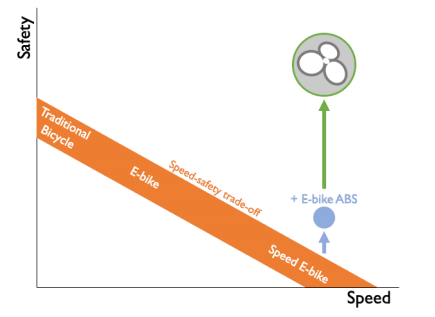
**Breder Kader: mobiliteitsvraagstuk**

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 (tabel ##). 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. Hierdoor nemen de risico’s toe. 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.

IntuEdrive’s *CoSaR* is een snelle elektrische fiets die veiliger is dan de klassieke mechanische fiets, dankzij hun innovatie 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 CoSaRde perfecte e-bike voor woon-werkverkeer: makkelijk, veilig en betrouwbaar.

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.

**Online machine learning voor geïndividualiseerde cadanscontrole.**

Deze thesis werkt verder op het prototype geleverd door IntuEdrive, met als doel het aantrekkelijker maken van hun e-bike. 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 naar de huidige omstandigheden en de individuele gebruiker. Dit zorgt ervoor 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 naar de huidige omstandigheden zal een machine learning algoritme ontwikkeld worden dat de toestand van de fiets als input binnenkrijgt en hiermee de trapsnelheid berekent. Naast het voorspellen van de cadans, moet de fietser ook kunnen aangeven dat de huidige cadans onaangenaam is. Zo zullen de knoppen die momenteel gebruikt worden om de trapsnelheid aan te passen, opnieuw gebruikt worden om aan te geven dat het algoritme moet bijleren.

Om deze doelstellingen te bereiken zal er eerst een simulatie opgezet worden die de toestand van de fiets zo goed mogelijk benadert. De berekeningen hiervoor zijn geleverd door Jorrit en zullen gecodeerd worden in python. Vervolgens worden een aantal machine learning algoritmes getest om te zien of deze het probleem aankunnen. De algoritmes maken deel uit van 2 libraries: Keras en scikit learn.

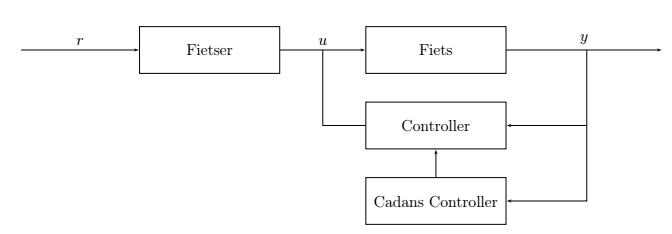
De cadanscontrole moet aan verschillende eisen voldoen. Het algoritme moet draaien op een Raspberry Pi, samen met het controleprogramma van de fiets. Dus het algoritme moet vrij lightweight zijn. De voorspellingen moeten in real time berekend worden. Het doel is om aan 10Hz de cadans aan te passen, maar hoe meer voorspellingen per seconde, hoe beter. Tragere voorspellingen kunnen hinderlijk zijn voor het rijgedrag. Tenslotte moet er natuurlijk ook rekening gehouden worden met de veiligheid van de fietser. Opeenvolgende voorspellingen mogen niet te veel van elkaar verschillen. Bovendien moet er een limiet staan op het maximum aantal toeren dat voorspeld kan worden.

De algoritmes worden geëvalueerd op basis van de mean squared error. Omdat we niet exact weten wat de optimale cadans is moet er ook een notie zijn van een fietsersmodel, een actor die naar gelang van de gesimuleerde toestand beslist wat de optimale trapsnelheid is. Simpel gezegd is het fietsersmodel een functie met als input de toestand van de fiets en output een “optimale cadans”. Deze functie is speculatief en kan makkelijk vervangen worden. Ten tweede moet het algoritme kunnen bijleren met een kleine hoeveelheid data. Aanpassingen gemaakt door de gebruiker zijn enkel relevant in de huidige context. Te veel data gebruiken kan een negatieve invloed hebben op reeds correcte voorspellingen. Ten slotte moet het algoritme snel bijleren. Elke verandering moet zo snel mogelijk doorgevoerd worden en moeten een betekenisvolle impact hebben.

