KU LEUVEN

FACULTEIT COMPUTERWETENSCHAPPEN

Real-Time Cadansaanpassing in een Automatische fietstransmissie

Arno Cools

Promotor:
Prof. M. Moens

Begeleider:
Ir. Tomas Keppens
Ir. Jorrit Heidbuchel
Ir. Rugen Heidbuchel

Proefschrift ingediend tot het
behalen van de graad van
Master in de Toegepase Informatica

Academiejaar 2018-2019

Voorwoord

A preface

Abstract

An abstract.

Inhoudstafel

Lijst van figuren

Afkortingen

CVT	continu variabele transmissie. 3
DT	decision tree. 16
ES	Exponential Smoothing . 17
FCC	freely chosen cadence. 3
MA	Moving Average . 17
PA	passive aggressive. 15
RF	random forest. 16

Symbolen

A_{aero}	Frontaal oppervlak fietser. 11
FCC_{est}	Schatting van FCC geleverd door de cadanscontroller. 4
FCC_{pred}	Voorspelde FCC. 17
F_{aero}	Luchtweerstand. 11
$F_{friction}$	Wrijvings last. 11
F_{grav}	Gravitationele last. 11
F_{load}	Totale last. 11
K	Agressiviteits parameter voor proportionele regelaar. 6
P	Vermogen. 11
S	Ondersteuningsniveau. 9
T_{MG2}	Koppel geleverd door motor op het voorwiel. 9
$T_{cy,m}$	Gemeten koppel geleverd door de fietser. 4
T_{cy}	Koppel geleverd door de fietser. 3
$T_{dc,max}$	Maximum DC koppel dat geleverd kan worden. 6
T_{dc}	Het DC koppel van de fietser (gemiddeld koppel). 5
T_{rw}	Koppel op het achterwiel. 8
α	Helling. 4
ω_{cr}	Cadans in rpm. 6
ρ_{aero}	Luchtdichtheid. 11
θ_{cr}	Hoek van de trapas. 4
c_d	Luchtweerstand coëfficiënt. 11
c_r	Rolweerstand coëfficiënt. 11
f_h	Factor voor fietsersmodel in functie van de helling. 9
f_k	Factor voor fietsersmodel in functie van het gemiddeld koppel. 9
f_v	Factor voor fietsersmodel in functie van de snelheid. 9
g	Gravitationele constante. 11
$k_{cr,r}$	Verhouding overbrenging trapas-ringwiel. 8
m	Totaal gewicht van fiets en fietser. 11
nr	Aantal tanden op het ringwiel. 8
ns	Aantal tanden op het zonnewiel. 8
r_w	Straal van het voor- en achterwiel. 13
r	Input vector fietser. 3
sf	Smoothing factor. 17
u_c	De staat van de knop die de cadans aanpast. 3
u_{contr}	Input vector fietser geleverd door de controller. 4
u_{cy}	Input vector fiets geleverd door de fietser. 3

v_{bike}	Snelheid van de fiets. 4
v_{ref}	Referentie snelheid van de fietser. 3
y	Output vector fiets. 4

Hoofdstuk 1

Probleemstelling

1.1 Mobiliteitsvraagstuk

De auto is het slachtoffer geworden van zijn eigen succes: we staan meer dan ooit in de file en de CO2 van personenverkeer stijgt jaar na jaar. De belg neemt al snel de auto voor korte afstanden (< 25 km). In deze auto zit meestal maar 1 persoon. Het Belgische wagenpark blijft groeien (figuur 2). Hier zien we wel een trend ontstaan. Er worden steeds meer elektrische en hybride wagens verkocht, maar die staan natuurlijk net zo goed in de file. Mobiliteit op twee wielen kan hier een oplossing bieden.

Mobiliteit op twee wielen kennen we al lang: fietsen bestaan al sinds de 19de eeuw. Elektrische fietsen hebben het potentieel van deze tweewielers enorm verhoogd: fietsen wordt moeiteloos en stukken sneller. Spijtig genoeg neemt het risico op ongevallen ook toe bij hogere snelheid. Dat komt omdat e-bikes en speed e-bikes precies dezelfde technologie gebruiken als normale fietsen – grote wielen met smalle banden, kettingaandrijving met manuele versnellingen, mechanische handremmen – bij veel hogere snelheden. IntuEdrive noemt dit de snelheid-veiligheid trade-off. De veiligheid kan beperkt worden verhoogd door componenten toe te voegen (bv. Bosch e-bike ABS), maar de functionaliteit van deze systemen blijft beperkt. Er is een meer holistische aanpak nodig. Bovendien bieden elektrische fietsen vandaag nog niet het gebruiksgemak en de betrouwbaarheid die de consument gewend is van zijn wagen.

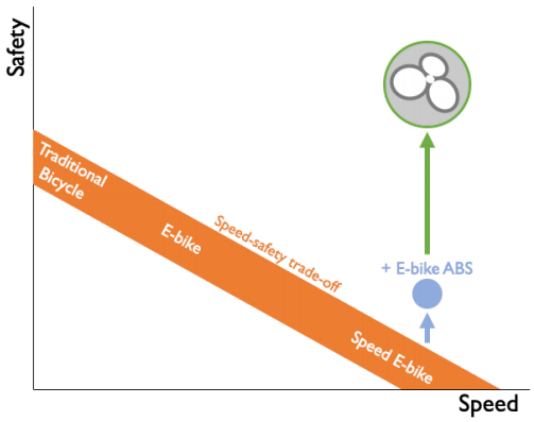


Figure 1: Snelheid-veiligheid trade-off (bron: IntuEdrive)

IntuEdrive's *CoSaR* is een snelle elektrische fiets die veiliger is dan de klassieke mechanische fiets, dankzij hun innovatieve tweewielaandrijving en elektrische remfunctie. Dit systeem reduceert de stopafstand met 60% en maakt schakelen overbodig (automatische versnellingen). Het stapt ook af van de onderhoudsintensieve fietscomponenten (ketting, tandwielen, mechanische remmen). Dit maakt CoSaR de perfecte e-bike voor woon-werkverkeer: makkelijk, veilig en betrouwbaar.

Op 1 augustus van het jaar	2014	2015	2016	2017	2018
Personenwagens	**5.555.499**	**5.623.579**	**5.712.061**	**5.785.447**	**5.853.782**
- rijdend op benzine	2.029.688	2.091.327	2.199.038	2.335.349	2.518.942
- rijdend op diesel	3.458.424	3.457.526	3.424.592	3.339.034	3.193.658
- rijdend op gas	22.051	18.967	17.238	15.965	15.500
- met elektrische motor	1.792	2.871	4.368	6.552	9.244
- hybride	23.444	32.151	44.364	63.740	87.012
- niet nader bepaald	20.100	20.737	22.461	24.807	29.426
Inwoners per personenauto op 1 augustus (d)	2,01	1,99	1,97	1,96	1,94

Figure 2: Grootte van het voertuigenpark 2014-2018 (bron: statbel.fgov.be)

Door automatisch te schakelen zorgt CoSaR ervoor dat de fietser in elke situatie precies zo snel trapt als hij of zij wil. Deze gewenste trapsnelheid – of beter trapcadans – varieert van persoon tot persoon en hangt af van omstandigheden zoals helling, tegenwind en rijsnelheid. Omdat deze gewenste cadans niet op voorhand gekend is, schakelt de transmissie momenteel op basis van een vaste wetmatigheid die tijdens testen getuned is om voor zoveel mogelijk gebruikers comfortabel aan te voelen. Wijkt deze wetmatigheid af van de gewenste cadans van een specifieke gebruiker, dan kan deze gebruiker via knoppen op het stuur tijdens het fietsen zijn of haar cadans manueel aanpassen.

1.2 Online machine learning voor geïndividualiseerde cadanscontrole

Deze thesis werkt verder op het prototype geleverd door IntuEdrive. Zoals reeds aangehaald schakelt de fiets automatisch. De trapcadans wordt hierdoor stabiel gehouden, ook wanneer de fietser harder of zachter trapt. Het doel is om deze instelling te veranderen door de cadans van de e-bike in real time te voorspellen aan de hand van de toestand van de fiets, zodat de trapsnelheid zich aanpast aan de huidige omstandigheden en de individuele gebruiker. Deze implementatie zal ervoor zorgen dat de fietser meer aandacht kan besteden aan de weg, waardoor gevaarlijke situaties kunnen vermeden worden. Om de cadans te personaliseren en dynamisch te maken, zal een machine learning algoritme ontwikkeld worden dat de toestand van de fiets als input binnenkrijgt en hiermee de trapsnelheid berekent. Wanneer de fietser besluit om de cadans manueel aan te passen, interpreteert het algoritme dit als een signaal om bij te leren.

Om de performantie van het machine learning algoritme te testen zal het volledig systeem fiets-fietser-cadanscontrole gesimuleerd worden. Het fietsmodel wordt geleverd door IntuEdrive en zal geïmplementeerd worden in python. Vervolgens worden een aantal machine learning algoritmes vergeleken op basis van een aantal vooraf gedefinieerde performance indicatoren. De machine learning algoritmes zijn afkomstig uit scikit-learn, een machine learning library.

De cadanscontrole moet aan verschillende eisen voldoen. Het algoritme moet draaien op een Raspberry Pi, samen met het controleprogramma van de fiets. Door deze beperkte resources moet het algoritme zo efficiënt mogelijk zijn. De voorspellingen moeten bijna in real time berekend worden. Het doel is om aan 10Hz de cadans aan te passen, maar hoe meer voorspellingen per seconde, hoe beter. Tragere voorspellingen kunnen hinderlijk zijn voor het rijgedrag. Ten slotte moet er ook rekening gehouden worden met de veiligheid van de fietser. Opeenvolgende voorspellingen mogen niet te veel van elkaar verschillen, anders zou de fietser erdoor gestoord kunnen worden en zijn concentratie verliezen. Bovendien mag de cadans nooit hoger dan een bepaalde maximum limiet ingesteld worden.

De algoritmes worden geëvalueerd op basis van de mean squared error tussen de ingestelde cadans - afkomstig van het machine learning algoritme - en de zogenaamde *freely chosen cadence (FCC)* van de fietser. Die freely chosen cadence is niet precies gekend en wordt in de simulatie bepaald aan de hand van een fietsersmodel. Dit is een functie die de toestand van de fiets en de fietser (rijsnelheid, helling,...) afbeeldt op de trapsnelheid die de fietser in dat geval het meest comfortabel vindt. Simpel gezegd is het fietsersmodel een functie met als input de toestand van de fiets en als output een "optimale cadans". Deze functie is speculatief en kan makkelijk aangepast worden. Op welke basis de fietser precies zijn freely chosen cadence bepaalt is voor dit onderzoek weinig relevant. Het gaat er hier vooral om dat het machine learning algoritme het fietsersmodel kan achterhalen.

Fietsersmodel: fcc = f(snelheid,koppel,vermogen,helling,...)

