus door één of meer individuen gedomineerd. Om ook de overige individuen te sorteren (deze zouden namelijk nog gebruikt kunnen worden in het verdere verloop van het algoritme), wordt de volgende strategie gevolgd: ieder individu dat net een rang toebedeelt heeft gekregen, heeft nog steeds zijn verzameling S_p. Van alle individuen die zich in deze verzameling S_p bevinden, wordt de n_p met één gereduceerd. Deze strategie wordt voor ieder individu uitgevoerd dat net gesorteerd is, dus net een rang toebedeeld heeft gekregen. Vervolgens wordt er gekeken voor welke individuen ditmaal de n_p nul is, dit zijn de individuen die tweede niet-gedomineerde front vormen en zij krijgen rang 2.
Bovenstaande procedure wordt net zolang herhaald tot alle individuen in een niet-gedomineerd front zijn verwerkt en daardoor hun betreffende rang hebben gekregen.

3.5.3 De crowding afstand

Om de spreiding op het Pareto front te bevorderen en het doen vervangen van de sharing parameter van NSGA, maakt NSGA2 gebruik van de zogenoemde crowding afstand bij ieder front. Voor het berekenen van de crowding afstand wordt ieder individu in oplopende waarde gesorteerd per doelfunctie, dat wil zeggen dat het individu met de kleinste waarde voor een doelfunctie als eerste in de rij staat en het individu met de grootste waarde als laatste. In deze gesorteerde rij wordt de waarde van de doelfunctie van het eerste en het laatste individu vervangen door oneindig. Vervolgens worden voor de overige individuen het verschil in waarde berekend tussen het individu dat net een iets kleinere waarde heeft en het individu dat een net iets grotere waarde heeft. Dus als de individuen bijvoorbeeld in oplopende grootte in een array zouden staan voor een doelfunctie, dan zou de crowding afstand van een individu i worden berekend door de waarde voor de doelfunctie van individu i-1 af te halen van de waarde voor de doelfunctie van individu i+1. De totale crowding afstand voor een individu is nu de sommatie van alle crowding afstanden van dat individu voor alle doelfuncties, gedeeld door het totaal aantal doelfuncties.

Alle individuen van een populatie kunnen nu dus vergeleken worden, zowel qua rang als qua crowding afstand. De individuen waar nu naar gezocht wordt, zijn die individuen met de laagste rang en met de grootste crowding afstand, om zo de twee hoofddoelen van multi-objective optimalisatie te bereiken; een geconvergeerd en gespreid Pareto front.

3.5.4 De crossover methode

NSGA2 maakt gebruik van de Simulated Binary Crossover (SBX), welke gebaseerd is op de blend crossover (BLX-α) [12], waarbij er twee kinderen uit twee ouders gecreëerd worden. De BLX-α creëert kinderen door alle genen een waarde te geven die in het volgende interval liggen [11]:

$$[x_i^{(1,t)} - \alpha(x_i^{(2,t)} - x_i^{(1,t)}), x_i^{(2,t)} + \alpha(x_i^{(2,t)} - x_i^{(1,t)})].$$

Waarbij $x_i^{(1,t)}$, $x_i^{(2,t)}$ het gen i van respectievelijk de eerste ouder en de tweede ouder voorstellen bij generatie t en α neemt hier een waarde aan tussen de nul en één.
De locatie van het kind zal dus afhangen van het verschil van de genen van de ouders. Bovendien heeft het bovenstaande gedefinieerde interval tot gevolg dat wanneer het verschil van de oplossingen van de ouders klein is, dan ook het verschil tussen de kinderen en de ouders klein zal zijn. Op dit principe is de SBX crossover gebaseerd, alleen maakt de SBX crossover gebruik van een distributie index die bepaalt in hoeverre de kinderen van de ouders kunnen afliggen [14].

Alvorens aan te kunnen tonen wat voor een invloed de distributie index heeft, zal eerst een zogenaamde "spread factor", worden gedefinieerd:

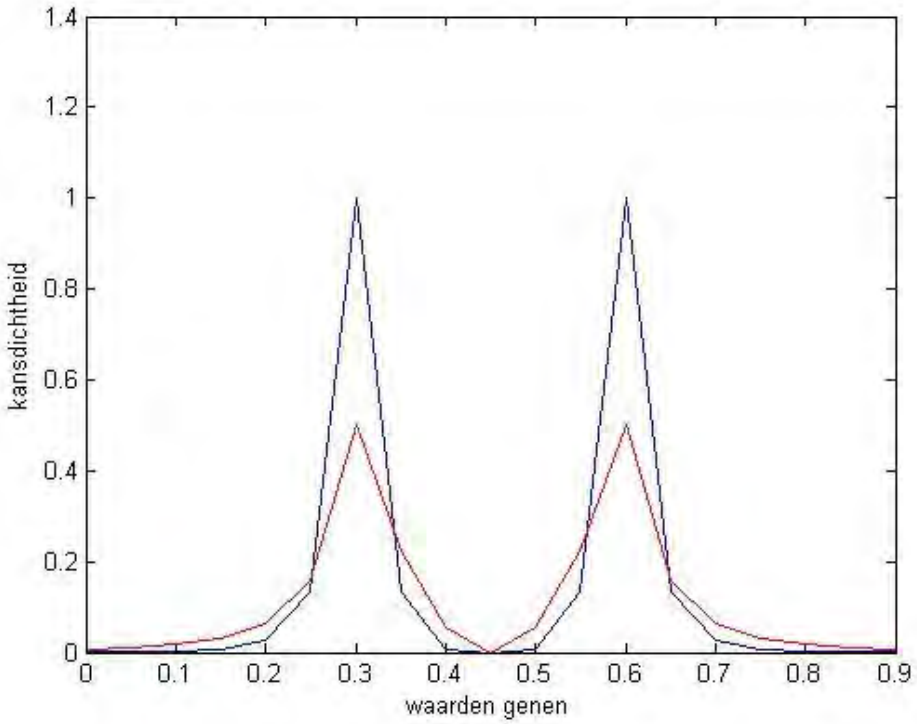
$$\beta_i = \left| \frac{x_i^{(2,t+1)} - x_i^{(1,t+1)}}{x_i^{(2,t)} - x_i^{(1,t)}} \right|.$$

Waarbij $x_i^{(1,t+1)}$ en $x_i^{(2,t+1)}$ het gen i van respectievelijk het eerste kind en het tweede kind voorstellen bij generatie t+1. Deze β_i geeft dus het absolute verschil van de waarden van de genen van de kinderen tot de waarden van de genen van de ouders.

Om te kunnen bekijken welke waarden de genen van de kinderen met de grootste waarschijnlijkheid aan zullen nemen, wordt er gebruik gemaakt van een kansverdeling.

$$\mathrm{P}(\beta_i) = \begin{cases} 0.5(\eta+1)\beta_i^{\eta} & als \quad \beta_i \le 1 \\ 0.5\dfrac{\eta+1}{\beta_i^{\eta+2}} & anders \end{cases}.$$

De verschillende waarden van β_i die hier worden gebruikt, zijn de waarden die bereikt worden door die waarden voor de genen van de kinderen te nemen, die liggen tussen de ondergrens en bovengrens zoals deze gedefinieerd zijn voor die genen. De parameter η stelt de distributie index voor, welke aan zal geven in hoeverre de kinderen van de ouders af kunnen komen te liggen. Het zal blijken dat de kinderen dichter bij de ouders zullen komen te liggen wanneer de distributie index een hoge waarde heeft, dan wanneer de distributie index een lage waarde heeft.
De invloed van de distributie index voor crossover is weergegeven in figuur 3.5, door de kansdichtheid behorende bij twee verschillende waarden van de distributie index af te beelden voor één gen. De waarde van dit gen behorende bij ouder 1 is 0.3 en ouder 2 heeft een waarde van 0.6 voor het gen, terwijl de ondergrens op 0 ligt en de bovengrens op 0.9. De blauwe lijn in figuur 3.5 geeft de kansdichtheid weer behorende bij η = 5 en de rode lijn behoort bij de kansdichtheid waarbij η = 2.



Figuur 3.5: kansdichtheid met distributie index 2 en 5

In figuur 3.5 is goed de invloed te zien van de distributie index, dat wil zeggen dat uit figuur 3.5 duidelijk blijkt dat een kind de meeste kans zal hebben om een waarde voor het gen te krijgen die in de buurt van een ouder ligt naarmate de distributie index groter is.

Om de waarden van de genen van een kind daadwerkelijk te berekenen, wordt β gedefinieerd op basis van de waarden van de genen van de ouders. Hiervoor wordt in eerste instantie de waarde van het gen van de eerste ouder vergeleken met de waarde van het gen van de tweede ouder. De ouder die nu de kleinste waarde voor het gen bezit krijgt de aanduiding BV1 en de andere ouder krijgt de aanduiding BV2. Verder zal er ook gebruik worden gemaakt van de ondergrens en bovengrens van het gen, deze worden respectievelijk aangeduid als BVmin en BVmax. De waarde van β kan nu berekend worden door:

$$\beta = \begin{cases} 1 + \left(2 \dfrac{(BV1 - BV\min)}{(BV2 - BV1)} \right) & voor \quad kind \quad 1 \\ 1 + \left(2 \dfrac{(BV\max - BV2)}{(BV2 - BV1)} \right) & voor \quad kind \quad 2 \end{cases}.$$

De waarde van β wordt vervolgens gebruikt om de waarde van de variabele α te berekenen:

$$\alpha = 2 - \beta^{(-(\eta+1))},$$

welke weer gebruikt wordt voor de berekening van de variabele βqi:

$$\beta_{qi} = \begin{cases} (u\alpha)^{\frac{1}{\eta+1}} & als \quad u \le \frac{1}{\alpha} \\ \left(\frac{1}{2-u\alpha}\right)^{\frac{1}{\eta+1}} & anders \end{cases}.$$

Waarbij u een random number is tussen nul en één. Na de berekening van al deze bovenstaande variabelen, kunnen tot slot de waarden van de desbetreffende genen van de kinderen bepaald worden [15]:

$$\text{gen van kind 1} = 0.5((BV1 + BV2) - \beta_{qi}(BV2 - BV1)),$$
$$\text{gen van kind 2} = 0.5((BV1 + BV2) + \beta_{qi}(BV2 - BV1)).$$

3.5.5 De mutatie methode

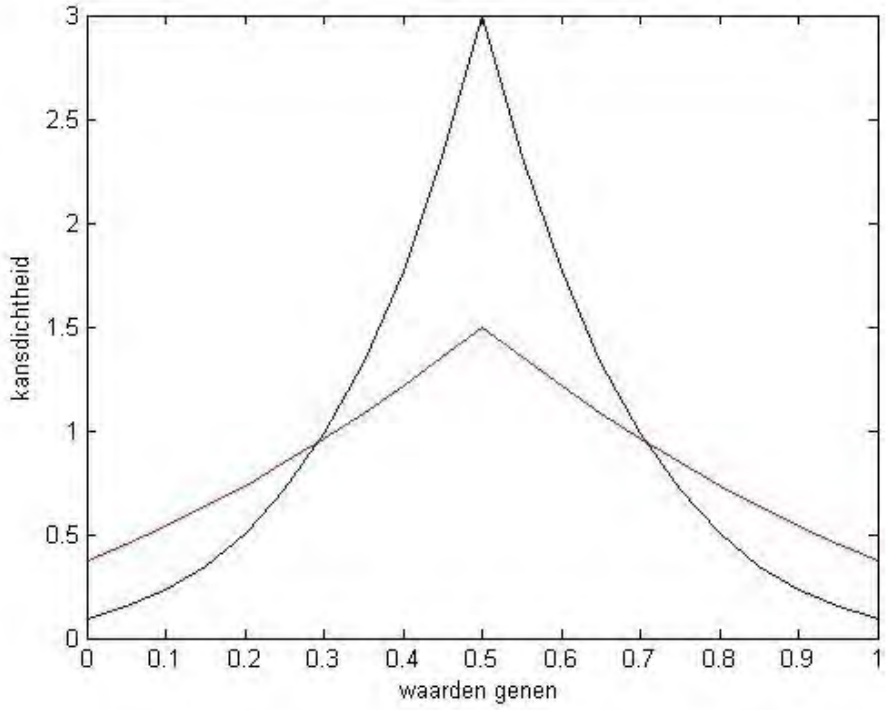
De mutatie methode waar NSGA2 gebruik van maakt, is de parameter-based mutatie waarbij de huidige waarde van een gen van een ouder wordt veranderd in een 'in de buurt liggende' waarde met behulp van een distributie index [13]. Evenals in de SBX crossover geldt ook hier dat hoe hoger de waarde van de distributie index des te meer kans dat de waarden van de genen van de kinderen in de buurt van de waarden van de genen van de ouders zullen komen te liggen. Echter wordt er bij de mutatie methode geen gebruik gemaakt van de spreidingsfactor β_i, maar van een storingsfactor δ:

$$\delta = \frac{x_i^{(1,t+1)} - x_i^{(1,t)}}{BV\max - BV\min}.$$

Om te kunnen bekijken welke waarden de gemuteerde genen van de kinderen met de grootste waarschijnlijk aan zullen nemen, kan er wederom gebruik worden gemaakt van een kansverdeling. Voor de bepaling van δ worden de mogelijke waarden voor de genen van de kinderen gebruikt, die liggen tussen de ondergrens en bovengrens van de waarden van deze genen. De kansverdeling kan daarna als volgt berekend worden:

$$\mathrm{P}(\delta) = 0.5(\eta + 1)(1 - |\delta|)^{\eta}.$$

De invloed van de distributie index voor deze mutatie methode is te vinden in figuur 3.6 waar een afbeelding voor één gen is geschetst. De waarde van dit gen, die dus tot de ouder behoort, is 0.5, terwijl de ondergrens op nul ligt en de bovengrens op één. De blauwe lijn in figuur 3.6 stelt wederom de kansdichtheid behorende bij $\eta = 5$ voor en de rode lijn staat weer voor de kansdichtheid bij $\eta = 2$.



Figuur 3.6: kansdichtheid met distributie index 2 en 5

Uit figuur 3.6 blijkt dat ook de distributie index voor de mutatie een grote invloed heeft op de kans in hoeverre de waarden van de gemuteerde genen van de kinderen af komen te liggen van de waarden van de genen van de ouders.

Om nu de waarden van de gemuteerde genen voor de kinderen daadwerkelijk te berekenen, wordt een δ berekend op basis van de waarde van het gen van de ouder (BV):

$$\delta_1 = \frac{BV - BV\min}{BV\max - BV\min},$$
$$\delta_2 = \frac{BV\max - BV}{BV\max - BV\min}.$$

De reden dat δ hier is opgesplitst in twee delen, heeft te maken met de random variabele, u, die ook in deze mutatie methode wordt gebruikt. Bovendien dient ook hier, net als bij de SBX crossover, een aantal variabelen berekend te worden alvorens de waarde van het gen van het kind bepaald kan worden:

$$\alpha = \begin{cases} 1-\delta_1 & als \quad u \le 0.5 \\ 1-\delta_2 & als \quad u > 0.5 \end{cases},$$

$$\beta = \begin{cases} 2u + (1-2u)\alpha^{\eta+1} & als \quad u \le 0.5 \\ 2(1-u) + 2(u-0.5)\alpha^{\eta+1} & als \quad u > 0.5 \end{cases},$$

$$\delta_q = \begin{cases} \beta^{\frac{1}{\eta+1}-1} & als \quad u \le 0.5 \\ 1-\beta^{\frac{1}{\eta+1}} & als \quad u > 0.5 \end{cases}.$$

Tot slot kan nu de waarde van het gen van het kind als volgt worden berekend [15]:

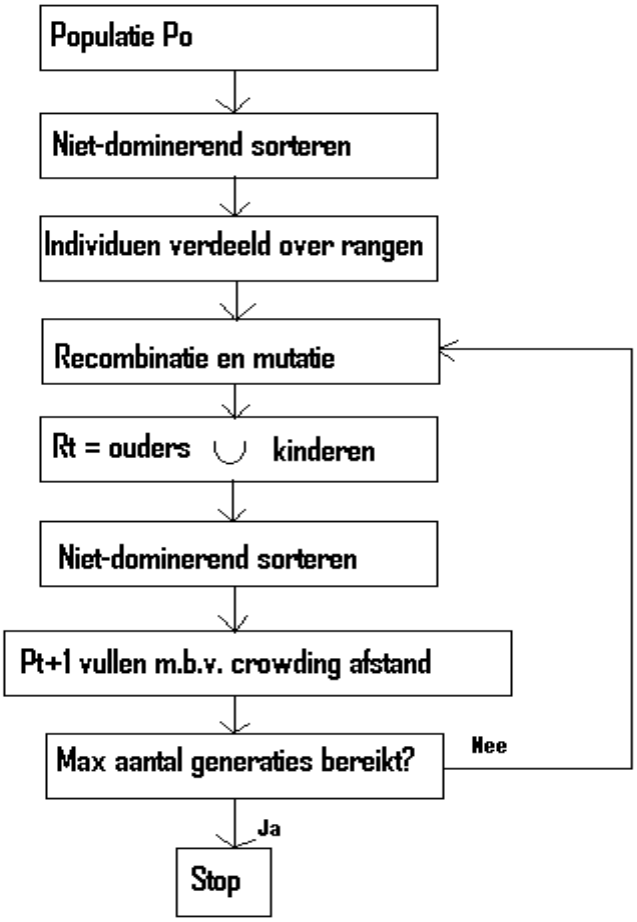
$$\text{gen van kind} = BV + \delta_q (BV\,\max - BV\,\min).$$

3.5.6 NSGA2 van begin tot eind

Na de beschrijving van de ranking, de crowding afstand en de evolutionaire mechanismen van NSGA2, kan nu uitgelegd worden hoe het algoritme in zijn geheel te werk gaat. Dit proces staat tevens afgebeeld in figuur 3.7.
NSGA2 begint met een willekeurige populatie P_0, bestaande uit N individuen. Door ieder individu een rang toe te wijzen op basis van het niet-gedomineerde front waaraan ze toebehoren, kan de populatie vervolgens op een dominerende wijze gesorteerd worden, dat wil zeggen dat alle individuen met rang 1 voorop staan, daarachter komen de individuen met rang 2, etc.

Vanaf hier wordt het algoritme voor iedere generatie op dezelfde manier uitgevoerd, totdat het aantal opgegeven T generaties bereikt is.
Door middel van de evolutionaire mechanismen crossover en mutatie worden er N kinderen gecreëerd, welke in de verzameling Q_{t+1} worden gezet, uit de N ouders. Vervolgens wordt er een gecombineerde populatie gecreëerd door de populaties P_t, de ouders, en Q_{t+1}, de kinderen, te verenigen tot een populatie R_t die dus 2N individuen bevat, welke gesorteerd zal worden op de ondertussen bekende dominerende wijze. Uit de gesorteerde verzameling zullen de individuen die rang 1 bezitten (mits dit aantal kleiner is dan N), overgezet worden naar de verzameling P_{t+1}. Deze laatste stap is zeer belangrijk want deze zorgt ervoor dat alle goede oplossingen (individuen) behouden blijven, doordat zowel ouders als kinderen in P_{t+1} gezet kunnen worden zolang ze maar de beste oplossingen bevatten.

Mocht het aantal individuen met rang één kleiner blijken te zijn dan N, dan zal de verzameling worden aangevuld met individuen met een hogere rang (beginnend bij rang 2) net zolang totdat er N individuen geselecteerd zijn. Het zal bijna nooit voorkomen dat alle individuen, die in het laatste front zitten dat voor de selectie in aanmerking komt, geselecteerd kunnen worden (omdat het aantal geselecteerde individuen exact N moet zijn), daarom zal bij het laatste front dat toegevoegd wordt gekeken moeten worden welke individuen het beste zijn van dat desbetreffende front. Dit gebeurd door middel van de crowding afstand; alleen die individuen met de grootste crowding afstand zullen hier geselecteerd worden.



Figuur 3.7: NSGA2 [16]

3.6 Conclusie

Door het nabootsen van de biologische evolutie, waar de algoritmen behorend tot de evolutionary computing op gebaseerd zijn, is het mogelijk een set van verschillende oplossingen te vinden voor een multi-objective optimalisatie probleem. Met name NSGA2 lijkt een zeer geschikt en efficiënt algoritme te zijn, doordat hierbij alle individuen met elkaar worden vergeleken zodat de allerbeste individuen doorgegeven kunnen worden naar de volgende generatie waardoor een geconvergeerd Pareto front behaald kan worden en door de crossover methode, de mutatie methode en de crowding afstand die voor de spreiding zullen zorgen op een Pareto front.

4. Multi-objective optimalisatie tools

In de afgelopen jaren zijn er naast de vele algoritmen ook optimalisatie tools ontwikkeld met als doel het beter en makkelijker kunnen benaderen van een optimale oplossing voor een optimalisatieprobleem, dit mede omdat het oplossen van een optimalisatie vraagstuk vaak een moeilijk probleem blijkt te zijn.
De meeste algoritmen en tools zijn ontwikkeld voor het oplossen van single-objective optimalisatie problemen, op het gebied van multi-objective optimalisatie problemen zijn er veel minder tools ontwikkeld. Desondanks is er op het NLR een aantal tools aanwezig die ter ondersteuning kunnen dienen voor het vinden van oplossingen voor multi-objective optimalisatie problemen. Zo heeft het NLR de tools Multi-Objective Parameter Synthesis (MOPS) en Multi-objective Evolutionary Algorithm (MOEA) in handen. De algoritmen die in deze tools gebruikt worden, zijn algoritmen met bepaalde zoekcapaciteiten, dat wil zeggen dat de algoritmen dienen te zoeken naar de beslissingvariabelen die de oplossing van één of meerdere doelfuncties naar een zo optimaal mogelijk punt, of punten, weten te brengen. In hoeverre de optimale punten bereikt worden in de tools MOPS en MOEA, zal in dit hoofdstuk getoond worden aan de hand van een testfunctie. Behalve het presteren van de algoritmen waar de twee tools gebruik van maken, zal er ook een vergelijking tussen MOPS en MOEA gemaakt worden, waaruit tevens de positieve en negatieve kanten van de tools naar voren zullen komen.

4.1 De testfunctie

De testfunctie die gebruikt zal worden om de prestaties van MOPS en MOEA mee te bespreken is een multi-objective optimalisatie probleem, bestaande uit twee doelfuncties, twee bijvoorwaarden en twee beslissingsvariabelen:

min $f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2,$

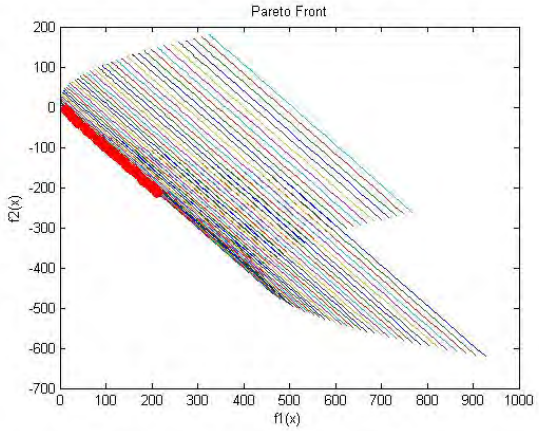
$f_2(x) = 9x_1 - (x_2 - 1)^2,$

onder $g_1(x) = x_1^2 + x_2^2 \le 225,$

$g_2(x) = x_1 - 3x_2 \le -10,$

$x_1, x_2 \in [-20,20].$

In figuur 4.1 wordt de feasible ruimte weergegeven van deze testfunctie waarbij de x-as voor f_1 staat en de y-as voor f_2. De rode lijn toont het daadwerkelijke optimale Pareto front vormt, dat deze rode lijn niet doorloopt naar beneden komt doordat dan de bijvoorwaarden overschreden zouden worden en punten waar de bijvoorwaarden overschreden worden behoren niet tot het optimale Pareto front.

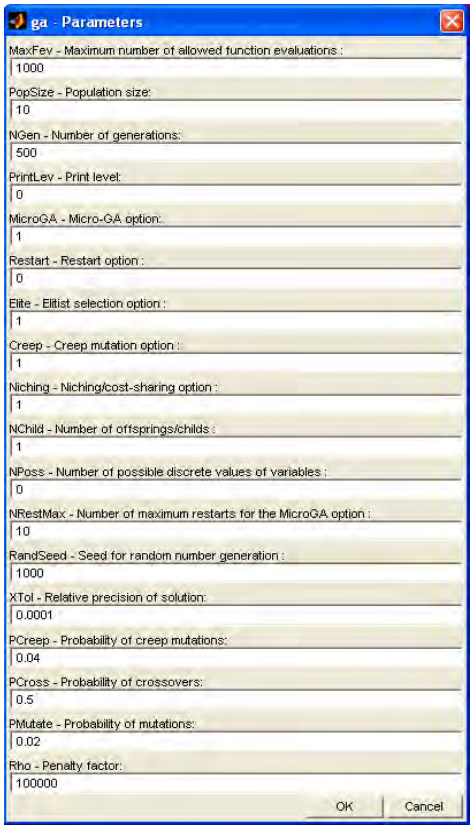


Figuur 4.1: feasible ruimte

4.2 Multi-Objective Parameter Synthesis

MOPS is een multi-objective optimalisatie tool die draait binnen Matlab 6.1. Door het invoeren van de doelfuncties, de bijvoorwaarden en de beslissingsvariabelen in een runscript van Matlab met een specifieke MOPS structuur, kan het multi-objective optimalisatie probleem in MOPS geladen worden. MOPS biedt vervolgens de keuze uit vijf verschillende algoritmen om een oplossing te zoeken voor het optimalisatie probleem: BOUNDS, Pattern Search, Simplex methode, SQP en Genetisch Algoritme. Aangezien dit onderzoek zich specifiek op de evolutionary computing richt, zullen hier alleen de resultaten besproken worden die met het Genetisch Algoritme (GA) worden behaald.

