THE HYDRO CRUISERS

Expertisecentrum TIS Haagse Hogeschool, Delft

Een optimalisatiealgoritme voor de rijstrategie

Afstudeerstage bedrijfswiskunde
Maarten Loef





Expertisecentrum TIS Haagse Hogeschool, Delft

Een optimalisatiealgoritme voor de rijstrategie

Afstudeerstage bedrijfswiskunde
Maarten Loef

Studentnummer 30406
Periode: 7 mei t/m 31 augustus 2012

Paragraaf begeleider
Dhr Ir. P. Menger

Voorwoord

Dit verslag is geschreven als afstudeerscriptie voor de opleiding bedrijfswiskunde aan de Haagse Hogeschool. Ik ben deze opleiding in 2003 begonnen en ik ben dan ook blij dat ik na enkele tegenslagen deze opleiding uiteindelijk in 2012 af kan ronden.

Deze afstudeeropdracht vervul ik voor the HydroCruisers van het Expertisecentrum TIS aan de Haagse Hogeschool.

Het voordeel van afstuderen bij the HydroCruisers is dat er gewerkt wordt met een multidisciplinair team van studenten die anders niet met elkaar in contact komen. Hierbij werken studenten aan afstudeeropdrachten, stages, minors en gewoon projecten, maar allemaal aan hetzelfde hoofddoel: De Shell Eco Marathon.

Nadeel bij the HydroCruisers is dat er elk kwartaal een andere samenstelling is van studenten en hierbij kan er bij de overdracht kennis verloren gaan. De kennis die overgedragen wordt via bijvoorbeeld rapporten is verdeeld en staat herhaald in vele documenten. Dit maakt het inlezen in het algemene project zeer chaotisch. Dit rapport is zo geschreven dat er geen dingen overgenomen worden uit andere rapporten.

Hierbij wil ik de begeleiders van mijn afstudeerstage, dhr. Peter Menger (Expertisecentrum TIS) en mw. Cathy Liem (stagebegeleider) bedanken voor de ondersteuning die ze mij hebben gegeven gedurende dit traject.

Ik heb tijdens mijn stage voornamelijk zelfstandig moeten werken. Ik hoop dan ook dat mijn algoritme een nuttige bijdrage zal leveren aan de toekomstige races van de Shell Eco Marathon.

Maarten Loef, augustus 2012

Inhoud

Samenvatting 5
Summary 6
Hoofdstuk 1 - Inleiding 7
1.1 De Shell Eco Marathon 7
1.2 De uitdaging van de Shell Eco Marathon 7
1.3 The HydroCruisers 9
1.4 De opdrachtomschrijving 9
1.5 Leeswijzer 9
Hoofdstuk 2 - Het basisalgoritme 10
2.1 De systeemdefinitie 10
2.2 Het globale ontwerp voor het programma voor de rijstrategie 11
2.3 Waarom Java? 11
2.4 Fysische probleemstelling 12
2.5 Het basisalgoritme 13
2.6 De stopcriteria 14
Hoofdstuk 3 - Het algoritme toetsen op eenvoudige banen 15
3.1 Aannamen voor het toetsen van het basisalgoritme 15
3.2 De baanbeschrijvingen 15
3.2.1 De rechthoekige baan 15
3.2.2 Baanbeschrijving cirkelvormige baan 16
3.3 De exacte berekeningen van minimale arbeid 17
3.3.1 De exacte, minimale arbeid op de rechthoekige baan 17
3.3.2 De exacte, minimale arbeid op de cirkelvormige baan 19
3.4 De minimale arbeid berekend met behulp van het algoritme 20
3.4.1 Algoritmebeschrijving 20
3.4.2 Het algoritme uitgevoerd 21
3.5 De uitkomsten van het algoritme vergeleken met de exacte berekening 22

Hoofdstuk 4 - Het algoritme uitgewerkt en uitgebreid 23
4.1 Vectoren in het kort 23
4.2 Het inlezen van het circuit 24
4.3 De relatieve windhoek en de meeste tegenwind 24
4.4 De positie van de auto en de basissnelheid voor iteratiestap 0 24
4.5 De benodigde kracht vanuit de snelheid 25
4.6 De versnelling (Stop & Go) 26
4.7 De optimale snelheid in de bochten 26
4.8 De iteratiestappen 27
4.9 De resultaten tonen 27
Hoofdstuk 5 - Het circuit van de Ahoy' Rotterdam 28
5.1 De verdeling van het circuit 28
5.2 De rijstrategie 30
5.3 De uitvoer van het programma 30
5.3.1 De nieuwe uitvoer van de rechthoekige baan 30
5.3.2 De uitvoer van het circuit van de Ahoy' Rotterdam 31
Hoofdstuk 6 - Conclusie 32
Hoofdstuk 7 - Aanbevelingen 33
7.1 De database en de weersomstandigheden 33
7.2 Variabele beginpositie 33
7.3 Versnellen en uitrollen 33
7.4 Bochten- en inhaalstrategie 34
7.5 Uit het lokale minimum ontsnappen 34
7.6 De code splitsen 34

Samenvatting

De Haagse Hogeschool doet sinds 2007 mee met de Shell Eco Marathon met het team “The HydroCruisers”. De Shell Eco Marathon is een race waar een parcours afgelegd dient te worden binnen een gestelde tijd. Hierbij is het de bedoeling zo zuinig mogelijk te rijden. The HydroCruisers doen mee in de “Urban Concept” klasse, waarbij de situatie op de weg in steden wordt gesimuleerd.

Naast het ontwerp en de technische aspecten van de auto heeft de rijstrategie ook een grote invloed op het energieverbruik. Tot op heden is deze strategie opgezet met vuistregels en is nauwelijks wetenschappelijk onderbouwd.

Ik wil tijdens deze afstudeerstage dan ook het begin maken van een onderbouwd algoritme die rekening houdt met de omgevingsfactoren en hieruit een optimale rijstrategie berekent.

Hierbij hebben we een algoritme ontwikkeld dat rekening houdt met de rol- en windwrijving en het versnellen van de auto. Dit algoritme is flexibel als het gaat om de baan waar op gereden moet worden. Tevens hebben we dit toegepast op de baan van de Shell Eco marathon van de jaren 2012-2014, de Ahoy’ in Rotterdam, Nederland.

We hebben dit algoritme getoetst op eenvoudige banen, zodat we de uitkomst kunnen toetsen aan exacte berekeningen. Hieruit blijkt dat het mogelijk is een circuit te modelleren met een algoritme.

Om zuinig te kunnen rijden zijn we van de natuurkundige aanname uitgegaan dat er bij tegenwind zo langzaam mogelijk gereden wordt. Hierdoor zoekt het algoritme de baanvakken die relatief ten opzichte van de auto de meeste tegenwind heeft en verlaagt die de snelheid. De tijd die dit extra kost wordt gecompenseerd met de baanvakken waar de meeste wind mee is. Als alle baanvakken tegenwind hebben neemt het algoritme deze met de minste tegenwind.

In principe werkt het algoritme naar behoeven, maar er is een mogelijkheid dat deze niet de meest optimale rijstrategie berekent. Dit is onder andere het geval als de te compenseren tijd te groot is voor een baanvak, zodat er daar te hard gereden dient te worden.

Al met al is het ontwikkelde algoritme een goede start voor een theoretische rijstrategie om zo zuinig mogelijk te rijden. Het algoritme is er klaar voor om uitgebreid te worden.

Summary

The Hague University participated in the Shell Eco Marathon since 2007 with the team “The HydroCruisers”. The Shell Eco Marathon is a race where the team has to complete a track within a certain amount of time. This race has the intention to finish as energy efficient as possible. The HydroCruisers are participating in the “Urban Concept” class, where the real situation on the roads of cities are simulated.

Not only the design and the technical aspects of the car, but also the driving strategy is influential to the use of energy. Until now that driving strategy has always been a rule of thumb, without any scientific reasoning.

During my internship I want to focus on developing an algorithm that is taking into account the environmental factors and calculate an optimal driving strategy”.

Thus we have developed an algorithm that takes into account the roll- and air friction and the acceleration of the car. The algorithm can handle a diversity of tracks. We inserted the track of the Shell Eco Marathon of the years 2012-2014, The Ahoy’ in Rotterdam, the Netherlands.

We tested our algorithm on some easier tracks, so we could compare the outcome with the exact calculations. The results of the test runs were that is possible to modulate a race circuit with an algorithm.

The base of the algorithm is the physical assumption that if you are driving crosswind, you want to minimize your speed. Therefore the algorithm searches for the sections where the influence of the crosswind is the most and in those section the speed is lowered. The extra time this lowering costs has to be compensated in the sections where the wind is the least problem.

The algorithm works with these few environmental factors, but there is a possibility that the calculated driving strategy is not the most optimal one. This can happen when the compensated time is too high for the length of the section. In that case the necessary speed of that section becomes too high.

This algorithm is a good start for the theoretical driving strategy to drive as frugal as possible. This algorithm is ready to be expanded.

Hoofdstuk 1 - Inleiding

1.1 De Shell Eco Marathon

In deze paragraaf is een selectie gemaakt van de website van de Shell(*).

De oorsprong van het concept ‘zo ver mogelijk rijden met zo weinig mogelijk brandstof’ gaat ver terug in de tijd. In 1930 lanceerde het Shell-laboratorium in het Amerikaanse Illinois het idee van een ‘fuel economy’-competitie. Pas in 1977 werd het idee concreet uitgewerkt in de vorm van een wedstrijd. In dat jaar werd in Groot-Brittannië namelijk de voorloper van de Eco-marathon, de Shell Mileage Marathon georganiseerd. Op Nederlandse bodem is tussen 1978 en 1981 de Shell Zuinigheid Marathon georganiseerd. Het Nederlandse record uit deze periode is 783 kilometer op 1 liter benzine, behaald door de TH Twente.

De eerste Europese Shell Eco-marathon in de huidige opzet, vond in 1985 plaats op het Paul Ricard-circuit in Le Castellet in Frankrijk. Twintig teams verschenen aan de start. De winnaars uit Zwitserland wisten omgerekend 680 kilometer op één liter benzine te rijden. In 2005 vestigden wederom Zwitserse deelnemers het huidige record van 3836 kilometer. Sinds de 25ste editie vorig jaar vindt de duurzaamheidscompetitie plaats in het meer centraal gelegen Duitse Lausitz.
Het spektakel groeit jaarlijks in omvang. In de afgelopen 25 jaar heeft het evenement meer dan 40.000 jongeren tussen de 13 en 26 jaar bij elkaar gebracht: een groep van (toekomstige) duurzame denkers en ontwikkelaars. Inmiddels is de zoektocht naar duurzame mobiliteit ook naar andere continenten uitgebreid. In 2008 werd de eerst Amerikaanse Shell Eco-marathon gelanceerd. In 2010 vond in juli de eerste Aziatische editie plaats.

De Shell Eco-marathon neemt bij een groot aantal deelnemende scholen en universiteiten een belangrijke plaats in het onderwijsprogramma in. Dat blijkt uit het grote aantal instellingen dat ieder jaar aanwezig is met een nieuw team.