Het algoritme moet kunnen bijleren met een kleine hoeveelheid data. De gebruiker zal immers niet vaak manuele aanpassingen doen aan de cadans. Te veel data gebruiken kan een negatieve invloed hebben op reeds correcte voorspellingen. Het algoritme moet ook snel bijleren. Elke verandering moet zo snel mogelijk doorgevoerd worden en moeten een betekenisvolle impact hebben.

1.3 Huidige systeem

De fiets van intuEdrive gebruikt een elektrische continu variabele transmissie (CVT) die ervoor zorgt dat er naadloos geschakeld kan worden tussen versnellingen, in tegenstelling tot het traditionele ketting-en-tandwiel systeem. Dit oude systeem schakelt in discrete trappen, waardoor de fietser tijdens het schakelen een discontinuïteit voelt. Het CVT-systeem gebruikt 2 motoren en schakelt traploos. Eén van de motoren regelt de trapcadans, de andere motor regelt het ondersteuningsniveau. Het ondersteuningsniveau bepaald hoeveel extra elektrisch vermogen er geleverd wordt, bovenop wat de fietser zelf levert.

Figuur 3 toont een blokdiagram van het systeem fiets-fietser-controller. We gaan ervan uit dat de fietser op elk moment een bepaalde referentiesnelheid (v_{ref}) probeert te halen, hier aangeduid met r. Deze kan variëren naar gelang de situatie, maar is voor elke gebruiker anders. Tijdens het fietsen geeft de fietser input aan de fiets (u_{cy}). Zo kan hij of zij het geleverde koppel variëren (T_{cy}), i.e. meer of minder kracht op de pedalen zetten of de cadans aanpassen met de knoppen (u_c). Inputs en fysische toestand van

de fiets worden gemeten door sensoren op de fiets: het koppel ($T_{cy,m}$), de hoek van de trapas (θ_{cr}), snelheid (v_{bike}), helling (α), etc. T_{cy} en $T_{cy,m}$ zijn niet hetzelfde, want er kunnen fouten gebeuren tijdens het meten. De vector van meetwaarden (y) is input voor de fietscontroller. De fietscontroller stuurt de motoren in de E-bike aan (u_{contr}) op basis van de metingen y en de ingestelde referentiecadans. De cadanscontroller die in deze thesis uitgewerkt zal worden zal op basis van dezelfde metingen een gepersonaliseerde referentiecadans (FCC_{est}) voorspellen die als input dient voor de controller.

$$r = \begin{bmatrix} v_{ref} \end{bmatrix} \qquad u_{cy} = \begin{bmatrix} T_{cy} \\ u_c \end{bmatrix} \qquad cc = \begin{bmatrix} FCC_{est} \end{bmatrix} \qquad y = \begin{bmatrix} \theta_{cr} \\ T_{cy,m} \\ v_{bike} \\ \alpha \end{bmatrix}$$

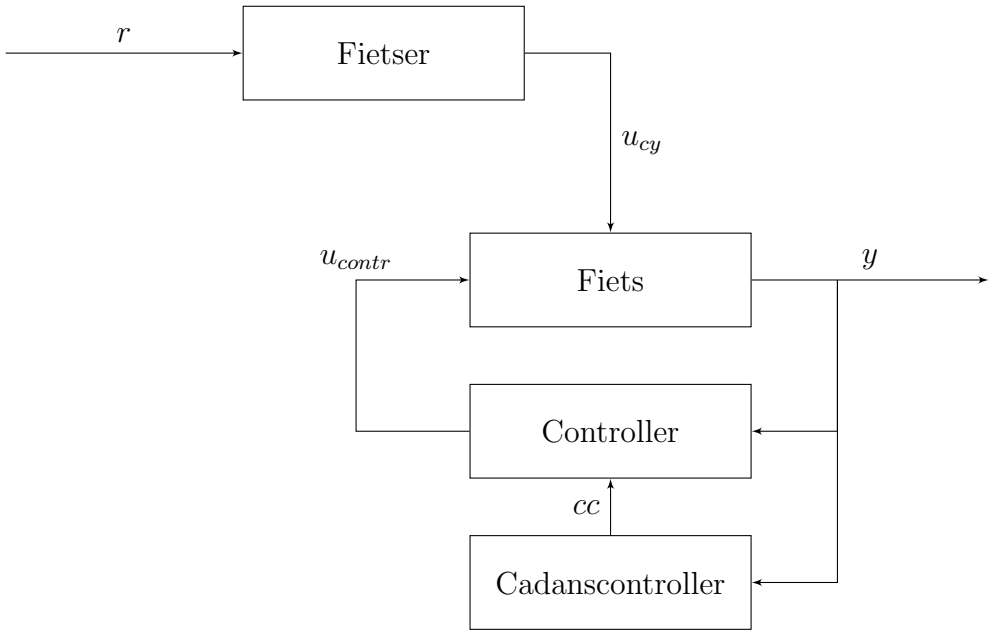


Figure 3: Blokdiagram van het fiets-fietser-controller systeem

Hoofdstuk 2

Methode

2.1 De fietssimulatie

Er wordt een simulatie gemaakt die de toestand van de fiets zo goed mogelijk probeert te benaderen. Er zal geen rekening gehouden worden met het manoeuvreren van de fiets of van tegenwind. Enkel de relevante meetwaarden worden bijgehouden. De simulatie is geparametriseerd om eenvoudig verschillende scenario's te testen.

Het voordeel van de fietssimulatie is de enorme flexibiliteit. Uren aan data kunnen in een moment tijd gegenereerd worden, waardoor het makkelijk is om verschillende tests uit te voeren. Hiervoor moeten slechts enkele instellingen aangepast worden. Het is ook mogelijk om slechts een enkele parameter aan te passen tijdens tests terwijl de rest constant blijft (*ceteris paribus*), wat praktisch onmogelijk is in een veldtest. Omdat de simulatie bovendien een duidelijke referentie genereert voor de FCC (output van het fietsersmodel), kan de performantie van de cadanscontroller op een kwantitatieve manier worden geëvalueerd. Tijdens een veldtest zou de fietser alleen kwalitatief kunnen aangeven of hij of zij de voorspelde cadans goed vind.

2.2 Modelleren van het fietserkoppel

Het fietserkoppel wordt gemodelleerd als een sinusfunctie met twee pieken per omwenteling van de trapas (2 benen), met het DC koppel van de fietser als parameter.

$$T_{cy} = T_{dc}(1 + sin(2\theta_{cr} - \frac{\pi}{6}))$$

Uit deze formule is het ook meteen duidelijk dat het DC koppel ook het vermogen-equivalent koppel is. Dat wil zeggen dat het DC koppel gedurende een volledige omwenteling van de trapas evenveel arbeid levert als het fietserkoppel.

$$\begin{aligned}\int_0^{2\pi} T_{cy}(1 + sin(2x - \frac{\pi}{6}))dx &= T_{cy} \int_0^{2\pi} (1 + sin(2x - \frac{\pi}{6}))dx \\ &= T_{cy} \left[x - \frac{1}{2} sin(\frac{1}{3}(6x + \pi)) \right]_0^{2\pi} \\ &= 2\pi \ T_{cy}\end{aligned}$$

Het gemiddelde koppel geleverd door de fietser wordt gemodelleerd als een proportionele regelaar. Het doel is om een bepaalde snelheid, v_{ref}, te behalen. Hoe groter het verschil is tussen de referentie snelheid en de eigenlijke snelheid, hoe meer kracht er geleverd zal worden. Als deze referentie snelheid overschreden wordt, dan zal er geen koppel meer geleverd worden. Dit wordt ook wel freewheelen genoemd. Om de kracht van de actor te limiteren, wordt er een maximum koppel ingesteld ($T_{dc,max}$) naar gelang de huidige cadans (ω_{cr}). Zo wordt er meer kracht geleverd wanneer de cadans laag is, net zoals in de werkelijkheid. K bepaalt de agressiviteit van de regelaar. De formules zien er als volgt uit:

$$T_{dc,max} = \frac{-\omega_{cr}}{2} + 60 \qquad (Figuur\ 4)$$
$$T_{dc} = min(T_{dc,max}, max(0, -K * (v_{bike} - v_{ref}))$$

Figuren 5a en 5b tonen een menselijk koppelverloop en gesimuleerd koppelverloop, gesampled aan 10Hz. Zoals te zien is het gesimuleerde koppel heel consistent. Het menselijk koppel volgt duidelijk een cyclische functie, maar toont vormen van inconsistentie. Merk wel op dat er telkens een afwisseling is van een hoge en een lage piek. Dit wijst op een dominant been. Figuur 5c toont een gesimuleerd koppelverloop van een fietser met een dominant been.