# Huidige systeem

Het huidige systeem gebruikt een e-CVT – een elektrische continu variabele transmissie – dat ervoor zorgt dat er naadloos kan geschakeld worden tussen versnellingen, in tegenstelling tot het traditionele ketting en tandwiel systeem. Wanneer men schakelt met het oude systeem, voelt de fietser een schok, in tegenstelling tot wat de CVT brengt. Het CVT-systeem gebruikt 2 motoren die elk invloed hebben op de trapcadans. Een van de motoren bepaald de eigenlijke snelheid van de trapas, de andere motor bepaald de hoeveelheid kracht – ook wel koppel genoemd – dat geleverd moet worden. Samen zorgen deze motoren ervoor dat de fiets natuurlijk aanvoelt.

Het huidige CoSaR is al enkele iteraties ouder dan het prototype van Jorrit zijn masterproef, maar in essentie is het nog steeds dezelfde fiets. Figuur ## toont een blokdiagram van het huidige systeem. We gaan ervan uit dat de fietser op elk moment een bepaalde snelheid zoekt te halen, hier aangeduid met r. Deze kan variëren naar gelang de situatie. Tijdens het fietsen wordt de toestand van de fiets door de fietser aangepast. Zo kan hij of zij het geleverde koppel variëren (t\_cy), i.e. meer of minder kracht op de pedalen zetten of zelfs zijn cadans aanpassen met de knoppen (u\_c). Een groot deel van de fietstoestand wordt gemeten door sensoren op de fiets zoals: het koppel (t\_cy,m), de hoek van de trapas (θcr), snelheid (v\_bike), helling (α), etc. Deze output wordt gebruikt door de controller om de fiets verder aan te sturen. Deze controller draait op een Raspberry Pi en is gecodeerd in C. De cadans controller, wat in deze thesis uitgewerkt zal worden, zal deze metingen gebruiken om de cadans te voorspellen die dan gebruikt kan worden door de controller.



R=[v\_ref] u=[Tcy,uc] y=[θcr,Tcy,m,vbike,α] (TODO: deftig afbeelden)

Hoofdstuk 1: Simulatie

**Waarom een simulatie  
 voor en nadelen**

Zoals reeds aangehaald wordt er een simulatie gemaakt die de toestand van de fiets zo goed mogelijk probeert te benaderen. Om de werkdruk reëel te houden is de simulatie beperkt tot hetgeen wat nodig is. Zo zal er geen rekening gehouden worden met het manoeuvreren van de fiets, wind tegen of mee. Er zal ook enkel de toestanden berekend worden die gemeten worden door de fiets. Het is immer onnodig om data te verzamelen waar de fiets geen weet van heeft.

Het voordeel van de fietssimulatie is de enorme flexibiliteit. Uren aan data kan in een moment tijd gegenereerd worden, waardoor het makkelijk is om verschillende tests uit te voeren. Er is wel een hoop gelogde data beschikbaar. Maar deze heeft veel nul waarden, aangezien er niet constant door gefietst kan worden. Hierdoor is het moeilijk om een lange reeks van opeenvolgende data punten te verkrijgen. Bovendien kan deze data een bias hebben voor een bepaalde route. Waardoor we niet genoeg data hebben in verschillende omstandigheden. Met de fietssimulatie kunnen makkelijk verschillende situaties gegenereerd worden, waardoor we over enorm gevarieerde data kunnen beschikken. Tenslotte is het mogelijk om slechts 1 parameter aan te passen tijdens tests, wat praktisch onmogelijk is in realiteit.

Het voornaamste nadeel is dan toch wel de onnatuurlijke consistentie van de fietssimulatie. De geprogrammeerde actor is zeer consistent met de hoeveelheid vermogen dat hij levert. Er kan wel wat ruis toegevoegd worden, of zelfs een dominant been. Maar er is toch nog wel een verschil met een echte fietser. Zo zullen opeenvolgende cycli van de simulatie niet ver uit elkaar liggen en een trend vormen. Deze trend is bijvoorbeeld het versnellen van de cadans. Een menselijke fietser daarentegen trapt inconsistent. Er zal bijvoorbeeld wel een gelijkaardige trend ontstaan, maar opeenvolgende cycli tonen een eerder inconsistente versnelling of vertraging.