In vijfentwintig jaar is er veel veranderd in de race. Zo werd op verzoek van deelnemers en toeschouwers de Urban Concept-klasse aan de wedstrijd toegevoegd. Hiermee wordt een meer realistische stadssituatie benaderd. Ook zijn er in de loop van tijd steeds nieuwe brandstoffen bijgekomen. De deelnemers richten zich daarnaast steeds meer op het groene karakter van de voertuigen.

1.2 De uitdaging van de Shell Eco Marathon

In deze paragraaf is een selectie gemaakt van de website van de Shell(*).

De wagens waar de teams mee rijden kunnen niet zomaar de openbare weg op. Ze zijn klein van stuk, bieden ruimte aan slechts 1 bestuurder en gaan niet veel harder dan de minimaal gestelde 25 kilometer per uur.

Om energie te kunnen besparen wordt er bij de ontwikkeling van het voertuig aandacht besteed aan aerodynamica, energie efficiëntie, gewicht en een zuinige rijstijl. Aandacht voor al deze elementen levert in de testomgeving van de Shell Eco-marathon verbazingwekkende resultaten op. Menig automobilist zou met een fractie van de resultaten meer dan tevreden zijn.

(*) De directe link naar deze website is te vinden in bijlage II

Wellicht is dit dichterbij dan we denken. De ontwikkelingen en innovaties zullen in de toekomst mogelijk hun weg vinden in onze voertuigen. Gerenommeerde universiteiten en hogescholen zijn tenslotte bij de competitie betrokken. Daarnaast leren de deelnemende studenten en scholieren op een bijzonder positieve manier nadenken over duurzame mobiliteit. Wie weet welke oplossingen we in de toekomst van hen kunnen verwachten? De verwachtingen liggen hiervoor in het inzicht in gewicht, aerodynamica, energie efficiëntie en een zuinige rijstijl.

Inzicht in gewicht
Onze huidige auto's zijn zwaar. Dit is het gevolg van de steeds hogere veiligheidseisen die gelden voor personenvoertuigen. In de afgelopen 20 jaar is het gemiddelde gewicht van een auto verdubbeld. De Shell Eco-marathon voertuigen zijn licht. De 'body' is veelal van kunststof (hars of carbonfiber) gemaakt. Er zijn teams die alles, tot aan de wielen toe, van composieten (een vederlicht maar ijzersterk materiaal) maken.

Aerodynamica
Een lage luchtweerstand heeft een enorm positief effect op het energieverbruik. Daarom zijn veel voertuigen in de prototype klasse gebaseerd op de ideale vorm van een vallende waterdruppel. De ombouw van de wagens wordt ontworpen door te experimenteren met schaalmodellen en testsessies in de windtunnel. Glad materiaal doet de rest. Tenslotte is veel winst te behalen door bijvoorbeeld de wielen te integreren in de ombouw van de wagen.

Energie efficiëntie
De gebruikte motoren zijn over het algemeen erg klein. Voor de vloeibare brandstoffen zijn er brommermotoren die voldoende vermogen hebben voor dergelijke lichte voertuigen en lage snelheden. Een wrijvingloze overbrenging van de beweging van de motor naar de wielen levert ook besparing op, al is dit hoogstwaarschijnlijk niet realistisch. De brandstofcel van de waterstof aangedreven voertuigen moet een zo hoog mogelijk rendement hebben. Bij sommige ontwerpen wordt een deel van de waterstof niet omgezet in elektriciteit. Winst wordt hier behaald door in het ontwerp recycling op te nemen van de waterstof naar de brandstofcel.

Zuinige rijstijl
Naast de eigenschappen van de auto zelf halen de teams veel voordeel uit een uitgekiend rijgedrag. Door in het rijgedrag rekening te houden met de lokale omstandigheden op het circuit kan er veel voordeel behaald worden. Het circuit heeft veel bochten. Een goede rijstrategie vraagt op het juiste moment bij een bocht gas te geven, uit te laten rollen en zodoende gebruik te maken van de middelpuntvliedende kracht. Vooraf gemaakte wiskundige berekeningen leveren de bestuurder een optimale handleiding voor het gehele circuit.

1.3 The HydroCruisers

In 2006 is het idee ontstaan om vanuit de Haagse Hogeschool mee te doen met de Shell Eco Challenge. Hieruit zijn the HydroCruisers in het leven geroepen. Sinds 2007 doen ze mee met deze wedstrijd en met succes. In het eerste jaar wonnen ze de race en hebben tot en met 2011 altijd een podiumplaats veroverd. Dit laatste jaar was het resultaat 104 km/kWh, wat vergeleken met een elektrische stadsauto 14 maal zuiniger is. Dit jaar (2012) echter hebben the HydroCruisers door technische problemen de race niet op tijd uit kunnen rijden.

Het team van the HydroCruisers bestaat uit een multidisciplinaire samenstelling studenten van de Haagse Hogeschool. Eind mei, net na de SEM 2012, zijn de meeste studenten klaar met hun (afstudeer)opdracht en is er nog een team van rond de 8 studenten actief. Deze mensen zijn hun opdracht nog aan het afronden. Het volgende kwartaal beginnen er namelijk alleen maar nieuwe mensen.

Het team is opgedeeld in management, engineering en een onderliggende R&D groep. De ontwikkeling van de rijstrategie behoort tot het laatste team en kan dus ook voor het andere SEM team, the ElectroCruisers, gebruikt worden.

1.4 De opdrachtomschrijving

Zoals in paragraaf 1.2 is beschreven is een van de uitdagingen van de Shell Eco Marathon het ontwikkelen van een zuinige rijstijl. Tot op heden wordt de rijstrategie van tevoren bepaald met vuistregels als leidraad. Wat er mist is een analytisch systeem, die met de omgevingsomstandigheden rekening houdt en hieruit automatisch de optimale rijstrategie bepaalt.

Deze uitdaging is te groot en ingewikkeld om door één persoon binnen een (afstudeer)periode op te lossen. Aangezien er net begonnen is met de opzet van deze nieuwe soort rijstrategie houden we de opdrachtomschrijving breed. Voor dit algoritme nemen we de technische aspecten en de aerodynamica van de auto als een gegeven. Dit wordt beschreven in de volgende paragraaf.

De opdrachtomschrijving luidt als volgt: “ontwikkel een algoritme die rekening houdt met de omgevingsfactoren en hieruit een optimale rijstrategie berekent”.

De door de opdrachtgever beoogde oplossing is een implementatie van het algoritme in een softwarepakket dat geïntegreerd is in een programma dat data logt, analyseert, interpreteert en uiteindelijk een advies genereert voor de rijstrategie.

1.5 Leeswijzer

De opbouw van dit verslag is als volgt:

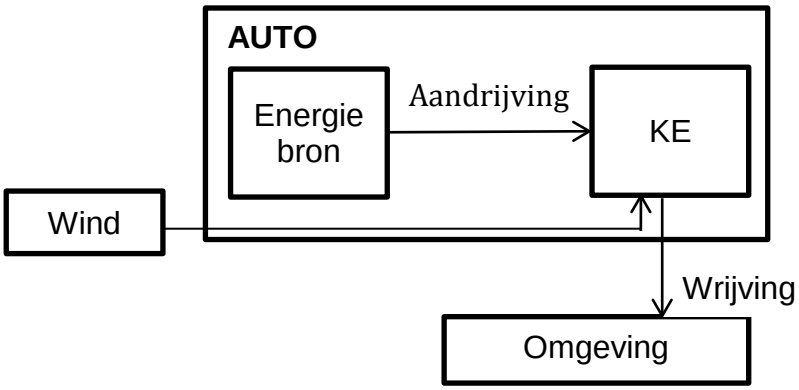
Hoofdstuk 2: het basisalgoritme wordt besproken
Hoofdstuk 3: het basisalgoritme wordt getoetst
Hoofdstuk 4: de uitwerking van en toevoegingen op het algoritme
Hoofdstuk 5: de reële situatie van de Ahoy’ Rotterdam
Hoofdstuk 6: de conclusie
Hoofdstuk 7: de aanbevelingen

Hoofdstuk 2 - Het basisalgoritme

2.1 De systeemdefinitie

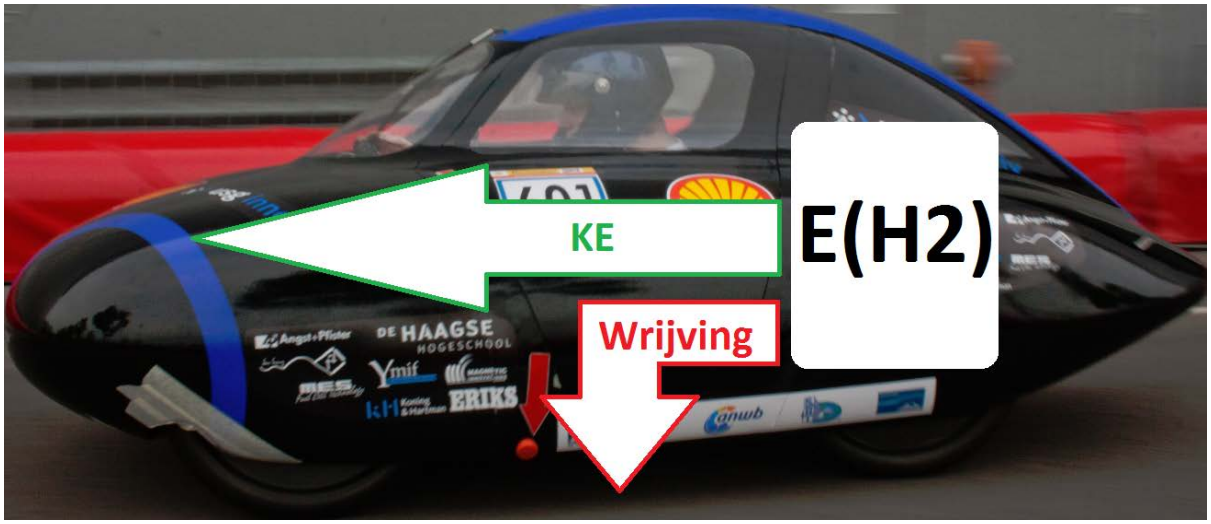
Voordat we kunnen beginnen dienen we het systeem te definiëren. Het systeem is de verzameling van objecten die samen de auto vormen samen met de weerstanden (o.a. wrijving en wind). Het systeem bevat onder andere de chauffeur, energiebron, brandstofcel, de wagen zelf en de weerstandseigenschappen.