Het GA in MOPS heeft verscheidene instel parameters, zoals afgebeeld in figuur 4.2, die door de gebruiker ingevuld dienen te worden.



Figuur 4.2: instel parameters GA

Behalve deze GA gebonden instel parameters, bevat MOPS ook nog andere instellingen die door de gebruiker ingevuld kunnen worden. Zo kan de gebruiker instellen naar welke specifieke waarden voor de doelfuncties gezocht dient te worden of tussen welke grenzen, om zo de oplossingsruimte te verkleinen. Een andere mogelijkheid die MOPS biedt, is om bepaalde gewichten in te stellen voor de doelfuncties, zodat de gebruiker kan bepalen welke doelfunctie het belangrijkste zal zijn om te optimaliseren. Een instelling die echter altijd ingevuld dient te worden door de gebruiker zijn de defaultwaarden, ofwel de startwaarden, van de inputvariabelen van het optimalisatieprobleem.

Nadat een simulatie van het GA is afgelopen, hebben er bijna altijd meerdere iteraties plaatsgevonden, waarbij iedere iteratie één oplossing geeft. Hoewel de resultaten in een grafiek kunnen worden afgebeeld, kan het resultaat van slechts één iteratie opgeslagen worden door MOPS, zodat maar één gevonden punt van het Pareto front bewaard kan blijven.

4.2.1 Resultaten

Hieronder worden de resultaten getoond die zijn behaald met MOPS voor de hierboven beschreven testfunctie, waarbij de defaultwaarden voor x1 en x2 op nul zijn gezet.

	f1	f2	g1	g2
GA(10)	4.3527	4.2037	1.0908	-2.33
GA(40)	4.3146	4.3144	1.1161	-2.344

Tabel 4.1: resultaten MOPS

Tabel 4.1 toont de resultaten die GA heeft gevonden waarbij het aantal individuen op tien (GA(10)) en op 40 (GA(40)) is gezet. MOPS heeft duidelijk naar een evenwichtig resultaat gezocht tussen de beide doelfuncties, echter is er niet aan de tweede bijvoorwaarde voldaan. Daarom zal de tweede bijvoorwaarde nu als een gelijkheids bijvoorwaarde worden ingesteld (g2=-10) in plaats van een

ongelijkheids bijvoorwaarde. De resultaten hiervan staan weergegeven in tabel 4.2 voor 10 en 20 individuen.

	f1	f2	g1	g2
GA(10)	146.56	-90.82	99.646	-10.01
GA(20)	10.988	9.0194	19.988	-9.993

Tabel 4.2: resultaten MOPS met gelijkheidsbijvoorwaarde

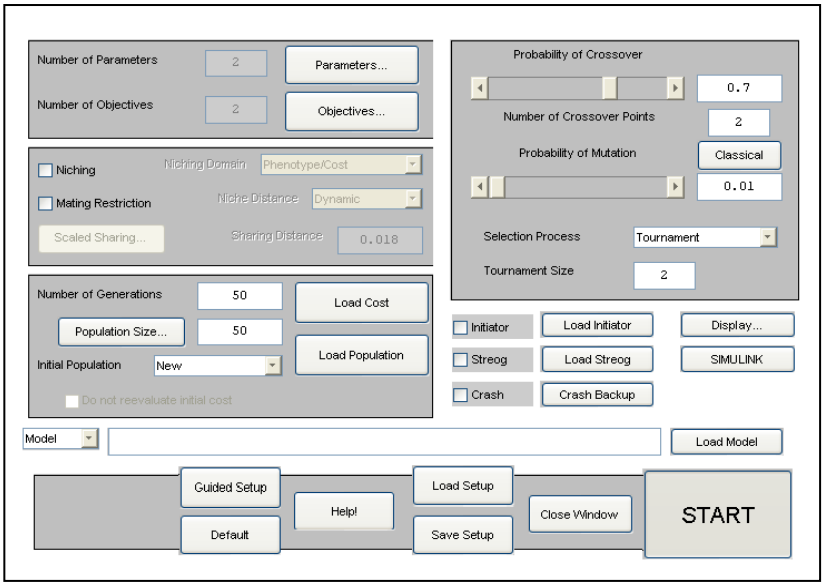
Door deze instellingen zijn er dus resultaten gevonden die veel dichter bij de eisen van de bijvoorwaarden liggen. Echter wel met het gevolg dat de waarden van de doelfuncties minder optimaal zijn.

4.2.2 Conclusie

MOPS heeft laten zien om te kunnen gaan met een multi-objective optimalisatie probleem. Echter door het geven van een enkel gevonden Pareto punt per iteratie wordt er geen geheel Pareto front getoond per iteratie, terwijl het Pareto front toch een belangrijke output is voor een multi-objective optimalisatie probleem. Bovendien bestond er de noodzaak de tweede bijvoorwaarde anders te definiëren voordat MOPS een passende oplossing kon vinden voor dit testprobleem.

4.3 Multi-objective Evolutionary Algorithm

MOEA is ontwikkelt binnen “National University of Singapore” en draait in Matlab 7.0.4. Na het invoeren van de doelfuncties in een runscript van Matlab met een specifieke MOEA structuur, kunnen de benodigde parameters, zoals deze staan afgebeeld in figuur 4.3 voor dit algoritme stapsgewijs ingevuld worden.



Figuur 4.3: MOEA

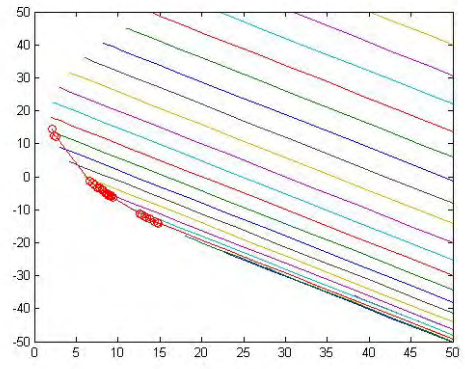
Bij het invullen van deze parameters dienen ook de bijvoorwaarden en de beslissingsvariabelen gedefinieerd te worden en dus niet in het runscript van Matlab zoals bij MOPS het geval is. Echter blijkt het invullen van de bijvoorwaarden niet te werken, waardoor de bijvoorwaarden ook als doelfuncties ingegeven dienen te worden, waar vervolgens wel een bovengrens aan gegeven kan

worden zodat er toch enigszins beperkingen aan de als doelfuncties gedefinieerde bijvoorwaarden gesteld kunnen worden.

In tegenstelling tot MOPS, levert MOEA wel meerdere resultaten (een Pareto front) aan het einde. Het is mogelijk deze resultaten na de simulatie in verschillende figuren te bekijken. Zo kan een grafiek getoond worden waarin twee of drie doelfuncties tegen elkaar zijn geplot, maar ook een grafiek waarin twee of drie beslissingsvariabelen tegen elkaar worden geplot.

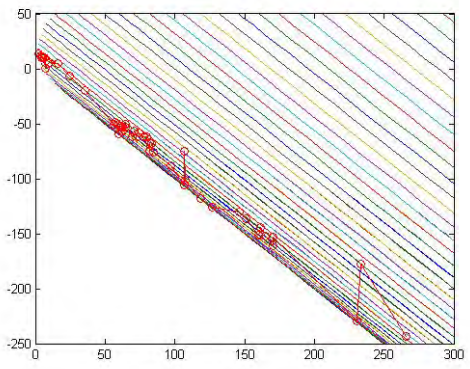
4.3.1 Resultaten

Daar MOEA in staat is een Pareto front als resultaat te bieden, zullen de resultaten van de testfunctie zoals ze zijn behaald door MOEA getoond worden in figuren waarin tevens de feasible ruimte van de testfunctie in staat afgebeeld. Als eerste wordt in figuur 4.4 het resultaat getoond dat verkregen is bij de instel parameters zoals zij door MOEA standaard ingesteld staan.

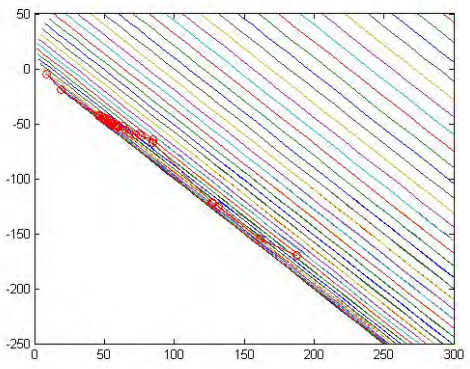


Figuur 4.4: instel parameters standaard

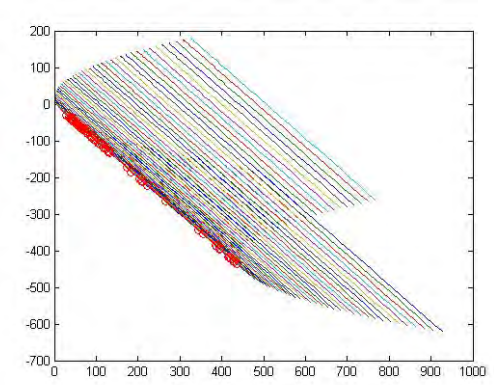
Uit figuur 4.4 blijkt duidelijk dat er een niet al te verspreid Pareto front wordt behaald, daarom wordt er geprobeerd door achtereenvolgens het verhogen van de kans op mutatie, het verhogen van het aantal generaties en het verhogen van zowel het aantal generaties als het aantal individuen, een meer gespreid Pareto front te bereiken.



Figuur 4.5: verhogen kans op mutatie



Figuur 4.6: verhogen aantal generaties



Figuur 4.7: verhogen aantal generaties en aantal individuen

Bij de figuren 4.5 en 4.6 blijkt er nu wel een meer gespreid Pareto front gevonden te zijn, alleen zijn deze Pareto fronten niet meer optimaal geconvergeerd. In figuur 4.7 wordt er een Pareto front getoond dat zowel gespreid is als geconvergeerd en lijkt daarom dus goede oplossingen te bevatten.
Echter is bij alle bovenstaande gepresenteerde Pareto punten, zoals deze zijn behaald door MOEA, de tweede bijvoorwaarde vaak overschreden.

4.3.2 Conclusie

MOEA lijkt goed om te kunnen gaan met multi-objective optimalisatie problemen en biedt bovendien een Pareto front dat uit meerdere oplossingen bestaat. Door het veranderen van de instel parameters van MOEA kan er een ander Pareto front tevoorschijn komen (dit wil zeggen, meer of minder gespreid en/of geconvergeerd). Echter lukt het MOEA niet altijd om aan de gestelde bijvoorwaarden te voldoen, wat ook het gevolg kan zijn van het feit dat de bijvoorwaarden als doelfuncties ingevuld dienen te worden daar de functie waar de bijvoorwaarden eigenlijk gedefinieerd zouden moeten worden in MOEA niet werkt.

4.4 Vergelijking MOPS en MOEA

Beide optimalisatie tools draaien binnen Matlab en vragen om een in Matlab gespecificeerde structuur van het optimalisatieprobleem. Bij het oplossen van een multi-objective optimalisatie probleem, laten beide tools zien in staat te kunnen zijn met een dergelijk probleem om te gaan. Echter laten ook beide tools problemen zien bij het voldoen aan de bijvoorwaarden. Wanneer er niet aan de bijvoorwaarden wordt voldaan, wordt dit ook niet goed aangegeven. MOPS geeft namelijk wel van de eerste bijvoorwaarde die overschreden wordt aan, in welke mate er niet aan de beperking voldaan is, maar voor de overige bijvoorwaarden wordt er door MOPS niet aangegeven hoe ernstig de overschrijdingen van die bijvoorwaarden zijn. Doordat de bijvoorwaarden niet specifiek gedefinieerd kunnen worden in MOEA, wordt er hier helemaal geen maat van overschrijding gegeven en dient dit daarom door de gebruiker zelf uitgezocht te worden.

Een groot nadeel van MOPS is dat er slechts één resultaat gepresenteerd wordt in plaats van een heel Pareto front, hetgeen toch zeer wenselijk is bij een multi-objective optimalisatie probleem. Nu biedt MOPS wel de mogelijkheid aan de gebruiker om een bepaalde richting te geven waar de oplossingen gezocht dienen te worden voor de doelfuncties, maar beter lijkt toch een grotere oplossingsruimte waar alle verschillende oplossingen gepresenteerd kunnen worden.

Verder is gebleken, door de optimalisatieberekeningen zoals deze hierboven zijn beschreven uit te voeren, MOPS altijd dezelfde resultaten weergeeft na iedere run, terwijl MOEA duidelijk gebruik maakt van een random number generator, waardoor de uitkomst iedere keer iets zal verschillen.

5. MNSGA

Uit literatuuronderzoek is gebleken dat het Nondominated Sorting Genetic Algorithm 2 (NSGA2) een efficiënt algoritme voor multi-objective optimalisatie is [8], [15], [16], [18], [24], [25]. In vergelijking met andere algoritmen convergeert NSGA2 goed naar het gezochte multi-objective optimalisatie resultaat: het optimale Pareto front. Om deze reden is besloten tot een implementatie van NSGA2 in Matlab 7.1 op basis van [15]. Deze implementatie zal verder aangeduid worden als MNSGA.

5.1 Beschrijving MNSGA

De stappen die MNSGA doorloopt om tot een Pareto front te komen voor een multi-objective optimalisatie probleem zullen hieronder in het kort toegelicht worden.

5.1.1 Eerste generatie

Alvorens de waarden van de individuen (de waarden van de doelfuncties) van de eerste generatie te kunnen berekenen, dienen eerst de waarden van de genen (de beslissingsvariabelen) bepaald te worden. De waarden van de genen kunnen opgegeven worden door de gebruiker, er kan dus voor gekozen worden om met bepaalde startwaarden te beginnen, of de waarden van de genen worden door middel van een random number generator bepaald, waardoor de waarden van de genen willekeurige startwaarden verkrijgen die tussen de opgegeven ondergrens en bovengrens liggen. De waarden van de individuen worden vervolgens berekend door de waarden van de genen in de doelfuncties te stoppen zoals deze zullen worden gedefinieerd door de gebruiker.
Indien er voor het ingevoerde optimalisatieprobleem ook sprake is van bijvoorwaarden, worden de waarden van de genen tevens gebruikt om de zogenoemde “constraint violation” per individu te berekenen. Deze constraint violation geeft aan in hoeverre de bijvoorwaarden overschreden worden en zal later worden gebruikt voor onder andere de rangbepaling van de individuen.

Behalve de constraint violation, worden ook de waarden voor de doelfuncties die de individuen bezitten gebruikt voor de rangbepaling, waarbij rang 1 gegeven wordt aan de individuen zonder (of met de laagste) constraint violation en met de meest optimale waarden voor de doelfuncties.
Nadat alle individuen een rang toebedeeld hebben gekregen, kan de crowding afstand van de individuen per rang bepaald worden, hetgeen op exact dezelfde manier geschiedt als beschreven is in Hoofdstuk 3, paragraaf 5.3. De berekeningen voor de eerste generatie zijn hiermee nu voltooid, waardoor er gestart kan worden met de tweede generatie.

5.1.2 Tweede generatie

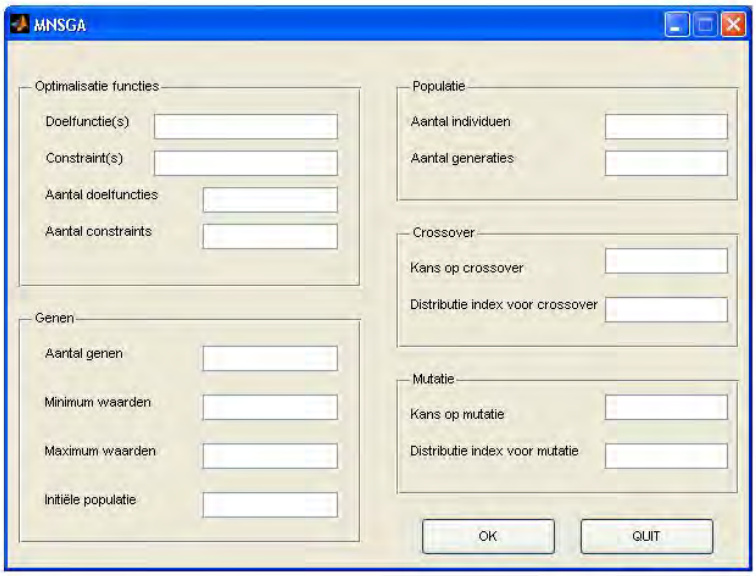
Voor de vorming van de tweede generatie worden de evolutionaire mechanismen selectie, crossover en mutatie uitgevoerd. Als selectiemethode wordt de tournament selectie gebruikt waarbij er steeds twee willekeurige individuen met elkaar vergeleken worden om te bepalen welk individu als ouder zal fungeren. Na de selectiemethode, zullen achtereenvolgens de crossover methode en de mutatie methode uitgevoerd worden, waarbij met een zekere kans, zoals deze door de gebruiker ingevuld zal worden, kinderen gecreëerd worden analoog aan de methoden als beschreven in Hoofdstuk 3 paragraven 5.4 en 5.5.

Door middel van het uitvoeren van de evolutionaire mechanismen is er nu een nieuwe verzameling individuen ontstaan: de kinderen. Deze kinderen zullen worden samengevoegd met de ouders en van al deze individuen tezamen wordt opnieuw de rang en crowding afstand per individu bepaald om daarmee alle individuen op niet-gedomineerde wijze te kunnen sorteren. Uit de gesorteerde verzameling individuen zal nu de beste helft door gaan als tweede generatie, dat wil zeggen de individuen die de laagste rang en de grootste crowding afstand bezitten.

Het proces zoals deze beschreven is voor de tweede generatie, zal nu net zolang herhaald worden tot het aantal generaties dat door de gebruiker ingevuld is, is bereikt.

5.2 De instel parameters

Bij de beschrijving van MNSGA is meerdere keren gemeld dat een methode afhankelijk is van een door de gebruiker ingevulde doelfunctie, kans of getal. Alle instel parameters voor MNSGA zijn weergegeven in figuur 5.1.



Figuur 5.1: GUI van MNSGA

Afgezien van de bijvoorwaarden en de initiële populatie dienen alle instel parameters ingevuld te worden.

5.3 Weergave van de resultaten

Wanneer alle generaties berekend zijn door MNSGA, zullen de resultaten zoals deze in de laatste generatie behaald zijn door Matlab getoond worden. Deze resultaten bevatten de volgende gegevens:

- Waarden van de doelfuncties
- Waarden van de bijvoorwaarden
- Waarden van de beslissingsvariabelen
- Constraint violation
- Rang
- Crowding afstand

Dat alleen de resultaten van de laatste generatie gepresenteerd worden, heeft te maken met het feit dat de beste individuen altijd doorgaan naar een volgende generatie en zich daardoor ook in de laatste generatie bevinden.

5.4 Verificatie

Om te kunnen verifiëren of MNSGA in staat is om het gewenste geconvergeerde en gespreide Pareto front te behalen, is er gebruik gemaakt van de testfunctie zoals deze beschreven is in Hoofdstuk 4 paragraaf 1.
De conclusie zoals deze hieronder beschreven is, is mede gebaseerd op de resultaten van deze testfunctie welke te vinden zijn in Appendix B.

5.5 Conclusie

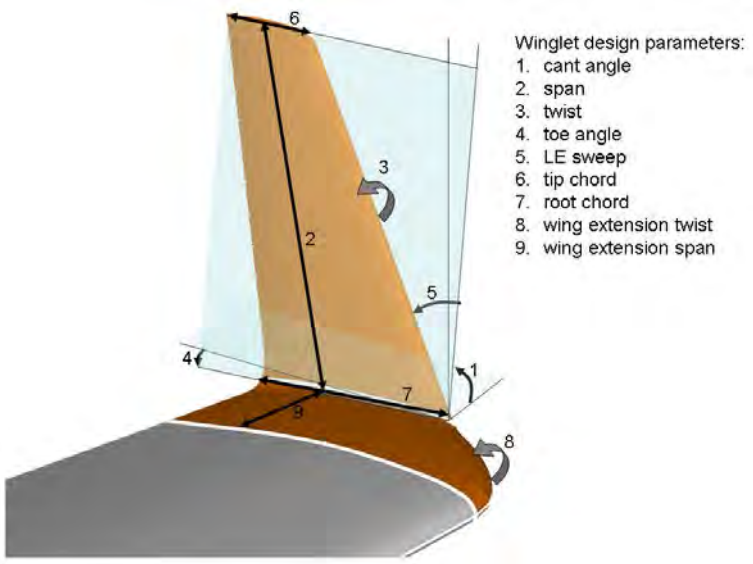
De instel parameters die ingevuld dienen te worden voor MNSGA, geeft de gebruiker veel vrijheid, zoals door de invulling van het aantal individuen en aantal generaties kan de gebruiker zelf aangeven hoe uitgebreid er gezocht dient te worden naar een optimaal Pareto front en daardoor heeft de gebruiker impliciet ook invloed op de rekentijd van MNSGA.

De uitkomst die wordt gegeven door MNSGA is zeer overzichtelijk doordat alleen de laatste generatie gepresenteerd wordt, welke de meest optimale oplossingen bevat zoals deze gevonden zijn door MNSGA. Hierdoor hoeft de gebruiker dus niet de resultaten van verschillende generaties te vergelijken voor de zoektocht naar de beste oplossingen.

Uit de resultaten zoals deze in Appendix B zijn gepresenteerd, blijkt dat MNSGA aan de twee hoofddoelstellingen van multi-objective optimalisatie voldoet door geconvergeerde en gespreide Pareto fronten te behalen in relatief weinig tijd. Bovendien worden de beperkingen die opgelegd worden door de bijvoorwaarden ook keurig nageleefd.

6. Winglet ontwerp optimalisatie studie

MNSGA is toegepast in een praktisch ontwerp optimalisatie studie: geometrische optimalisatie van een gegeneraliseerde vliegtuig-winglet (verticaal gericht uiteinde van een vliegtuigvleugel; i.e. figuur 6.1).



Figuur 6.1: winglet geometrie en ontwerp parameters

Het doel van deze studie is om de winglet afmetingen te vinden waarvoor bepaalde weerstanden en belastingen, waar het vliegtuig mee te maken heeft, zo laag mogelijk worden. De weerstanden en belastingen die optreden voor een bepaalde winglet geometrie kunnen berekend worden met behulp van simulatieprogramma's van ondermeer de luchtstroming rondom het vliegtuig. In deze studie wordt uitgegaan van een dataset van 126 verschillende winglet geometriën met de daarbij behorende weerstanden en belastingen, welke in een eerdere studie berekend zijn.

Hieronder zal worden uitgelegd hoe er vanuit de gegeven dataset, tot de door MNSGA gevonden optimale oplossingen wordt gekomen. De dataset zal hiervoor eerst nader toegelicht worden, waarna er zal worden beschreven hoe geschikte benaderingsfuncties (interpolaties of fits) worden gemaakt voor deze dataset, welke met MNSGA vervolgens geoptimaliseerd kunnen worden. Om aan te tonen dat de resultaten die door MNSGA geleverd zijn, daadwerkelijk goede resultaten bevatten, zal er een aantal van deze resultaten worden vergeleken met andere optimalisatieprogramma's. Voor de betrouwbaarheid van de resultaten zullen de fit-residuen worden geanalyseerd en tot slot zal er van de vele Pareto optimale resultaten die MNSGA levert, gezocht worden naar een klein aantal geselecteerde optimale ontwerppunten voor de winglet.

6.1 De dataset

De dataset waar in deze studie vanuit gegaan wordt, bevat negen onafhankelijke- of inputvariabelen (de winglet afmetingen of ontwerpparameters zoals deze in figuur 1 staan afgebeeld) en drie afhankelijke- of outputvariabelen (de weerstanden en belastingen). Deze drie outputvariabelen, die verder aangeduid worden met respectievelijk y_1, y_2 en y_3, betreffen de weerstand van de winglet in twee verschillende vliegcondities en de toename van buigende belasting in de vliegtuigvleugel ten gevolge van de winglet. Voor een aantal van de 126 verschillende winglet geometriën in de dataset zijn er ongeldige resultaten uit de berekeningen gekomen welke in de dataset zijn opgenomen met 'Not a Number' (NaN) als waarde. Deze geometriën (dat wil zeggen de rijen uit de dataset), zijn uit de dataset gefilterd, waardoor

er 104 rijen met geldige waarden resteren. Met andere woorden, er zijn nu nog 104 combinaties van afmetingen over waarvoor de bijbehorende weerstanden bekend zijn. Verder is er van de dataset bekend dat de laatste vijf à tien rijen (de winglet geometriën) in gunstige ontwerpgebieden liggen, daarom zal er extra aandacht uitgaan naar deze ontwerppunten.

Voor de resterende dataset wordt er, met behulp van het multi-dimensionale fitting programma MultiFit van het NLR, gezocht naar een geschikte benaderingsfunctie.