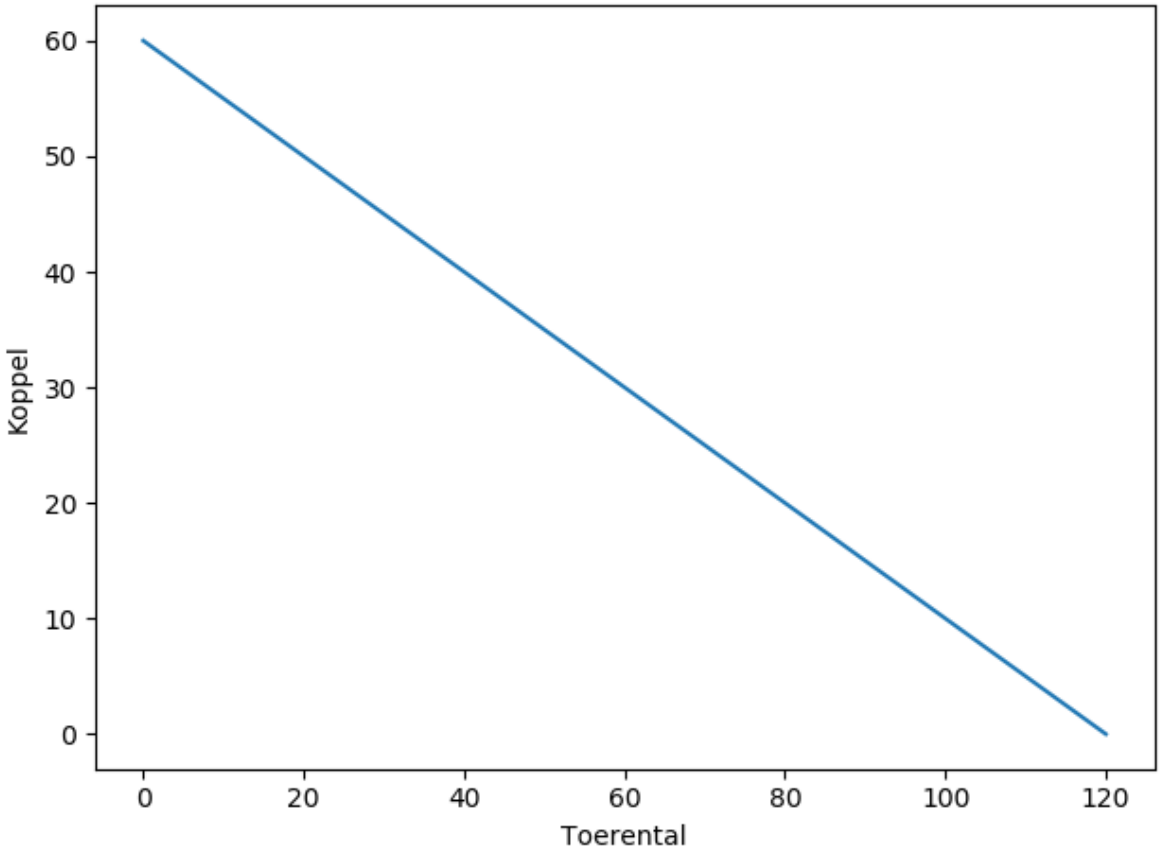
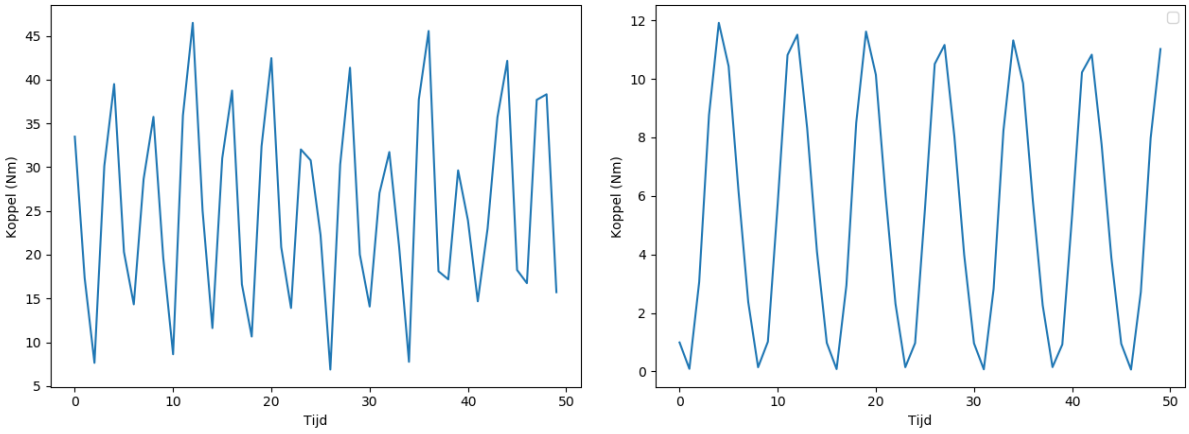
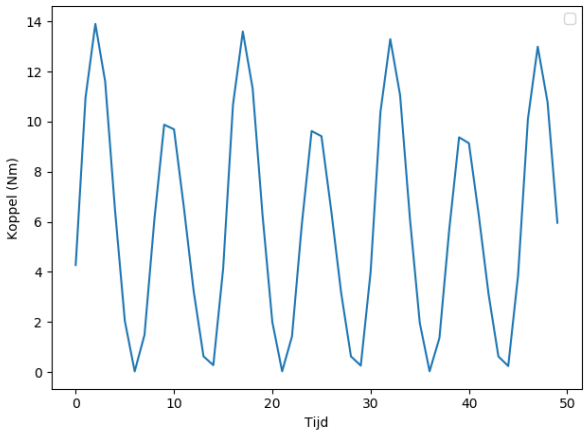


Figure 4: Het koppel-toerentalkarakteristiek



(a) Menselijk koppelverloop

(b) Gesimuleerd koppelverloop



(c) Gesimuleerd koppelverloop met dominant been

Figure 5: Het koppelverloop van een mens (linksboven), de simulatie (rechtsboven) en een gesimuleerd dominant been (onderaan)

T_{cy} is het koppel op de trapas. Dit moet nog overgebracht worden op het achterwiel. CoSaR maakt gebruik van een planeetwielmechanisme (figuur 6). Dit mechanisme laat toe om een grote overbrengingsverhouding te voorzien in een kleine ruimte. Het achterwiel-koppel wordt beïnvloed door het aantal tanden op het zonnewiel (1; ns) en het ringwiel (2; nr) en de overbrengingsverhouding tussen de trapas en het ringwiel ($k_{cr,r}$). Het koppel op het achterwiel (T_{rw}) ziet er als volgt uit:

$$T_{rw} = T_{cy} * k_{cr,r} * \frac{nr + ns}{nr}$$

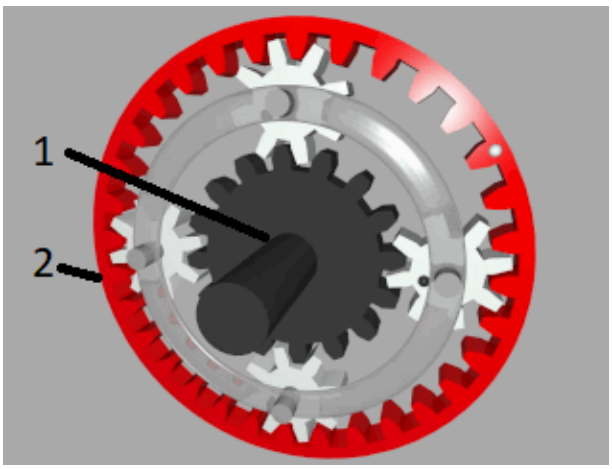


Figure 6: Planeetwielmechanisme (bron: wikipedia)

Bovenop het vermogen geproduceerd door de fietser, levert CoSaR extra ondersteuning a.d.h.v. een motor (T_{MG2}) gekoppeld aan het voorwiel. De fietser kan zelf een ondersteuningsniveau (S) instellen tussen 0 en 5. Hoe hoger dit ondersteuningsniveau, hoe minder inspanning de fietser moet leveren.

$$T_{MG2} = min(35, S\,.\,T_{cy})$$

2.3 Het fietsersmodel

Hoe kiest een fietser zijn cadans? Dit is voor elke fietser verschillend en er is nog nauwelijks onderzoek naar gebeurd. Wielrenners trainen om sneller te kunnen trappen omdat dit efficiënter is. Ze kunnen een gemiddeld vermogen leveren van 300 Watt. De doorsnee fietser levert gemiddeld ongeveer 75 Watt tijdens een normale fietstocht. Het fietsersmodel zal hierop worden afgesteld, aangezien wielrenners niet de voornaamste doelgroep zijn voor CoSaR.

Het fietsersmodel is een functie die op verschillende manieren uitgedrukt kan worden: op basis van de helling, gemiddeld koppel, of snelheid. Wat het correcte model is wordt in deze thesis niet uitgewerkt. Het is vooral van belang dat de cadanscontroller het model zo snel en zo nauwkeurig mogelijk kan achterhalen, ongeacht wat het model precies is. Hier wordt de volgende aanname gemaakt: hoe hoger het koppel geleverd door de fietser, hoe hoger de gewenste cadans. Wanneer de fietser bijvoorbeeld een helling oprijdt schakelt hij of zij een versnelling omlaag zodat de kracht die op de pedalen gezet moet worden aangenaam blijft. We stellen hier volgende eenvoudige modellen voor:

$$\begin{aligned} Gemiddeld\ koppel: &\qquad fcc = f_k.T_{dc} \\ Helling: &\qquad fcc = f_h.\alpha \\ Snelheid: &\qquad fcc = f_v.v_{bike} \end{aligned}$$

Er wordt verder aangenomen dat de fietser ook een zeker lineariteit verwacht bij lage snelheden. Dat wil zeggen dat een fietser het niet comfortabel vindt wanneer hij of zij snel moet trappen wanneer de fiets nog stilstaat of heel traag rijdt, ook al moet er op dat moment veel koppel geleverd worden om te kunnen vertrekken. Daarom wordt bij lage snelheden de FCC begrensd door een lineair oplopend maximum, te vergelijken met een mechanische fietsversnelling. Omdat de doorsnee fietser niet heel traag of heel snel trapt wordt de FCC begrensd tussen de 40 en 120 rpm.

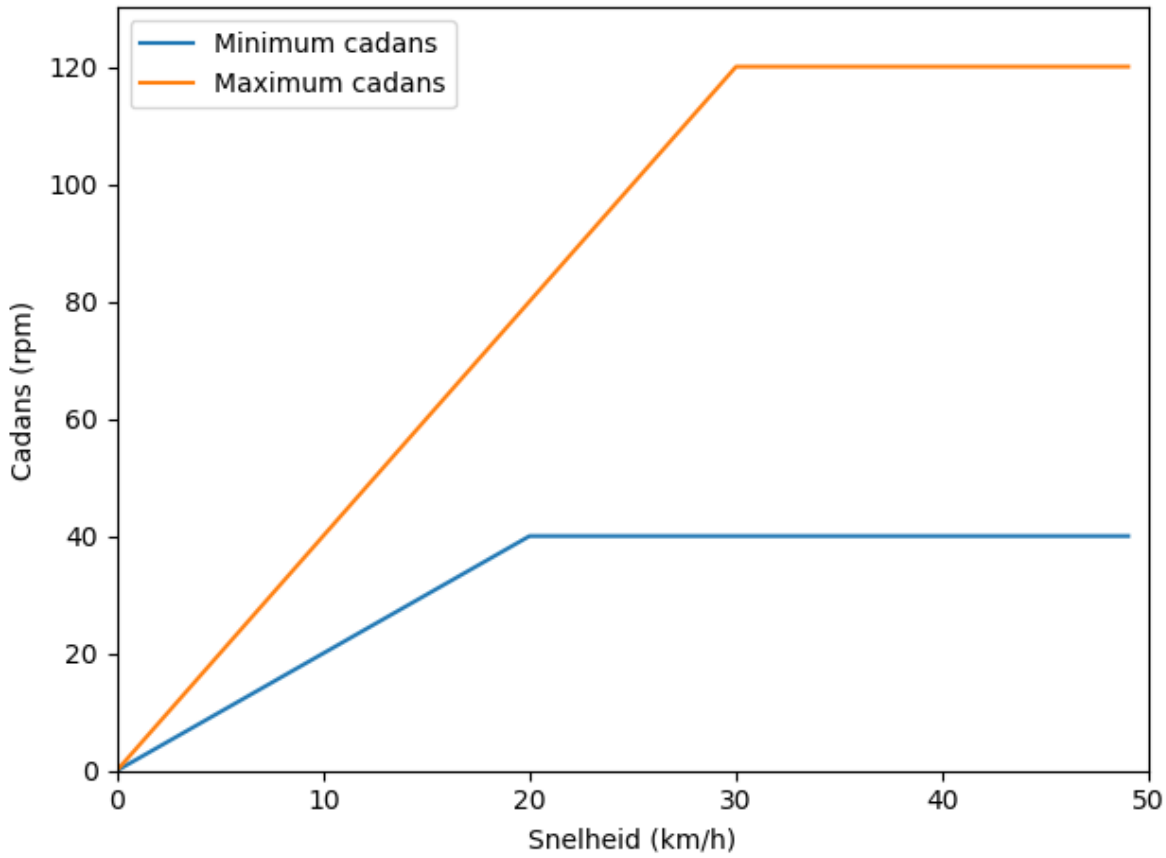


Figure 7: Verwacht cadansverloop in functie van de snelheid.