De waterstof levert de energie dat door de motor omgezet wordt in kinetische energie (KE). Een deel van deze energie gaat verloren doormiddel van wrijving. Dit is in figuur 2.1 visueel gemaakt. Dit is als volgt schematisch weergegeven:



Figuur 2.1 De systeemdefinitie

Zoals in figuur 2.1 te zien is kan de wind zowel kinetische energie leveren als wrijving genereren. Hier komen we later op terug. Globaal levert dit de situatie op zoals in figuur 2.2. Versimpeld weergegeven gaat de energie die de waterstof (H2) levert naar wrijving en kinetische energie:

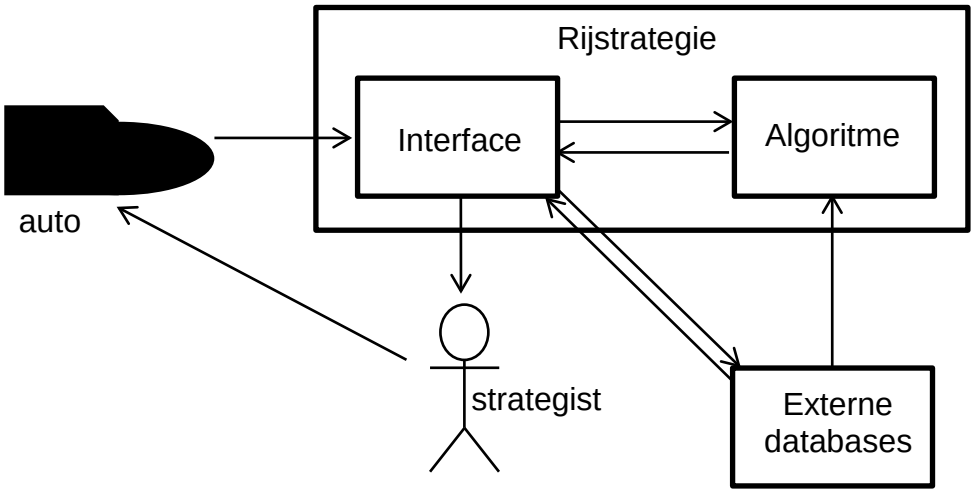


Figuur 2.2 Een versimpelde weergave van het energieplaatje

Deze definiëring zorgt ervoor dat we bijvoorbeeld de interne, mechanische wrijving als gegeven beschouwen en niet meenemen in het energieplaatje. De andere, externe factoren (bijvoorbeeld frontale oppervlak en wrijvingscoëfficiënten) worden wel meegenomen.

2.2 Het globale ontwerp voor het programma voor de rijstrategie

Het algoritme is een onderdeel van de software dat een rijstrategie weergeeft aan de gebruiker van het programma. In het algemeen de persoon die de bestuurder instrueert, vanaf nu strategist genoemd. Het opslaan van de omgevingsfactoren samen met het weergeven van deze factoren en de uitkomst van het algoritme vindt plaats in de interface. Het totale systeem geeft de optimale rijstrategie weer aan de strategist, die deze dan doorgeeft aan de bestuurder. Schematisch kan dit als volgt voorgesteld worden:



Figuur 2.3 schematische weergave van het totale systeem

Het gedeelte “interface” in de rijstrategie, beschreven in figuur 2.3 wordt elders gemaakt en in de toekomst worden deze twee samengevoegd tot een enkel programma. De opzet hiervan is beschreven in het verslag “Strategie voor the HydroCruisers”[(*)].

De “externe databases” zijn onder andere de baanbeschrijving, parameters van de auto, weersomstandigheden en prestaties tot op heden.

2.3 Waarom Java?

Om het algoritme te kunnen implementeren moeten we eerst beslissen in welke programmeertaal de code geschreven gaat worden. Hierbij moet transparantie en overzicht de boventoon hebben. Het ook leesbaar te zijn voor iedereen die enige kennis van programmeren en het moet makkelijk op een embedded systeem gezet kunnen worden. Verder is bij de keuze ook rekening gehouden met de aanwezige programmeerervaring. Deze eisen komen samen in de programmeertaal Java.

Java is een platformonafhankelijke taal wat het makkelijk maakt om door iedere computer gelezen te kunnen worden. De taal is open source en dus door iedereen vrij te gebruiken en de taal kan geschreven worden in het gratis te verkrijgen programma Netbeans IDE. Tevens is er van Java op dit moment de meeste kennis in huis. Dit programma wordt dan ook gebruikt om te programmeren.

[(*)] Zie de literatuurlijst in bijlage II

2.4 Fysische probleemstelling

Nu we het systeem gedefinieerd hebben willen we de fysische probleemstelling behandelen. Deze formules zijn nodig om het uiteindelijke algoritme te kunnen gaan schrijven. Zoals in paragraaf 2.1 is besproken gaat er energie verloren door wrijving. De opdracht is de totale arbeid te minimaliseren. Dit natuurlijk onder de voorwaarde dat de auto de afgelegde weg binnen een bepaalde tijd uitrijdt.

Om een rijstrategie te kunnen bedenken moeten we eerst van te voren het te berijden circuit op delen in logische baanvakken, zoals bijvoorbeeld rechte stukken en bochten.

Daarna moeten we per baanvak de wrijving bepalen die overwonnen moet. Deze wrijving bepaalt de arbeid van de motor die nodig is om een bepaalde snelheid te kunnen rijden. Als we eenmaal deze arbeid hebben gedefinieerd kunnen we gaan optimaliseren zodat we de ideale snelheid per baanvak kunnen berekenen. Als basis nemen we alleen de rol- en windwrijving.

Er zijn meerde manieren om de wrijving die op de auto werkt te berekenen. Dit kan bijvoorbeeld door te kijken naar de snelheden of te kijken naar de krachten. In ons geval hebben we er voor gekozen alleen te kijken naar het snelheidsspectrum. Hierbij hebben we de volgende formules uit de mechanica nodig, waarbij de symbolen in bijlage I worden uitgelegd:

Algemene formules uit de basismechanica:

Arbeid: $W = \int \vec{F}\,\overrightarrow{ds}$ (2.1)

Kracht: $\vec{F} = m \cdot \vec{a}$ (2.2)

De formules voor de weerstanden en arbeid:

Rolweerstand: $\vec{F}_{rol} = \eta mg$ (2.3)

Luchtweerstand: $\vec{F}_L = \frac{1}{2}\rho A C_w (v_v - \cos\theta\, v_w)^2$ (2.4)

Arbeid: $W = s \cdot \left(\vec{F}_{rol} + \vec{F}_L\right)$ (2.5)

Als we de formules 2.3 en 2.4 invullen in formule 2.5 geeft dit ons de volgende formule, waarbij de arbeid van alle baanvakken (N) is gesommeerd, beginnende bij het eerste baanvak (i=0):

$$W_{totaal} = \sum_{i=0}^{N-1} s_i \left(\eta mg + \frac{1}{2}\rho A C_w \left(v_{v(i)} - \cos\theta\, v_{w(i)}\right)^2\right)$$

Deze formule levert ons de totale arbeid op. Dit is echter onder de voorwaarde dat er binnen de maximaal gestelde tijd (t_{max}) over de eindstreep gereden dient te worden. De formule voor de totaal gereden tijd is:

$$t_{max} = \sum_{i=0}^{N-1} v_i s_i \quad (2.6)$$

Dit geeft ons een algemeen beeld welke formules er nodig zijn voor het algoritme. Nu we deze hebben bepaald kunnen we het algoritme gaan bespreken.

2.5 Het basisalgoritme

In deze paragraaf bespreken we stap voor stap het globale idee van het algoritme. In hoofdstuk 4 gaan hier we verder op. We gaan er vanuit dat de auto altijd direct de benodigde snelheid rijdt, zodat we geen rekening hoeven te houden met de tijd die versnellen en afremmen kost.

Stap 1 – Het inlezen van de gegevens

Het programma leest de parameters van de auto, het circuit en de weersomstandigheden in. Ook de in het verleden behaalde resultaten worden uitgelezen. De relatieve windsnelheden en de basissituatie worden op alle baanvakken direct berekend.

Stap 2 – De baanvakken selecteren waar de snelheid aangepast gaat worden

Het algoritme gaat alle baanvakken langs en selecteert hieruit de baanvakken waar de relatieve windsnelheid het meest tegenwerkt (tegenwind) en het meest meewerkt (wind mee). Hierbij kunnen voor beide gevallen 1 of meerdere baanvakken geselecteerd worden.

Stap 3 – De snelheid aanpassen

In de geselecteerde baanvakken wordt eerst de totaal benodigde tijd uitgerekend. Daarna wordt de snelheid in de baanvakken met de meeste tegenwind met een vooraf gestelde factor verlaagd. Deze verlaging betekent een verlies in tijd en dit moet gecompenseerd worden met de andere geselecteerde baanvakken om toch op tijd over de eindstreep te rijden. Hierdoor wordt deze verloren tijd evenredig verdeeld over de baanvakken met de meeste wind mee en zodoende wordt daar de snelheid verhoogd.

Stap 4 – Arbeid berekenen

Nu wordt de arbeid berekend van de nieuwe situatie.

Stap 5 – Stopcriteria

In paragraaf 2.6 bepreken we enkele stopcriteria. Als de nieuwe situatie voldoet aan een van deze stopcriteria, wordt deze situatie over boord gegooid en is de oude situatie dus het meest optimaal. Als er geen stopcriteria gehaald worden wordt de nieuwe situatie opgeslagen en begint het algoritme weer bij stap 2.

Stap 6 – Het tonen van de gewenste snelheid per baanvak

De functie van het algoritme is een zo zuinig mogelijke rijstrategie berekenen. Als er een optimum is bereikt moet het algoritme de snelheden per baanvak terugkoppelen aan de gebruiker van het programma. Hierna is de functie van het programma klaar en stopt het algoritme.

2.6 De stopcriteria

Om het algoritme te laten beëindigen hebben we stopcriteria nodig. Dit zijn zowel logische als aangenomen voorwaarden waar het eindresultaat aan dient te voldoen. Deze zijn in ons algoritme als volgt gedefinieerd:

- Een iteratiestap levert nauwelijks een verbetering op

  Computers kunnen bijna eindeloos nauwkeurig berekeningen voor optimalisaties maken. Echter zijn de verbeteringen die er uitgerekend worden niet altijd groot genoeg om van enig nut te zijn. Er kan gekozen worden voor een procentuele of exacte toetsing van de berekende arbeid. Als deze verbetering kleiner is dan deze factor dan heeft het voortzetten van het algoritme geen toevoeging op de optimalisatie.

- De snelheden op de baanvakken moeten realistisch zijn

  Als een berekende kracht in een van de baanstukken bijvoorbeeld negatief wordt dan betekent het, dat om de tijd te compenseren, er terug in de tijd een kracht geleverd dient te worden. Dit is fysiek niet mogelijk, dus als dit voorkomt stopt het algoritme. Hier komen we in paragraaf 3.2.1 met een voorbeeld op terug.

- Minimaal benodigde of maximum haalbaar vermogen

  Minimaal benodigde vermogen: Het algoritme berekent te ideale situatie en geeft uiteindelijk als terugkoppeling het minimaal benodigde vermogen dat nodig is om de rijstrategie te kunnen uitvoeren

  Maximum haalbaar vermogen: De gebruiker voert in de specificaties van de auto een maximum vermogen in. Het algoritme neemt deze variabele mee als stopcriterium en gaat hier niet overheen

  De ideeën voor het vermogen zijn op dit moment nog niet geïmplementeerd in het algoritme.

In een later stadium kunnen andere stopcriteria toegevoegd worden aan de code van het algoritme.

Hoofdstuk 3 - Het algoritme toetsen op eenvoudige banen

In paragraaf 2.5 hebben we een beschrijving gemaakt van hoe we het algoritme gaan toepassen. Om de werking hiervan te kunnen toetsen gaan we dit algoritme toepassen op eenvoudige banen, waar we tevens een exacte berekening van de minimale arbeid kunnen maken.

Als de uitkomst van het algoritme overeenkomt met de exacte berekening kunnen we stellen dat dit algoritme betrouwbaar is en kunnen we deze uit gaan breiden.