6.2 MultiFit

MultiFit is een op het NLR in Matlab ontwikkeld programma [23] waarmee relatief eenvoudig met verschillende methoden (zoals multi-dimensionale polynomiale regressie en kriging modellen) benaderingsfuncties gemaakt kunnen worden voor multi-dimensionale datasets. Bovendien biedt MultiFit verschillende mogelijkheden om te bepalen met welke methode de data het best benaderd wordt.

Eén van de mogelijkheden is om de residuen die voorkomen uit een benaderingsmethode te evalueren. Hierbij wordt er een subset van rijen uit de dataset beschouwd als verificatieset en wordt er met de overige punten een fit gemaakt waarmee de outputwaarden in de weggelaten verificatieset voorspeld worden. De residuen (de verschillen tussen de voorspelde waarden voor de outputs en de waarden in de verificatieset) worden vervolgens verwerkt tot een gemiddelde foutmaat: root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}((y_i^* - y_i)^2}{N}} .$$

Hierbij is y de output waarde in de verificatieset, y* de voorspelde waarde voor y zoals deze wordt gevonden door een benaderingsmethode van MultiFit en N het aantal punten in de verificatieset.
Een andere mogelijkheid is de ‘leave-k-out’ methode. Hierin worden er steeds k punten (k dient door de gebruiker ingevuld te worden) uit de dataset gelaten en wordt er voor de overige punten een fit gemaakt waarmee de outputwaarden in de weggelaten k punten voorspeld worden. Dit wordt net zo lang herhaald totdat alle mogelijke k punten geëvalueerd zijn met een fit die gemaakt is op de overige punten. Hierdoor kan er voor alle punten worden berekend in hoeverre het geëvalueerde resultaat verschild met het resultaat zoals deze door de dataset wordt gegeven, hetgeen wordt uitgedrukt in een gemiddelde van de gevonden RMSE waarden.
Verder bestaat er ook nog de mogelijkheid om gebruik te maken van zogenaamde p-fold cross validation. Hierbij worden er p (niet overlappende) subsets gemaakt die als verificatiesets dienen. Ook hiervan kunnen vervolgens de verschillen bekeken worden van de waarden van de verificatiesets en de daadwerkelijke waarden.

Voor de winglet ontwerpstudie zijn de verschillende mogelijkheden van MultiFit uitgeprobeerd om de beste benaderingsfunctie voor de dataset te bepalen. In de eerste plaats is er geprobeerd een globaal beeld te krijgen van de prestaties van de verschillende benaderingsmethoden voor de dataset. Hiervoor is er gebruik gemaakt van de leave-1-out, leave-2-out, leave-3-out en de 10-fold cross validation. Ter illustratie worden de leave-1-out resultaten hieronder gepresenteerd, waarbij de best gevonden waarden voor RMSE in het vet gedrukt staan. De overige resultaten worden in appendix C gegeven.

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,30619	0,3993	0,48783
KriginglG	0,29797	0,3538	0,35206
KrigingqG	0,33735	0,48983	**0,31514**
KrigingcE	0,38071	0,43413	0,72596
KriginglE	0,38452	0,44527	0,65782
KrigingqE	0,31033	0,42335	0,31954
KrigingcC	0,34612	0,26649	0,50484
KriginglC	**0,27287**	**0,23702**	0,31684
KrigingqC	0,34345	0,42044	0,31712
Poly1	0,9438	0,86215	1,2339
Poly2	0,35442	0,4712	0,31541
Poly3	1,4832	1,3124	1,9532

Tabel 6.1: RSME's van de output voor de verschillende benaderingsmethoden bij leave-1-out

Uit de tabel hierboven en Appendix C, blijkt kriginglC de beste fit te leveren. Alvorens een definitieve keuze te maken voor de benaderingsmethode, zal er eerst nog meer in detail naar de dataset worden gekeken met behulp van MultiFit. De resultaten daarvan zijn eveneens in Appendix C opgenomen. De conclusie uit al deze analyses volgt hieronder.

6.2.1 Conclusie

De verschillende mogelijkheden van MultiFit die zijn toegepast op de dataset tonen goede resultaten voor de kriginglC fitmethode. Hoewel kriginglC niet altijd het beste resultaat heeft laten zien, dat wil zeggen de kleinste RMSE waarden, behaalt kriginglC de meest goede resultaten.

Op basis van deze resultaten die met MultiFit zijn behaald, wordt kriginglC geselecteerd als meest geschikte benaderingsmethode voor deze dataset en zal verder worden gebruikt voor alledrie de outputvariabelen in de hieronder beschreven ontwerpoptimalisatie berekeningen.

6.3 MNSGA

Alvorens MNSGA in deze studie te kunnen gebruiken, zal eerst bepaald moeten worden op welke waarden de instel parameters (Aantal individuen, Aantal generaties, Kans op crossover, Distributie index voor crossover, Kans op mutatie, Distributie index voor mutatie) het best gezet kunnen worden. Deze analyse hiervoor en de gevonden resultaten zijn opgenomen in Appendix D.

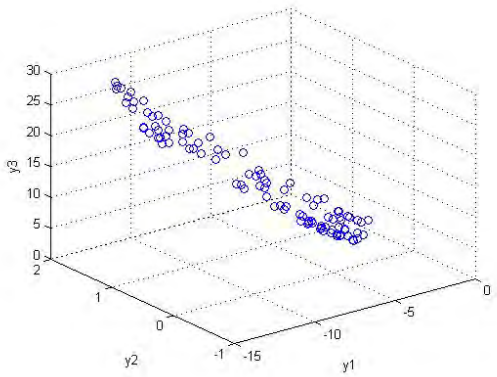
Om er zeker van te zijn dat de gevonden resultaten in een realistisch ontwerpgebied zullen liggen, worden er aan de inputvariabelen (de ontwerpparameters) een ondergrens en een bovengrens meegegeven. Deze grenzen zijn bepaald door aan te nemen dat het maximale zoekgebied voor de inputvariabelen wordt bepaald door de minimale en maximale waarden van de inputvariabelen zoals deze voorkomen in de dataset. Het zoekgebied wordt vervolgens op 70% ingesteld van het maximale zoekgebied, omdat de dataset van 104 punten een zeer ijle representatie is van het negen dimensionale ontwerpdomein waardoor de kans op slechte voorspellingen door de fits sterk toeneemt naarmate een punt meer bij de rand van dit gebied ligt. De ondergrens en bovengrens van de inputvariabelen voor dit toegepaste zoekgebied staat eveneens weergegeven in Appendix D.

Doordat deze winglet studie uit drie outputvariabelen bestaat, die elk een zo klein mogelijke waarde zouden moeten hebben, kan het optimalisatieprobleem op verschillende manieren geformuleerd worden. Ten eerste kan, rechttoe rechtaan, naar het minimum van elk van de drie functies tegelijkertijd gezocht worden met behulp van een multi-objective optimalisatie algoritme zoals MNSGA, waarbij dan de outputs als doelfuncties worden beschouwd. Deze formulering, welke verder aangeduid zal worden als OPT1, zal leiden tot een globaal resultaat, dat wil zeggen dat er een Pareto front gevonden zal worden in een 3D oplossingsruimte met mogelijk vrij grote ranges van de drie doelfuncties.

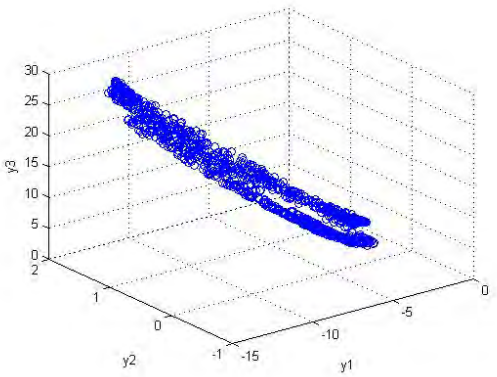
Om specifieker te zoeken kunnen ook één (verder aangeduid als OPT2) of twee doelfuncties (aangeduid als OPT3) als bijvoorwaarden worden voorgeschreven. Een dergelijke probleemformulering kan mogelijk efficiënter opgelost worden en betere resultaten opleveren. Bovendien kan volgens de aanpak voor OPT3 ook gebruik gemaakt worden van single-objective optimalisatie algoritmen zoals GA en fmincon.

6.3.1 Resultaten van OPT1

Hieronder worden de resultaten getoond in 3D zoals zij zijn behaald door alle outputvariabelen, die met krigingIC bepaald worden, als doelfuncties te definiëren. Onder de figuren staat vermeld op welk aantal individuen en generaties de resultaten gebaseerd zijn.

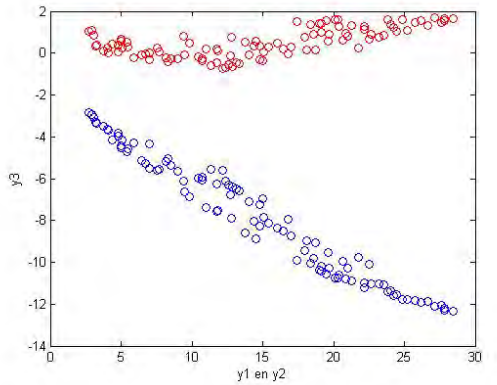


Figuur 6.2: aantal individuen = 100 aantal generaties = 100

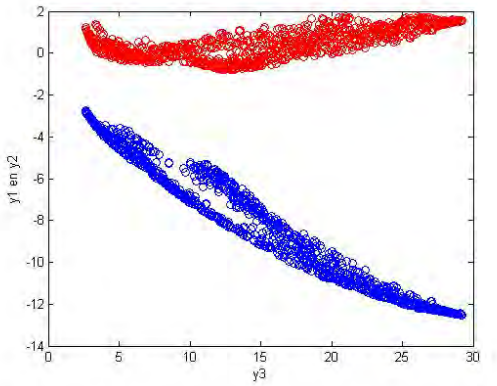


Figuur 6.3: aantal individuen = 1000, aantal generaties = 1000

Ten behoeve van een beter inzicht zullen de gevonden resultaten (Pareto punten) nu ook in een 2D-plot worden weergegeven met y3 op de horizontale as en zowel y1 (blauwe punten) als y2 (rode punten) op de verticale as.



Figuur 6.4: aantal individuen = 100 aantal generaties = 100



Figuur 6.5: aantal individuen = 1000 aantal generaties = 1000

De bovenstaande figuren geven een Pareto front weer in een redelijk grote oplossingsruimte. Er wordt hier dus getoond dat MNSGA in staat is een brede set van oplossingen te geven voor deze winglet ontwerpstudie. Ook voor OPT2 en OPT3 is er getracht een optimaal Pareto front te vinden, deze resultaten zijn opgenomen in Appendix E. De conclusie uit alle gevonden resultaten van MNSGA volgt hieronder.

6.3.2 Conclusie

Voor deze winglet ontwerp optimalisatiestudie is nog niet bekend in welke gebied van de oplossingsruimte er precies gezocht moet worden. Waarschijnlijk daarom levert het resultaat van MNSGA waar alle outputs geminimaliseerd worden de beste resultaten. De resultaten waarbij y2 als bijvoorwaarde wordt gedefinieerd (zie Appendix E) zouden echter ook bruikbaar kunnen zijn, maar de overige resultaten leveren een Pareto front dat onvoldoende gespreid is waardoor niet alle mogelijke oplossingen in beschouwing genomen kunnen worden.

6.4 Genetic Algorithm toolbox en fmincon

De resultaten, zoals deze door MNSGA bereikt zijn (zie Appendix E), zullen worden vergeleken met de resultaten die door de optimalisatiealgoritmen Genetic Algorithm (GA) en fmincon van Matlab behaald kunnen worden. Deze vergelijking heeft als doel er zeker van te kunnen zijn dat de resultaten van MNSGA daadwerkelijk goede Pareto punten zijn.
Echter zijn beide tools (GA en fmincon) single-objective optimalisatie tools, waardoor dus alleen het single-objective – twee constraints (OPT3) probleem opgelost kan worden.

De resultaten, zoals deze zijn opgenomen in Appendix F, laten zien dat beide algoritmen niet altijd in staat blijken te zijn om lagere waarden te vinden voor de doelfunctie dan MNSGA. Bovendien is er vaak sprake van een overschrijding van de bijvoorwaarden, terwijl MNSGA resultaten vindt zonder deze overschrijdingen. Ook vereisen zowel GA als fmincon soms meer rekentijd om tot de resultaten (zoals vermeld in Appendix F) te komen dan MNSGA, hetgeen waarschijnlijk voornamelijk komt door de twee negen dimensionale niet-lineaire constraint functies waaraan moeilijk voldaan kan worden.
Uit deze resultaten blijkt dat GA en fmincon geen wezenlijk betere punten kunnen vinden dan MNSGA. De verdere analyses zullen daarom op de, door MNSGA behaalde, resultaten gebaseerd worden.

6.5 Residuen

De bovenstaande optimalisatie berekeningen zijn allemaal uitgevoerd binnen het hierboven beschreven 70% zoekdomein. Uit de verkregen resultaten blijkt dat veel van de interessante ontwerppunten op de rand van dit domein liggen. Dit wekt de verwachting dat een verdere verruiming van het zoekdomein nog interessantere ontwerppunten zal opleveren. Daarom is besloten om het zoekgebied achtereenvolgens te vergroten naar 80, 90 en 100%, waarvoor de bijbehorende onder- en bovengrenzen opgenomen zijn in tabel D1 in Appendix D. Echter, bij deze ruimere zoekdomeinen zullen de punten aan de randen op grotere afstand van de punten in de dataset liggen. Bovendien is de kans groter dat deze punten buiten de “wolk van datapunten” zullen liggen en dus op basis van ruimtelijke extrapolatie (in plaats van interpolatie) met de benaderingsfuncties voorspeld worden. De nauwkeurigheid van deze voorspellingen zal afnemen naar mate een voorspeld punt verder verwijderd is van de punten in de dataset, in het bijzonder in geval van extrapolatie (in plaats van interpolatie). In de optimalisatieberekeningen in de 80, 90 en 100% zoekdomeinen zal daarom extra aandacht besteed worden aan de nauwkeurigheid van de benaderingsfuncties, hetgeen hieronder eerst toegelicht zal worden.

Een schatting voor de benaderingsfouten (residuen) voor de outputs in elk van de datapunten kan verkregen worden met leave-one-out analyses, zoals in het bovenstaande al beschreven is. Van deze geschatte residuen kan weer een fit gemaakt worden, welke een globale representatie geeft (in de negen dimensionale ontwerpruimte) van de “onnauwkeurigheid” van de benaderingsfuncties van de outputvariabelen. Met MultiFit kan vervolgens relatief eenvoudig geanalyseerd worden welke fitmethode het beste gebruikt kan worden om de residuen te benaderen. Vervolgens kan de beste “fit van de residuen” gebruikt worden om de gevonden interessante ontwerppunten te beoordelen op hun “betrouwbaarheid”. Deze schatting van de fit fout is echter nogal grof en zal alleen in de beoordeling

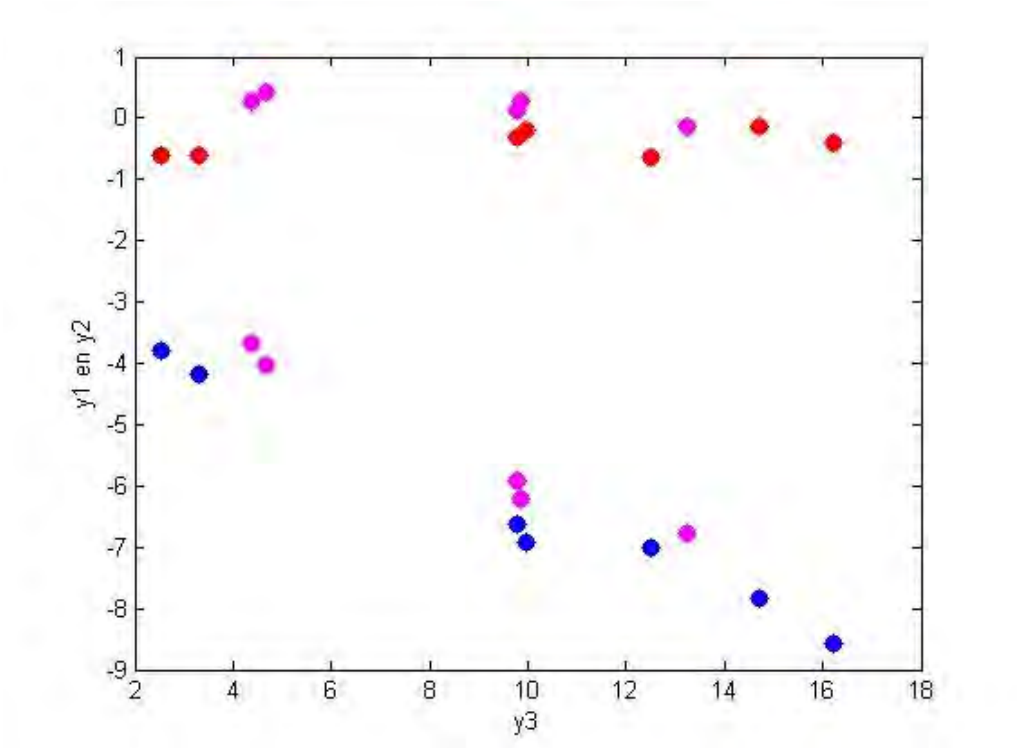
van de gevonden interessante ontwerppunten achteraf meegenomen worden. Een toelichting op hoe deze “fit van de schatting van de fit fout” met behulp van MultiFit bepaald kan worden is gegeven in Appendix G. Uit deze analyses kon geconcludeerd worden dat ook voor de “fit van de schatting van de fit fout” de kriginglC methode de beste resultaten biedt, dus is deze gebruikt in de verdere optimalisatie analyses.

De optimalisaties voor de vergrote zoekdomeinen zijn uitgevoerd met MNSGA met alledrie de outputvariabelen als doelfuncties. De resultaten hiervan zijn te vinden in Appendix H waarin eveneens de schattingen van de fit fouten zijn opgenomen.

6.6 De meest interessante ontwerppunten

Tot nu toe is gebleken dat MNSGA goed in staat is om een Pareto front te vinden voor de winglet ontwerp studie, wanneer de output als doelfuncties wordt ingesteld en wordt benaderd door kriginglC. Door het vergroten van het zoekgebied bleek er een nog breder Pareto front gevonden te kunnen worden waarvoor tevens de schatting van de fit fouten bepaald kan worden. Met deze kennis en uitgaande van de resultaten die tot nu toe behaald zijn, zal getracht worden om een aantal goede en betrouwbare ontwerppunten te vinden voor de winglet studie.

Uit de analyse om tot de meest belovende ontwerppunten te komen zoals deze is opgenomen in Appendix I, zijn de zeven meest interessante punten die gevonden zijn door MNSGA hieronder weergegeven in figuur 6.6 en in tabel 6.2. In figuur 6.6 zijn er tevens extra punten toegevoegd (de roze punten), dit zijn de laatste vijf punten van de gegeven dataset om zo een duidelijk beeld te krijgen in hoeverre de nieuwe punten (de rode en blauwe punten), zoals deze door MNSGA gevonden zijn, geoptimaliseerd zijn.



Figuur 6.6: De 7 meest belovende ontwerppunten

X1	X2	X3	X4	X5	X6	X7	X8	X9	Y1	Y2	Y3
292,71	-1,534	600,02	200,07	39,994	-5,956	3,937	1687,2	86,443	-3,787	-0,603	2,536
338,45	-0,786	616,33	211,91	39,894	-5,96	3,795	1612,2	86,429	-4,191	-0,596	3,296
405,22	-0,712	699,51	230,03	36,921	-3,014	-0,040	1845,9	76,379	-6,915	-0,185	9,960
383,31	-0,713	690,54	230,92	33,416	-3,138	-0,110	1845,6	77,918	-6,627	-0,308	9,765
412,30	-3,425	798,48	200,01	27,828	-0,629	-0,323	1000,3	30,088	-7,004	-0,625	12,450
423,72	-0,832	880,64	238,60	24,274	-1,757	-0,012	1151,6	39,810	-7,849	-0,145	14,725
586,46	-0,588	849,14	200,06	27,054	-0,580	-0,232	1002,0	30,531	-8,568	-0,396	16,222

Tabel 6.2: De 7 meest belovende ontwerppunten

6.7 Conclusie

Door gebruik te maken van de benaderingsmethode kriginglC, kan de output van de winglet studie geoptimaliseerd worden door MNSGA. Het Pareto front dat MNSGA hierbij levert lijkt zeer interessante ontwerppunten te bevatten, waarvan de betrouwbaarheid door middel van de fit fouten bekeken kan worden.

7. Conclusie

Een optimalisatie probleem waarbij meerdere doelstellingen tegelijkertijd beschouwd dienen te worden, staat bekend als een multi-objective optimalisatie probleem. Voor het oplossen van multi-objective optimalisatie problemen is er een algoritme nodig dat in staat is een set van goede benaderingen te geven van het optimale Pareto front. Zoals blijkt uit de literatuur en is beschreven in hoofdstuk 3, lijken de algoritmen die behoren tot de evolutionary computing hier zeer geschikt voor te zijn. Dit komt onder meer door de zoekkracht (bereikt door de evolutionaire mechanismen) die deze algoritmen bezitten en door het kunnen leveren van meerdere oplossingen tegelijk (alle individuen in een generatie leveren een mogelijke oplossing).

Het Nondominated Sorting Genetic Algorithm 2 (NSGA2) blijkt een zeer efficiënt evolutionary computing algoritme te zijn, hetgeen gebleken is uit literatuurstudies en testberekeningen en uit vergelijkingen met resultaten van andere algoritmen, zoals beschreven is in de hoofdstukken 3 en 4 van dit verslag. Het zeer geconvergeerde Pareto front dat NSGA2 behaalt, wordt bereikt door alle individuen binnen een generatie altijd met elkaar te vergelijken en de beste individuen hiervan altijd door te geven aan een volgende generatie. Daarnaast is NSGA2 ook in staat om een zeer gespreid Pareto front te behalen door de crossover methode, de mutatie methode en de crowding afstand waar NSGA2 gebruik van maakt.

Zoals beschreven is in hoofdstuk 5, is MNSGA het resultaat van een succesvolle implementatie van NSGA2 in Matlab. Daar waar andere optimalisatie tools in het algemeen iets meer moeite mee bleken te hebben, worden de twee hoofddoelstellingen van multi-objective optimalisatie; het behalen van een geconvergeerd en gespreid Pareto front, door MNSGA in aanzienlijk weinig rekentijd behaald. Bovendien kan MNSGA ook goed met bijvoorwaarden overweg en levert resultaten die aan alle bijvoorwaarden voldoen.

Om de kwaliteit en efficiency van MNSGA te meten voor een complex ontwerpprobleem, is getracht om met MNSGA een set van oplossingen te vinden voor een winglet ontwerp studie met negen vrije ontwerpparameters en drie ontwerpdoelstellingen, zoals in hoofdstuk 6 van dit verslag beschreven is. Voor de bepaling van de doelfunctie waarden is er gebruik gemaakt van een benaderingsmethode van MultiFit, waaruit is gebleken dat MNSGA ook in staat is een efficiënte en flexibele combinatie te vormen met MultiFit. Behalve de ontwerpdoelstellingen als doelfuncties te definiëren, zijn de ontwerpdoelstelling ook op verschillende manieren als bijvoorwaarden gesteld, waarvan de waarden ook weer door MultiFit gegeven worden. Door deze verschillende variaties in de definities van doelfuncties en bijvoorwaarden is gebleken dat MNSGA verschillende optimale resultaten kan leveren voor de doelfuncties en ondertussen ook aan de bijvoorwaarden kan voldoen. Door de verschillende variaties voor de ontwerpdoelstellingen en ook door de variaties in de instel parameters van MNSGA, heeft MNSGA een enorme set van mogelijke oplossingen weten te verzamelen voor dit ontwerpprobleem. Wanneer er tot slot een kleine selectie wordt gemaakt uit de enorme verzameling resultaten die MNSGA geleverd heeft, blijkt dat de combinatie van MNSGA en MultiFit heeft geleid tot verbeteringen in de ontwerp doelfuncties van globaal ongeveer 5%.

Appendix A

In deze Appendix worden de processen van de evolutionaire mechanismen uitgebreid beschreven en worden er enkele voorbeelden gegeven van de verschillende methoden die in de loop der tijd voor deze mechanismen ontwikkeld zijn.

Selectie
Bij een selectie worden de ouders in de meeste algoritmen geselecteerd volgens één van de volgende methoden [20]:

- Roulette-wheel selectie
- Stochastic universal sampling
- Local selectie
- Truncation selectie
- Tournament selectie

Bij de roulette-wheel selectie is de kans dat een individu wordt geselecteerd als een ouder gerelateerd aan de fitness of de rang van het individu, dus des te meer fitness of des te beter de rang, des te meer kans dat dit individu gekozen zal worden als ouder.

De stochastic universal sampling verdeelt alle individuen over een lijn, hierbij krijgt een individu meer afstand tot het volgende individu naarmate zijn fitness groter is. De lijn wordt vervolgens opgedeeld in een bepaald aantal gelijke stukken (dit aantal kan door de gebruiker zelf bepaald worden) en van ieder stuk lijn wordt dan het eerste individu gekozen.