2.4 Het lastmodel

De simulatie is voorzien van een lastmodel. Zoals in realiteit, werken lasten in op de fiets. Zwaartekracht, wrijving met de weg en luchtweerstand zijn gemodelleerd als volgt:

$$F_{grav} = m \;.\; g \;.\; sin\ \alpha$$
$$F_{friction} = m \;.\; g \;.\; c_r \;.\; cos\ \alpha$$
$$F_{aero} = \frac{c_d \;.\; \rho_{aero} \;.\; A_{aero} \;.\; v_{bike}^2}{2}$$

Samen vormen ze de totale belasting op de fiets.

$$F_{load} = F_{grav} + F_{friction} + F_{aero}$$

Deze lasten zorgen ervoor dat de simulatie een realistische hoeveelheid vermogen nodig heeft om een bepaalde snelheid te halen. Er wordt hier geen rekening gehouden met de wind. Ten eerste zou dit extra complexiteit toevoegen aan de simulatie. En ten tweede vermoeden we volgende hypothese:

> De freely chosen cadence hangt af van de hoeveelheid last, van welke bron dan ook, die de gebruiker ondervindt en de gebruiker zelf.

Het voorgestelde lastmodel omvat deze vereiste. Door de helling en referentie snelheid te variëren ondergaat de fietser een veranderende last. Zoals in de realiteit zoeken mensen een bepaalde snelheid te halen. Wanneer de fietser een te hoge last ondervindt, bijvoorbeeld door een berg op te rijden, moet hij of zij meer vermogen genereren om zijn of haar gewenste snelheid te behouden. Hiervoor zijn 2 mogelijkheden: het verhogen van het koppel of de trapsnelheid. Mensen zijn meer geneigd om hun gewenste cadans te behouden, ongeacht het koppel (binnen bepaalde grenzen). De formule voor mechanisch vermogen gaat als volgt:

$$P = T_{cy} \;.\omega_{cr}$$

Om het lastmodel correct te laten werken, moet er nog een helling gegenereerd worden. Om veel werk uit te sparen met het uitstippelen van parcours, wordt dit dynamisch gegenereerd met behulp van perlin noise. Perlin noise kan gebruikt worden om willekeurige getallen te genereren waarbij opeenvolgende getallen weinig van elkaar verschillen. Een perfecte kandidaat dus om terrein te simuleren. Om verschillende trajecten te creëren, kan de seed variabele aangepast worden. Een hellingsgraad wordt in de fietswereld vaak percentueel voorgesteld. Deze implementatie heeft echter radialen nodig. De helling zal beperkt worden tussen ≈ 0 en 10% (0 en 0.1 radialen). Ter vergelijking, de Koppenberg heeft een gemiddeld stijgingspercentage van 11.6%. Het minimum stijgingspercentage is zo gekozen dat de simulatie zo weinig mogelijk gaat freewheelen.

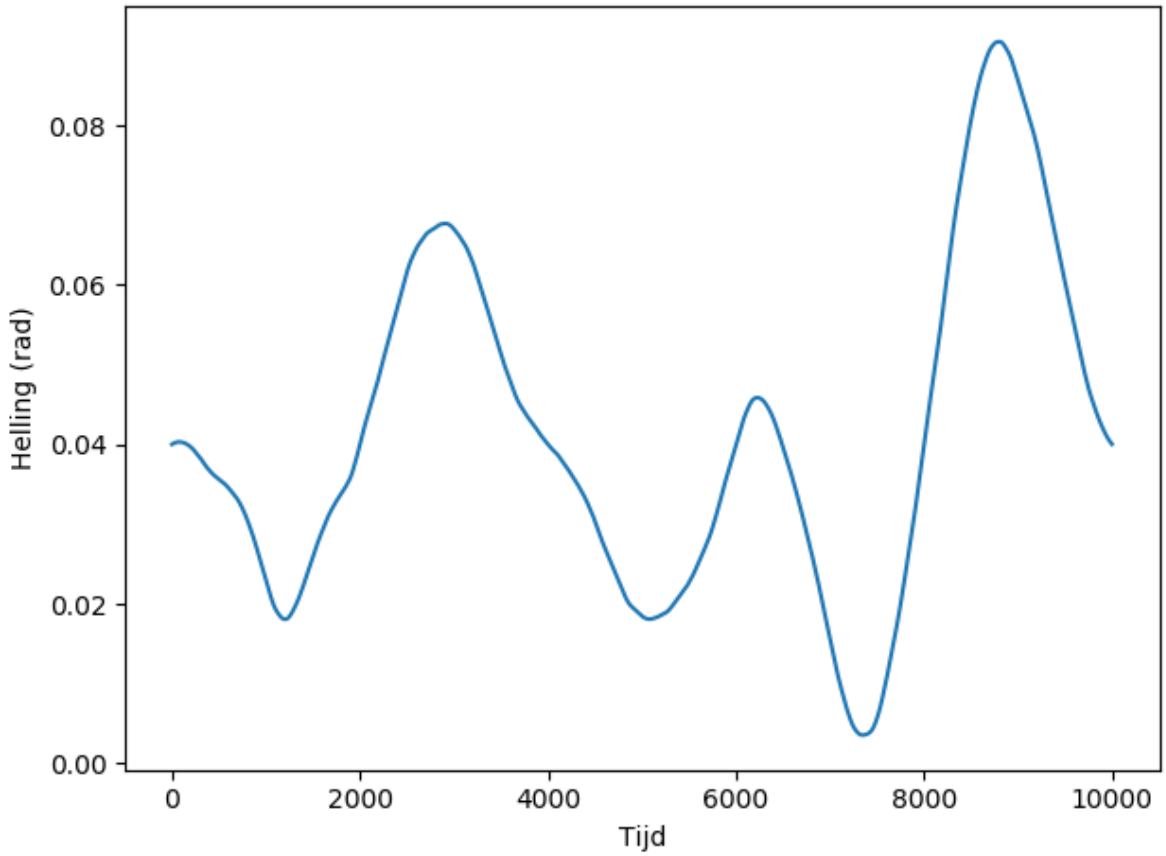


Figure 8: Voorbeeld helling verloop

2.5 Snelheidsvergelijking

De snelheid wordt berekend met een standaardformule: vorige snelheid plus acceleratie met respect tot de genomen tijdsprong. De acceleratie is in functie van de last, het totaalgewicht (m), het vermogen geleverd door de fietser op het achterwiel (T_{rw}) en het vermogen van een motor bevestigd op het voorwiel (T_{MG2}).

De bewegingsvergelijking van de fiets is, met inbegrip van het lastmodel en het fietsersmodel:

$$F \ = \ m \ . \ a$$

Deze vergelijking wordt elke tijdsstap geïntegreerd met behulp van een voorwaartse Euler methode:

$$F = m.(\frac{v_{bike}[h] - v_{bike}[h-1]}{\Delta t})$$

$$\frac{\Delta t.F}{m} = v_{bike}[h] - v_{bike}[h-1]$$

$$v_{bike}[h] = v_{bike}[h-1] + \Delta t.\frac{1}{m}.F$$

$$v_{bike}[h] \ = \ v_{bike}[h-1] \ + \Delta t \ .\frac{1}{m} \ . \ (\frac{T_{MG2} \ + \ T_{rw}}{r_w} \ - \ F_{load})$$

De volledige simulatie ziet er als volgt uit:

for h in 1..#tijdssprongen

$T_{dc,max} = \frac{-\omega_{cr}[h-1]}{2} + 60$

$T_{dc} = min(T_{dc,max},\ max(0, -K * (v_{bike}[h-1] - v_{ref}))$

$fcc = f(T_{dc})$

$\omega_{cr} = cadans(v_{bike}[h-1],\ T_{dc},\ fcc)$

$\theta_{cr} = \theta_{cr}[h-1] + \Delta t\ .\ \omega_{cr}$

$T_{cy} = T_{dc}(1 + sin(2\theta_{cr} - \frac{\pi}{6}))$

$T_{rw} = T_{cy} * k_{cr,r} * \frac{nr+ns}{nr}$

$T_{MG2} = min(35,\ S\ .\ T_{cy})$

$F_{grav} = m\ .\ g\ .\ sin\ \alpha$

$F_{friction} = m\ .\ g.\ c_r\ .\ cos\ \alpha$

$F_{aero} = \frac{c_d\ .\ \rho_{aero}\ .\ A_{aero}\ .\ v_{bike}[h-1]^2}{2}$

$F_{load} = F_{grav} + F_{friction} + F_{aero}$

$v_{bike} = v_{bike}[h-1] + \Delta t\ .\frac{1}{m}\ .\ (\frac{T_{MG2} + T_{rw}}{r_w} - F_{load})$

2.6 On-line voorspellen van de cadans

Een groot deel van de toestand van de fiets wordt berekend. Om een efficiënt algoritme te creëren moet enkel de relevante data bekeken worden. Zo worden de volgende attributen gebruikt: snelheid, koppel, hoek van de trapas en helling.

De helling is vanzelfsprekend. Dit is de voornaamste vorm van last en zal dus een impact hebben op de freely chosen cadence van de fietser. De hoek van de trapas op zich heeft niet veel betekenis. En enkel het koppel ook niet. Er kan bijvoorbeeld een koppel geleverd worden van 20Nm. Dit koppel kan op verschillende plaatsen geleverd worden in de trapcyclus. Als dit in de situatie 2 geleverd wordt, dan is dit waarschijnlijk het laagste koppel in de trapcyclus. Wordt dit geleverd gedurende de neergaande beweging (1), dan is dit het hoogste koppel. Deze verschillende situaties zullen een verschillende cadans nodig hebben. Tijdens de eerst situatie wordt er gemiddeld meer koppel geleverd, wat wijst op een grote last. Dus verwachten we hier een hoge cadans. De tweede situatie daarentegen zal gemiddeld een lagere last hebben. Daarom wordt in conjunctie met het koppel de hoek van de trapas gebruikt. Snelheid is ook een relevant attribuut. In situaties met verschillende snelheden en dezelfde last gaat de cadans verschillen.

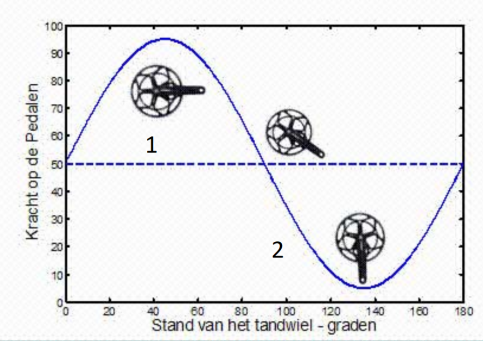


Figure 9: Evolutie koppel in functie van hoek trapas

2.7 Preprocessing

Niet elk algoritme heeft nood aan genormaliseerde input. Voor algoritmes die een afstandsfunctie gebruiken is standaardisatie cruciaal. De doelvariabele daarentegen moet niet genormaliseerd worden volgens Warren S. Sarle [4]. In zijn FAQ schrijft hij dat het standaardiseren van de output voornamelijk voordelige effecten heeft op de initiële gewichten. Wanneer er meerdere doelvariabele zijn en als deze ver uit elkaar liggen, kan het wel nuttig zijn om deze te normaliseren. In dit geval is dat niet nodig aangezien enkel de cadans voorspeld zal worden.