3.1 Aannamen voor het toetsen van het basisalgoritme

We nemen aan dat alleen de rol- en windwrijving invloed hebben op de arbeid die de motor moet verrichten. De auto rijdt aan het begin van een baanvak direct de gewenste snelheid, zodat er geen rekening gehouden wordt met onder andere de versnelling.

In paragraaf 2.4 hebben we de rol- en windwrijving gedefinieerd samen met de benodigde arbeid. We gebruiken eerst een rechthoekige baan en daarna een baan met een perfecte cirkel als vorm. Bij deze situaties kunnen we de benodigde snelheid om zo min mogelijk arbeid te verrichten exact uitrekenen. Hierdoor kunnen we de uitkomst van het algoritme toetsen aan deze berekende waarden.

In verband met het toetsen van het algoritme nemen we de volgende waarden voor de variabelen aan:

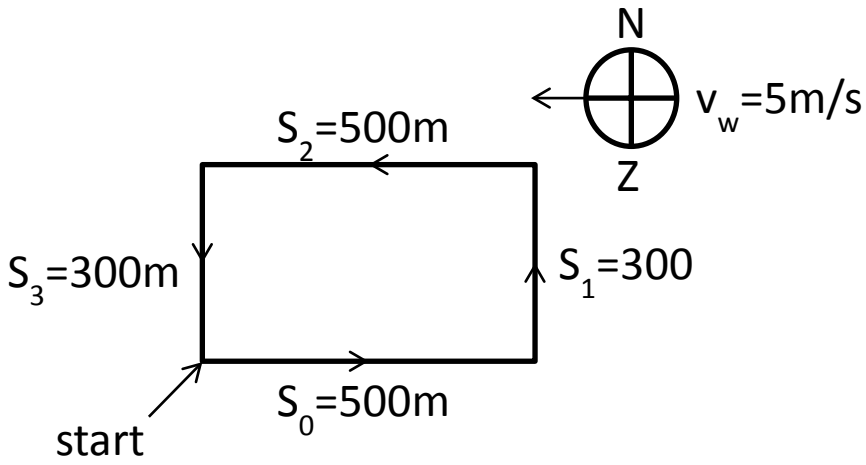
variabele	waarde
η	0.001
m	150
g	9.81
ρ	1.29
A	1
C_w	1
v_w	5
θ_w	π

In de praktijk worden deze waarden uitgelezen uit een externe database.

3.2 De baanbeschrijvingen

3.2.1 De rechthoekige baan

We nemen een rechthoekige baan van s_{totaal} meter lang waar we t_{max} seconden over mogen doen. We verdelen deze baan op in 4 gedeeltes waarbij we de volgende afstanden definiëren: $s_0 = s_2 = 500m$ en $s_1 = s_3 = 300m$ zodat de totale lengte van de baan $s_0 + s_1 + s_2 + s_3 = 1600m$ is. De start is op baanvak 0, we rijden tegen de klok in en de relatieve windrichting is π (tegenwind) en de maximale tijd voor de baan stellen we op $t_{max} = 240s$. De gemiddelde snelheid is dus $\frac{1600}{240} = \frac{20}{3} m/s$. Deze baan hebben we gevisualiseerd in figuur 3.1 op de volgende pagina.



Figuur 3.1 De eenvoudige, rechthoekige baan

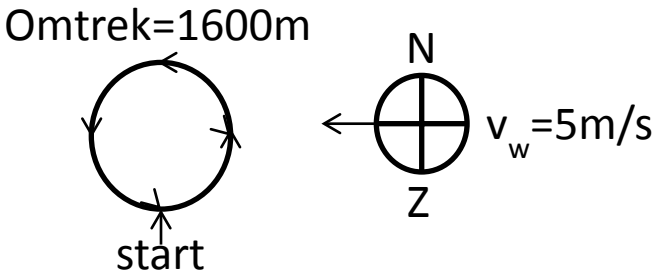
In de aannames hebben we gesteld dat de windhoek π rad is. Baanvak 0 bevindt zich in een hoek van 0 rad. Dit betekent dat als we de formule 2.4 invullen voor elk baanvak dat in de baanvakken 0 en 3 er geen windinvloed is. Dit op zijn beurt betekent dat we deze snelheden met rust laten. Gevolg hiervan is dat de snelheden in de overige twee baanvakken rechtstreeks afhankelijk van elkaar zijn. Deze afhankelijkheid is te beschrijven met formule 3.1:

$$v_2 = \frac{s_2}{t_0 + t_2 - \frac{s_0}{v_0}} \tag{3.1}$$

3.2.2 Baanbeschrijving cirkelvormige baan

Bij de volgende simplistische baan hebben we een perfecte cirkel genomen die een omtrek heeft van 1600 meter en er mag wederom 240 seconden over gedaan worden. Hierbij starten we, net zo als de eerdere baan met een relatieve windhoek van π rad en gaan we tegen de klok in rijden. We verdelen deze baan in 240 baanvakken van ieder 1 seconde.

We hebben deze cirkelvormige baan gevisualiseerd in figuur 3.2.



Figuur 3.2 De eenvoudige, cirkelvormige baan

3.3 De exacte berekeningen van minimale arbeid

In deze paragraaf bespreken we de exacte berekening van de arbeid op de beide simplistische banen. Hierdoor creëren we toetsingswaarden zodat we het algoritme kunnen testen.

3.3.1 De exacte, minimale arbeid op de rechthoekige baan

Omdat de arbeid (W) in deze situatie volledig afhankelijk is van de snelheid in baanvak 0 (v_0) kunnen we deze exact uitrekenen. De aannames eerdere meegenomen houden we op baanvak 1 en 3 de snelheid gelijk. Per iteratiestap wordt dus de snelheid v_0 verlaagd.

Aangezien de tijd die er over baanvak 1 en 3 gedaan wordt hierdoor altijd hetzelfde is (namelijk 45 seconden) kunnen we de snelheid op baanvak 2 direct afleiden uit de snelheid in baanvak 0. De totale tijd die we hebben voor de baanvakken 0 en 2 is $t_{max} - t_1 - t_3 = 150$ s. We definiëren de aangepaste formules van arbeid per baanvak.

$$W_0 = s_0\left(\eta mg + \tfrac{1}{2}\rho A C_w (v_0 + v_w)^2\right)$$
$$W_1 = s_1\left(\eta mg + \tfrac{1}{2}\rho A C_w {v_1}^2\right)$$
$$W_2 = s_2\left(\eta mg + \tfrac{1}{2}\rho A C_w (v_2 - v_w)^2\right)$$
$$W_3 = s_3\left(\eta mg + \tfrac{1}{2}\rho A C_w {v_3}^2\right)$$

Hierna vullen we de bekende waarden in en rekenen we de arbeid uit.

$$W_0 = 735.8 + 322.5(v_0 + 5)^2$$
$$W_1 = 9040.5$$
$$W_2 = 735.8 + 322.5\left(\frac{500}{150 - \frac{500}{v_0}} - 5\right)^2$$
$$W_3 = 9040.5$$

We zien hier inderdaad dat de arbeid in de baanvakken 0 en 3 onafhankelijk zijn van de snelheid v_0.

Nu bepalen we de vergelijking van de totale arbeid.

$$W_{totaal=} 19554.4 + 322.5(v_0 + 5)^2 + 322.5\left(\frac{500}{150 - \frac{500}{v_0}} - 5\right)^2$$

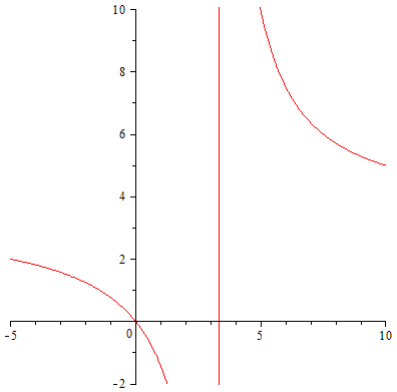
Nu differentiëren we deze vergelijking naar v_0 zodat we de nulpunten kunnen vinden en berekenen we de oplossing.

Nadat we deze berekeningen ingevoerd hebben in het rekenprogramma Maple, hebben we de volgende uitkomsten van v_0 gekregen:

$$\{v_0 = 5.357967742\}, \{v_0 = 2.533594075 + 2.5144581241i\}$$
$$\{v_0 = -5.357967742\}, \{v_0 = 2.533594075 - 2.5144581241i\}$$

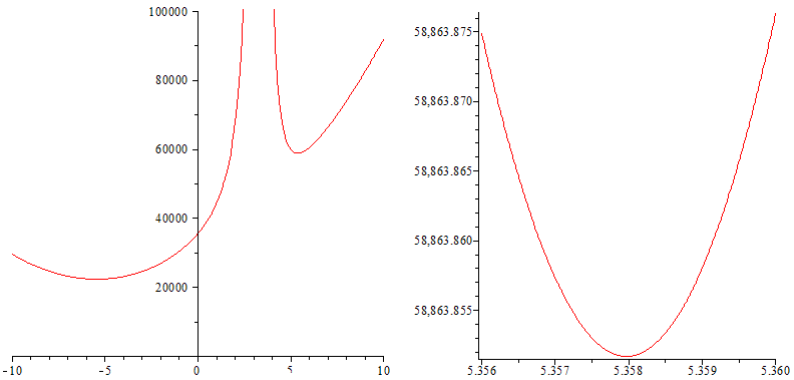
Twee van deze uitkomsten zijn imaginair en hebben dus geen reële betekenis. Voor de andere uitkomsten moeten we een test doen. Het blijkt dat als we v_1 en v_3 tegen elkaar afzetten dat er een punt is in deze vergelijking waar v_2 oneindig groot is. Dit punt wordt bereikt als t_0 even groot is als de totale tijd dat over de beide baanvakken gedaan mag worden. Dit punt is dus een verticale asymptoot. Als v_0 lager dan dit getal komt heeft de uitkomst van v_2 geen reële betekenis. Uitgerekend komen we op een snelheid van $v_0 = \frac{500}{150} = 3\frac{1}{3}$ m/s. Hieruit kunnen we concluderen dat het negatieve getal kleiner is dan deze snelheid en dus geen betekenis heeft.

In figuur 3.3 zien we de snelheidsvergelijking tussen v_3 en v_3 met de verticale asymptoot.



Figuur 3.3 de snelheden van v_0 op de horizontale as en v_2 op de vertical as

Nu weten we dus dat $v_0 = 5.36$ m/s. Deze berekende snelheid ingevuld komen we op een optimale arbeid van (afgerond) 58864 J. Dit hebben we visueel gemaakt in de grafieken in figuur 3.4.



Figuur 3.4 De arbeid afhankelijk van de snelheid v_0

In figuur 3.4 hebben we rechts een vergroting weergegeven van de berekende snelheid v_0, waar de berekende arbeid minimaal is.

3.3.2 De exacte, minimale arbeid op de cirkelvormige baan

Aangezien deze baan een perfecte cirkel beschrijft is de relatieve windhoek te beschrijven met een sinusfunctie. Ook in deze situatie rijden we tegen de wind in zo langzaam mogelijk, waardoor ook de snelheid met een sinusfunctie te beschrijven is. Zoals in paragraaf 3.2.2 besproken verdelen we deze baan in N baanvakken van elk een seconde.