Stochastic universal sampling kan ook weer gebruikt worden binnen de local selectiemethode. Bij deze methode vormt ieder individu met de individuen die binnen een bepaalde afstand van hem liggen een ‘local neighbourhood’, hierbinnen wordt een individu geselecteerd met behulp van een andere selectiemethode, zoals dus bijvoorbeeld stochastic universal sampling. Crossover en mutatie zullen hierbij alleen plaats vinden tussen individuen uit dezelfde ‘neighbourhood’.

De truncation selectie wordt vooral gebruikt bij grote populaties. Hierbij wordt de populatie gesorteerd in afnemende mate van fitness, om vervolgens van de gesorteerde rij de eerste ‘trunc’ individuen ervan af te halen. Trunc is hierbij het aantal individuen dat gebruikt zal worden voor de crossover methode en de mutatie methode.

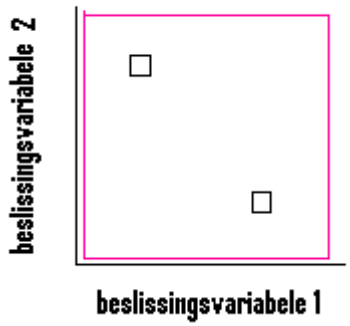
Bij de tournament selectie worden er steeds twee of meer (dit aantal zal worden bepaald door de gebruiker) individuen met elkaar vergeleken. Het beste individu van alle met elkaar vergeleken individuen wordt hier nu gekozen als ouder.

Crossover
Net als bij de selectie, kan ook de crossover volgens verschillende methoden plaatsvinden [20]:

- Intermediate crossover
- Line crossover
- Extended line crossover
- Single-point crossover
- Double-point crossover
- Multi-point crossover

Bij de eerste drie crossover methoden wordt een percentage bepaald in hoeverre de genen van de kinderen mogen afwijken van die van hen ouders, dus als bijvoorbeeld alle genen (dat wil zeggen alle beslissingsvariabelen) van de ouders de waarde vier hebben en het percentage dat de waarden voor deze genen mag afwijken op tien procent is gesteld, dan zullen de genen van het kind dus een waarde tussen 3,6 en 4,4 moeten aannemen. Door het vaststellen van een afwijkingspercentage is er een gebied ontstaan met waarden die de genen mogen aannemen voor de kinderen. Bij de eerste methode bestaat

dit gebied uit een vlak en bij de laatste twee methoden uit een lijn, beide gebieden zijn aangegeven met een rode lijn in de onderstaande figuren.



Figuur A1: Voorbeeld van een intermediate crossover gebied

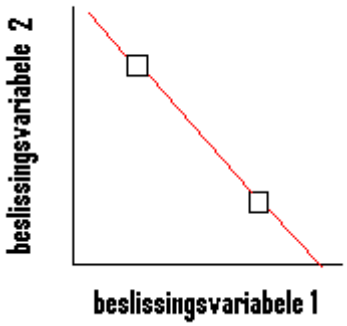
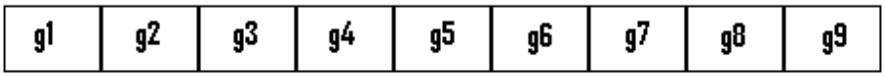


Fig. A2: Voorbeeld van een line crossover gebied

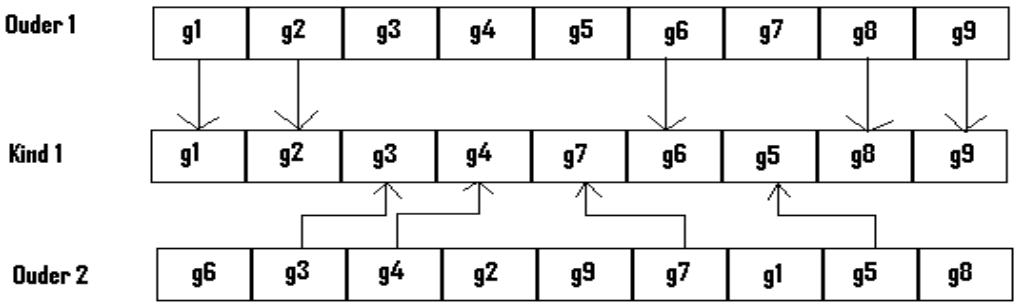
Bij evolutionaire en genetische algoritmen wordt er echter vooral gebruik gemaakt van de drie laatst genoemde crossover methoden. Bij deze methoden wordt de verzameling van genen van de ouders op één, twee of meerdere punten gesplitst om kinderen te creëren die de eigenschappen van meerdere ouders kunnen overnemen. Voor de verdere uitleg van de crossover procedure, zal er echter van uitgegaan worden dat twee ouders twee kinderen zullen voortbrengen.

Alle genen bij elkaar vormen het chromosoom van het individu. Het chromosoom kan als een array of een vector gepresenteerd worden, met op de indexplekken de genen van het individu:



Figuur A3: chromosoom van een individu met op de indexplekken de genen van het individu

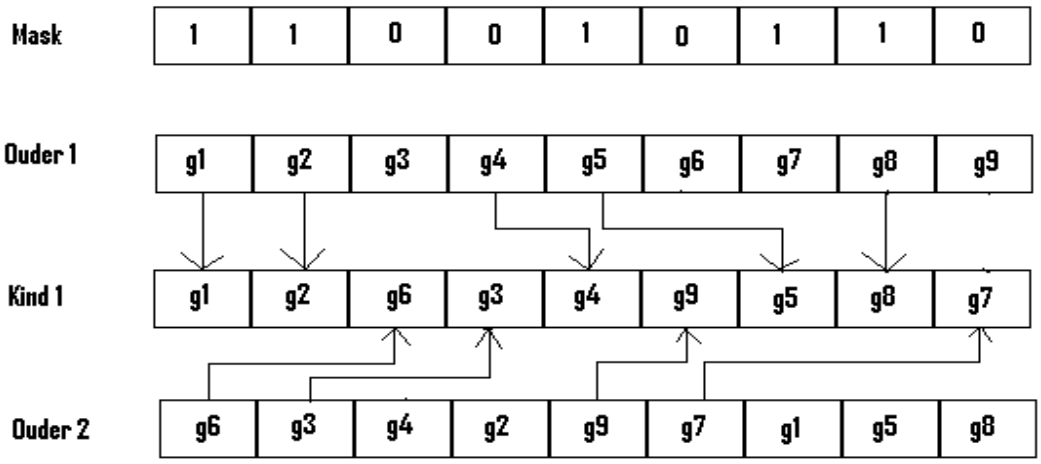
Een veel gebruikte operator om een dergelijke crossover methode uit te voeren is de OBX operator. Hierbij wordt een aantal genen en de bijbehorende indexplekken van de eerste ouder gereserveerd voor het eerste kind en worden de overige genen van de tweede ouder overgenomen.



Figuur A4: OBX operator voor negen genen

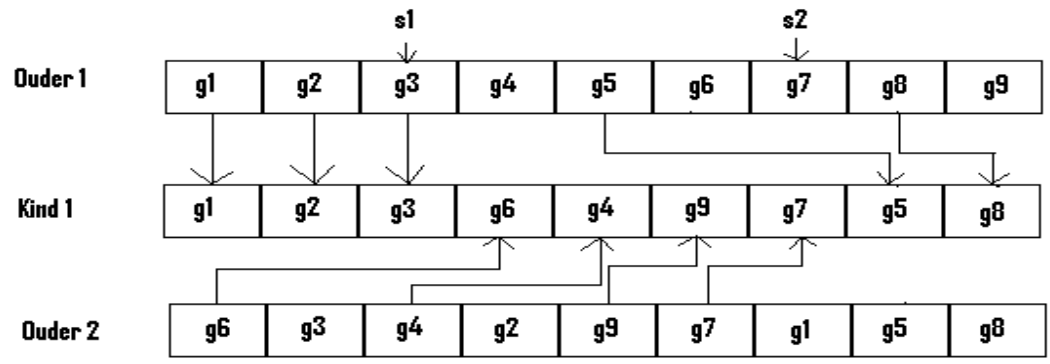
Alle genen die nu niet gebruikt zijn van beide ouders, zullen vervolgens het tweede kind vormen.

Twee andere operatoren om crossover mee uit te voeren, zijn de Precedence Preservative Crossover (PPX) en de One Segment Crossover (OSX).
Bij de PPX is een subset van de genen van de eerste ouder gereserveerd voor het eerste kind, alleen wordt er met een 'mask' bepaald op welke indexplekken deze genen terecht zullen komen in het kind. De overige indexplekken worden uiteraard opgevuld door de genen van de tweede ouder.



Figuur A5: PPX operator voor negen genen

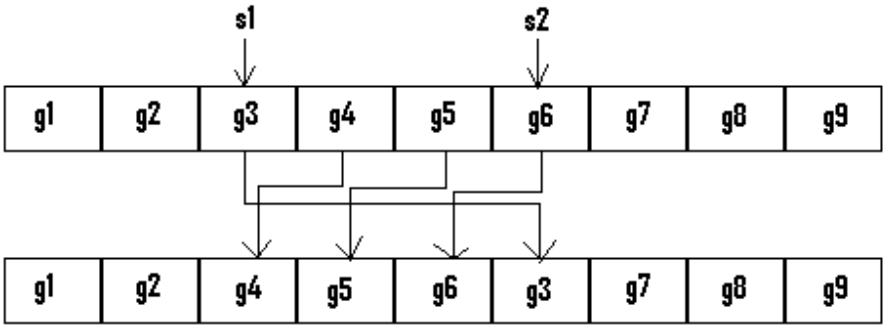
Bij de OSX wordt eigenlijk alleen het principe van double-point crossover uitgevoerd. Hier worden namelijk twee willekeurige punten geselecteerd, waarna de genen van het punt index één tot en met het eerste geselecteerde punt van de eerste ouder komen. Vervolgens worden de indexplaatsen van het kind van het indexpunt dat zich na het eerste geselecteerde punt bevindt, tot en met het tweede geselecteerde punt gevuld met waarden van de genen van de tweede ouder. Echter worden slechts die genen overgebracht die het kind nog niet had meegekregen van de eerste ouder. Ten slotte worden de overige indexpunten van het kind opgevuld met de waarden van de genen van de eerste ouder die nog niet aan het kind gegeven waren.



Figuur A6: OSX operator voor negen genen

Mutatie
Ook bij het mutatie proces bestaan er verschillende operatoren, zoals insert, swap en switch [6].

Bij insert worden er twee indexen willekeurig geselecteerd, de index die de laagste waarde bevat gaat hierbij naar de index die de hoogste waarde bevat en alle tussenliggende indexen schuiven één plekje op in de richting waar voor het begin van de mutatie de index met de laagste waarde stond.



Figuur A7: Insert operator voor negen genen

De operator die swap uitvoert, selecteert twee willekeurige genen en verwisselt deze.
Bij switch wordt er maar een enkele index geselecteerd waarna het gen van die index wordt verwisseld met het gen van de daaropvolgende index, indien bij deze operator de laatste index wordt geselecteerd, dan zal het gen van die index worden verwisseld met het gen van de eerste index.

Appendix B

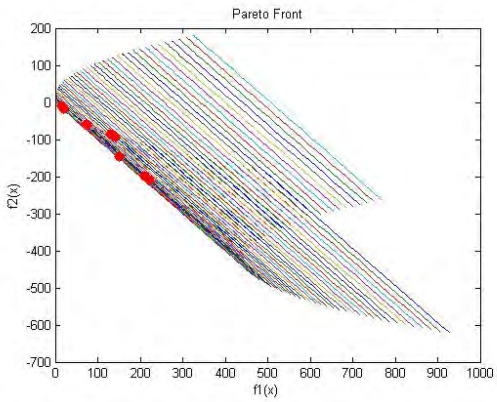
Hieronder worden de resultaten gepresenteerd die behaald zijn door MNSGA voor de testfunctie zoals deze beschreven is in Hoofdstuk 4 paragraaf 1.

De testfunctie zal in eerste instantie getest worden met waarden voor de instel parameters zoals deze staan vermeld in tabel B1. Deze instel parameters zullen vervolgens gevarieerd worden (als in de derde kolom van tabel B1), om zo een vollediger beeld van de mogelijke resultaten van MNSGA te verkrijgen.

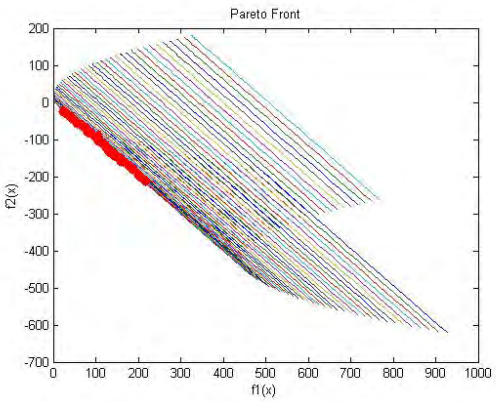
Instel parameter	Begin instellingen	Gevarieerde instellingen
Populatiegrootte	10	40, 100
Aantal generaties	10	50
Mutatie distribution	5	20
Crossover distribution	5	20
Kans op mutatie	0.01	0.5
Kans op crossover	0.7	0.9

Tabel B1: defaultwaarden van de instel parameters

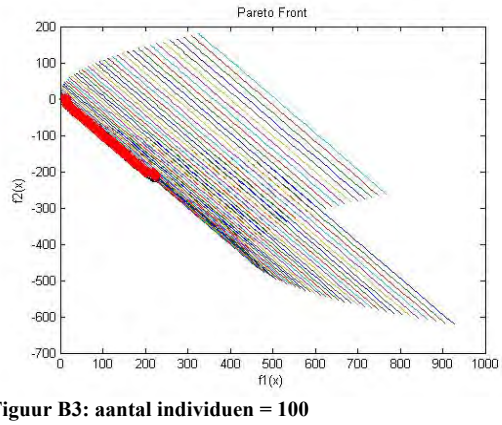
De resultaten voor de verschillende waarden van de instel parameters zullen hieronder getoond worden door middel van grafieken waarbij de twee doelfuncties tegen elkaar geplot zijn in een figuur waarin tevens de feasible ruimte (zonder de inachtneming van de bijvoorwaarden) getoond wordt van de testfunctie. Onder de figuren zal steeds vermeld staand welke instel parameter veranderd is, de overige instel parameters zullen hierbij altijd op de begin instellingen staan.



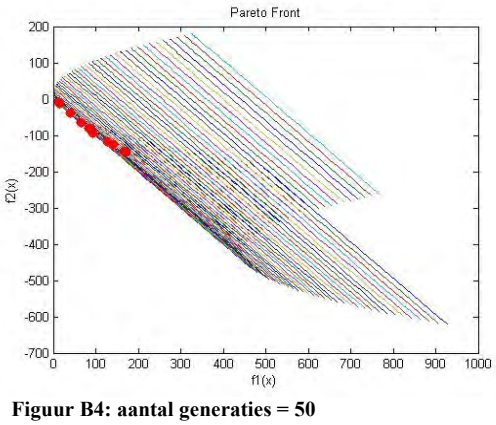
Figuur B1: begin instellingen



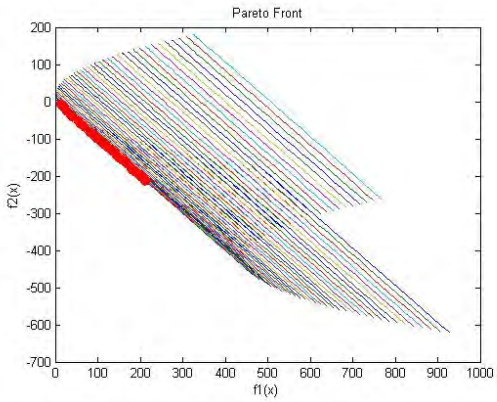
Figuur B2: aantal individuen = 40



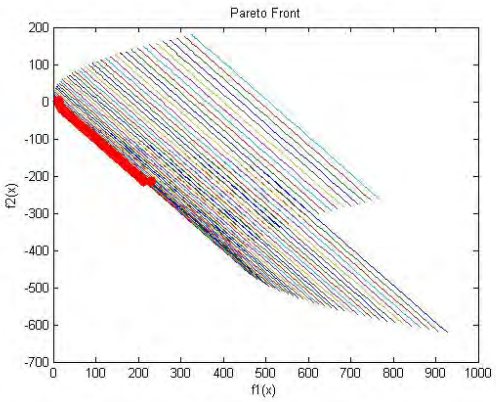
Figuur B3: aantal individuen = 100



Figuur B4: aantal generaties = 50

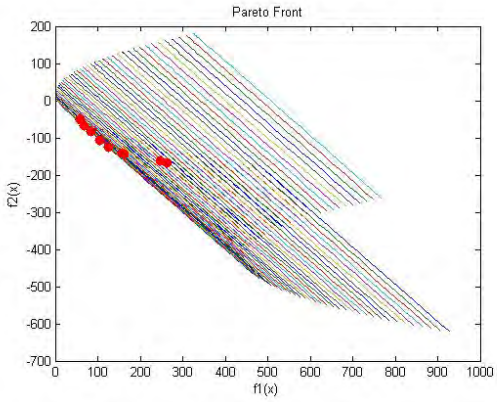


Figuur B5: aantal individuen = 40
aantal generaties = 50

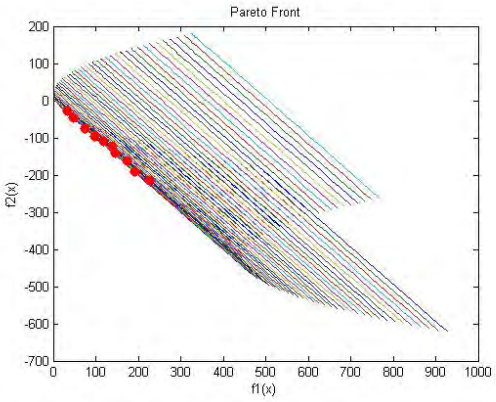


Figuur B6: aantal individuen = 100
aantal generaties = 50

Het verhogen van het aantal individuen en het aantal generaties leidt duidelijk tot een optimaler Pareto front.

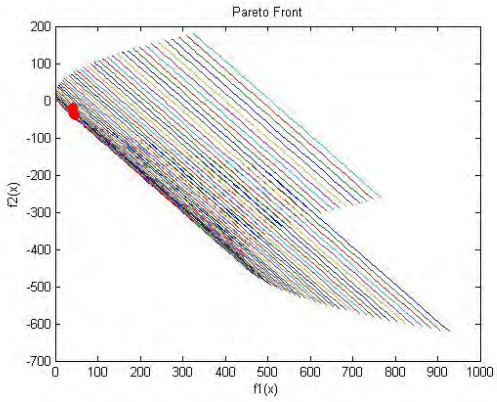


Figuur B7: kans op crossover = 0.9

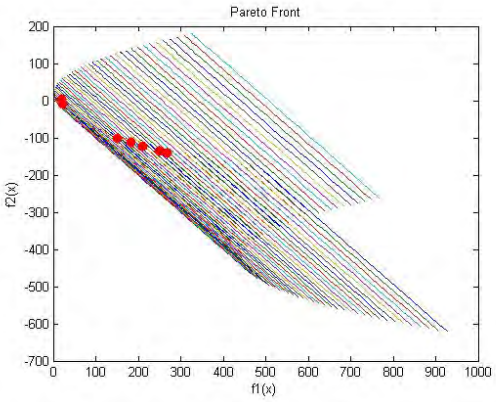


Figuur B8: kans op mutatie = 0.5

Door de kans op mutatie te verhogen blijkt er een meer gespreid Pareto front behaald te kunnen worden, terwijl de verhoging van de kans op crossover voor deze functie tot een mindere spreiding leidt.



Figuur B9: distributie index crossover = 20



Figuur B10: distributie index mutatie = 20

De verhoging van de distributie index voor zowel crossover als mutatie (waardoor de waarden van de genen van de kinderen dichter bij de waarden van de ouders gezocht worden) leidt tot een mindere spreiding op het Pareto front.

De rekentijden die nodig waren voor het behalen van de Pareto fronten, worden hieronder in tabel B2 gepresenteerd.

Variatie instel parameter	rekentijd
Begin instellingen	2.3 seconden
Aantal individuen = 40	3.7 seconden
Aantal individuen = 100	8.4 seconden
Aantal generaties = 50	5.8 seconden
Aantal individuen = 40, aantal generaties = 50	12.1 seconden
Aantal individuen = 100, aantal generaties = 50	43.4 seconden
Kans op crossover = 0.9	2.5 seconden
Kans op mutatie = 0.5	2.6 seconden
Distributie index crossover = 20	2.9 seconden
Distributie index mutatie = 20	2.5 seconden

Tabel B2: benodigde rekentijden bij verschillende variaties van de instel parameters

Appendix C

Deze appendix bevat de resultaten zoals die zijn behaald met MultiFit voor de verschillende benaderingsmethoden.

Hieronder volgen in eerste instantie de resultaten voor de leave-2-out, leave-3-out en de 10-fold cross validation, welke gebruikt zijn om een globale indruk te krijgen van de prestaties van de verschillende benaderingsmethoden.

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,36899	0,48459	0,50833
KriginglG	0,33177	0,38454	0,38187
KrigingqG	0,37346	0,55142	0,34728
KrigingcE	0,40686	0,43204	0,88346
KriginglE	0,46464	0,46408	0,69775
KrigingqE	0,3441	0,50837	0,32771
KrigingcC	0,38967	0,29291	0,6072
KriginglC	0,30348	0,23725	0,37255
KrigingqC	0,40289	0,47865	0,35977
Poly1	1,1819	1,0778	1,4104
Poly2	0,41652	0,52837	0,34077
Poly3	1,7885	1,4069	2,2782

Tabel C1: RSME's van de output voor de verschillende benaderingsmethoden bij leave-2-out

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,36612	0,45668	0,59382
KriginglG	0,37112	0,35362	0,39412
KrigingqG	0,41236	0,57226	0,39755
KrigingcE	0,46899	0,47441	0,82771
KriginglE	0,44593	0,47622	0,78307
KrigingqE	0,38251	0,54825	0,40041
KrigingcC	0,38189	0,31994	0,59976
KriginglC	0,33559	0,28568	0,3436
KrigingqC	0,41519	0,49855	0,42462
Poly1	1,1662	1,1212	1,4733
Poly2	0,41015	0,58552	0,39607
Poly3	1,7711	1,5514	2,6754

Tabel C2: RSME's van de output voor de verschillende benaderingsmethoden bij leave-3-out

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,41802	0,75811	0,64875
KriginglG	0,37595	0,48544	0,49443
KrigingqG	0,42463	0,73973	0,46676
KrigingcE	0,5382	0,79346	1,1107
KriginglE	0,57815	0,79666	1,0435
KrigingqE	0,53676	0,75276	0,53467
KrigingcC	0,41344	0,54386	0,63799
KriginglC	0,37455	0,55019	0,45499
KrigingqC	0,5158	0,79016	0,5317
Poly1	1,3578	1,3783	1,6356
Poly2	0,50034	0,77819	0,4872
Poly3	6,1423	4,4003	3,4916

Tabel C3: RSME's van de output voor de verschillende benaderingsmethoden bij p-fold-10 cross validation

Met name de methode kriginglC lijkt goede resultaten te boeken.

De tabellen die nu volgen, geven de resultaten weer van de verschillende benaderingsmethoden wanneer er in meer detail naar de dataset gekeken wordt.
In eerste instantie zal er gekeken worden naar welke waarden voor de RSME's er behaald kunnen worden met de benaderingsmethoden, wanneer er een willekeurige set van verificatie punten uit de dataset wordt gehaald. Dit wil zeggen dat er een fit wordt gemaakt met een benaderingsmethode op basis van de dataset waarbij er één of meerdere punten ontbreken. De RMSE kan dan vervolgens worden berekend door de ontbrekende punten van de dataset met behulp van de fit te evalueren en deze te vergelijken met de daadwerkelijke waarden. In de tabellen C4 tot en met C6, zijn de waarden van de RSME's weergegeven die zijn bereikt wanneer er respectievelijk één, drie en vijf willekeurige punten uit de dataset zijn weggelaten voor de fit.