Ten eerste zal de data in sequentie gegoten worden. Een sequentie wordt gezien als een aantal vectoren (x_t) over verschillende tijdstippen. Een enkele data meting op zich is niet genoeg om een accurate voorspelling te maken, maar te veel data gebruiken is ook niet goed aangezien de voorspellingen tijdig moeten geleverd worden.

$$x_t = \begin{bmatrix} \theta_{cr} \\ T_{cy,m} \\ v_{bike} \\ \alpha \end{bmatrix} \qquad sequentie = \begin{bmatrix} x_t \\ x_{t-1} \\ \dots \\ x_{t-n} \end{bmatrix}$$

Ten tweede zal de hoek van de trapas niet in zijn zuivere vorm gebruikt worden. De hoek van de trapas is een variabele tussen nul en twee pi. Het probleem hier is dat het begin en het einde van een cyclus ver uit elkaar liggen. Voor de mens is het evident dat nul en twee pi hetzelfde zijn, maar voor de computer is dit een groot verschil. Daarom zal de sinus en de cosinus van de hoek genomen worden zodat het begin en einde dicht bij elkaar liggen (figuur 11). Zo leren we het algoritme bij dat de data zich cyclisch gedraagt.

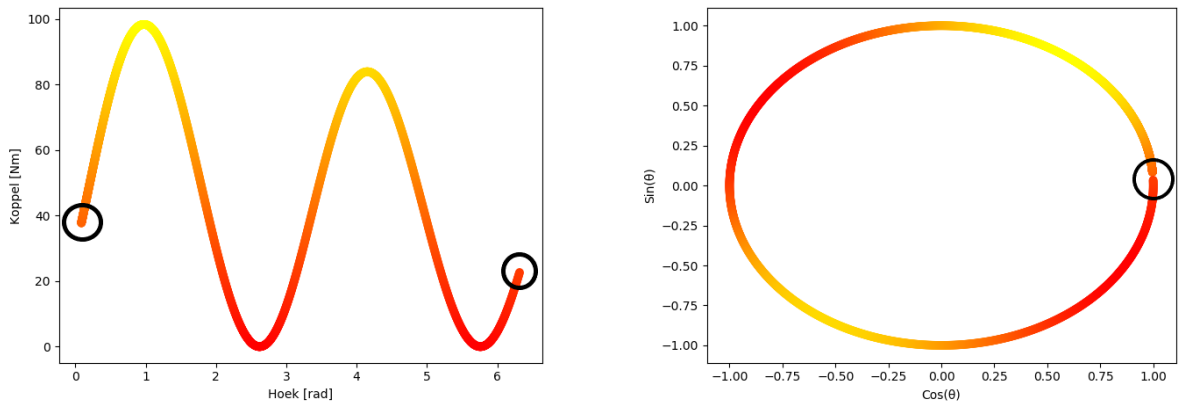


Figure 10: Figuur links toont dat het begin en einde van een trapcyclus ver uit elkaar liggen. Figuur rechts toont dat beide punten van de linkse figuur dicht bij elkaar liggen.

Ten slotte kan er ruis zitten op de metingen. Er zal altijd wel een klein foutje zitten op de data omdat de meetapparatuur niet perfect is. Het is mogelijk dat trillingen van de motor of het wegdek een impact kunnen hebben, voornamelijk op het geleverde koppel. Voor deze iteratie zal hier geen rekening mee gehouden worden. Ruis van het wegdek komt voor op een frequentie van ongeveer 20 Hz. Dit kan nog opgemerkt worden als er op

voorhand data wordt gesampled op hogere frequentie. Ruis van de motoren daarentegen komt voor op 13000 Hz. Ver boven de gewilde sample frequentie en zal dus afgebeeld worden op lagere frequenties. Het huidige systeem zal geen rekening houden met beide soorten ruis.

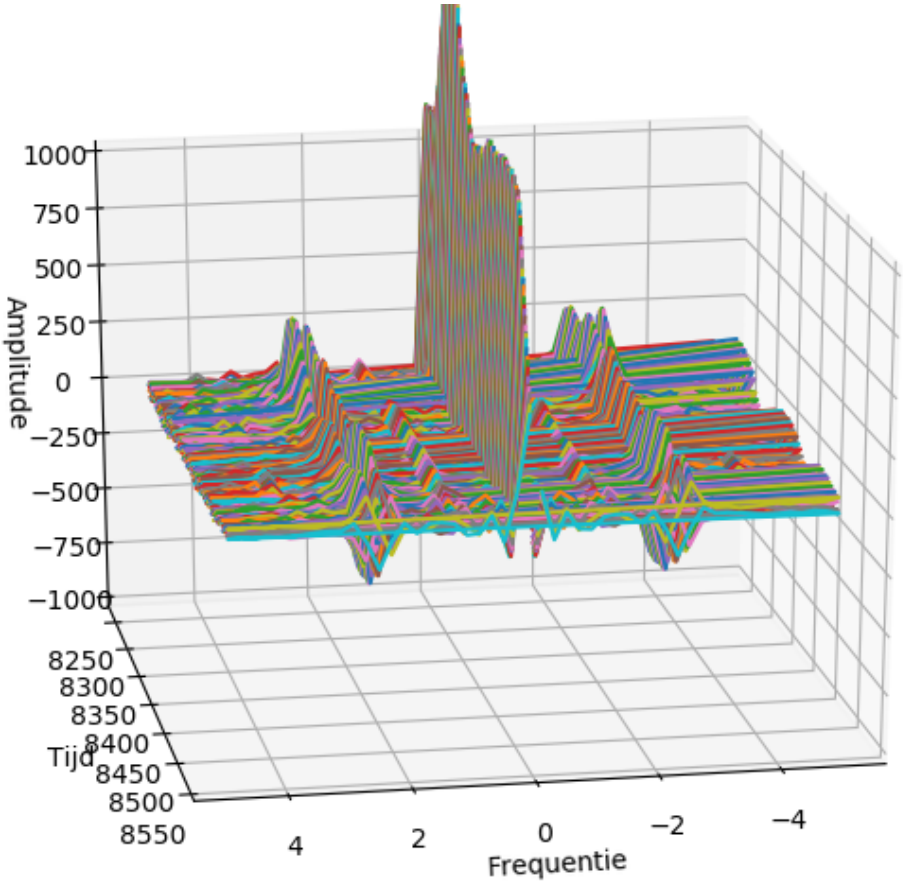


Figure 11: Een Fast Fourier Transformatie van het menselijk koppelverloop.

2.8 Algoritmes

2.8.1 Passive Aggressive Algorithm

Het passive aggressive (PA) algoritme, beschreven in de paper van Crammer et al. [5], is een online algoritme gelijkaardig aan een perceptron. Net zoals de perceptron, doet PA de matrixvermenigvuldiging $y_t = w_t \cdot x_t$ om de voorspelling te berekenen. Het grootste verschil tussen beide is hoe de gewichten geüpdatet worden. Het PA algoritme is bruikbaar voor classificatie, regressie, uniclass voorspellingen en multiclass problemen.

Het PA algoritme, zoals de naam weggeeft, kan zich zowel passief als agressief gedragen. Het trainen van PA bestaat uit twee stappen. In de eerste stap wordt er een voorspelling $y_{t,p}$ gemaakt a.d.h.v. de input vector x_t en gewichtenmatrix w. Hierna wordt het echte label y_t bekendgemaakt. Als de fout kleiner is dan een voorgedefinieerde waarde ϵ, dan zullen de gewichten niet geüpdatet worden. Als de error toch groter is dan deze marge,

dan zullen de gewichten w aangepast worden zodat de fout voor de huidige instantie nul wordt. PA past de gewichten aan zodat het verschil tussen het vorige gewicht en het nieuwe gewicht minimaal is.

$$loss_\epsilon(w_t; (x_t, y_t)) = \begin{cases} 0 & |w \cdot x - y| \leq \epsilon \\ |w \cdot x - y| - \epsilon & anderzijds \end{cases}$$

$$w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) = 0$$

Door de agressiviteit van het standaard PA algoritme kunnen er problemen ontstaan wanneer er veel ruis zit op de data. Daarom heeft Crammer et al. twee extra versies, PA-I en PA-II, gemaakt die dit probleem oplost. Beide versies voegen een slack variabele ξ toe. Deze variabele zorgt ervoor dat bij het aanpassen van de gewichten, de fout kleiner of gelijk moet zijn aan ξ in plaats van nul. Beide versies hebben ook een agressiviteits parameter C die deze ξ beïnvloedt. Dit is een vorm van regularisatie om overfitting te voorkomen.

$$PA - I \qquad w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 + C\xi \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) \leq \xi$$

$$PA - II \qquad w_{t+1} = \underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}||w - w_t||^2 + C\xi^2 \qquad zodat\ loss_\epsilon(w_{t+1}; (x_t, y_t)) \leq \xi$$

De nieuwe gewichtenmatrix w_{t+1} wordt als volgt berekent:

$$w_{t+1} = w_t + \tau_t y_t x_t$$

$$\begin{aligned} \tau_t &= \frac{loss_t}{||x_t||^2} && (PA) \\ \tau_t &= min\{C, \frac{loss_t}{||x_t||^2}\} && (PA - I) \\ \tau_t &= \frac{loss_t}{||x_t||^2 + \frac{1}{2C}} && (PA - II) \end{aligned}$$

2.8.2 Decision Tree en Random Forest

Een decision tree (DT) is een rule-based model. Dit algoritme is snel (greedy), maar kan niet goed om met ruis. Dit model is gekend om makkelijk te overfitten. Daarom wordt het random forest (RF) algoritme simultaan bekeken.

Een DT is een binaire boom. In elke knoop wordt een binair keuzepunt gemaakt op basis van een attribuut. Dit keuzepunt is gekozen zodat de data optimaal gesplitst is over beide takken. Sci-kit learn biedt de mogelijkheid aan om de maximum diepte van de boom te beperken. Wanneer deze parameter niet ingesteld is, zal de DT blijven groeien totdat alle blad nodes "puur" zijn. Een correcte diepte kiezen is een enorm moeilijke taak.

RF is een ensemble. Dit wilt zeggen dat meerdere algoritmes, in dit geval meerdere

DT's, gebruikt worden om een betere voorspelling te maken. L. Breiman [2] beweert is zijn paper dat alle RF's convergeren zodat overfitting geen probleem is. In tegenstelling tot DT's, kiest een RF geen optimaal attribuut wanneer een node gesplitst wordt. De variabele worden at random gekozen, waardoor geen enkele boom dezelfde is. De verschillende bomen trainen niet met exact dezelfde trainingsdata. Ze passen Bootstrap Aggregating toe, of bagging. Dit houdt in dat uit de originele trainingsset data wordt gesampled at random. Een instantie kan meerdere keren gesampled worden. Deze techniek reduceert de variantie.