We willen nog steeds de gemiddelde snelheid rijden om op tijd binnen te komen en krijgen hierdoor een snelheidsfunctie van $v(i) = -\cos(\theta_i) + v_{gem}$. De enige variatie die we op deze formule kunnen toepassen is de amplitude met een factor K vergroten. De functie voor de snelheid is dus als volgt:

$$v_v(i) = -K\cos(\theta_i) + v_{gem} \tag{3.2}$$

Hieruit weten we dat de kracht de volgende functie heeft:

$$F = \eta mg + \frac{1}{2}\rho A C_w\left(-Kcos(\theta_i) + v_{gem} - cos(\theta_i)v_w\right)^2 \tag{3.3}$$

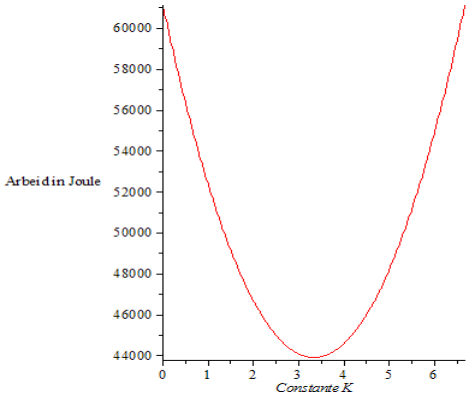
Zoals we beschreven hebben is de snelheid v_v van het voertuig af te leiden uit het baanvak i. We gebruiken hiervoor de volgende formule:

$$v_v = -K\cos\left(\frac{2\pi}{t_{ronde}} \cdot i\right) + \frac{20}{3} \tag{3.4}$$

De afgelegde afstand s_i is te berekenen door de snelheid op het moment i te vermenigvuldigen met het aantal seconde per i. In dit geval dat dus $s_i t_i = s_i \cdot 1 = s_i$. De totale arbeid die verricht moet worden is de sommatie van alle W_i, in formulevorm als volgt weergegeven:

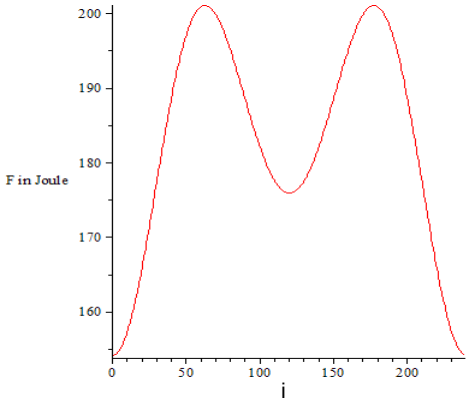
$$W_{totaal} = \sum_{i=0}^{N-1} s_i F_i \tag{3.5}$$

Om deze functie te minimaliseren moet in formule 3.3 de constante K aangepast worden. Om de optimale K uit te rekenen moeten we de functie 3.5 twee maal differentiëren en dan het nulpunt berekenen. We komen hierbij op een waarde van $K = \frac{10}{3}$. In figuur 3.5 is de arbeid afgezet ten opzichte van de verschillende waarde van de constante K. De minimale arbeid in deze situatie is 43921 J.



Figuur 3.5 - De arbeid als functie van de constante K

Nu de ideale constante K is uitgerekend, kan functie $f(i)$ uitgezet worden in een grafiek. Deze grafiek laten we zien in figuur 3.6 op de volgende pagina.



Figuur 3.6 – De benodigde arbeid F(i)

In Figuur 3.6 goed is te zien is dat er aan het begin zeer langzaam gereden wordt en daarna steeds sneller. Bij i=60 is het punt dat de wind loodrecht staat op de auto. De snelheid neemt daarna wel toe, maar wind zorgt voor meer energie en dus de totaal benodigde kracht van de auto neemt af. Vanaf i=120 hebben volledige wind mee en rijden we met steeds minder wind mee totdat er bij i =180 de wind weer loodrecht staat. Hierna gaan we weer steeds langzamer rijden.

3.4 De minimale arbeid berekend met behulp van het algoritme

3.4.1 Algoritmebeschrijving

In deze paragraaf beschrijven we de algoritmes die we op de twee eenvoudige banen hebben gebruikt.

3.4.1.1 De beschrijving van het algoritme op de rechthoekige baan

Als basis voor het algoritme stellen we de snelheid op ieder baanvak op v_{gem}. Deze snelheid berekenen we door $\frac{s_{totaal}}{t_{max}} = \frac{1600}{240} = \frac{20}{3} m/s$.

De benodigde arbeid is een kwadratische functie van de snelheid. Dit meegenomen wil je altijd op de maximum aangegeven tijd aankomen op de finish. Bij tegenwind kan er het beste zo langzaam mogelijk gereden worden. Op de baanvakken 1 en 3 is de relatieve windhoek loodrecht op de auto en heeft de wind, in dit geval, geen invloed op de wrijving.

De stappen die we nemen in het algoritme zijn de volgende:

Stap 1: Alle bekende variabelen worden ingevuld en de gemiddelde snelheid wordt bepaald en ingevuld op alle baanvakken. In deze situatie wordt het energiegebruik berekend.
Stap 2: Het geselecteerde baanvak is in dit geval altijd baanvak 0
Stap 3: De snelheid v_0 wordt verlaagd met een factor Δv en de snelheid v_2 wordt berekend.
Stap 4: De arbeid wordt berekend.
Stap 5: Deze arbeid wordt vervolgens vergeleken met de eerdere waarde. Als deze waarde hoger is dan de oude wordt deze verworpen. Zo niet, wordt deze aangenomen en gaan beginnen we weer bij stap 3.
Stap 6: De uitkomst wordt op het scherm getoond.

3.4.1.2 De beschrijving van het algoritme op de cirkelvormige baan

Om te beginnen delen we de baan op in segmenten van elk een seconde lang. In de vorige paragraaf hebben we besproken wat de formule voor de arbeid is en deze kunnen we per segment toepassen.

Aangezien we weten dat de snelheid van de auto met een sinusfunctie te benaderen is kunnen we deze per seconde berekenen. Hieruit weten we de startpositie van het segment en dus ook de relatieve windhoek aan het begin van dit segment. Hierdoor kunnen we de benodigde arbeid berekenen over dit stukje cirkel. Uiteindelijk levert de sommatie van al deze segmenten de totale arbeid op.

Zoals eerder besproken is de enige manier om deze functie te optimaliseren de constante K aanpassen. Hierbij beginnen we met K=1 en verhogen deze steeds met een klein getal, in dit geval met 0.1. Hierna berekenen we de totale arbeid en vergelijken deze met de eerder verkregen antwoorden.

3.4.2 Het algoritme uitgevoerd

In deze paragraaf voeren we het algoritme uit en verkrijgen hierdoor een minimale arbeid.

3.4.2.1 Het algoritme uitgevoerd op de rechthoekige baan

Zoals eerder beschreven definiëren we alle bekende variabelen. Hierna schrijven we de arbeid als functie van snelheid in een IF functie.

```
if(H == 0.5*Math.PI || H == 1.5*Math.PI) E = n*m*g*L + Rho*A*Cw*L*Math.pow(V,2);
else if((0.5*Math.PI < H && H < 1.5*Math.PI) || V > Windsnelheid ) E = n*m*g*L +
        0.5*Rho*A*Cw*L*Math.pow((V - Math.cos(H)*Windsnelheid),2);
else E = n*m*g*L;
```

Waarbij L de lengte van het baanvak is, V de relatieve snelheid van de auto en H de relatieve windhoek ten opzichte van de auto.

Na de basiswaarde uitgerekend te hebben, dus overal de gemiddelde snelheid, krijgen we de volgende uitkomst:

```
Iteratiestap 0
Snelheid op baanvak 0 is: 6.667 m/s en hij doet er 75.0 s over
Snelheid op baanvak 1 is: 6.667 m/s en hij doet er 45.0 s over
Snelheid op baanvak 2 is: 6.667 m/s en hij doet er 75.0 s over
Snelheid op baanvak 3 is: 6.667 m/s en hij doet er 45.0 s over
Arbeid is: 64346 J
```

Nu verkleinen we snelheid van baanvak 0 steeds met 0.01 m/s en bereken we de nieuwe waarde. Deze vergelijken we met de oude en op het moment dat één van de stopcriteria is bereikt geeft dit algoritme de uitkomst in afgeronde joules:

```
Optimale bereikt na 132 iteratiestappen
==================EINDE ALGORITME=======================
Snelheid op baanvak 0 is: 5.357 m/s en hij doet er 93.34 s over
Snelheid op baanvak 1 is: 6.667 m/s en hij doet er 45.0 s over
Snelheid op baanvak 2 is: 8.825 m/s en hij doet er 56.66 s over
Snelheid op baanvak 3 is: 6.667 m/s en hij doet er 45.0 s over
Arbeid is: 58864 J
```

De snelheid v_0 is dus 5.36 m/s en hierbij verbruikt de motor 58864 J. De complete uitvoer is te vinden in bijlage III.

3.4.2.2 Het algoritme uitgevoerd op de cirkelvormige baan

De manier van arbeidsberekening is hetzelfde als bij de rechthoekige baan, alleen de functie voor de snelheid is nu een sinus die elke stap vermenigvuldigd wordt met een constante K zodat de amplitude van deze functie toeneemt. De aanpassing van deze K wordt in Java dan per segment:

```
do
{
  i++;                          // iteratiestap verhogen
  Wnieuw = TijdStappen();   // Arbeid berekenen met deze K

  double Kr;                  // variabel om K afteronden op 1 cijfer achter de komma
  Kr=K*10;
  Kr=Math.round(Kr);
  Kr=Kr/10;

  System.out.println("Arbeid = "+Math.round(Wnieuw)+"J met K="+Kr);
  if(Math.round(Woud) - Math.round(Wnieuw) > 1) Woud = Wnieuw;
  else{
    Abort = 1;
    System.out.println("Optimale bereikt na "+i+"iteratiestappen");}

  K = K + Kplus;              // K verhogen met aangegeven waarde
}while(Abort < 1 && Vgem-K>0);
```

Nu kunnen we de arbeid berekenen per K, welke een optelling is van de 240 delen van elk 1 seconde lang. De berekende K wordt als volgt weergegeven:

```
Arbeid = 52349J met K=1.0
Arbeid = 51642J met K=1.1
…
Arbeid = 43949J met K=3.2
Arbeid = 43923J met K=3.3
Arbeid = 43928J met K=3.4
Optimale bereikt na 34 iteratiestappen
```

Hieruit zien we dat de optimale K de waarde 3.3 heeft en hierbij een arbeid nodig heeft van 43923 J.

3.5 De uitkomsten van het algoritme vergeleken met de exacte berekening

De berekende minimale arbeid op de rechthoekige baan bij zowel de exacte berekening als de berekening met het algoritme arbeid is 58864 J. Hierbij kunnen we concluderen dat het algoritme werkt en we deze kunnen gebruiken om verder uit te breiden tot het gebruikte circuit in de race.

De uitkomst van de cirkelvormige baan in de exacte berekening is 43921 J en die van het algoritme 43923 J. Om te kunnen concluderen dat dit in verband gebracht kan worden met een afrondingsfout herhalen we het algoritme in sub paragraaf 3.4.2.2 met een kleinere verandering in K (0.01), wat resulteert in een langere rekentijd, 13 seconde in plaats van 1 seconde. Hierbij genereert het algoritme een K van 3.33 en een bij behorende arbeid van 43921 J. We kunnen nu ook concluderen dat het algoritme voor de cirkelvormige baan te gebruiken is voor de bochten in het circuit van de race.