**modelleren van trapkracht = sinus (foto sinus tov echte cadans), reëel?**

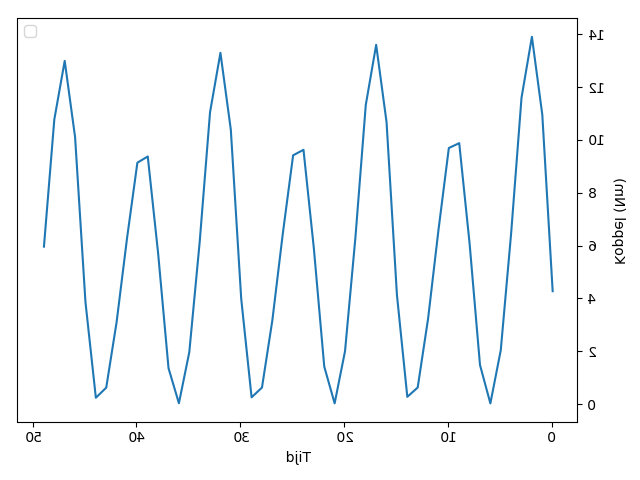
De modellering van het koppel is in functie van het gemiddelde koppel en een sinus functie. We gaan ervan uit dat 1 cyclus een volledige omwenteling is van de trapas. Het gegenereerde koppel vormt 2 pieken, 1 piek voor elk been. De functie ziet er als volgt uit:

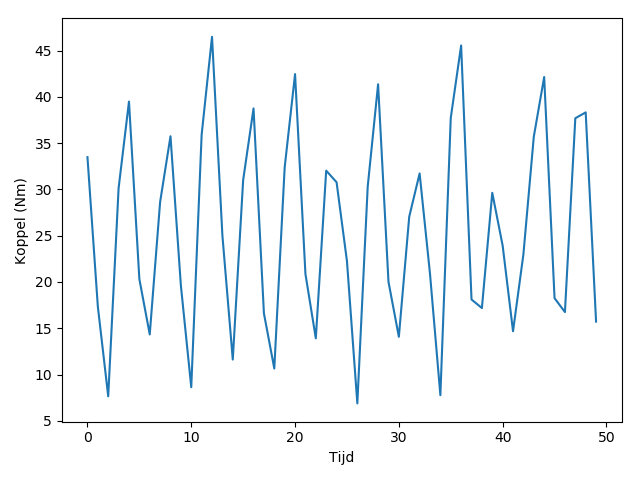
t\_cy=t\_dc\*(1+sin(2θ-pi/6))

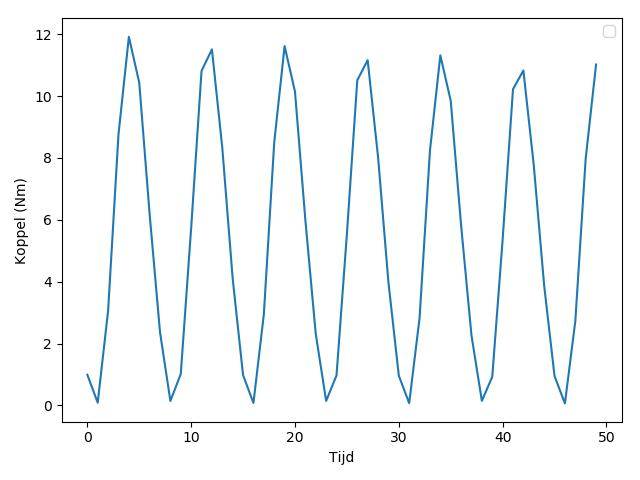
Het gemiddelde koppel geleverd door de fietser wordt gemodelleerd als een proportionele regelaar. Het doel is om een bepaalde snelheid, v\_ref, te behalen. Hoe verder de simulatie af is van deze doelstelling, hoe meer kracht er zal geleverd worden. Als deze snelheid overschreden is, dan wordt er geen koppel meer geleverd, ook wel freewheelen genoemd. Er wordt dus geen koppel meer geleverd. Om de kracht van de actor de limiteren, wordt er ook een maximum koppel ingesteld naar gelang de huidige cadans. De agressiviteit dat de actor doorvoert om de snelheid te halen wordt hier bepaald door K. De formules zien er als volgt uit:

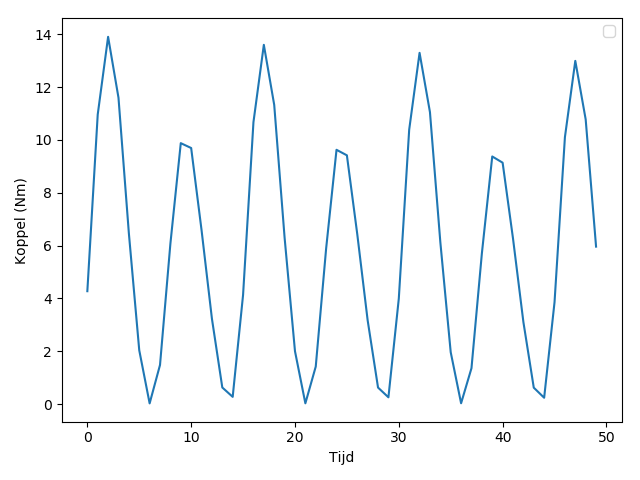
t\_dc\_max = (-omega\_crank[h - 1]) / 2 + 60  
 t\_dc = min(t\_dc\_max, max(0, -K \* (v\_fiets[h - 1] - v\_fiets\_ref)))

Hoe realistisch is deze modellering juist? Figuren ## en ## tonen het gesimuleerde en menselijk koppel respectievelijk, beide gesampled aan 10Hz. Zoals te zien valt is het gesimuleerde koppel enorm 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 ## toont een gesimuleerde dominant been. We stellen hier vast dat de gekozen modellering van het koppel realistisch genoeg is.