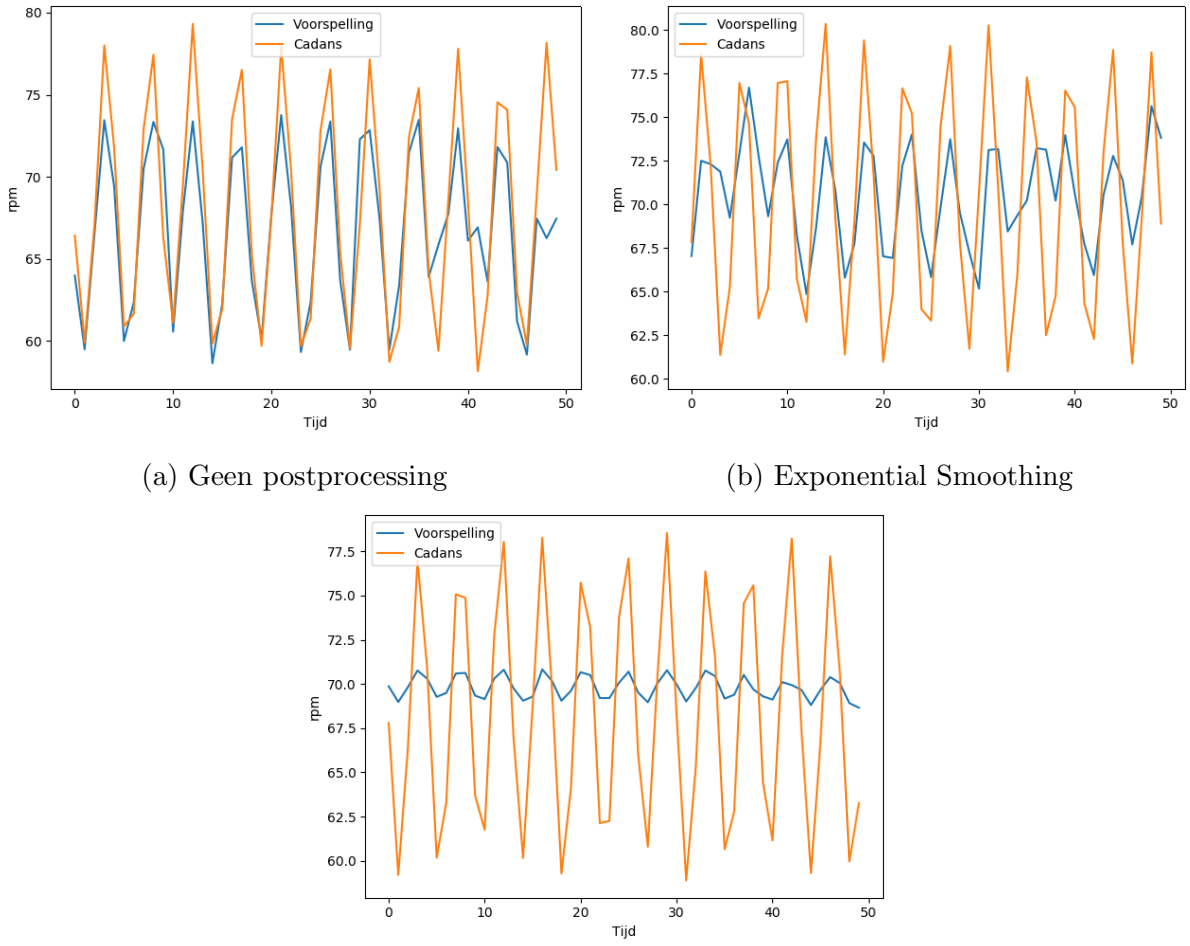
2.9 Postprocessing

De cadans variëert lichtjes doorheen een trapcyclus, zowel in het echt als in de simulatie. De voorspellingen zullen dezelfde trend vertonen. Bovendien kan een beetje ruis of het gedrag van de fietser voor afwijkingen zorgen, bijvoorbeeld een grote sprong tussen twee voorspellingen. Dit kan ergerend zijn voor de fietser.

Dit probleem kan op verschillende manieren opgelost worden. Gebruikmakend van de huidige en vorige voorspellingen (FCC_{pred}) of de vorige schatting (FCC_{est}). De vorige instelling is de cadans die is doorgegeven aan de fiets controller.

$$\begin{aligned} \textit{Moving Average } (MA) \qquad & FCC_{est,t} = \frac{\sum_{i=0}^{n} FCC_{pred,t-i}}{n} \\ \textit{Exponential Smoothing } (ES) \qquad & FCC_{est,t} = sf.FCC_{pred,t} + (1 - sf).FCC_{est,t-1} \end{aligned}$$

MA lost beide problemen goed op. Meer voorspellingen leidt tot stabielere schattingen, maar dit introduceert een vertraging (lag) op de cadans. I.e. wanneer de omstandigheden veranderen, zal de ingestelde cadans slechts na enkele iteraties optimaal zijn, in plaats van onmiddellijk. ES verminderd de amplitude van de oscillaties slechts in mindere mate. De onderstaande figuren tonen wat de impact is van ES en MA op ruizige voorspellingen (20% kans op ruis tussen -10 en +10).



(a) Geen postprocessing

(b) Exponential Smoothing

(c) Moving Average

Figure 12: De effecten van verschillende postprocessing technieken.

Hoofdstuk 3

Resultaten

De algoritmes worden getest in een on-line situatie gegenereerd door de simulatie. Er is enkel ruis toegevoegd op het koppel geleverd door de fietser. De modellen beginnen van nul. Gedurende een startperiode leren de algoritmes bij en zal de cadans bepaald worden door het fietsersmodel. Na de startperiode nemen de algoritmes deze instelling over. Na elke voorspelling wordt de mean squared error berekend tussen de voorspelling en het fietsersmodel. Elke 30 iteraties wordt er afgewogen of de algoritmes te ver afwijken van het fietsersmodel. Wanneer de absolute fout de grens van 5 rpm overschrijdt, zal er geleerd worden. De data gebruikt om bij te leren bestaat uit de toestand van de fiets van de afgelopen 100 iteraties (10s). Deze data wordt verwerkt tot een set van 50 training instanties, zijnde data van iteratie 0-49,1-50,..., 50-99. Deze set wordt toegevoegd aan de trainingsset die gebruikt wordt door de algoritmes. Met deze evaluatiemethode wordt er nagegaan hoe snel en hoe vaak het algoritme bijleert. Alle algoritmes worden geëvalueerd in exact dezelfde omstandigheden.

3.1 Sequentie preprocessing

De lengte van de sequenties heeft een invloed op de resultaten. Zoals te zien op figuur 13 heeft een te kleine sequentie negatieve invloeden op de resultaten van PA. Hoe groter de lengte van de sequentie, hoe accurater de voorspellingen worden. De tijd die het algoritme nodig heeft om de testen te voltooien stijgt ook naar gelang de grootte van de sequenties. De error en uitvoeringstijd bij DT en RF ondervinden een kleine impact bij het variëren van de lengte van de sequenties.

Om goede resultaten te krijgen zetten we de lengte van de sequenties boven de 20. Wat juist de beste optie is, is moeilijk te zeggen. Een kleine sequentie omvat maar enkele omwentelingen van de pedalen. Als er hier een grote inconsistentie voordoet kan dit slechte resultaten opleveren. Daarom zullen alle tests vanaf dit punt een constant lengte van 50 hebben.

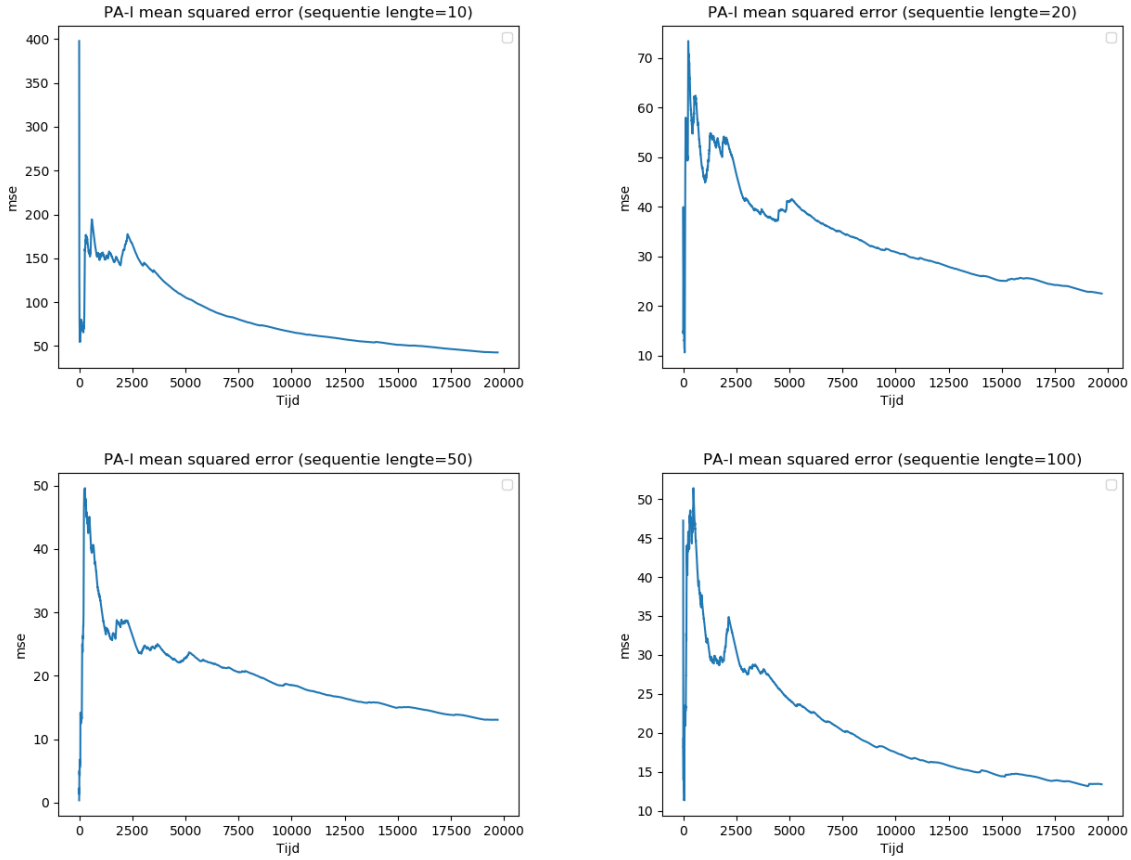


Figure 13: De invloed van sequentielengte op de error

3.2 Algoritmes

3.2.1 Passive Aggressive Algorithm

Er zijn enkele hyperparameters die ingesteld kunnen worden. Max_iter is het maximum aantal iteraties dat het algoritme probeert bij te leren. Tol is een parameter die bepaalt of het algoritme vroegtijdig stopt. Dit gebeurt wanneer de fout na een leercyclus met minder dan tol verbeterd. In de testomgeving, met $max\_iter = 25$ en $tol = 0.1$, wordt deze vroegtijdige stop altijd behaalt. Met andere woorden, na 25 iteraties heeft het algoritme de trainingsdata geleerd.