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,09	0,0319	0
KriginglG	0,366	0,0793	0,29
KrigingqG	0,169	0,2597	0,59
KrigingcE	0,035	0,1422	0,31
KriginglE	0,147	0,2849	0,59
KrigingqE	0,114	0,1373	0,64
KrigingcC	0,424	0,1615	0,17
KriginglC	0,12	0,0876	0,53
KrigingqC	0,173	0,0406	0,61
Poly1	0,41	0,2336	1,59
Poly2	0,05	0,0258	0,66
Poly3	1,476	1,183	1,33

Tabel C4: RSME's gevonden door de benaderingsmethoden bij de verificatieset van 1 punt (rij 22 van de dataset)

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,418825341	0,568049883	0,486477817
KriginglG	0,546568995	0,411352489	0,315648539
KrigingqG	0,294816327	0,560700274	0,282938156
KrigingcE	0,322131443	0,440924786	0,41412639
KriginglE	0,354355753	0,409219175	0,39584593
KrigingqE	0,208234643	0,483139759	0,270617812
KrigingcC	0,036013886	0,18902014	0,515209343
KriginglC	0,154894588	0,206562146	0,259230914
KrigingqC	0,25622386	0,501225366	0,277634052
Poly1	0,934988235	0,966966907	0,48653263
Poly2	0,247797632	0,430392085	1,088013787
Poly3	2,303475779	2,71399128	5,71374419

Tabel C5: RSME's gevonden door de benaderingsmethoden bij de verificatieset van 3 punten (rijen 8, 45 en 101 van de dataset)

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,350686185	0,18421301	0,84354336
KriginglG	0,427566603	0,110270395	0,462178104
KrigingqG	0,602595884	0,914675175	0,374408066
KrigingcE	0,532483239	0,48664642	0,839530226
KriginglE	0,479787453	0,565552086	1,412096145
KrigingqE	0,844304921	0,788311416	0,328089927
KrigingcC	0,42411555	0,235944731	0,739237715
KriginglC	0,437740334	0,188901876	0,489467874
KrigingqC	0,798175294	0,613558072	0,342466349
Poly1	0,900054887	1,001864487	1,659191493
Poly2	0,776191729	0,915248721	0,32489506
Poly3	2,097784927	0,853198168	6,872190262

Tabel C6: RSME's gevonden door de benaderingsmethoden bij de verificatieset van 5 punten (rijen 18, 33, 57, 78 en 113 van de dataset)

Uit de bovenstaande tabellen blijkt een aantal benaderingsmethoden een redelijk resultaat te geven; krigingcG, kriginglG en kriginglC

Daar de laatste vijf tot tien winglet geometriën van de dataset redelijk goede ontwerppunten lijken te zijn, is het van belang dat er in dit ontwerpgebied niet teveel fitfouten zitten, om deze reden is er ook gekeken naar de waarden van de RSME's, zoals deze worden behaald door de verschillende benaderingsmethoden met als verificatiesets de laatste vijf en de laatste tien punten.

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,135673874	0,241002822	0,47264744
KriginglG	0,133134518	0,177728251	0,32744771
KrigingqG	0,505395291	1,062512366	0,27220103
KrigingcE	0,06933109	0,341747514	0,75862758
KriginglE	0,256681125	0,341675831	0,30145016
KrigingqE	0,532041916	0,992424772	0,27361323
KrigingcC	0,299346956	0,297561062	0,58725855
KriginglC	0,168859705	0,203956951	0,19330649
KrigingqC	0,532041916	1,062512366	0,27421524
Poly1	1,17032551	0,826352557	1,4899202
Poly2	0,532041916	1,062512366	0,27568787
Poly3	1,746641806	0,498189093	0,74054872

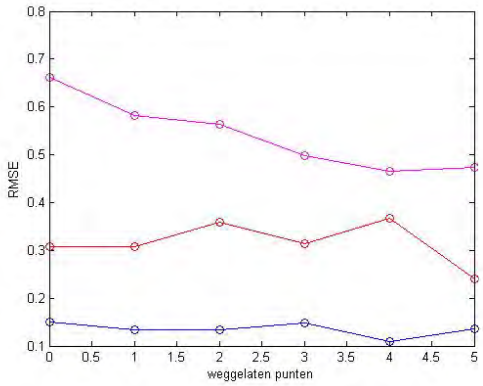
Tabel C7: RSME's gevonden door de benaderingsmethoden bij de verificatieset van de laatste 5 punten van de dataset

Methode	RMSE y_1	RMSE y_2	RMSE y_3
KrigingcG	0,268782998	0,282444559	0,430341841
KriginglG	0,160199875	0,096365487	0,320661504
KrigingqG	0,395366033	0,781531419	0,281053732
KrigingcE	0,377170784	0,381005728	0,736643401
KriginglE	0,268423918	0,2859574	0,577424887
KrigingqE	0,416964147	0,925482611	0,272144631
KrigingcC	0,291519296	0,189304696	0,436762292
KriginglC	0,215768394	0,161233579	0,359380439
KrigingqC	0,416964147	0,758700664	0,269824758
Poly1	1,090894725	0,830380872	1,738752484
Poly2	0,416964147	0,816574144	0,272144631
Poly3	3,254005442	2,203793248	1,485009158

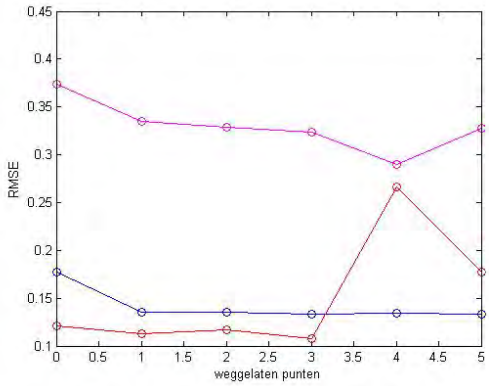
Tabel C8: RSME's gevonden door de benaderingsmethoden bij de verificatieset van de laatste 10 punten van de dataset

Wanneer de tabellen C7 en C8 in beschouwing worden genomen, blijken de benaderingsmethoden kriginglG, krigingcC en kriginglC redelijke resultaten te leveren.

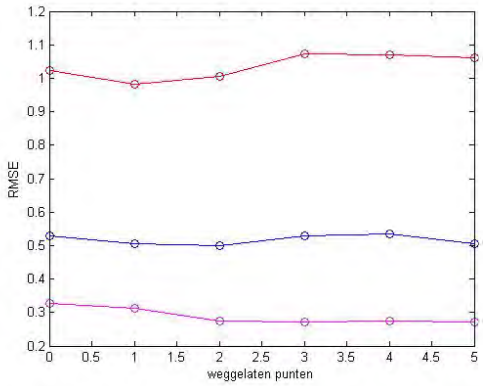
Tot slot zal er gekeken worden naar de convergerende krachten van de benaderingsmethoden. Deze convergerende krachten zullen gebaseerd worden op de laatste vijf ontwerppunten van de dataset, door deze punten als vaste verificatieset te nemen. Daarnaast zijn er nog vijf andere willekeurige ontwerppunten uit de dataset gekozen om een tweede verificatieset mee te vormen. Samen vormen deze twee verificatiesets nu de convergentieset.
Er zal nu eerst een fit gemaakt worden op de dataset zonder de convergentieset. Vervolgens zal er steeds één van de willekeurige verificatiepunten weer toegevoegd worden aan de dataset waarvoor dan opnieuw een fit wordt gemaakt voor de verschillende benaderingsmethoden. Wanneer uiteindelijk alle vijf de willekeurige verificatiepunten zijn toegevoegd, blijft dus alleen nog de set van de laatste vijf ontwerppunten van de dataset over als verificatieset.
De resultaten van deze convergentie worden hieronder in de figuren C1 tot en met C12 weergegeven, waarbij de x-as het aantal verificatiepunten is dat uit de convergentieset is gehaald, met andere woorden bij x=0 zijn er nog 10 punten in de convergentieset en bij x=5 bestaat de convergentieset alleen nog uit de laatste vijf ontwerppunten van de dataset. Op de y-as zijn de bijbehorende RMSE waarden te vinden, waarbij de blauwe punten in de blauwe lijn de resultaten van y_1 weergeven, de rode punten in de rode lijn de resultaten van y_2 en y_3 worden gerepresenteerd door de roze punten in de roze lijn.



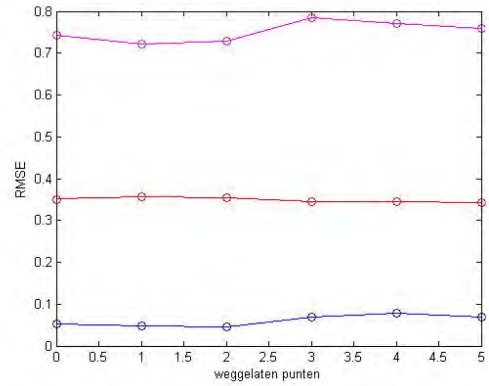
Figuur C1: convergerende kracht krigingcG



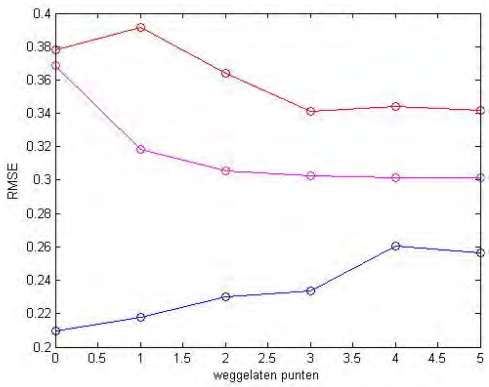
Figuur C2: convergerende kracht kriginglG



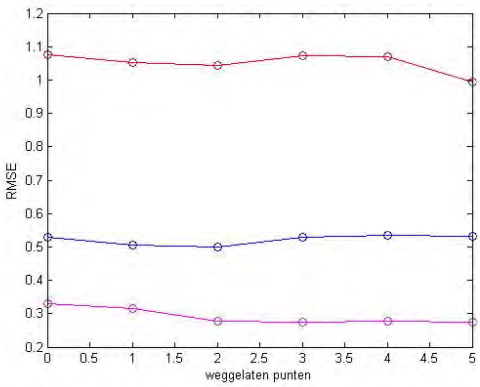
Figuur C3: convergerende kracht krigingqG



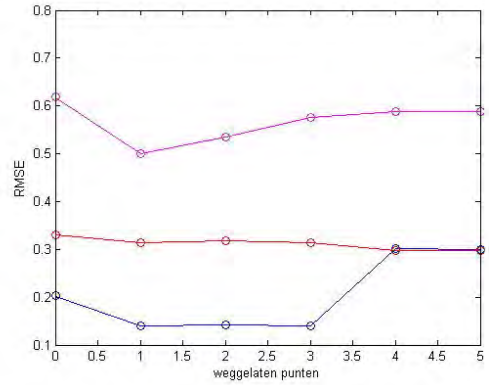
Figuur C4: convergerende kracht krigingcE



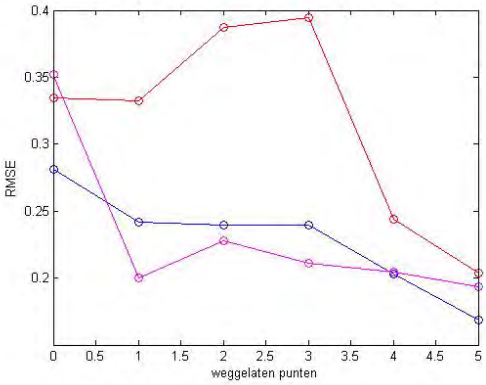
Figuur C5: convergerende kracht kriginglE



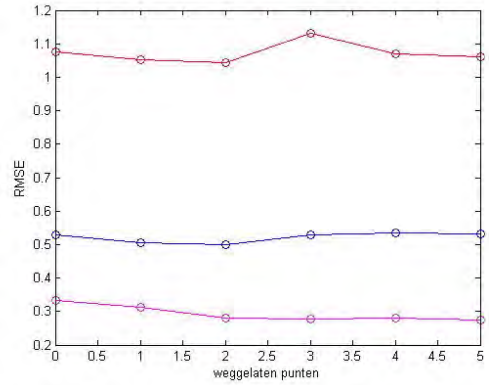
Figuur C6: convergerende kracht krigingqE



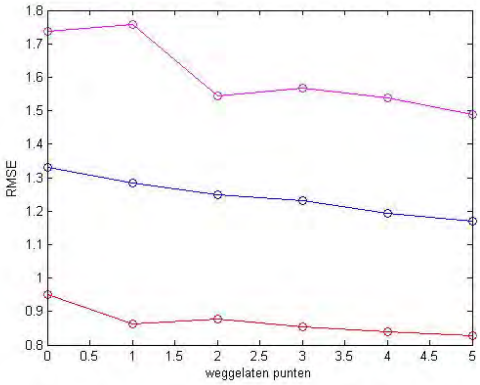
Figuur C7: convergerende kracht krigingcC



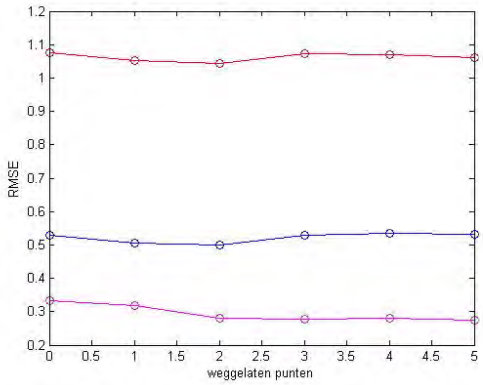
Figuur C8: convergerende kracht kriginglC



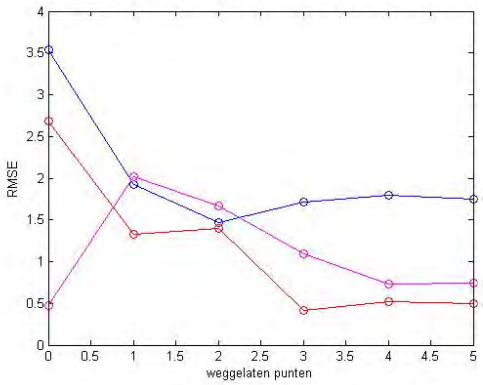
Figuur C9: convergerende kracht krigingqC



Figuur C10: convergerende kracht poly1



Figuur C11: convergerende kracht poly2



Figuur C12: convergerende kracht poly3

Een duidelijke convergentie voor alle outputvariabelen is er niet echt te vinden voor de verschillende benaderingsmethoden. Wanneer de convergerende krachten van de benaderingsmethoden met elkaar vergeleken worden, wordt het duidelijk dat de beste resultaten gegeven worden door kriginglG en kriginglC. Hoewel beide benaderingsmethoden geen eenduidige convergentie laten zien, zijn de RMSE waarden voor de outputvariabelen (behalve bij y2 van kriginglG) lager voor vijf punten in de convergentieset zitten dan voor 10 punten in deze set. Bovendien zijn de RMSE waarden voor alle outputvariabelen bij deze benaderingsmethoden vrij laag in vergelijking met de andere benaderingsmethoden.

Appendix D

Deze Appendix geeft de ondergrenzen en bovengrenzen weer zoals deze voor verschillende optimalisatieberekeningen in de winglet ontwerp studie worden gebruikt. Verder wordt er tevens een overzicht gegeven van de waarden van de instel parameters die gebruikt zijn in MNSGA en hoe de beslissing tot deze waarden tot stand is gekomen.

Grenzen voor de inputvariabelen
De verschillende ondergrenzen en bovengrenzen voor de data zijn hieronder in tabel D1 opgenomen, deze grenzen behoren bij de verschillende zoekdomeinen van de inputvariabelen, zoals deze gebruikt zijn in de winglet ontwerp studie.

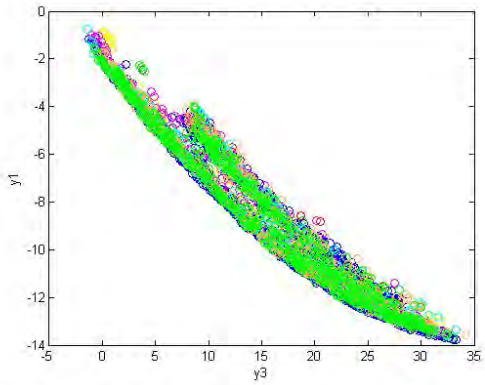
	x1	x2	x3	x4	x5	x6	x7	x8	x9
Ondergrens 70%	260	-3.412	689.595	229.865	23.689	-4.772	-2.806	1149.325	38.475
Bovengrens 70%	540	-0.711	1107.050	369.235	37.121	0.772	2.766	1846.175	78.025
Ondergrens 80%	240	-3.613	659.73	219.91	22.729	-5.168	-3.204	1099.55	35.65
Bovengrens 80%	560	-0.517	1137.57	379.19	38.081	1.168	3.164	1895.95	80.85
Ondergrens 90%	220	-3.807	629.865	209.496	21.770	-5.564	-3.602	1049.775	32.825
Bovengrens 90%	580	-0.324	1167.435	389.145	39.041	1.564	3.562	1945.725	83.675
Ondergrens 100%	200	-4	600	200	20.81	-5.96	-4	1000	30
Bovengrens 100%	600	-0.13	1197.3	399.1	40	1.96	3.96	1995.5	86.5

Tabel D1: onder- en bovengrens input parameters voor de verschillende zoekdomeinen

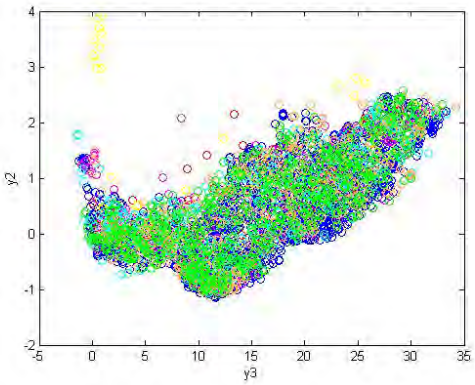
De instel parameters
Om optimaal gebruik te kunnen maken van MNSGA, zijn de volgende instel parameters van belang om ingesteld worden:

- Aantal individuen
- Aantal generaties
- Kans op crossover
- Distributie index voor crossover
- Kans op mutatie
- Distributie index voor mutatie

Het aantal individuen zal dusdanig ingesteld moeten worden dat een verdere convergentie van het Pareto front niet of nauwelijks meer mogelijk is. Er zal daarom gekeken worden bij welke aantal individuen de convergentie voor het oog optimaal is. Hiervoor zal er begonnen worden met een run van MNSGA waarbij het aantal individuen op 2000 wordt gezet (blauwe punten in figuren D1 en D2). Vervolgens wordt het aantal op achtereenvolgens 50 (rode punten), 100 (gele punten), 200 (roze punten), 400 (lichtblauwe punten), 600 (oranje punten) en 750 (groene punten) gezet. Het aantal generaties wordt hiervoor steeds op 20 gezet, zodat dit aantal niet teveel invloed kan uitoefenen op de te behalen Pareto fronten.



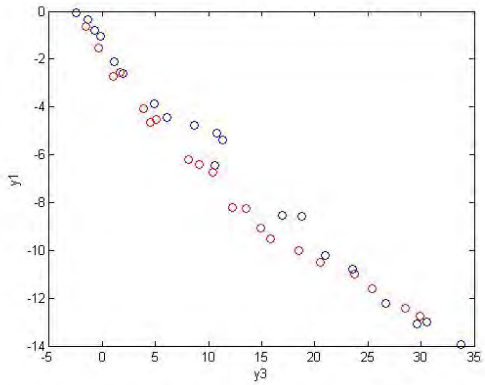
Figuur D1: convergentie aantal individuen



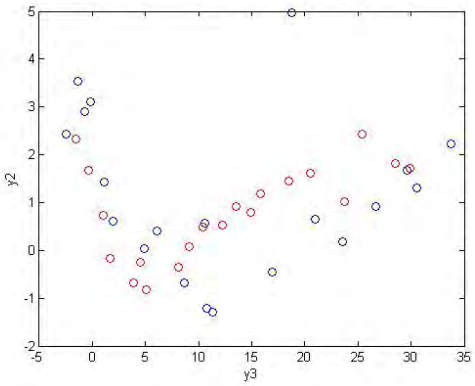
Figuur D2: convergentie aantal individuen

Zoals in figuur D1 en D2 te zien is vindt er, wanneer het aantal individuen op 750 is gesteld, ongeveer dezelfde convergentie plaats als bij 2000 individuen.

Net als voor het aantal individuen zal er ook worden gekeken bij welk aantal generaties het Pareto front voor het oog optimaal geconvergeerd zal zijn. Er zal hier begonnen worden met een run van MNSGA waarbij het aantal generaties op 2000 wordt gezet (blauwe punten in figuren D3 en D4). Vervolgens wordt dit aantal op 50 gezet (rode punten). Het aantal individuen wordt hiervoor steeds op 20 gezet, om dit aantal niet van een al te grote invloed te laten zijn voor de convergentie van het Pareto front.



Figuur D3: convergentie aantal generaties



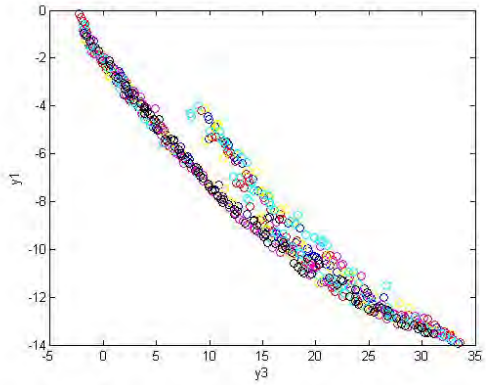
Figuur D4: convergentie aantal generaties

Uit figuur D3 en D4 is er geen groot verschil te zien tussen de verschillende waarde voor het aantal generaties. Blijkbaar speelt het aantal individuen een grotere rol dan het aantal generaties voor de convergentie van het Pareto front.

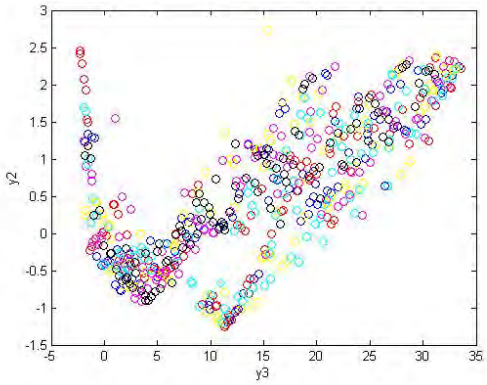
De overige instel parameters zijn niet alleen bedoeld om een zo geconvergeerd mogelijk Pareto front te vinden, maar tevens om de oplossingen zo gespreid mogelijk in het Pareto front te vinden.
Voor het bepalen van de waarden van deze instel parameters zullen, net als voor het aantal individuen en het aantal generaties, verschillende varianten geprobeerd worden. Welke varianten gebruikt worden zal steeds vermeld staan in een tabel en ook hoe deze varianten in de figuren herkenbaar zijn. Het aantal generaties en het aantal individuen zullen hierbij steeds op 100 worden gezet.

Kans op crossover	in de figuren D5 en D6 aangegeven door
0.1	Blauwe punten
0.3	Rode punten
0.5	Gele punten
0.7	Roze punten
0.8	Lichtblauwe punten
0.9	Zwarte punten

Tabel D2: verschillende varianten voor de kans op crossover



Figuur D5: kansen op crossover

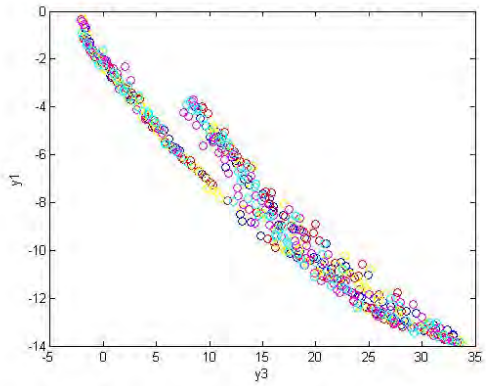


Figuur D6: kansen op crossover

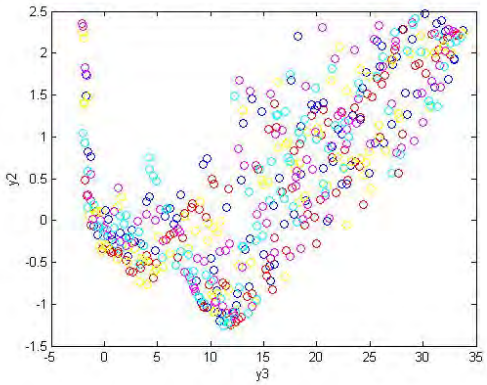
Het Pareto front met een kans op crossover van 0.7 geeft hier de meest gespreide punten en daardoor de beste resultaten weer.

Distributie index voor crossover	in de figuren D7 en D8 aangegeven door
2	Blauwe punten
5	Rode punten
10	Gele punten
20	Roze punten
50	Lichtblauwe punten

Tabel D3: verschillende varianten voor de distributie index voor crossover



Figuur D7: distributie index voor crossover

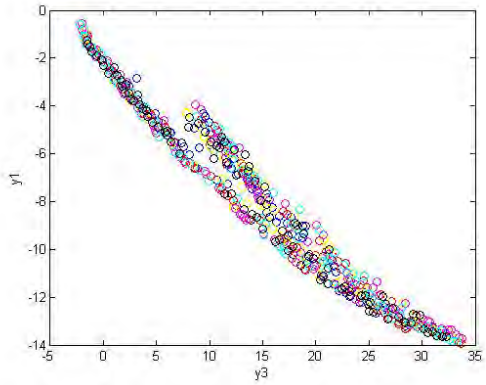


Figuur D8: distributie index voor crossover

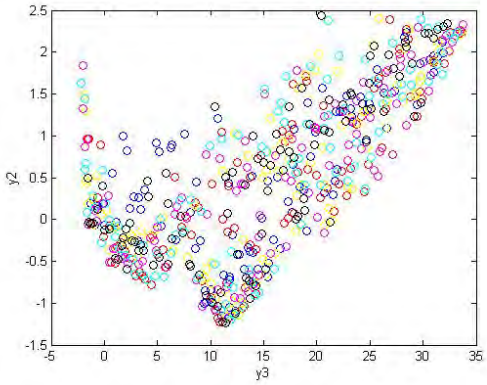
Het Pareto front met een distributie index voor crossover van 5 geeft hier de meest gespreide en geconvergeerde punten weer.