Hoofdstuk 4 - Het algoritme uitgewerkt en uitgebreid

Nu we weten dat het basisalgoritme toe te passen is kunnen we dit gaan uitbreiden. We hebben in paragraaf 2.4 besproken dat we de krachten uitrekenen vanuit de snelheidsvectoren. Hierbij berekenen we eerst de relatieve windsnelheid en daarna de benodigde kracht. We gaan in dit verslag niet in op de berekening vanuit het krachtspectrum, waar eerst de krachten relatief aan elkaar gemaakt worden en daarna de totale kracht berekend wordt. Om de werking van deze relatieve windsnelheid uit te leggen gaan we eerst kort in op de werking van vectoren. Hierna gaan we in op enkele, specifieke werkingen van het algoritme.

4.1 Vectoren in het kort

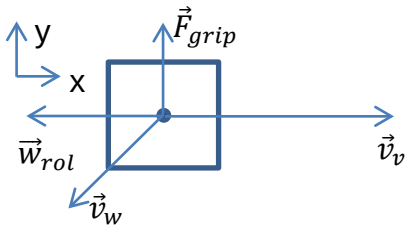
Tot nu toe hebben we voor de relatieve wind alleen formules gegeven en deze nog niet echt uitgewerkt. In dit hoofdstuk beschrijven we de snelheden en krachten die op de puntmassa (auto) werken. We leggen dit uit door middel van vectoren.

Een vector wordt weergegeven als een pijl en heeft een lengte, een hoek en een aanknopingspunt. We kunnen vectoren noteren zowel dikgedrukt (**v**) als met een pijl erboven ($\vec{v}$). Er is geen verschil tussen deze schrijfwijze en we kiezen voor het leesgemak de notatie met de pijl erboven. De hoek van deze vector wordt beschreven in radialen en we gebruiken hiervoor het symbool θ (Thèta).

In onze situatie gebruiken we vectoren met 2 dimensies: de invloed van deze vector in de richting van de x en y assen. We berekenen deze invloed op de x-as door de cosinus te vermenigvuldigen met de lengte van de vector. Hetzelfde doen we voor de invloed op de y-as, alleen dan met de sinus. We noteren dit als volgt:

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot v \\ \sin\theta \cdot v \end{pmatrix} \tag{4.1}$$

We beschrijven de situatie van de auto met de volgende schematische weergave:



Figuur 4.1 – schematische weergave van de vectoren

In figuur 4.1 zijn de factoren te zien die we tot nu toe hebben meegenomen:$\vec{v}_w$ (wind), $\vec{v}_v$ (voertuig) en $\vec{W}_{rol}$(rolweerstand). Tevens zien we de grip van de banden $\vec{F}_{grip}$, welke even groot is als de in vloed van de wind op de y-as. Aangezien deze kracht geen invloed heeft in de rijrichting laten we deze voortaan weg. We hebben dit schema zo getekend dat $\vec{v}_v$ zich onder een hoek van 0 rad bevindt, maar dit is niet noodzakelijk, aangezien de formules de relatieve windhoek gebruiken.

4.2 Het inlezen van het circuit

Aangezien er geen enkel circuit hetzelfde hoeft te zijn moet dit gedeelte van het algoritme flexibel zijn. Er moet hiervoor eerst bekeken worden hoeveel baanvakken het circuit heeft. Dit kan door in de database een variabel aan te maken die dit aantal bevat of het algoritme dient dit zelf te berekenen.

De database heeft voor elk baanvak de volgende informatie nodig:

- Het nummer van het baanvak (moet chronologisch)
- De hoek aan het begin van het baanvak op de kaart t.o.v. het oosten
- De hoek aan het eind van het baanvak op de kaart t.o.v. het oosten
- De lengte van het baanvak

Dit wordt dan in een array(*) gezet om te kunnen gebruiken in de rest van het algoritme. Met deze informatie kan de baan “geconstrueerd” worden. Dit wil zeggen of dit baanvak een bocht is en dus welk gedeelte van het algoritme dit baanvak het beste kan benaderen.

Het uitlezen van het circuit uit een database dient geautomatiseerd te worden in een later stadium. Op dit moment moeten we het circuit in de code definiëren.

4.3 De relatieve windhoek en de meeste tegenwind

Vervolgens berekent het algoritme de relatieve windhoek ten opzichte van de auto en zet deze tussen 0 en 2π. Deze laatste stap is nodig omdat dit verdere berekeningen in Java vergemakkelijkt. Dit doen we met de volgende code:

```
relHoekBegin[i] = baanVakHoekBegin[i] - windhoek;
if(relHoekBegin[i] < 0) relHoekBegin[i] = relHoekBegin[i] + 2*Math.PI;
if(relHoekBegin[i] >= 2*Math.PI) relHoekBegin[i] = relHoekBegin[i] - 2*Math.PI;
```

Hierna wordt er bekeken in welk(e) baanvak(ken) de meeste (en minste) tegenwind is. Hier begint het algoritme bij het eerste baanvak en slaat de relatieve windinvloed op. Daarna gaat hij naar het volgende baanvak en vergelijkt dit met de opgeslagen waarden. Als deze kleiner is gaat het algoritme door, bij gelijkspel slaat hij dit baanvak op naast het eerdere en als hij groter is vergeet hij alle eerdere baanvakken en slaat alleen deze op. Zo doorloopt het algoritme het circuit. Dit wordt ook gedaan met de minste relatieve windinvloed.

4.4 De positie van de auto en de basissnelheid voor iteratiestap 0

Als eerste dient het algoritme de positie van de auto te bepalen en hieruit de nog te rijden baanvakken. Hiervoor is de afgelegde afstand in deze ronde nodig. Als deze afstand 0 is betekent dit dat de auto zich aan het begin van het circuit bevindt. Het algoritme berekent wel in welk baanvak de auto zich bevindt, maar hier wordt geen gebruik van gemaakt.

De snelheid in elk baanvak voor de basis van het algoritme (iteratiestap 0) dient daarna gedefinieerd te worden. Op dit moment hebben we er voor gekozen om deze overal de gemiddelde snelheid te geven. Hierna wordt de arbeid voor deze beginsituatie berekend, zodat we een basis hebben om te optimaliseren.

(*)*eenvoudige, geordende rij dezelfde soort getallen in een programmeertaal*

4.5 De benodigde kracht vanuit de snelheid

In hoofdstuk 3 hebben we formules gebruikt om de benodigde kracht van de motor uit te rekenen om een bepaalde snelheid te kunnen rijden. Hierbij berekenen we direct de relatieve snelheid van de wind ten opzichte van het voertuig. Daarna berekenen we de kracht die er nodig is die snelheid rijden. We zien dit terug in de formule 2.4.

Omdat we in hoofdstuk 3 er vanuit gaan dat er op baanvak 0 tegenwind is, weten we dat de windrichting in deze situatie π radialen is. Dit betekent dat op baanvak 0 de relatieve hoek $\cos\pi = -1$ en op baanvak 2 dit $\cos 0 = 1$ is. Ook zien we dat bij de windrichtingen die loodrecht op de baan staan de wind geen invloed heeft, wat we kunnen zien door de relatieve windhoeken in te vullen: $\cos\left(\frac{1}{2}\pi\right) = \cos\left(\frac{3}{2}\pi\right) = 0$.

De algemene formule voor de benodigde kracht, gebruikmakend van de vector met snelheid en hoek, is:

$$\vec{F}_{totaal} = \frac{1}{2}\rho A C_w \begin{pmatrix} v_v \\ \theta_v \end{pmatrix}^2 - \frac{1}{2}\rho A C_w \begin{pmatrix} v_w \\ \theta_w \end{pmatrix}^2 + \begin{pmatrix} \eta m g \\ \theta_v - \pi \end{pmatrix} \tag{4.2}$$

Formule 4.2 is de algemene formule van de kracht in een algemeen stelsel. We willen alleen deze formule weten vanuit de auto. Hierdoor moeten we deze formule herschrijven zodat de rijrichting van de auto de nieuwe x-as wordt. Op deze manier kunnen we de invloed van de krachten berekenen op deze rijrichting. Dan kunnen we dus vector $\vec{v}_w$ hierop projecteren. Door de rijrichting de x-as te maken valt de richting van de vector $\vec{v}_v$ weg. Doordat we de kracht vanuit de auto schrijven weten we dat deze altijd naar voren gericht is en kunnen we de vector van de kracht $\vec{F}_{totaal}$ vervangen door de benodigde kracht F_{totaal} van de motor.

Deze invloed wordt berekend door de cosinus te nemen van de relatieve hoek tussen $\vec{v}_w$ en $\vec{v}_v$ en deze te vermenigvuldigen met de lengte van $\vec{v}_w$. Dit ingevuld in formule 3.2 levert dat onze formule van de benodigde kracht:

$$F_{totaal} = \eta m g + \frac{1}{2}\rho A C_w (v_v - \cos\theta_w\, v_w)^2 \tag{4.3}$$

Formule 4.3 is niet meer dan de optelling van de formules 2.3 en 2.4. Als uitzondering op deze formule is er een mogelijkheid dat er wind mee is en er bij de gewenste snelheid geldt dat $\eta m g < \frac{1}{2}\rho A C_w (v_v - \cos\theta_w\, v_w)^2$. In dit geval betekent dit dat de kracht die de wind levert groter is dan de te overwinnen wrijving, wat resulteert in een benodigde kracht van $F_{totaal} = 0$.

Dit wordt in het programma opgevangen door de situatie eerst te beoordelen op deze punten, voordat er de benodigde kracht uitgerekend wordt. Hierbij wordt de volgende statement getoetst:

$$v_w \cos\theta_w \geq \sqrt{\frac{\eta m g}{\frac{1}{2}\rho A C_w}}$$

Als dit statement waar is geeft het algoritme aan dat de benodigde kracht voor de gewenste snelheid 0 is.

4.6 De versnelling (Stop & Go)

Omdat we de verplichte Stop & Go (elke ronde dient er een moment stil te staan) willen we de verandering van snelheid meenemen in het algoritme. We beginnen met de aanname dat de auto ineens de snelheid rijdt die nodig is. Hierdoor is het niet mogelijk om dit via de kracht die daarbij nodig is (deze is dan ∞) uit te rekenen. We gebruiken hierbij de arbeid die gelijk is aan de verandering van de kinetische energie. Bij de kinetische energie en de verandering hiervan gebruiken we de volgende formules:

$$KE = \frac{1}{2}mv^2 \tag{4.4}$$

$$W_a = KE_{nieuw} - KE_{oud}$$

$$W_a = \frac{1}{2}m({v_{nieuw}}^2 - {v_{oud}}^2) \tag{4.5}$$

waarbij geldt: $v_{nieuw} > v_{oud}$. Als de auto vertraagt (remt) nemen we dit als verlies, aangezien we deze energie niet terugwinnen. In een later stadium van het algoritme kunnen we de versnelling en het uitrollen (per tijdseenheid) meenemen, zodat we deze ook kunnen optimaliseren.

Tegen het eind van een ronde (Stop & Go) dient de auto een moment stil te staan. De auto begint het volgende baanvak dan met een snelheid van 0 m/s.