Een laatste interessante parameter is C: de agressiviteit parameter. Hoe hoger deze is, hoe agressiever het algoritme de gewichten gaat bijwerken. De onderstaande figuur toont hoe de C parameter de mean squared error beïnvloed van zowel PA-I als PA-II. C is de enige parameter die aangepast wordt.

Bijna alle tests convergeren naar een mse tussen tien en vijftien, met een gemiddelde tussen dertien en veertien. Wat het verschil is tussen PA-I en PA-II valt niet duidelijk te zien. In de paper van Crammer et al. [5] worden beide versies vergeleken op basis van instance noise en label noise. In beide gevallen scoren PA-I en PA-II aanzienlijk beter dan het standaard algoritme. In dit experiment scoren PA-I en PA-II gelijkaardig.

Figuur 15 toont de invloed van C op het aantal keer trainen. Het verschil tussen de verschillende settings is klein. De “stappen” die genomen worden tijdens het trainen zullen dus vaak klein genoeg zijn zodat C=1 agressief genoeg is. Het is dus niet nodig om hoge C-waarden (5-10) te gebruiken. Deze data is genomen over tien sessies, van elk 20000 iteraties, en duurde telkens gemiddeld 90 seconden.

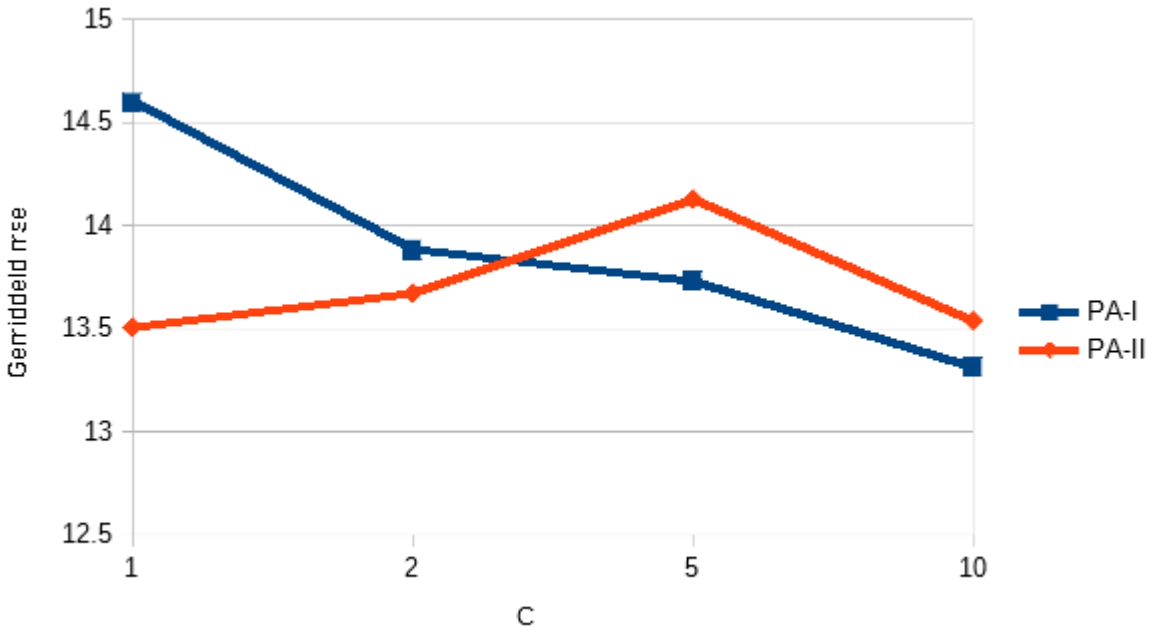


Figure 14: De invloed van C op mse van PA

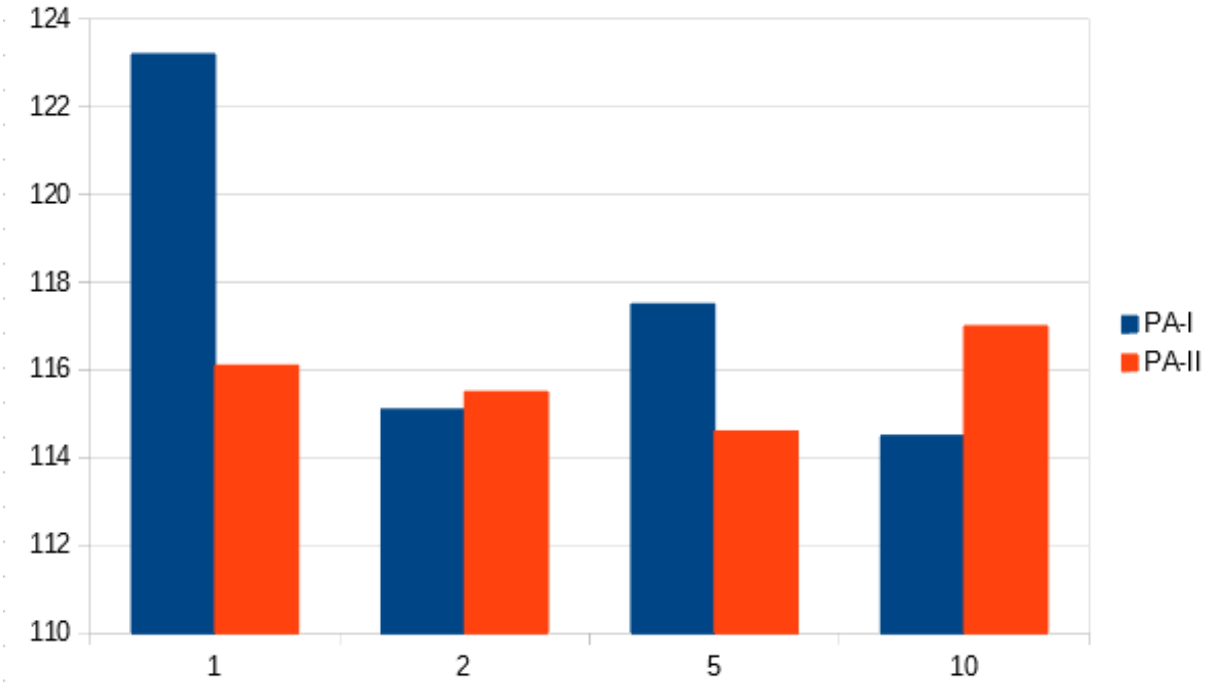


Figure 15: Gemiddeld aantal keer trainen PA

3.2.2 Decision Tree en Random Forest

De belangrijkste parameter bij deze rule-based learners is de *max_depth*. Dit is een moeilijk in te schatten parameter, vooral voor de DT. Diepere DT's maken betere voorspellingen, maar op een bepaald punt begint de DT te overfitten. Bij RF kan ook het aantal bomen ingesteld worden. Hoe meer bomen er gebruikt worden, hoe minder invloed ruis heeft op voorspellingen en hoe beter de voorspellingen worden.

DT's en RF's convergeren beide naar ongeveer dezelfde mean squared error. Diepere bomen leiden evident naar een lagere mean squared error. Algemeen zijn grotere RF's beter, maar in deze situatie is het verschil klein.

Voor DT, daalt het gemiddeld aantal trainingen spectaculair tussen diepte drie en vier (figuur 17). Een DT van diepte drie zal dus geen goede oplossing zijn. RF's daarentegen presteren wel goed met een diepte van drie. Een DT/RF dieper dan vier is niet nodig, aangezien dit geen groot voordeel oplevert en mogelijk overfit.

De uitvoeringstijd (figuur 18) lijkt geen probleem te vormen voor grotere RF. Het verschil tussen de uitvoeringstijden van een RF met diepte drie en de andere is te wijten aan het aantal keer dat getraind moest worden op diepte drie.

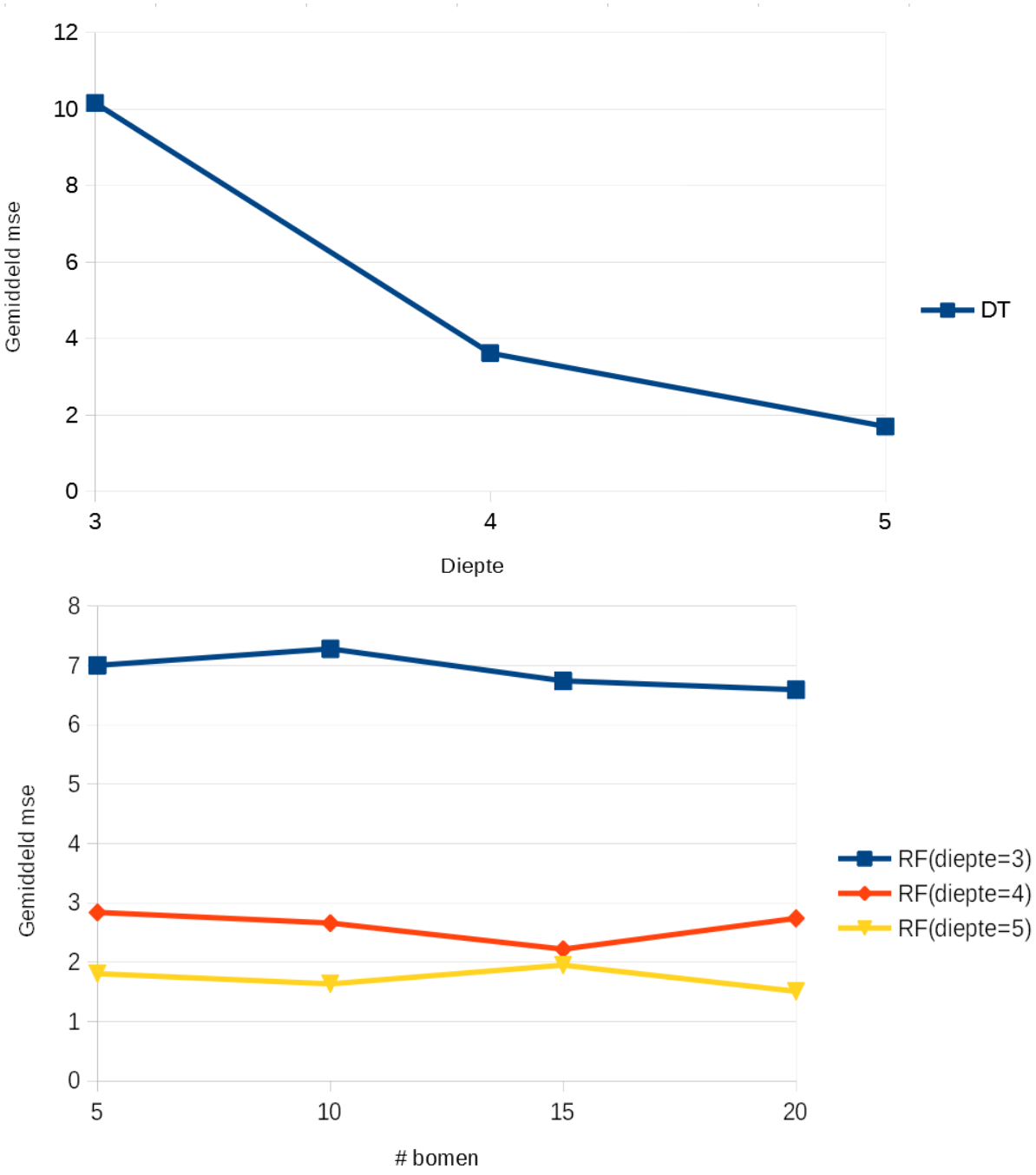


Figure 16: De invloed van diepte en aantal bomen op de gemiddelde mean squared error van DT en RF (10 keer 20 iteraties)