Kans op mutatie	in de figuren D9 en D10 aangegeven door
0.01	Blauwe punten
0.1	Rode punten
0.111	Gele punten
0.3	Roze punten
0.5	Lichtblauwe punten
0.7	Zwarte punten

Tabel D4: verschillende varianten voor de kans op mutatie



Figuur D9: kans op mutatie

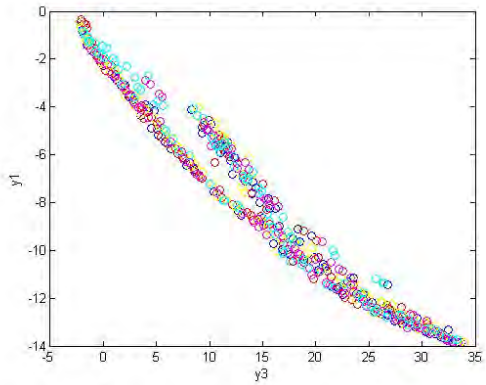


Figuur D10: kans op mutatie

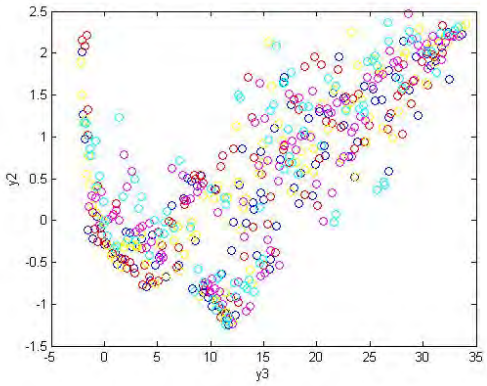
Het meest gespreide en geconvergeerde Pareto front wordt gegeven wanneer de kans op mutatie op 0.5 staat.

Distributie index voor mutatie	in de figuren D11 en D12 aangegeven door
2	Blauwe punten
5	Rode punten
10	Gele punten
20	Roze punten
50	Lichtblauwe punten

Tabel D5: verschillende varianten voor de distributie index voor mutatie



Figuur D11: distributie index voor mutatie



Figuur D12: distributie index voor mutatie

De resultaten waarbij de distributie index voor mutatie op 2 staat, bereiken hier het meest gespreide en geconvergeerde Pareto front.

Door de convergentie en de spreiding van de verschillende Pareto fronten zoals die hierboven zijn weergegeven, zullen de instel parameters de waarden verkrijgen voor de overige berekeningen van MNSGA zoals ze staan vermeld in tabel D6.

Instel parameter	waarde
Aantal individuen	100 en 1000
Aantal generaties	100 en 1000
Kans op crossover	0.7
Distributie index voor crossover	5
Kans op mutatie	0.5
Distributie index voor mutatie	2

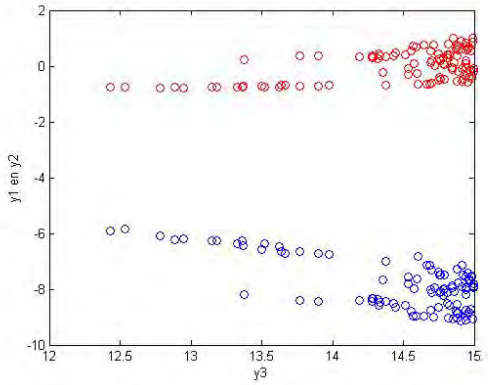
Tabel D6: waarden instel parameters

De waarden voor het aantal individuen en het aantal generaties dient hier nog enige uitleg te krijgen; het aantal 100 is gekozen om te kunnen bekijken hoe het Pareto front zich vorm bij relatief lage waarden voor deze instel parameters en het aantal 1000 zal worden gebruikt om een goed geconvergeerd Pareto front te verkrijgen (het aantal individuen staat hiervoor op 1000 in plaats van 750 om er zeker van te zijn dat er genoeg individuen in de berekeningen worden gebruikt) en om dezelfde reden is het aantal generaties hierbij op ook 250 gezet.

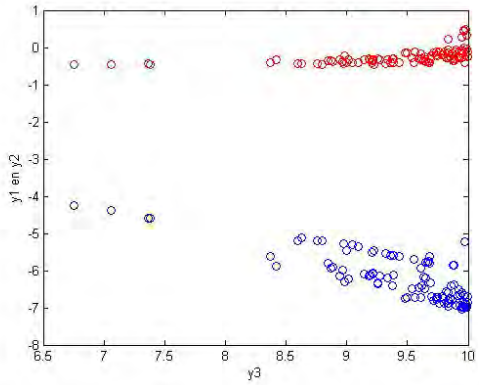
Appendix E

Hieronder worden de resultaten weergegeven zoals zij zijn behaald voor OPT2 en OPT3. Deze resultaten zijn behaald door MNSGA waarbij zowel het aantal individuen als het aantal generaties op 100 staat (tenzij anders vermeld).

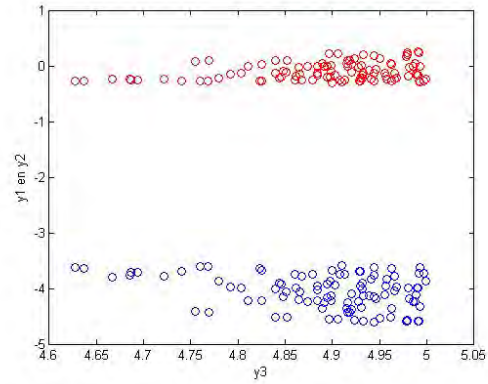
Resultaten OPT2
Hieronder zullen de resultaten getoond worden waarbij de output verdeeld is in twee doelfuncties en één bijvoorwaarde. Onder de grafiek zal steeds vermeld staan welke output als bijvoorwaarde gedefinieerd is en aan welke beperking deze bijvoorwaarde onderworpen is.



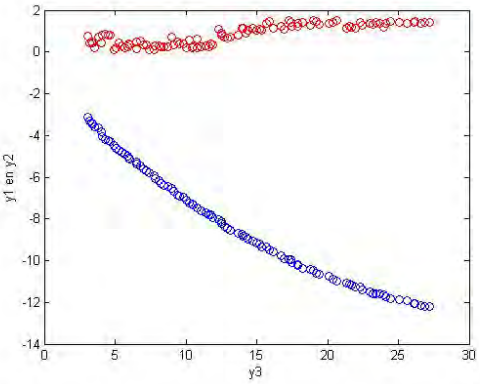
Figuur E1: y3 < 15 als bijvoorwaarde



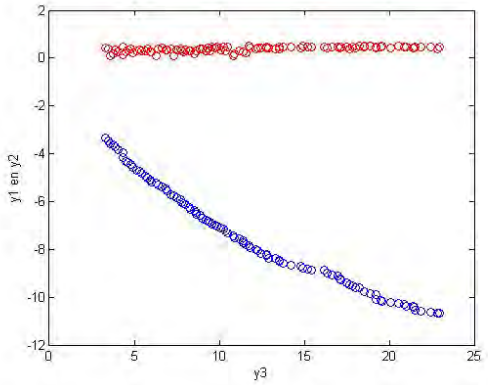
Figuur E2: y3 < 10 als bijvoorwaarde



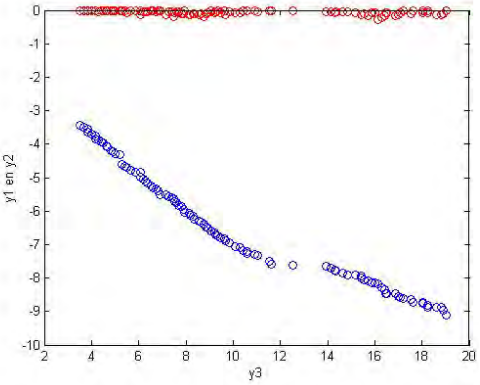
Figuur E3: y3 < 5 als bijvoorwaarde



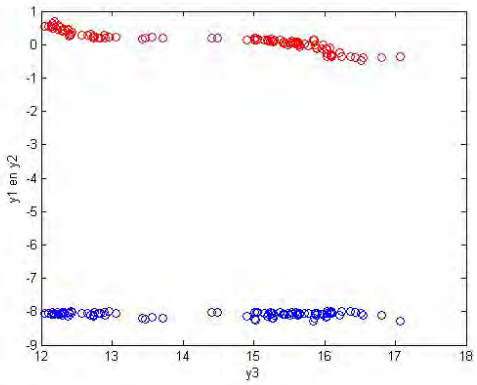
Figuur E4: y2 < 1.5 als bijvoorwaarde



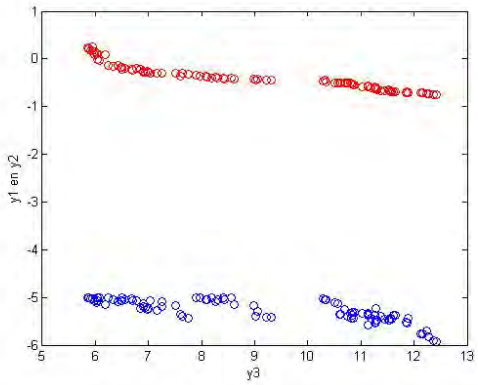
Figuur E5: y2 < 0.5 als bijvoorwaarde



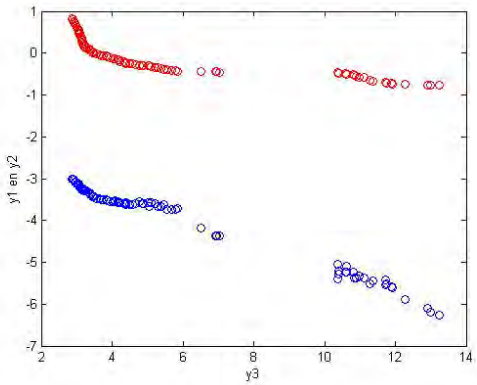
Figuur E6: y2 < 0 als bijvoorwaarde



Figuur E7: y1 < -8 als bijvoorwaarde



Figuur E8: y1 < -5 als bijvoorwaarde



Figuur E9: y1 < -3 als bijvoorwaarde

De figuren E2, E7, E8 en E9 bevatten lege ruimtes tussen de verschillende oplossingen. Een lege ruimte in een Pareto front is niet erg wanneer al bekend is dat oplossingen met waarden die tot een dergelijke lege ruimte behoren geen reële oplossingen of geen optimale oplossingen zijn. Aangezien zulke oplossingen voor deze winglet studie niet bekend zijn, zijn dit niet de Pareto fronten waarnaar gezocht wordt. Verder valt hier op dat, wanneer de derde output als bijvoorwaarde wordt gedefinieerd, er veel Pareto punten worden gevonden rond de waarde die als beperking voor deze bijvoorwaarde is gesteld. Wanneer nu de laatste vijf ontwerppunten van de dataset in beschouwing worden genomen, blijkt dat alleen de figuren E4 tot en met E6 deze vijf punten (met of zonder betere waarden voor één van de outputvariabelen) bevatten. Deze figuren behoren alledrie bij y_2 als bijvoorwaarde.

Resultaten OPT3

Om te bekijken in hoeverre een outputvariabele geminimaliseerd kan worden wanneer de rest van de output als twee bijvoorwaarden is gedefinieerd, zijn er meerdere combinaties uitgeprobeerd. In de tabellen E1, E2 en E3 staan de resultaten hiervan en staat tevens vermeld welke outputs als bijvoorwaarden dienen met welke beperkingen en wat het minimale punt is dat vervolgens voor de doelfunctie is behaald. Bij deze berekeningen staat het aantal individuen wederom op 100, net als het aantal generaties.

Bijvoorwaarden	Minimale waarde voor y1
y2 < 1.5 en y3 < 15	-9.236
y2 < 0.5 en y3 < 15	-8.879
y2 < 0 en y3 < 15	-7.980
y2 < 1.5 en y3 < 10	-7.145
y2 < 0.5 en y3 < 10	-7.061
y2 < 0 en y3 < 10	-7.015
y2 < 1.5 en y3 < 5	-4.617
y2 < 0.5 en y3 < 5	-4.629
y2 < 0 en y3 < 05	-4.205

Tabel E1: minimale waarden voor y1, met y2 en y3 als bijvoorwaarden

Bijvoorwaarden	Minimale waarde voor y2
y1 < -8 en y3 < 15	0.030
y1 < -5 en y3 < 15	-0.784
y1 < -3 en y3 < 15	-0.784
y1 < -8 en y3 < 10	0.401
y1 < -5 en y3 < 10	-0.462
y1 < -3 en y3 < 10	-0.473
y1 < -8 en y3 < 5	0.154
y1 < -5 en y3 < 5	0.139
y1 < -3 en y3 < 5	-0.319

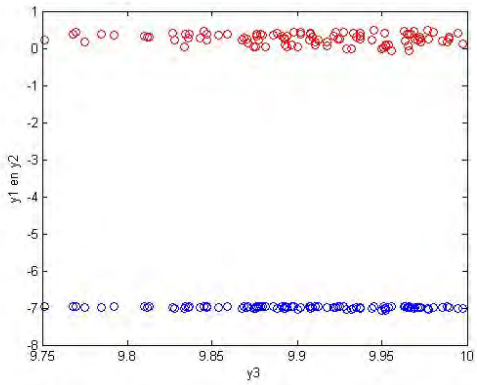
Tabel E2: minimale waarden voor y2, met y1 en y3 als bijvoorwaarden

Bijvoorwaarden	Minimale waarde voor y3
y1 < -8 en y2 < 1.5	11.976
y1 < -5 en y2 < 1.5	5.671
y1 < -3 en y2 < 1.5	2.870
y1 < -8 en y2 < 0.5	12.339
y1 < -5 en y2 < 0.5	5.720
y1 < -3 en y2 < 0.5	3.040
y1 < -8 en y2 < 0	15.324
y1 < -5 en y2 < 0	6.579
y1 < -3 en y2 < 0	3.485

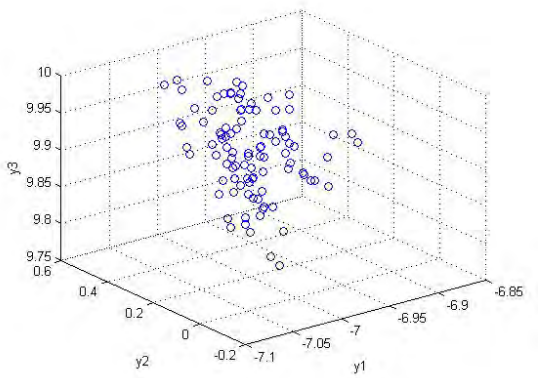
Tabel E3: minimale waarden voor y3, met y1 en y2 als bijvoorwaarden

Alleen in het geval dat y1 -8 als beperking opgelegd krijgt en y3 5 (in het rood aangegeven in tabel E2), is er sprake van een constraint overschrijding. Bij de overige instellingen is er altijd aan de bijvoorwaarden voldaan.

Door twee bijvoorwaarden te definiëren wordt het gebied waar de Pareto punten worden gevonden (de oplossingsruimte) vrij klein zoals te zien is in de figuren E10 en E11, waar een voorbeeld is gegeven. Echter worden er nog altijd wel verschillende Pareto punten gevonden.



Figuur E10: y2 < 0.5 en y3 < 10



Figuur E11: y2 < 0.5 en y3 < 10

Doordat de oplossingsruimte waarin het Pareto front gevonden wordt dusdanig is verkleind bij het gebruik van twee bijvoorwaarden, is het definiëren van twee bijvoorwaarden eigenlijk alleen maar effectief wanneer al bekend is in welk gebied de waarden van de doelfunctie zich ongeveer zou moeten bevinden.

Appendix F

Hieronder worden de resultaten getoond zoals deze behaald zijn door GA en fmincon.

Gentic Algorithm toolbox
Om de instel parameters in GA zoveel mogelijk gelijk te stellen aan de instel parameters van MNSGA, is besloten tot de instellingen zoals deze in tabel F1 te vinden zijn.

Instel parameter	instelling
Aantal individuen	10
Aantal generaties	10
Selectie methode	Tournament 2
Crossover methode	Two point
Kans op crossover	0.7
Mutatie methode	Uniform
Kans op mutatie	0.5

Tabel F1: instellingen instel parameters GADS

De reden dat het aantal individuen en aantal generaties maar op tien is gezet en niet op 100 is dat tijdens de berekeningen is gebleken dat GA de grootste moeite met dit probleem heeft en daardoor zeer traag werd als deze aantallen op meer dan tien werden gezet. Zelfs met deze tien individuen, duurde de runs nog altijd rond de dertien minuten.

In de tabellen F2, F3 en F4, zijn de minimale waarden te vinden die GADS heeft geproduceerd en de bijvoorwaarden met de bijbehorende beperkingen.
Verder is gebleken dat de bijvoorwaarden in een aantal gevallen overschreden zijn, wanneer dit het geval is geweest zal de bijvoorwaarde met zijn beperking in het rood gekleurd staan in de onderstaande tabellen.

Bijvoorwaarden	Minimale waarde voor y1
y2 < 1.5 en y3 < 15	-5.966
y2 < 0.5 en y3 < 15	-4.707
y2 < 0 en y3 < 15	-5.908
y2 < 1.5 en y3 < 10	-3.757
y2 < 0.5 en y3 < 10	-5.401
y2 < 0 en y3 < 10	-5.478
y2 < 1.5 en y3 < 5	-3.669
y2 < 0.5 en y3 < 5	-3.581
y2 < 0 en y3 < 5	-3.898

Tabel F2: minimale waarden voor y1, met y2 en y3 als bijvoorwaarden

Bij de eerste acht instellingen voor de bijvoorwaarden werd y3 vaak overschreden en y2 nooit, daarom zijn de bijvoorwaarden voor het laatste geval (y2 < 0 en y3 < 5) dusdanig ingevoerd zodat eerst y3 berekend is en daarna pas y2 (in alle andere gevallen is het dus andersom geweest). Het resultaat bleek vervolgens te zijn dat nu y2 overschreden is en niet y3.

Bijvoorwaarden	Minimale waarde voor y2
y1 < -8 en y3 < 15	0.658
y1 < -5 en y3 < 15	0.149
y1 < -3 en y3 < 15	0.022
y1 < -8 en y3 < 10	-
y1 < -5 en y3 < 10	-0.214
y1 < -3 en y3 < 10	-0.302
y1 < -8 en y3 < 5	-
y1 < -5 en y3 < 5	-0.027
y1 < -3 en y3 < 5	1.731

Tabel F3: minimale waarden voor y2, met y1 en y3 als bijvoorwaarden

In de gevallen waarbij y1 als beperking -8 had en y3 10 en 5, was GA niet in staat om minimale waarden voor y2 te vinden.

Bijvoorwaarden	Minimale waarde voor y3
y1 < -8 en y2 < 1.5	16.509
y1 < -5 en y2 < 1.5	7.691
y1 < -3 en y2 < 1.5	12.022
y1 < -8 en y2 < 0.5	14.887
y1 < -5 en y2 < 0.5	13.183
y1 < -3 en y2 < 0.5	7.792
y1 < -8 en y2 < 0	18.817
y1 < -5 en y2 < 0	9.707
y1 < -3 en y2 < 0	8.424

Tabel F4: minimale waarden voor y3, met y1 en y2 als bijvoorwaarden

Er is slechts één keer een resultaat gevonden dat een lagere waarde presenteert voor de doelfunctie dan MNSGA heeft geproduceerd (dit resultaat is in het blauw gekleurd in tabel F3). In alle andere gevallen is er door MNSGA altijd een lagere waarde gevonden. Verder zijn de bijvoorwaarden bij GA een aantal keer overschreden terwijl dit bij MNSGA maar één keer het geval is geweest. Tot slot viel ook nog op dat alle individuen in GA dezelfde waarden aannemen bij het uiteindelijke resultaat (dus na de tiende generatie), terwijl in MNSGA de meeste individuen verschillende waarden bevatten voor de te minimaliseren doelfunctie.

Fmincon

In de tabellen F5, F6 en F7 zijn de minimale waarden te vinden voor een doelfunctie die fmincon bij twee bijvoorwaarden en de daarbij behorende beperkingen heeft gehaald.
Tijdens de verschillende berekeningen is gebleken dat de bijvoorwaarden ook hier soms werden overschreden, wanneer dit het geval is geweest zal de bijvoorwaarde met zijn beperking ook weer in het rood gekleurd staan in de tabellen. Wanneer een minimale waarde in het blauw gekleurd staat, geeft dit aan dat de minimale waarde gevonden door MNSGA lager was dan de waarde die door fmincon gevonden is.

Bijvoorwaarden	Minimale waarde voor y1
y2 < 1.5 en y3 < 15	-9.2378
y2 < 0.5 en y3 < 15	-9.239
y2 < 0 en y3 < 15	-8.055
y2 < 1.5 en y3 < 10	-7.573
y2 < 0.5 en y3 < 10	-7.221
y2 < 0 en y3 < 10	-7.101
y2 < 1.5 en y3 < 5	-5.155
y2 < 0.5 en y3 < 5	-5.263
y2 < 0 en y3 < 5	-4.238

Tabel F5: minimale waarden voor y1, met y2 en y3 als bijvoorwaarden

Bijvoorwaarden	Minimale waarde voor y2
y1 < -8 en y3 < 15	0.060
y1 < -5 en y3 < 15	-1.013
y1 < -3 en y3 < 15	-1.013
y1 < -8 en y3 < 10	0.166
y1 < -5 en y3 < 10	-0.112
y1 < -3 en y3 < 10	-0.113
y1 < -8 en y3 < 5	-0.366
y1 < -5 en y3 < 5	-0.795
y1 < -3 en y3 < 5	-0.844

Tabel F6: minimale waarden voor y2, met y1 en y3 als bijvoorwaarden

Bijvoorwaarden	Minimale waarde voor y3
y1 < -8 en y2 < 1.5	12.015
y1 < -5 en y2 < 1.5	6.005
y1 < -3 en y2 < 1.5	2.208
y1 < -8 en y2 < 0.5	11.027
y1 < -5 en y2 < 0.5	5.188
y1 < -3 en y2 < 0.5	1.043
y1 < -8 en y2 < 0	13.915
y1 < -5 en y2 < 0	7.343
y1 < -3 en y2 < 0	6.274

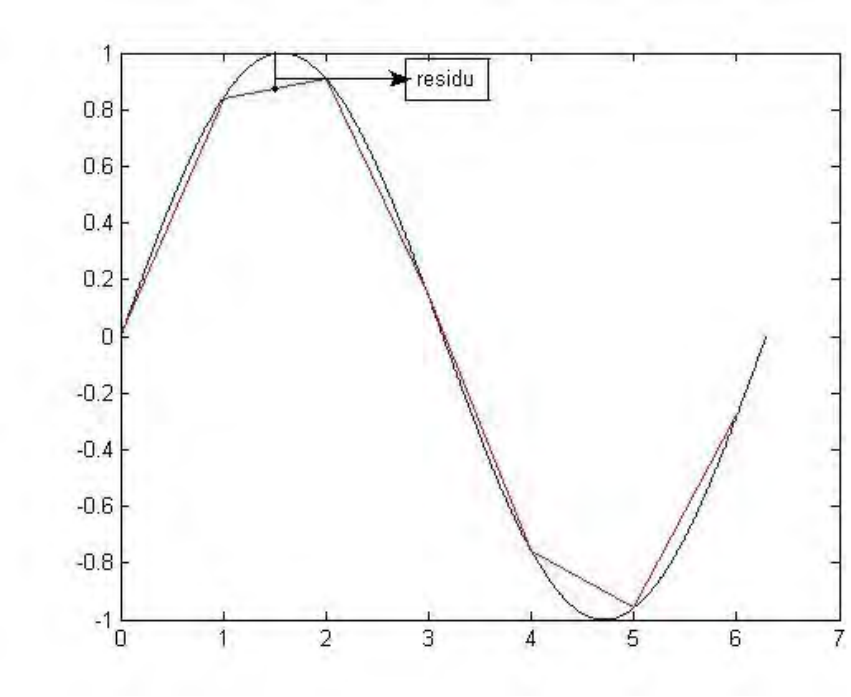
Tabel F7: minimale waarden voor y3, met y1 en y2 als bijvoorwaarden

Duidelijk is dat door fmincon de bijvoorwaarden nog wel eens overschreden worden en bovendien wordt er lang niet altijd een lagere waarde voor de doelfunctie bereikt dan door MNSGA, hetgeen je wel zou verwachten van een optimalisatie algoritme dat specifiek voor single-objective optimalisatie problemen bedoeld is. Ook blijkt fmincon veel tijd te vragen voor de berekening, deze lag namelijk steeds rond de dertien minuten (net zoals bij GA dus).

Appendix G

Hieronder zal het idee achter de residuen eerst nog iets nader toegelicht worden met behulp van een voorbeeld om daarmee duidelijk te kunnen maken welke waarden er nou precies verkregen worden en waarom deze belangrijk zijn. Daarna zal de analyse getoond worden hoe een geschikte benaderingsmethode gevonden is voor deze residuen met behulp van MultiFit. Tot slot zullen de resultaten bekeken worden voor al eerder behaalde resultaten van MNSGA, maar nu in combinatie met de geschatte residuen.

De Sinusfunctie
Een residu wordt gedefinieerd als het verschil tussen een voorspelde waarde en de daadwerkelijke waarde. Om duidelijk te kunnen maken wat deze residuen nu precies aangeven en hoe ze kunnen worden bepaald, zal er hier een voorbeeld van een sinusfunctie gebruikt worden
In figuur G1 staat een sinusfunctie afgebeeld met een blauwe lijn, maar tevens is er ook een rode lijn te zien, welke beschouwd kan worden als een benaderde sinusfunctie.