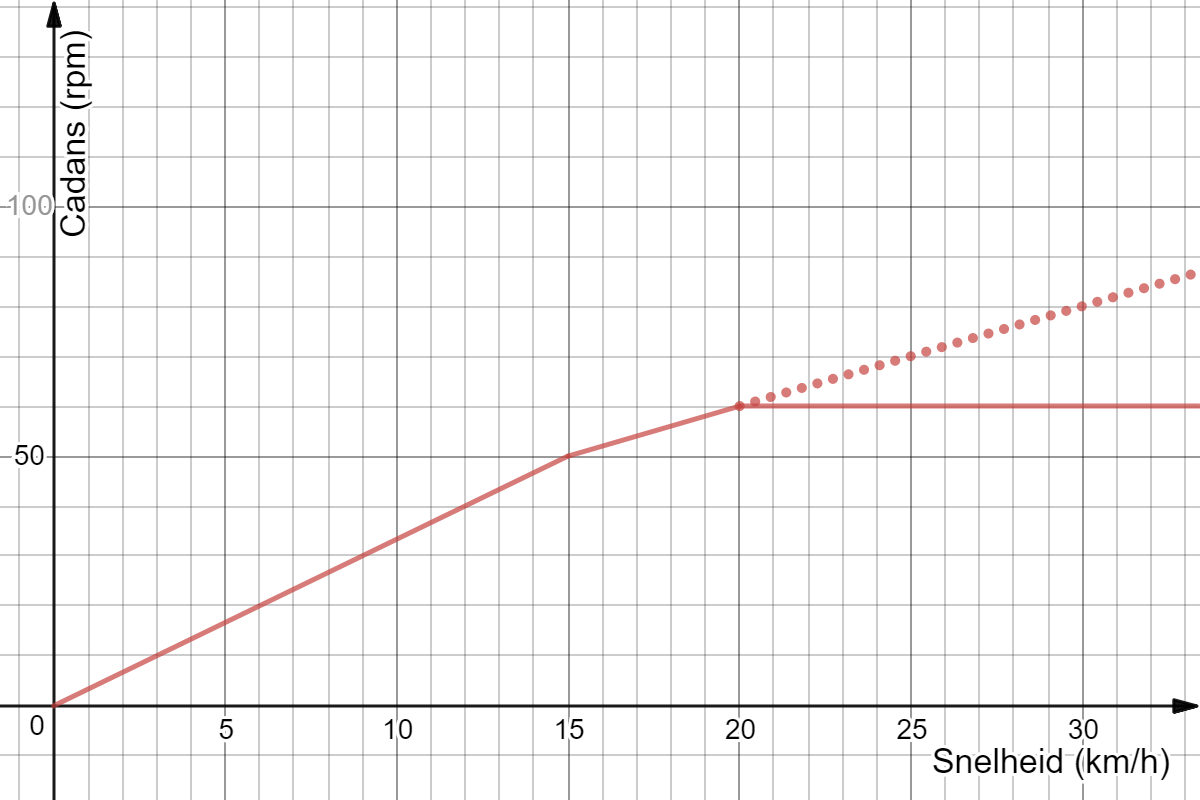
****

**fietsmodel,fcc  
 verschillende mogelijkheden, plug and play**

Hoe kiest een fietser zijn optimale cadans? Er is veel informatie beschikbaar over de optimale cadans – of freely chosen cadence (fcc) – bij wielrenners, maar dit is niet de voornaamste doelgroep voor CoSaR. Wielrenners kunnen een gemiddeld vermogen halen van 300 Watt. Ter vergelijking met de doorsnee fietser die gemiddeld ongeveer 75 Watt levert tijdens een normale fietstocht en 100 Watt wanneer hij of zij moeite doet. Het fietsersmodel zal hierop worden afgericht.

Het fietsersmodel kan een functie zijn die op verschillende manieren uitgedrukt kan worden: op basis van de helling, gemiddeld koppel, of een stabiel vermogen. Wat het correcte model is wordt hier niet bewezen. Er wordt een intuïtieve formule toegepast op basis van het gemiddeld fietserskoppel. De gedachte gaat als volgt: hoe hoger het koppel geleverd door de fietser, hoe hoger de optimale cadans is. Met andere woorden, wanneer de fietser bijvoorbeeld een helling oprijdt schakelt hij of zij een versnelling omhoog zodat de kracht die op de pedalen gezet moet worden aangenaam blijft. Dit zit verwerkt in de volgende lineaire formule:

fcc = 7,5 . t\_dc

Deze fcc wordt gebruikt in een functie afhankelijk van de snelheid om de eigenlijke cadans (ω\_crank) te berekenen. Het zou immers onrealistisch zijn om te vertrekken bij lage snelheid beginnende met een hoge cadans. De fcc zal een bovengrens vormen voor deze functie (figuur ##). Omdat de doorsnee fietser niet continu trapt, i.e. iemand gaat niet aan 5 rpm of trappen, wordt de fcc begrensd tussen 40 en 120 rpm. Figuur ## toont een voorbeeld van deze berekening waarbij de fcc 70 rpm is. Zoals te zien is, kapt fcc deze functie af.

**waarom een goed lastmodel nodig? Geen wind tegen? Hypothese: cadans in functie van last**

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

F\_grav = m.g.sin α

F\_friction = m.g.cos α

F\_aero = ½ . c\_d . ρ\_aero . A\_aero . V2

Samen vormen ze de totale belasting op de fiets dat kwadratisch is ten opzichte van de snelheid.

F\_load = F\_grav + F\_friction + F\_aero

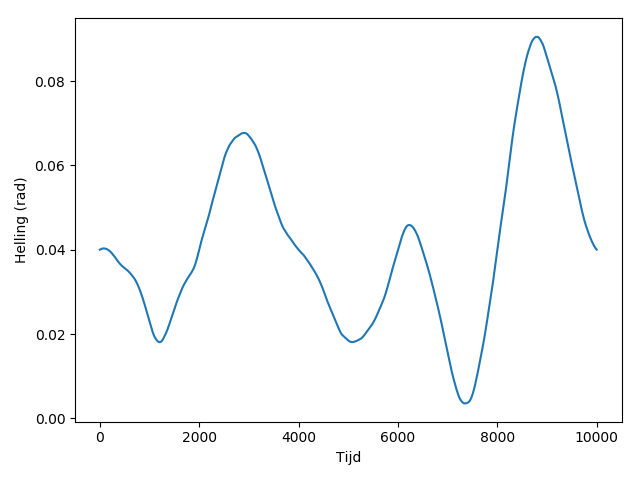
Deze lasten zorgen ervoor dat de simulatie een realistische hoeveelheid vermogen nodig heeft om een bepaalde snelheid te halen. Er wordt hier geen rekening gehouden met de wind. Ten eerste zou dit extra complexiteit toevoegen aan de simulatie. En ten tweede vermoeden we volgende hypothese:

De optimale cadans 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 meer last. Zoals in de realiteit zoeken mensen een bepaalde snelheid te halen. Wanneer de fietser een 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. De formule voor mechanisch vermogen gaat als volgt:

P=T\_cy . ω

Om het lastmodel correct te laten werken, moet er nog een helling gegenereerd worden. Om veel werk uit te sparen 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. Een hellingsgraad wordt in de fietswereld vaak percentueel voorgesteld. In dit geval zal de helling beperkt worden tussen ≈ -2% en 10%. 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.



Nu hebben we alle onderdelen om de fiets te laten rijden. De snelheid wordt berekend met een standaard formule, vorige snelheid plus acceleratie met respect tot de genomen tijdssprong. De acceleratie is in functie van de last, het totaal gewicht, het vermogen geleverd door de fietser op het achterwiel (T\_rw) en het vermogen van een motor bevestigd op het voorwiel (T\_MG2). T\_rw wordt berekend met behulp van het koppel geleverd door de fietser en verschilt van band tot band. T\_MG2 is ook in functie van het koppel, maar wordt “geholpen” door een ondersteuningsniveau ingesteld door de gebruiker.

v[h] = v[h-1] + Δt . 1/(m\_fiets+m\_fietser) . [((T\_MG2+T\_rw)/r\_w) – F\_load]

for h in 1..#tijdssprongen

t\_dc = min(t\_dc\_max, max(0, -K \* (v\_fiets[h - 1] – v\_fiets\_ref)))

fcc= optimale\_cadans(t\_dc)

ω\_crank = cadans(v[h-1],t\_dc,fcc)

θ\_crank[h] = θ\_crank[h-1] + Δt . ω\_crank

t\_cy = fietsers\_koppel(t\_dc, θ\_crank[h]) #wordt gebruikt om T\_MG2 en T\_rw te berekenen

f\_load = last(v[h-1], bereken\_helling(h))

v[h] = v[h-1] + Δt . 1/(m\_fiets+m\_fietser) . [((T\_MG2+T\_rw)/r\_w) – f\_load]

**Appendix**

# IntuEdrive

Voor het ontstaan van intuEdrive werkte Ir. Tomas als directeur bij Toyota. In 2010 begon hij met het CoSaR project. Hij startte meerdere masterproeven op aan de Universiteit KU Leuven om dit idee stap per stap uit te werken. Na jaren van samenwerking werpte dit vruchten af. Uit Jorrit zijn masterproef, in het schooljaar 2016-2017, ontstond het eerste prototype. Vanaf dan besloten ze samen IntuEdrive op te richten in December 2017.

De visie van IntuEdrive is het veranderen van mobiliteit door elektrische tweewielers aantrekkelijk te maken. Daarom is de voornaamste missie speed e-bikes produceren die veiliger, gebruiksvriendelijker en betrouwbaarder zijn dan de traditionele mechanische fiets. IntuEdrive wilt dit bereiken door oudere componenten te vernieuwen, aangezien deze lang geen innovatie gezien hebben.

Momenteel telt IntuEdrive 4 werknemers. CoSaR zal in productie komen in het najaar van 2019 en mikt om in 2020 duizend CoSaRs te bouwen. Naast de speed e-bike, zoekt IntuEdrive ook een 25 km/h versie van CoSaR uit te brengen.

Referenties

(<https://www.milieucentraal.nl/duurzaam-vervoer/fiets-ov-of-auto/>: consumptie/km)

(tabel van het voertuigenpark in belgië 2014-2018)

[https://statbel.fgov.be/nl/themas/mobiliteit/verkeer/voertuigenpark#figures](https://statbel.fgov.be/nl/themas/mobiliteit/verkeer/voertuigenpark" \l "figures)

(evolutie van de file zwaarte in Vlaanderen 2012-2018. (Filezwaarte = file lengte \* duur))

https://www.statistiekvlaanderen.be/filezwaarte