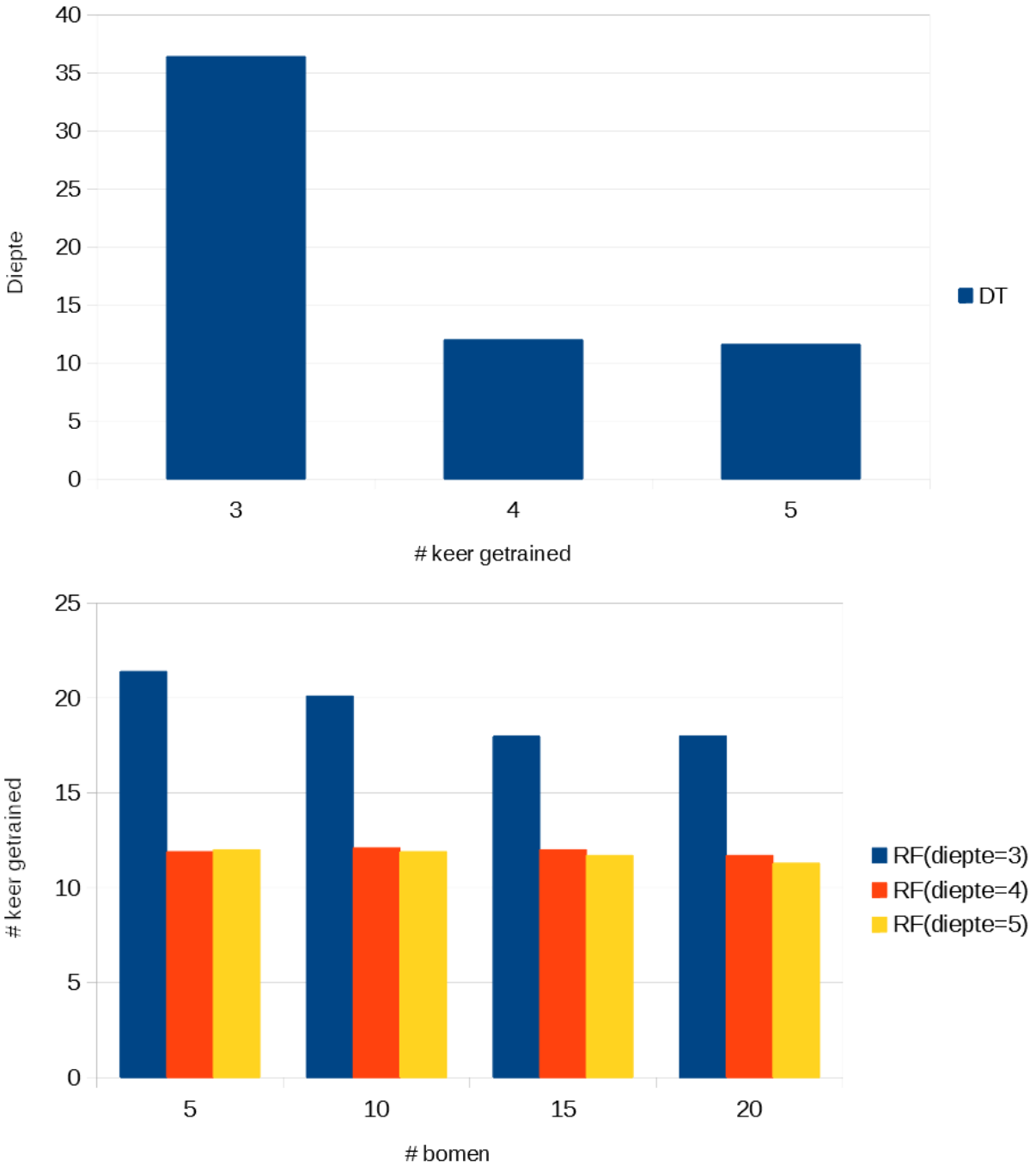


Figure 17: De invloed van diepte en aantal bomen op het gemiddeld aantal keer trainen van DT en RF (10 keer 20 iteraties)

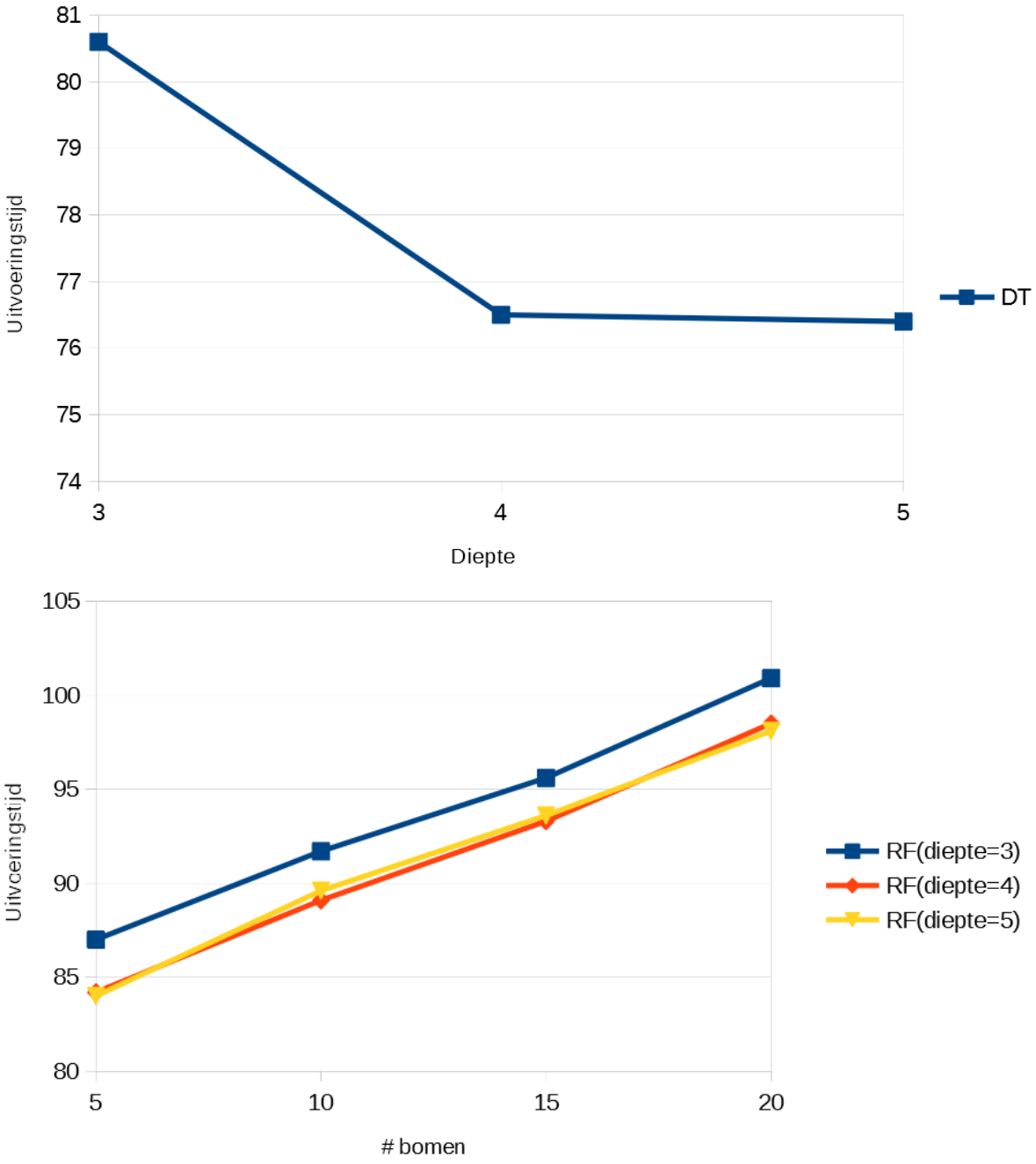


Figure 18: De invloed van diepte en aantal bomen op de uitvoeringstijd van DT en RF (10 keer 20 iteraties)

Hoofdstuk 4

Discussie

Appendix

IntuEdrive

Ir. Tomas Keppens werkte als departementshoofd bij Toyota toen hij in 2010 met het IntuEdrive project begon. In het kader van aan aantal masterproeven aan de KU Leuven werd het project stap per stap uitgewerkt.. In het academiejaar 2016-2017 werkte ingenieursstudent Jorrit Heidbuchel de aansturing van het intuEdrive CVT systeem uit. Halverwege 2017 was er het eerste prototype. Later dat jaar richtten Tomas Keppens en Jorrit Heidbuchel samen intuEdrive op.

IntuEdrive wil mobiliteit duurzamer en efficiënter maken door twee-wielmobiliteit veiliger te maken. Het bedrijf bouwt en ontwikkelt E-bikes die perfect passen in het leven van zijn klanten: makkelijk in gebruik, veilig en betrouwbaar.
Vandaag telt IntuEdrive 4 werknemers. CoSaR gaat in het najaar van 2019 voor het eerst in productie en mikt in 2020 op een verkoop van 1000 stuks in België.

Figure 19: Logo IntuEdrive

Bibliography

[1] Heidbuchel, J. (2017); Hardware Implementation and Control Strategy of a High Dynamic CVT Transmission for an E-Bike; KULEUVEN

[2] Breiman, L. (2001); Random Forests; UC BERKELEY

[3] London I. (2016); Encoding cyclical continuous features - 24-hour time; https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

[4] Sarle, W. comp.ai.neural-nets FAQ; http://www.faqs.org/faqs/ai-faq/neural-nets/part1/preamble.html

[5] Crammer, Dekel, Keshet, Shalev-Shwartyz, Singer (2006); Online Passive-Aggressive Algorithms; Hebrew University of Jerusalem

[6] https://en.wikipedia.org/wiki/Continuously_ variable_ transmission

[7] https://nl.wikipedia.org/wiki/Vermogen_%28natuurkunde%29

[8] https://en.wikipedia.org/wiki/Random_ forest

Computerwetenschappen
Celestijnenlaan 200 A bus 2402
3000 LEUVEN, BELGIË
tel. + 32 16 32 77 00
fax + 32 16 32 79 96
www.kuleuven.be