Figuur G1: sinusfunctie en de benaderde sinusfunctie

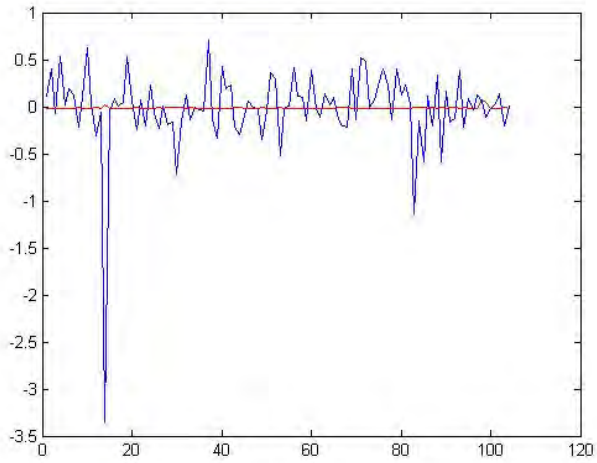
De benaderde sinusfunctie is hier gevonden door een klein aantal punten van de sinusfunctie te plotten en daardoorheen rechte lijnen van het ene naar het andere punt te trekken. Door deze benadering zijn er nu enkele verschillen waar te nemen tussen de blauwe en de rode lijn. Een duidelijk verschil is bijvoorbeeld te zien bij de x-waarde van 1.5. Dit verschil kan nu berekend worden door de waarde van de rode lijn in dit punt te nemen en de waarde van de blauwe lijn ervan af te halen (zie de zwarte lijn in figuur G1), dit is dan tevens de waarde van het residu dat behoort bij x = 1.5.
Wanneer nu waarden van residuen worden bepaald voor meerdere x-waarden, kan er een passende benaderingsmethode gezocht worden voor deze residuen met behulp van MultiFit.

Het uiteindelijke idee is nu dat wanneer er nieuwe x-punten genomen worden, bijvoorbeeld in het interval [2*PI, 4*PI], niet alleen de benaderde y-waarden met behulp van MultiFit in dit interval gevonden kunnen worden, maar ook de benaderde residuen. Met deze residuen kan dan een idee verkregen worden in hoeverre de benaderde y-waarden van de werkelijke y-waarden af zullen liggen.

Een benaderingsmethode voor de residuen

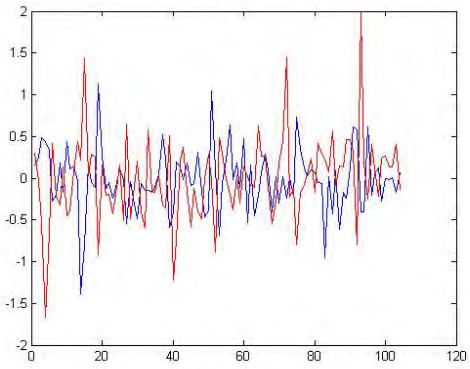
Voor het bepalen van de residuen (de fit fout) voor winglet studie zijn eerst de waarden voor y* bepaald met behulp van leave-one-out mogelijkheid van MultiFit. KriginglC is hier uiteraard gebruikt als benaderingsmethode, daar kriginglC ook gebruikt wordt door MNSGA om optimale waarden te vinden voor de output. Nu er voor iedere rij in de dataset (dus voor iedere combinatie van de x-waarden waarbij de y-waarden bekend zijn) de benaderde waarden van y, y*, bepaald zijn, kunnen de residuen berekend worden.
Welke benaderingsmethode nu het beste aansluit bij het benaderen van de residuen, zal worden bepaald door te kijken naar de resultaten van leave-one-out, de waarden van de benaderde residuen voor de laatste vijf punten van de dataset met de bijbehorende RMSE waarden en de RMSE waarden van de benaderde residuen voor drie verschillende verificatiesets die alledrie uit vijf willekeurige punten van de dataset bestaan.

Wanneer de daadwerkelijke residuen geplot worden tegen de benaderde residuen zoals die behaald zijn door middel van leave-one-out, kunnen er verschillende resultaten gepresenteerd worden. Hierdoor is ook duidelijk geworden dat een aantal benaderingsmethode niet goed in staat is om de residuen te schatten, daar door deze methoden voor bijna alle residuen dezelfde waarde wordt gegeven, terwijl de waarden van de daadwerkelijke residuen wel degelijk verschillend zijn. Een voorbeeld hiervan staat afgebeeld in figuur G2, waarbij de blauwe lijn de daadwerkelijke residuen van y2 beschrijft en de rode lijn de benaderde residuen voor y2 die gevonden zijn door krigingcE.

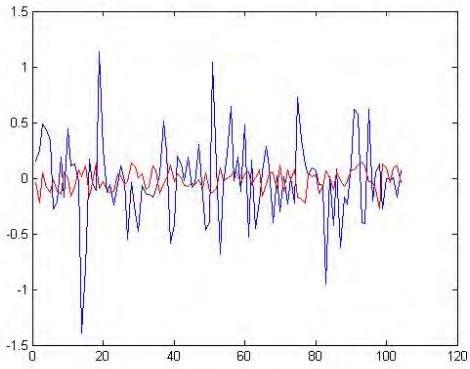


Figuur G2: benaderde en echte residuen van y2 met krigingcE

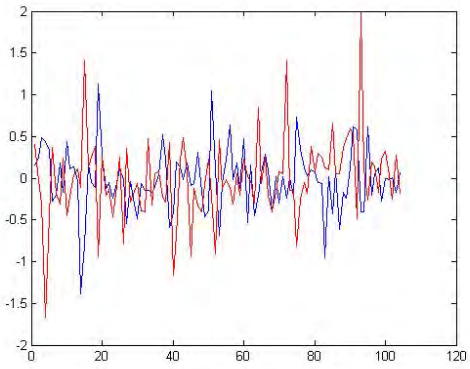
De benaderingsmethoden die het gedrag van de residuen beter beschrijven, zijn hieronder in de figuren G3 tot en met G13 afgebeeld.



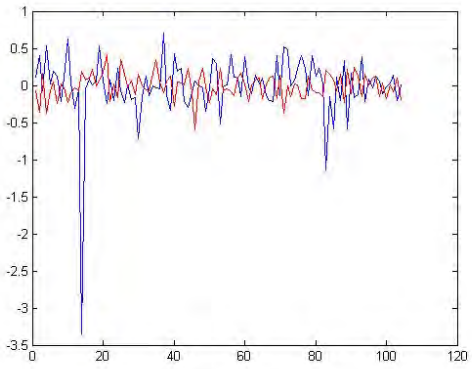
Figuur G3: y1 met krigingqE



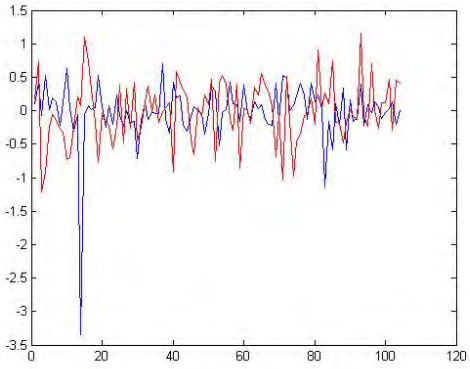
Figuur G4: y1 met kriginglC



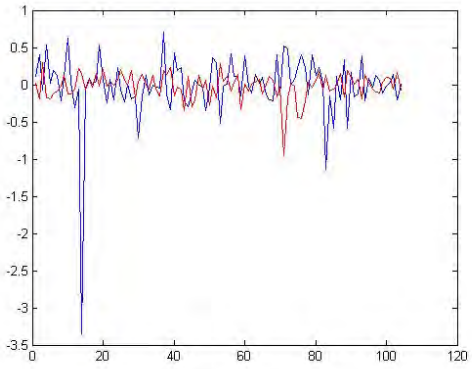
Figuur G5: y1 met krigingqC



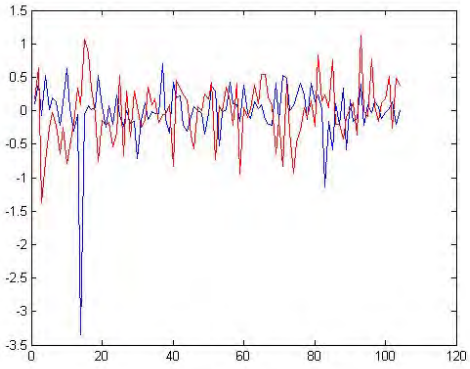
Figuur G6: y2 met kriginglG



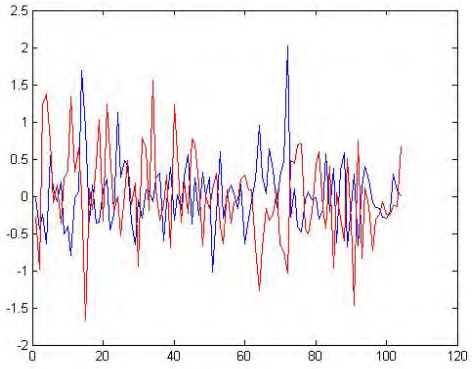
Figuur G7: y2 met krigingqE



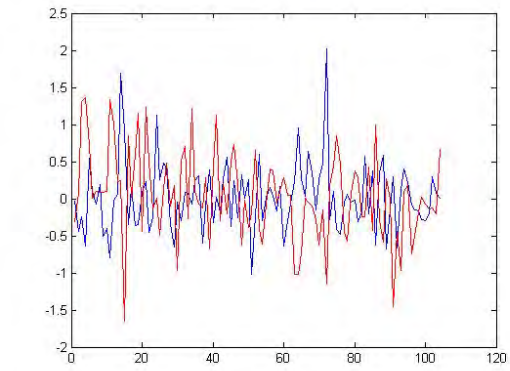
Figuur G8: y2 met kriginglC



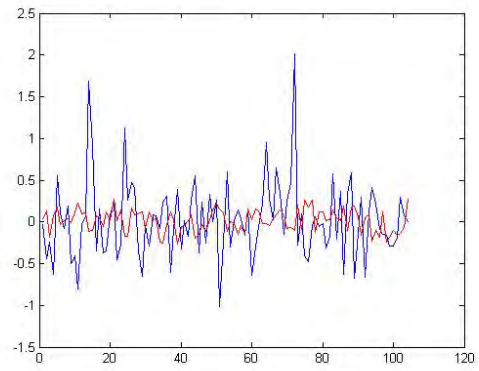
Figuur G9: y2 met krigingqC



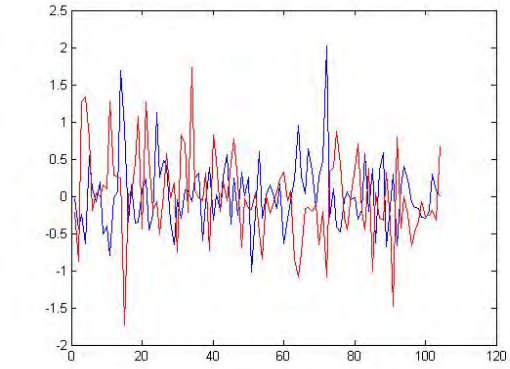
Figuur G10: y3 met krigingqG



Figuur G11: y3 met krigingqE

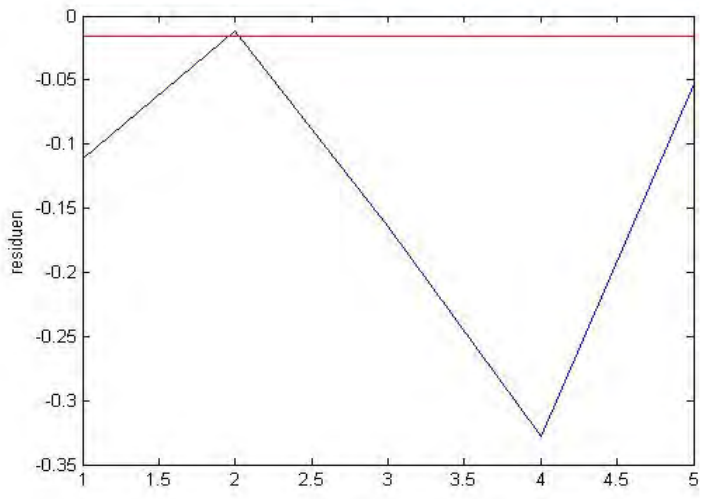


Figuur G12: y3 met kriginglC



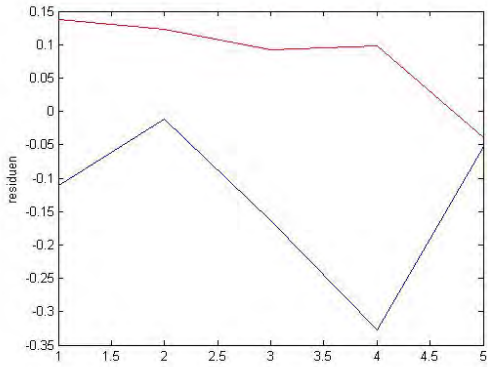
Figuur G13: y3 met krigingqC

Om te bekijken in hoeverre de benaderingsmethoden de residuen kunnen schatten in de laatste vijf ontwerppunten van de gegeven dataset, is nu opnieuw een fit voor de residuen bepaald met kriginglC, maar nu voor de eerste 99 inputwaarden en de daarbij behorende y-waarden van de dataset. Ook hierbij is er weer een aantal benaderingsmethoden dat voor alle residuen (vijf in totaal) ongeveer dezelfde waarde geeft, een voorbeeld is hieronder te zien in figuur G14 waar het resultaat van krigingcC voor y1 te vinden is, waarbij de blauwe lijn weer de daadwerkelijke residuen aangeeft en de rode lijn de benaderde residuen.

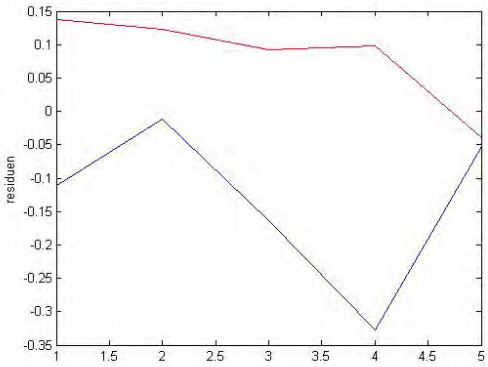


Figuur G14: y1 met krigingcC

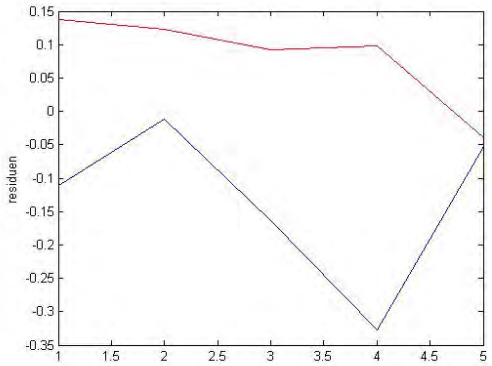
De figuren G15 tot en met G26 zoals die hieronder afgebeeld zijn, geven een veel beter resultaat. Voor welke output de resultaten behaald zijn en met welke benaderingsmethode zal onder de figuren vermeld staan.



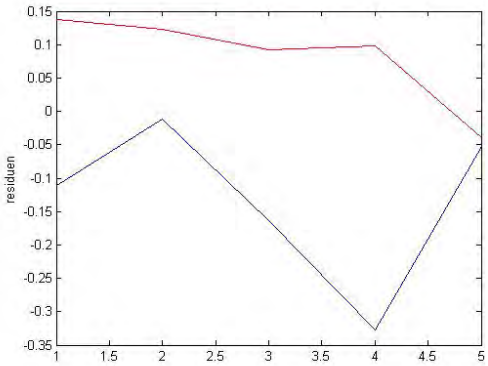
Figuur G15: y1 met kriginglG



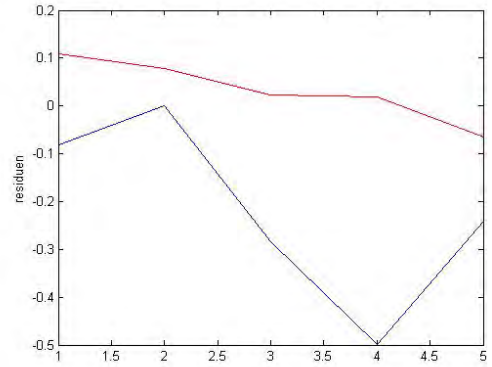
Figuur G16: y1 met kriginglE



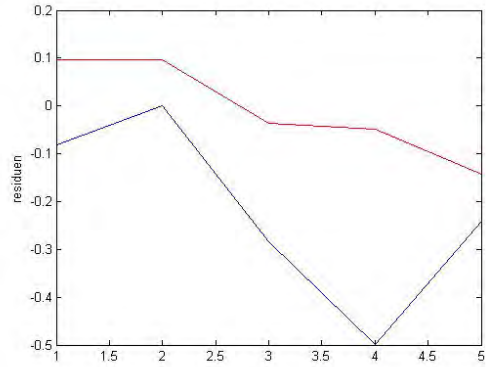
Figuur G17: y1 met kriginglC



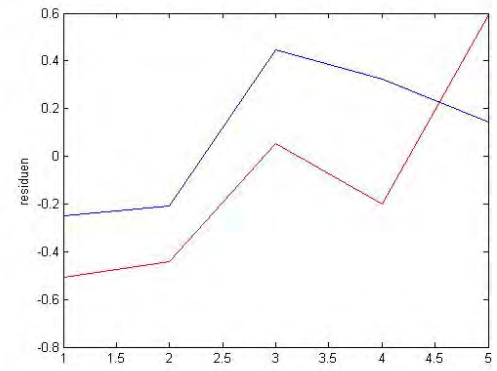
Figuur G18: y1 met poly1



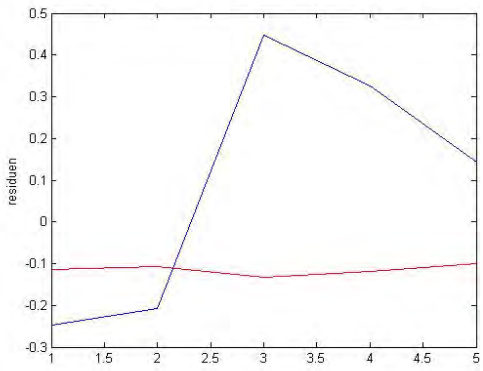
Figuur G19: y2 met kriginglE



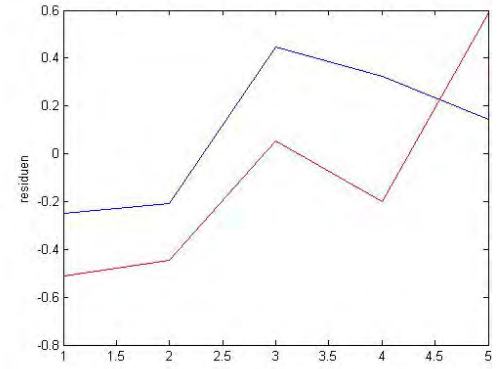
Figuur G20: y2 met kriginglC



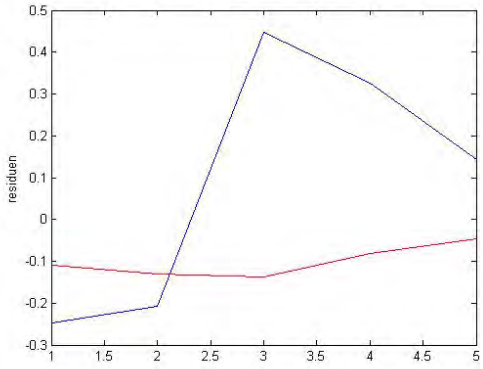
Figuur G21: y3 met krigingqG



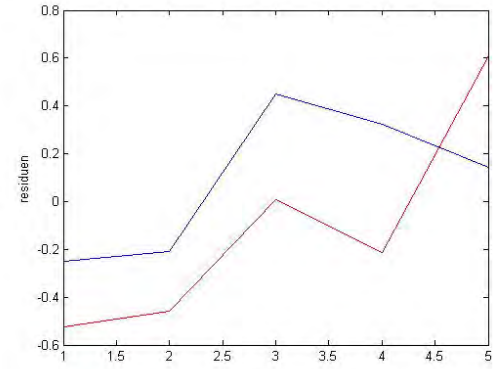
Figuur G22: y3 met kriginglE



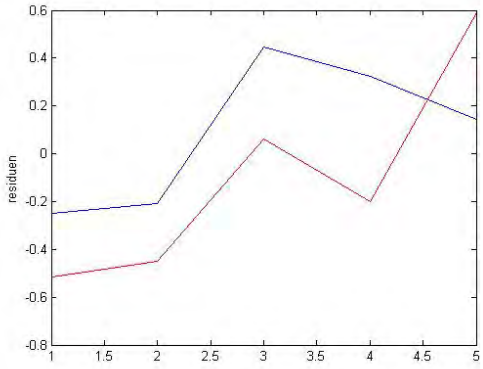
Figuur G23: y3 met krigingqE



Figuur G24: y3 met kriginglC



Figuur G25: y3 met krigingqC



Figuur G26: y3 met poly1

De RMSE waarden die horen bij de bovenstaande figuren, zijn hieronder weergegeven in tabel G1. Hierbij zijn de waarden die redelijk te noemen zijn in het vet aangegeven.

Methode	Output	RMSE
kriginglG	Y1	**0.256**
krigingqG	Y3	0.388
kriginglE	Y1	**0.256**
kriginglE	Y2	**0.295**
kriginglE	Y3	0.352
krigingqE	Y3	0.390
kriginglC	Y1	**0.256**
kriginglC	Y2	**0.251**
kriginglC	Y3	0.338
krigingqC	Y3	0.410
Poly1	Y1	**0.256**
Poly1	Y2	**0.284**
Poly2	Y3	0.389

Tabel G1: RSME's voor de residuen behorende bij de laatste vijf punten van de dataset

Opvallend is dat voor y1 alle vier de benaderingsmethode dezelfde waarde voor de RMSE geven. Voor zowel y2 en y3 is het kriginglC die de beste RMSE waarden levert.

Ditzelfde idee, dus een fit voor de residuen creëren op basis van 99 punten van de dataset en als verificatieset vijf punten nemen, is ook nog eens uitgevoerd voor drie verificatiesets met elk vijf willekeurige punten. De resultaten hiervan zijn hieronder in de tabellen G2, G3 en G4 te vinden. Echter zullen de resultaten van krigingcG, krigingcE, krigingcC en poly2 niet getoond worden, omdat deze benaderingsmethoden steeds een rechte lijn laten zien in de grafieken en de benaderde waarden van de residuen dus allemaal (bijna) hetzelfde zijn.

Methode	RMSE y1	RMSE y2	RMSE y3
kriginglG	**0,288896982**	0,425187876	**0,257148591**
krigingqG	**0,34729612**	0,477627728	0,436912843
kriginglE	**0,288896982**	**0,279723146**	**0,258237709**
krigingqE	0,411229794	0,445837723	0,611672033
kriginglC	**0,288838806**	0,370111327	0,254880562
krigingqC	0,363879508	0,424615581	0,533341026
Poly1	**0,288896982**	**0,299395133**	**0,273309559**
Poly2	0,35198905	0,424055193	0,490116094
Poly3	6,716244638	4,696660502	3,916870426

Tabel G2: RSME's voor de residuen behorende bij vijf willekeurige punten van de dataset (2, 50, 78, 97 en 103)

Methode	RMSE y1	RMSE y2	RMSE y3
kriginglG	**0,32196632**	0,430700283	0,39203564
krigingqG	**0,330771709**	0,59667438	0,647102944
kriginglE	**0,278737755**	**0,297953904**	0,370398231
krigingqE	0,396277559	0,590307809	0,692523016
kriginglC	**0,30282898**	0,421885418	0,386965193
krigingqC	0,357014904	0,639487079	0,702717508
Poly1	**0,290888295**	**0,33476359**	**0,316180934**
Poly2	0,399022207	0,693906766	0,645240707
Poly3	3,922687028	2,486425014	0,986915293

Tabel G3: RSME's voor de residuen behorende bij vijf willekeurige punten van de dataset (12, 37, 63, 81 en 103)

Methode	RMSE y1	RMSE y2	RMSE y3
kriginglG	**0,184040793**	0,382430707	**0,277480993**
krigingqG	**0,332711636**	0,879657725	0,49885278
kriginglE	**0,188007404**	0,423181815	**0,247168275**
krigingqE	0,352340337	0,625774807	0,639595213
kriginglC	**0,182291018**	0,892751794	**0,282835269**
krigingqC	**0,332621922**	0,611818488	0,40079844
Poly1	**0,177960654**	0,411612469	**0,258871186**
Poly2	**0,339524685**	0,534571041	0,396269024
Poly3	4,029993674	3,049590363	4,89249478

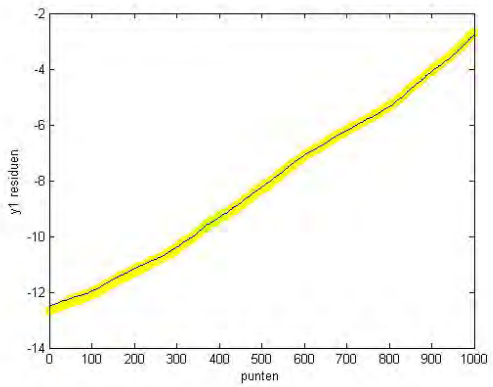
Tabel G4: RSME's voor de residuen behorende bij vijf willekeurige punten van de dataset (7, 28, 59, 71 en 104)

Wanneer alle bovenstaande resultaten in acht worden genomen, lijken kriginglE en kriginglC de beste benaderingsmethoden te zijn. Als gekeken wordt naar de RMSE waarden zoals deze behaald zijn voor de laatste vijf punten van de dataset, geeft kriginglC een beter resultaat dan kriginglE. Omdat deze vijf punten erg belangrijk zijn in het bepalen van de beste resultaten, zal er gekozen worden voor kriginglC als benaderingsmethode voor de residuen.