4.7 De optimale snelheid in de bochten

We hebben in hoofdstuk 3 de ideale cirkel als baan besproken. Deze theorie nemen we mee op de bochten van de baan. Op het circuit van de Ahoy' Rotterdam vinden we voornamelijk rechte hoeken. We willen het algoritme zo algemeen mogelijk houden zodat we ook rekening kunnen houden met andere bochten. Hiervoor registreren we voor elk baanvak de begin- en eind hoek en de lengte zodat we hier de (relatieve) cirkelhoek uit kunnen berekenen met de straal.

Met het eerder besproken algoritme kunnen we de vermenigvuldigingsfactor K per baanvak berekenen en zodoende de rijstrategie aanpassen. We gaan er in het algoritme vanuit dat de wagen in de bocht altijd in het midden van de baan rijdt. Voor verdere uitleg verwijzen we hiervoor naar paragraaf 5.2.

4.8 De iteratiestappen

Het optimalisatiealgoritme gaat er vanuit dat er de meeste winst gehaald kan worden door langzamer te rijden waar er veel tegenwind is. Het tijdverlies moet gecompenseerd worden. Dit kan door in de baanvakken met de minste tegenwind meer kracht te zetten, zodat er daar harder gereden wordt.

Hierbij wordt rekening gehouden met stopcriteria. Als een van deze criteria gehaald wordt stopt de iteratiestap direct en worden alle berekeningen in die iteratiestap niet onthouden.

Nadat de snelheden aangepast zijn wordt wederom de benodigde arbeid berekend en alleen als deze kleiner is dan de vorige iteratiestap (rekening houdend met het stopcriterium) worden deze snelheden onthouden en wordt de volgende iteratiestap ingezet.

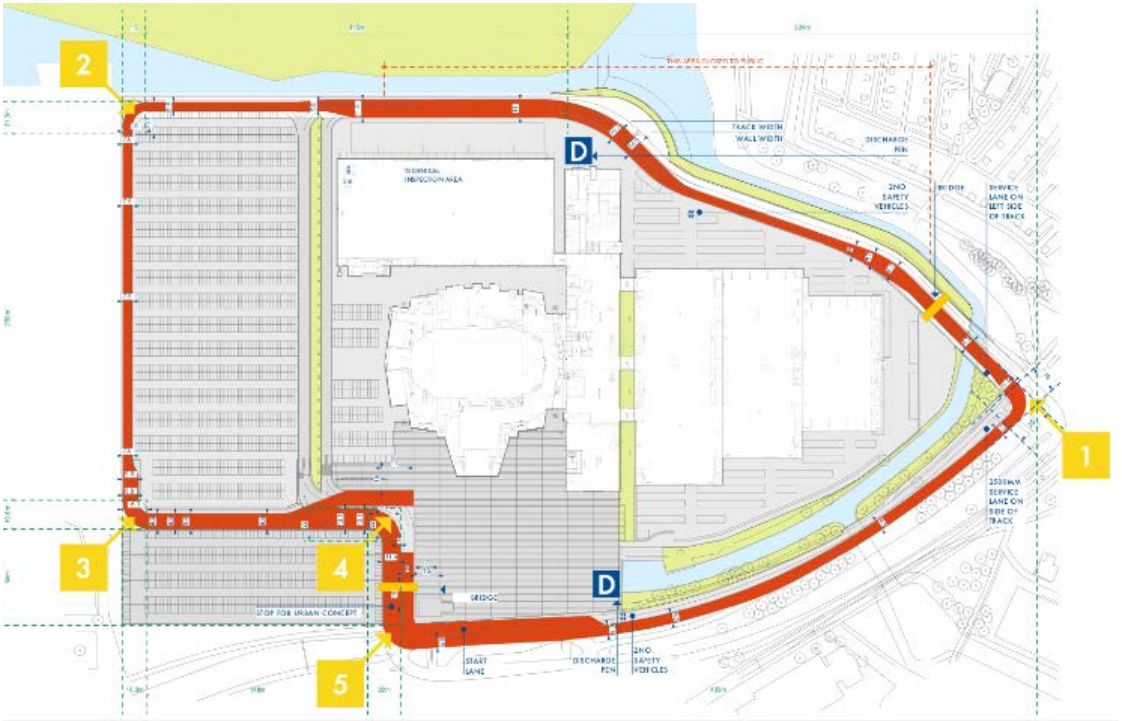
4.9 De resultaten tonen

Nadat er een stopcriterium is bereikt stopt het algoritme met de iteratiestappen en toont dan op het scherm de benodigde snelheden per baanvak. Dit moet in een later stadium optimaal doorgegeven worden, via de interface, aan de strategist. Deze interface staat los van het algoritme en wordt zodoende niet behandeld in dit verslag.

Hoofdstuk 5 - Het circuit van de Ahoy' Rotterdam

5.1 De verdeling van het circuit

We bekijken als eerste hoe het circuit van Rotterdam er uitziet en we verdelen deze naar ons beste inzicht. In dit plaatje zijn de 90-graden bochten genummerd van 1 t/m 5. De totale lengte per ronde is 1630 meter en er mag maximaal 39 minuten gedaan worden over 10 ronden (dus 234 s per ronde).



Afbeelding 5.1 – Het circuit Ahoy' Rotterdam, Shell Eco Marathon 2012

Voor de indeling van dit circuit nemen we dezelfde indeling die dhr. L. Houtenbos(*) gebruikt heeft in de simulatie. Hierdoor krijgen we de volgende indeling (hoeken in graden):

Baanvak	Hoeken		Lengte
I	0	45	410.15
II	45	135	12.25
III	135	135	113.28
IV	135	145	30.74
V	145	145	137.58
VI	135	135	59.11
VII	135	180	27.30
VIII	180	180	286.54
IX	180	270	16.12
X	270	270	250.00
XI	270	360	10.10
XII	360	360	150.69

Baanvak	Hoeken		Lengte
XIII	360	270	11.61
XIV	270	270	44.28
	Stop & Go		
XV	270	270	12.90
XVI	270	360	19.35
XVII	360	360	33.10
Overig			4.89
Totaal			163000

Tabel 5.1 De verdeling volgens de eerdere simulatie

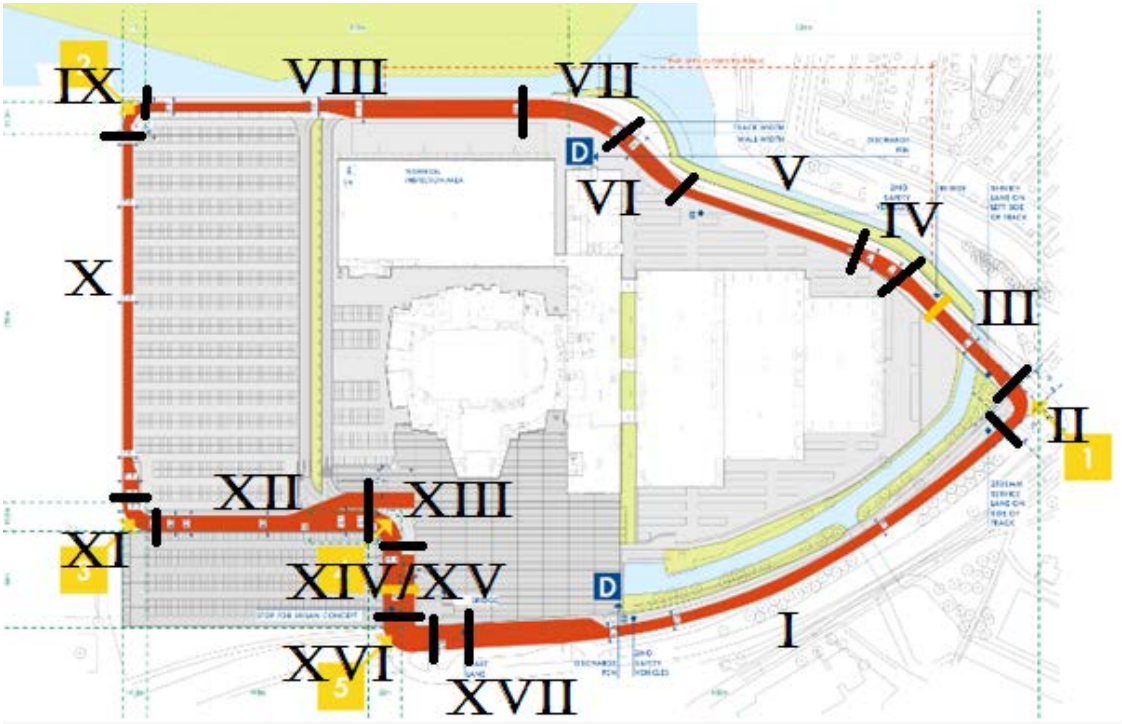
(*) *Zie de literatuurlijst in bijlage II*

Hierbij moet opgemerkt worden dat er, vergeleken met de simulatie, een kleine wijziging in hoeken is meegenomen tussen de baanvakken III en XIII. De afstanden staan in honderdste meters genoteerd. De nauwkeurigheid hiervan valt te betwijfelen, maar we houden deze afstanden van de simulatie aan.

Tevens hebben we opgemerkt dat de baan in de simulatie 4.89 meter te kort is. Dit staat equivalent aan ongeveer een seconde rijtijd, dus het maakt voor het algemene algoritme zeer weinig uit waar deze afstand zich bevindt. We voegen deze meters toe aan het laatste baanvak, zodat we er zeker van zijn dat deze afstand altijd meegenomen wordt met de berekeningen. Beter een seconde te vroeg dan te laat finishen, zodat we altijd een geldig resultaat kunnen neerzetten.

In het algoritme gebruiken we de hoeken in radialen, maar de simulatie gebruikt graden. Deze hoeken worden in het algoritme, indien nodig, omgerekend.

De indeling van het circuit ziet er dan globaal als volgt uit:



Afbeelding 5.2 – De verdeling van het circuit

Afbeelding 5.2 samengenomen met tabel 5.1 geeft ons een beeld over hoe het circuit is verdeeld. Het circuit ingevuld in de totaalformule voor de benodigde arbeid staat in bijlage V.

5.2 De rijstrategie

In het model gaan we er vanuit dat het voertuig altijd in het midden van de baan rijdt. Dit is waarschijnlijk niet de meest optimale rijstrategie. Ook zijn er andere auto's op de baan die eventueel inhalen of ingehaald dienen te worden. Hierdoor weten we dat er ook een energiebezuiniging kan plaatsvinden door de baanbenutting optimaal te maken.

In het eindrapport van de projectgroep "Matharon"[(*)] staat een rijstrategie beschreven voor de positie van de wagen op de baan bij inhaalmanoeuvres en de optimale baanbenutting bij bochten. In dit rapport staat beschreven dat bij het nemen van een bocht de insteek ervan zo ruim mogelijk moet zijn en daarna via de binnenkant van de bocht weer zo breed mogelijk de bocht uitrijden.

Bij de berekeningen in het algoritme wordt op dit moment geen rekening gehouden met de baanbenutting, zodat de berekende arbeid niet overeen komt met de werkelijke arbeid die de motor levert. Om dit te compenseren raden we aan in een later stadium hiervoor een wegingsfactor van een baanvak te gebruiken.