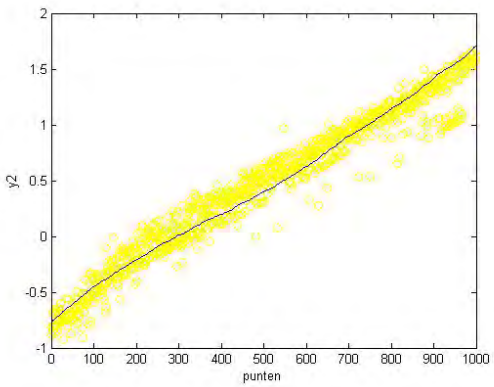
Posteriori check

De benaderde residuen zoals die gefit zijn met de kriginglC methode, zullen worden gebruikt als posteriori check. Dat wil zeggen dat de waarden van de ontwerpvariabelen die horen bij de beste gevonden waarden voor de outputvariabelen, welke gevonden zijn met behulp van MNSGA, ook zullen worden gebruikt om de waarden van de residuen mee te benaderen. Op deze manier zal getracht worden om de meest optimale waarden voor de output te vinden met de kleinste residuen, om zo de beste en meest betrouwbare resultaten te kunnen presenteren.

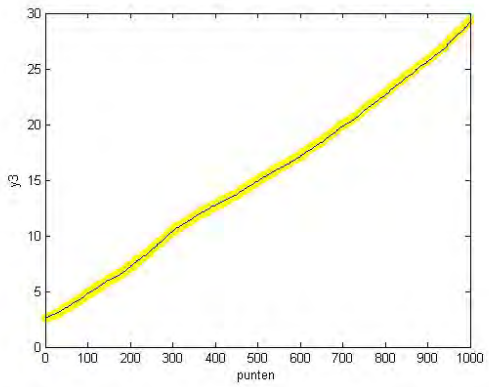
Om deze posteriori check nader toe te lichten, zal er hier een voorbeeld worden gegeven waarvoor de resultaten van MNSGA worden gebruikt die met 1000 individuen en 1000 generaties bereikt zijn en waarbij alle output geminimaliseerd werd. Bij deze resultaten zullen nu de benaderde residuen getoond worden door deze op te tellen bij de waarden die voor de outputvariabelen verkregen zijn. In de figuren G27, G28 en G29 zijn nu de outputvariabelen en de 'outputvariabelen + residuen' afgebeeld, waarbij de laatste in het geel worden aangegeven en de outputvariabelen door een blauwe lijn worden getoond.



Figuur G27: y1 en y1+residuen



Figuur G28: y2 en y2+residuen



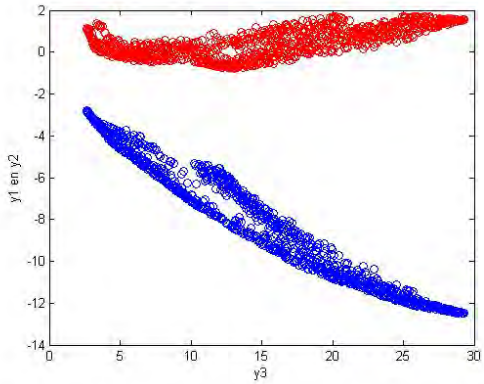
Figuur G29: y3 en y3+residuen

De residuen tonen dus een gebied, waarin de waarden van de daadwerkelijke outputvariabelen zich ook zouden kunnen bevinden.

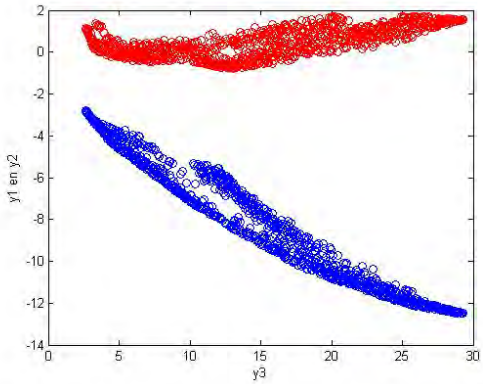
Appendix H

In deze appendix worden de resultaten weergegeven zoals deze zijn behaald wanneer het zoekgebied achtereenvolgens vergroot is naar 80%, 90% en 100% van het maximale zoekdomein zoals deze door de dataset is gegeven. Echter zullen ook steeds de resultaten van het gebied van 70% weer gepresenteerd worden om een duidelijk beeld te krijgen van de behaalde resultaten.

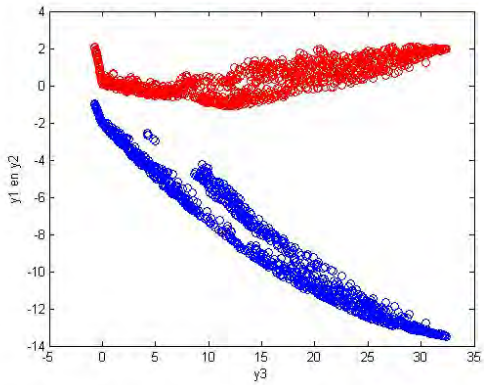
Als eerste wordt er gekeken naar de resultaten van de outputvariabelen zoals deze behaald zijn door MNSGA waarbij er gebruik gemaakt is van 1000 individuen en 1000 generaties.



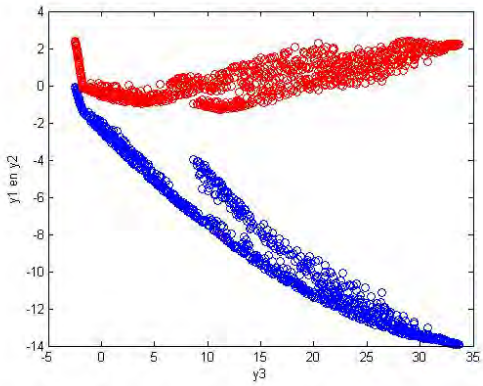
Figuur H1: output bij 70%



Figuur H2: output bij 80%



Figuur H3: output bij 90%



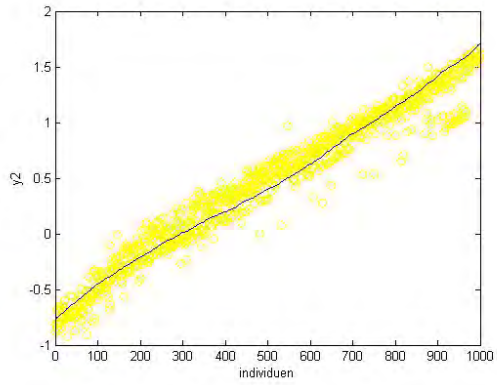
Figuur H4: output bij 100%

Door het vergroten van het zoekgebied wordt er dus wel degelijk een Pareto front gevonden dat een groter gedeelte in de oplossingsruimte bedekt. Dit blijkt ook uit tabel H1 waarin de minimale en maximale waarden van de behaalde outputvariabelen staan vermeld zoals deze behaald zijn bij de verschillende grootten van de zoekgebieden.

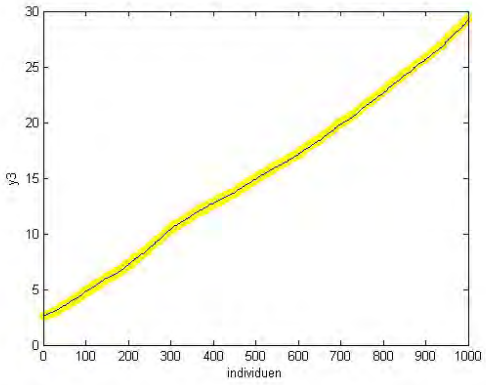
	y1	y2	y3
Minima 70%	-12.503	-0.766	2.642
Maxima 70%	-2.776	1.708	29.291
Minima 80%	-12.992	-0.942	0.990
Maxima 80%	-1.864	1.882	30.827
Minima 90%	-13.475	-1.097	-0.738
Maxima 90%	-0.950	2.092	32.337
Minima 100%	-13.933	-1.277	-2.508
Maxima 100%	-0.057	2.414	33.629

Tabel H1: minima en maxima van de outputwaarden bij de verschillende grootten van de zoekgebieden

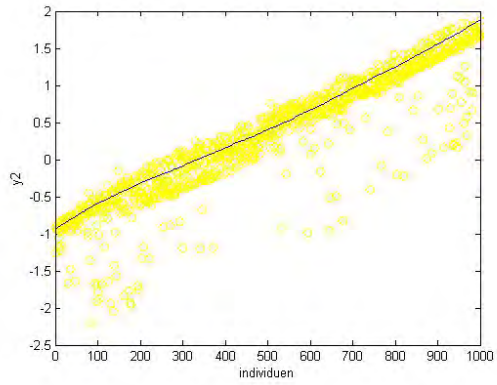
Met het in gebruik nemen van de benaderde residuen, kan er hier worden gekeken in hoeverre het zoekgebied van 100% betrouwbaar is ten opzichte van het zoekgebied van achtereenvolgens 70%, 80% en 90%. In de figuren H5 tot en met H12 zijn de resultaten van deze posteriori check te vinden voor y2 en y3 (y1 is hier niet bijgevoegd omdat deze outputvariabelen voor alle resultaten hetzelfde beeld laat zien als voor y3).



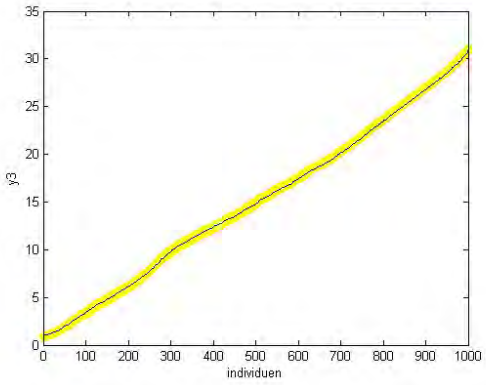
Figuur H5: y2 en y2+residuen voor 70%



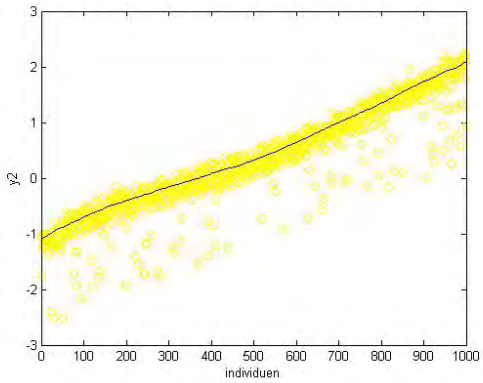
Figuur H6: y3 en y3+residuen voor 70%



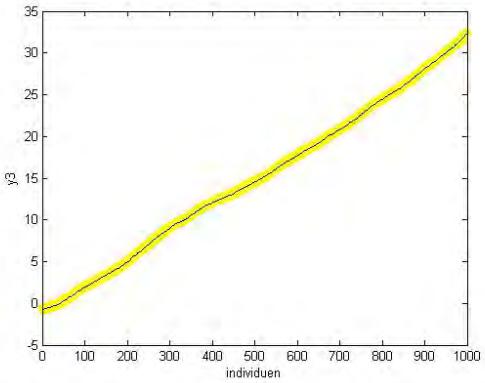
Figuur H7: y2 en y2+residuen voor 80%



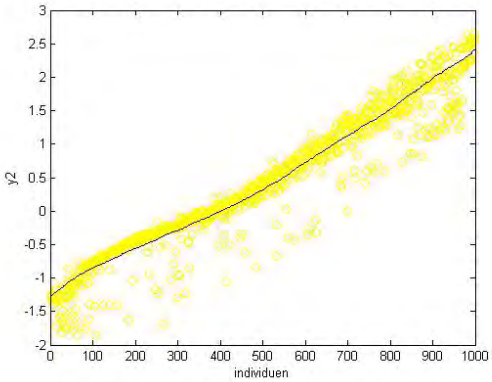
Figuur H8: y3 en y3+residuen voor 80%



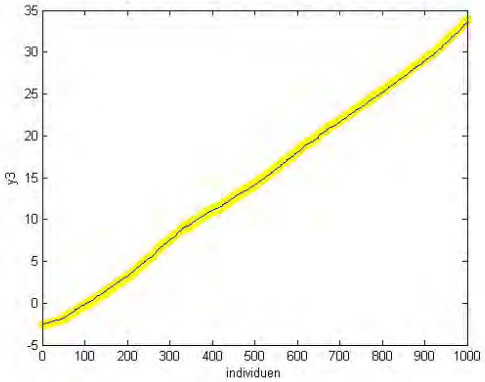
Figuur H9: y2 en y2+residuen voor 90%



Figuur H10: y3 en y3+residuen voor 90%

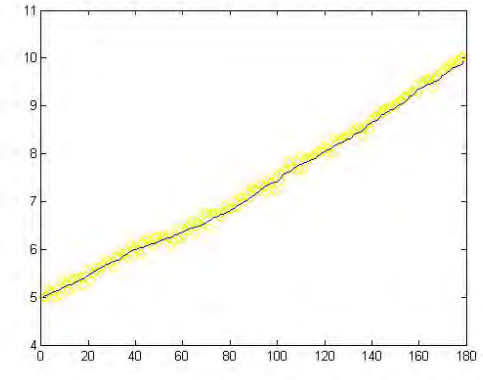


Figuur H11: y2 en y2+residuen voor 100%

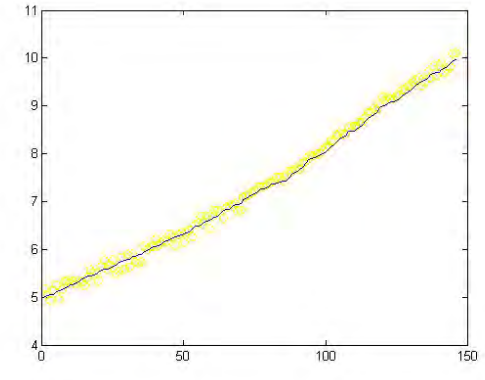


Figuur H12: y3 en y3+residuen voor 100%

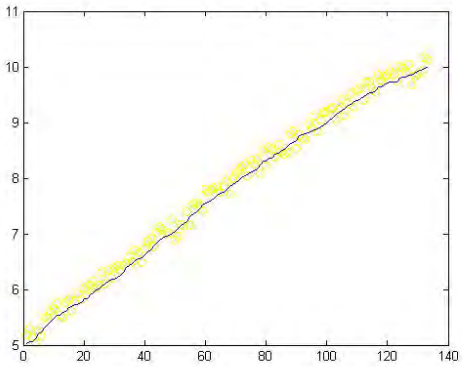
Duidelijk is te zien dat de waarden van de residuen voor y2 zich meer uitspreiden in de oplossingsruimte naarmate het zoekgebied groter wordt, dat wil dus zeggen dat de daadwerkelijke waarden van y2 in een groter zoekgebied waarschijnlijk meer zullen verschillen dan in het zoekgebied van 70%. Voor y3 is er geen verandering te zien, dit kan het gevolg zijn van het interval dat de waarden van y3 beslaat. Er zal daarom nu worden gekeken naar een kleiner gebied van de oplossingsruimte van y3, namelijk naar het gebied waar y3 de waarden tussen vijf en tien aanneemt.



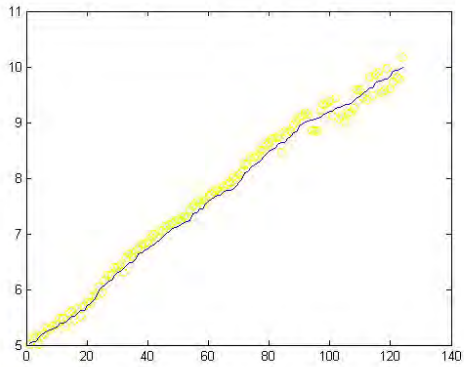
Figuur H13: y3 en y3+residuen voor 70%



Figuur H14: y3 en y3+residuen voor 80%



Figuur H15: y3 en y3+residuen voor 90%



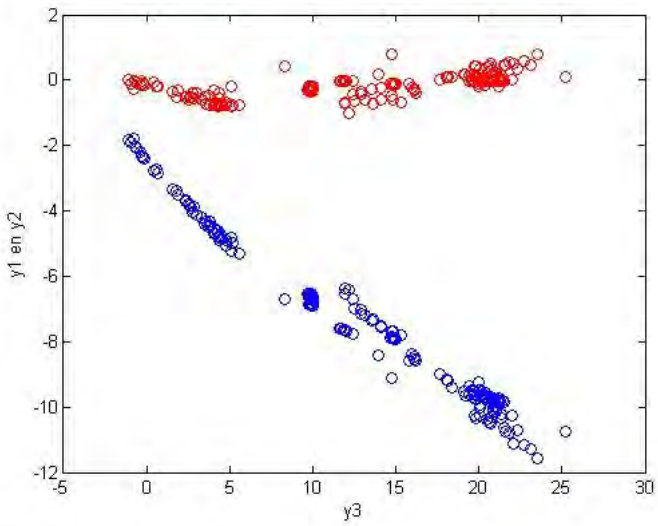
Figuur H16: y3 en y3+residuen voor 100%

Hoewel minder dan voor y2, beslaan hier nu ook de waarden voor de benaderde residuen van y3 een groter gebied bij een zoekdomein van 100%, dan bij 70%.

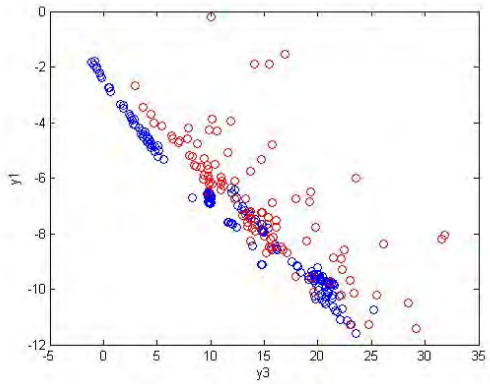
Appendix I

Hieronder wordt beschreven hoe er van alle resultaten die zijn behaald, zoals deze in het verslag en in de appendices zijn opgenomen, wordt gekomen tot zeven zeer interessante ontwerppunten.

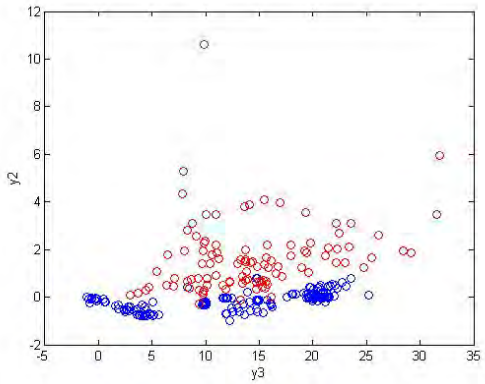
Aangezien er naar de allerbeste ontwerppunten gezocht wordt, zijn alle tot nu toe behaalde resultaten bij elkaar gevoegd, dit levert een dataset op van 15.600 rijen. Op deze enorme dataset is vervolgens MNSGA losgelaten voor één generatie om zo te kunnen bepalen wat de beste punten hiervan zijn, dat wil zeggen dat er wordt gekeken naar de ontwerppunten die rang 1 verkrijgen. Daar de ontwerppunten die gevonden zijn in het 70% zoekdomein de meest betrouwbare punten zullen zijn, zijn al deze ontwerppunten die behoren bij dit gebied en bovendien rang 1 blijken te hebben eruit gefilterd, hetgeen een set oplevert met 161 punten. Deze set is vervolgens aangevuld tot 250 ontwerppunten met de beste gevonden punten, gebaseerd op de rang en de crowding afstand waar MNSGA gebruik van maakt, van de overige zoekgebieden (dus van het 80%, 90% en 100% zoekdomein). In figuur I1 worden deze 250 punten weergegeven en de figuren I2 en I3 laten respectievelijk deze set van 250 punten voor y3 tegen y1 en y3 tegen y2 zien samen met de punten zoals deze gegeven zijn door de dataset (weergegeven door de rode punten).



Figuur I1: de 250 meest belovende ontwerppunten



Figuur I2: de 250 punten voor y3 tegen y1 met dataset



Figuur I3: de 250 punten voor y3 tegen y2 met dataset

Daar 250 ontwerppunten nog steeds een aanzienlijke set is, is er weer een selectie gemaakt uit deze 250 punten om zo een kleinere set, die de meest interessante ontwerppunten bevat, te kunnen realiseren.

Deze selectie is in eerste instantie gemaakt op basis van de waarden voor de verschillende outputs; alle ontwerppunten die een zeer extreem hoge waarde bleken te hebben voor één van de outputvariabelen zijn uit de dataset gehaald. Vervolgens is er gekeken naar de waarden van de bijbehorende benaderde residuen. Hierbij is gezocht naar de ontwerppunten die hoogstwaarschijnlijk een zo klein mogelijke fit fout bezitten. Tot slot is er ook nog gekeken naar de waarden van de ontwerpvariabelen. Alle ontwerppunten die veel inputvariabelen bezitten die op (of vlakbij) de rand liggen van het toegestane zoekgebied liggen zijn er uitgefilterd, omdat het ontwerppunt voor deze inputvariabelen waarschijnlijk door extrapolatie in plaats van interpolatie voorspeld is.
Na deze vergelijking blijkt er een set van zeven meest interessante ontwerppunten over te blijven, die een verbetering van globaal 5% zijn van de output ten opzicht van de gegeven dataset.

Literatuurlijst

1. www.nlr.nl
2. Informatie en Communicatie Technologie afdeling (2001); *Evaluatie van MOPS – Multi-Objective Parameter Synthesis*, NLR memorandum 2003-013
3. A.H. Aguirre, S.B. Rionda, G.L. Lizárraga, C.A.C. Coello; *IS-PAES: A Constraint-Handling Technique Based on Multiobjective Optimization Concepts*, EMO 2003 Conference, Faro
4. T.P. Bagchi; *Pareto-Optimal Solutions for Multi-objective Production Scheduling Problems*, EMO 2001 Conference, Zurich
5. M. Bijvank (2004); *Evolutionaire Methoden voor het Job Shop Scheduling Probleem*, Master thesis Vrije Universiteit, Amsterdam
6. C.A. Brizuela, R. Aceves; *Experimental Genetic operators Analysis for the Multi-objective Permutation Flowshop*, EMO 2003 Conference, Faro
7. D. Büche, S. Müller, P. Koumoutsakos; *Self-Adaptation for Multi-objective Evolutionary algorithms*, EMO 2003 Conference, Faro
8. C.A.C. Coello; *An Updated Survey of GA-Based Multiobjective Optimization Techniques*, Mexico
9. C.A.C. Coello (1996); *An empirical study of evolutionary techniques for multiobjective optimization in engineering design*
10. C.A.C. Coello (1999); *A comprehensive survey of evolutionary-based multiobjective optimization techniques*, Mexico
11. K. Deb, www.iitk.ac.in/kangal/resources.shtml
12. K. Deb, R.B. Agrawal (1994); *Simulated binary crossover for continuous search space*
13. K. Deb, M.Goyal; *A combined genetic adaptive search (GeneAS) for engineering design*
14. K. Deb, S.Gulati; *Design of truss-structures for minimum weight using genetic algorithms*
15. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan (2002); *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*, IEEE transactions on evolutionary computation, vol. 6 no. 2
16. J.L. Dorn, S.R. Ranjithan; *Evolutionary Multiobjective Optimization in Watershed Water Quality Management*, EMO 2003 Conference, Faro
17. D.E. Goldberg: *Genetic algorithms in Search, Optimization and Machine Learning,* 1st edition -1989 – Addison-Wesley Longman Publishing Co., Inc.Boston, MA, USA
18. B. Khare, X. Yao, K. deb; *Performance Scaling of Multi-objective Evolutionary Algorithms*, EMO 2003 Conference, Faro
19. S. De Kleermaeker (2000); *Multiobjective design optimisation and genetic algorithms: a literature survey*, NLR memorandum IW-2000-020
20. H. Pohlheim (2004); *Evolutionary Algorithms: Overview, Methods and Operators*, www.geatbx.com

21. I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologische Evolution,* Frommann-Holzboog verlag, Stuttgard, 1973

22. W.J. Vankan (2000); *Genetic algorithms in multiobjective design optimisation*, NLR memorandum 2000-027

23. W.J. Vankan, W.F. Lammen, J. Kos, R. Maas (2004); *Complementary approximate modeling in Matlab and Modelica,* Eurosim 2004 conference, Parijs

24. L. Willmes, T. Bäck; *Multi-criteria Airfoil Design with Evolution Strategies*, EMO 2003 Conference, Faro

25. E. Zitzler, M. Laumanns, L. Thiele; *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Swiss Federal Institute of Technology, Zurich