Deze wegingsfactor zal berekend worden naar aanleiding van de theoretische tijd die een baanvak vak kost uit te zetten tegen de in het verleden gereden werkelijke tijden van de baanvakken. Deze berekende factor kan meegenomen worden in de tijd die een baanvak kost en zodoende worden deze theoretische bochten gecompenseerd met de werkelijkheid.

5.3 De uitvoer van het programma

5.3.1 De nieuwe uitvoer van de rechthoekige baan

Nu we het een en ander in de code hebben veranderd moeten we deze opnieuw uitvoeren om een basis voor deze code te krijgen. De uitvoer is nu:

```
Optimum bereikt! iteratiestap: 138
================EINDE ALGORITME=========================
V(0) is: 5.23 m/s, t(0): 94.40 s
V(1) is: 6.67 m/s, t(1): 45.0 s
V(2) is: 8.99 m/s, t(2): 55.60 s
V(3) is: 6.67 m/s, t(3): 45.0 s
totale arbeid: 60991
```

Dit is vergeleken met de eerdere berekening 2127 J meer, wat onder andere te wijten is aan de toevoeging van de versnelling.

(*) Zie de literatuurlijst in bijlage II

5.3.2 De uitvoer van het circuit van de Ahoy' Rotterdam

Na de voorwaarden van tijd en afstand veranderd te hebben en het circuit ingevoerd te hebben krijgen we de volgende uitvoer voor de basissituatie dat er continu v_{gem}gereden wordt:

```
V(0) is: 6.97 m/s,     t(0): 58.86
V(1) is: 6.97 m/s,     t(1): 1.72
V(2) is: 6.97 m/s,     t(2): 16.22
V(3) is: 6.97 m/s,     t(3): 4.45
V(4) is: 6.97 m/s,     t(4): 19.81
V(5) is: 6.97 m/s,     t(5): 8.47
V(6) is: 6.97 m/s,     t(6): 3.88
V(7) is: 6.97 m/s,     t(7): 41.20
V(8) is: 6.97 m/s,     t(8): 2.30
V(9) is: 6.97 m/s,     t(9): 35.89
V(10) is: 6.97 m/s,    t(10): 1.44
V(11) is: 6.97m/s,     t(11): 21.68
V(12) is: 6.97 m/s,    t(12): 1.72
V(13) is: 6.97 m/s,    t(13): 6.32
V(14) is: 6.97 m/s,    t(14): 1.87
V(15) is: 6.97 m/s,    t(15): 2.73
V(16) is: 6.97 m/s,    t(16): 5.46
totale arbeid: 77295 J
```

Hierna kan de optimalisatie beginnen. We krijgen als we het algoritme uitvoeren een minimale verbetering ten opzichte van de basissituatie. Een van de redenen hiervoor kan zijn dat de te compenseren tijd op een te klein baanvak wordt geprojecteerd en zodoende er op dat baanvak zeer hard gereden dient te worden.

Op dit moment genereert het algoritme al naar 3 iteratiestappen een minimum. De snelheden liggen echter nog erg dicht rond het gemiddelde. Hierdoor is er het vermoeden dat dit niet dicht bij het globale maximum ligt. We zijn op dit moment hard aan het werk om hiervoor een oplossing te vinden en uit dit lokale minimum te kunnen ontsnappen.

Tot die tijd is de volgende output het lokale minimum:

```
V(0) is: 6.955811965811965 m/s,     t(0): 58.943514001695696     F:87.89213327173375
V(1) is: 6.965811965811965 m/s,     t(1): 1.7226993865030675     F:32.76853594126671
V(2) is: 6.965811965811965 m/s,     t(2): 16.22208588957055 F:9.061090881417282
V(3) is: 6.965811965811965 m/s,     t(3): 4.450306748466257 F:7.813090387674822
V(4) is: 6.965811965811965 m/s,     t(4): 19.811042944785274     F:6.784492074896948
V(5) is: 6.965811965811965 m/s,     t(5): 8.469938650306748 F:9.061090881417282
V(6) is: 6.965811965811965 m/s,     t(6): 3.8760736196319017     F:88.04151797593973
V(7) is: 6.975923626084386 m/s,     t(7): 41.14150546701072 F:3.989756843596412
V(8) is: 6.975923626084386 m/s,     t(8): 2.2936030922375314     F:9.105901520108713
V(9) is: 6.965811965811965 m/s,     t(9): 35.88957055214724 F:32.76853594126671
V(10) is: 6.965811965811965 m/s,    t(10): 1.4355828220858897    F:72.60098100111615
V(11) is: 6.955811965811965 m/s,    t(11): 21.708464912819633    F:93.6687286463949
V(12) is: 6.955811965811965 m/s,    t(12): 1.725176019561825     F:72.46557813937062
V(13) is: 6.965811965811965 m/s,    t(13): 6.316564417177914     F:32.76853594126671
V(14) is: 6.965811965811965 m/s,    t(14): 1.8662576687116563    F:32.76853594126671
V(15) is: 6.965811965811965 m/s,    t(15): 2.72760736196319      F:72.60098100111615
V(16) is: 6.955811965811965 m/s,    t(16): 5.463057395279112     F:93.6687286463949
totale arbeid: 77170
```

Hoofdstuk 6 - Conclusie

We hebben het algoritme algemeen besproken, in het kort komt het hier op neer:

- Lees de omstandigheden
- Zoek de baanvakken die veranderd worden
- Verander de snelheden
- Controleer of deze snelheden voldoen
- Geef uitvoer op scherm

We hebben dit algoritme in Java geprogrammeerd en getoetst op eenvoudige banen, waar we de minimale arbeid exact konden berekenen. We hebben de uitkomsten van het algoritme getoetst en deze kwamen overeen met de exacte waarden. We weten dus dat deze basis van het algoritme gebruikt kan worden.

Hierna hebben we onder andere de versnelling toegevoegd en hebben we er voor gezorgd dat de code voor de baan flexibel is en geschikt voor verschillende aantallen baanvakken. Daarna hebben we het circuit van de Ahoy' Rotterdam ingevoerd.

Hierdoor weten we dat we een algoritme hebben bedacht die rekening houdt met omgevingsfactoren en hieruit een optimale rijstrategie beschrijft. Hierbij is de opdrachtomschrijving behaald. Al met al kunnen we zeggen dat we een goed begin hebben gemaakt voor het algoritme van de rijstrategie.

Hoofdstuk 7 - Aanbevelingen

We hebben een opzet gemaakt voor dit algoritme, maar vinden dat deze niet overal optimaal of compleet is. In dit verslag hebben we de ontwikkeling en werking van het optimalisatiealgoritme besproken. Hierbij zijn ook hier en daar extra aanbevelingen gedaan.

In dit hoofdstuk nemen we de opmerkingen door en geven aanbevelingen over wat er in de toekomst in ieder geval toegevoegd dient te worden aan het algoritme voordat deze in gebruik genomen kan worden bij een eventuele volgende Shell Eco marathon.

7.1 De database en de weersomstandigheden

Op dit moment dienen de gegevens van de baanvakken in de code opgenomen te worden. Deze gegevens dienen echter extern (dus buiten dit algoritme en de interface) opgeslagen te worden aangezien we niet telkens de code willen aanpassen als er op een ander circuit gereden wordt. Dit dient ook gedaan te worden met de specificaties van de auto, aangezien deze in de loop van de jaren ook kan gaan verschillen.

De weeromstandigheden dienen meegegeven te worden door de interface, als deze het algoritme aanroept. De meegegeven gegevens dienen geïmplementeerd te worden in de code.

Kortom de code dient zo aangepast te worden dat alle benodigde, externe gegevens uit een externe bron komen. Hierbij moet dan wel in de code toegevoegd worden dat de databases uitgelezen kunnen worden.

7.2 Variabele beginpositie

Tot nu toe zijn we er vanuit gegaan dat de auto altijd aan het begin van het circuit begint en er maar een enkele ronde gereden wordt. Het algoritme heeft al een functie die het baanvak van de startpositie bepaald, maar gebruikt op dit moment alleen de beginpositie 0.

Dus is het aan te raden het algoritme zo te veranderen dat er op elk gewenst moment dit algoritme aangeroepen kan worden en deze een optimale rijstrategie berekend voor de nog af te leggen weg.

7.3 Versnellen en uitrollen

In paragraaf 3.4 wordt het trucje van de verandering van kinetische energie toegepast. Hierbij wordt er dus vanuit gegaan dat de auto in elk baanvak constant rijdt. De strategie voor optrekken en uitrollen is hierbij niet geïmplementeerd in deze code.

Echter kan verstandig versnellen en uitrollen wel een verschil maken in de totale benodigde arbeid. Daarnaast kan de motor maar een maximaal vermogen leveren en dus ook maar een maximum versnelling. Hier moet dus ook rekening mee gehouden dient te worden (stopcriterium).

7.4 Bochten- en inhaalstrategie

Zoals in hoofdstuk 4 besproken is gaan we er in het algoritme er altijd vanuit dat de auto in het midden van de baan rijdt. We hebben ook aangeraden de bochten- en inhaalstrategie over te nemen uit een eerder onderzoek. Hierbij moeten we het verlies in tijd voor inhalen voor lief nemen, aangezien dit geen constante factor is.

De bochtenstrategie kan wel geïmplementeerd worden in de code om een nog nauwkeurigere, theoretische weergave van de benodigde arbeid te berekenen.

Zoals in paragraaf 4.2 aangegeven kunnen we deze factoren ook benaderen door een berekende wegingsfactor per baanvak mee te nemen. Hierbij kunnen we de benodigde tijd (en dus ook snelheid en arbeid) per baanvak benaderen. Deze wegingsfactor dient dan wel per baanvak extern berekend te worden en moet dan ook uitgelezen kunnen worden door het algoritme.

7.5 Uit het lokale minimum ontsnappen

Op dit moment gaat het algoritme er vanuit dat de arbeidsfunctie een dal parabool beschrijft en zodoende vindt het algoritme een enkel minimum. Het kan zijn dat de arbeidsfunctie een andere, onbetrouwbaardere vorm heeft. Het algoritme zoekt vanuit de gemiddelde snelheid op elk baanvak naar het eerste minimum op deze functielijn, het lokale minimum. Hierdoor kan het zijn dat er een ander punt op de functielijn een nog lager benodigde arbeid aangeeft, het globale minimum.

Als er, nadat het lokale minimum gevonden is, een mogelijkheid is om alles opnieuw uit te rekenen vanuit bijvoorbeeld een andere basispositie. Dan kan het algoritme opnieuw er op los gelaten worden en is er een kans dat er een lager minimum gevonden wordt.

Doordat er veel variabelen invloed hebben op de functieomschrijving is het niet altijd mogelijk om het globale maximum te vinden. Maar op het moment dat er een lokaal minimum gevonden wordt dat lager is dan het eerder berekende minimum, dan weten we in ieder geval zeker dat dit dichterbij het globale minimum ligt.

Enige mogelijkheden hiervoor staan beschreven in het boek “Introduction to operations research”(*)

7.6 De code splitsen

We hebben tot nu toe alle codes in een enkele bestand gezet. Dit is niet de meest duidelijke manier om de codes weer te geven. We raden dan ook aan om in de toekomst deze codes te splitsen naar meerdere bestanden om de leesbaarheid te verbeteren

(*) *Zie de literatuurlijst in bijlage II*