Technische Universität Dresden

Fakultät Verkehrswissenschaften „Friedrich List“

Institut für Wirtschaft und Verkehr

Studienarbeit

Implementierung eines Verbrauchsmodells in einem Java-basierten Verkehrssimulator

eingereicht von Fabian Selzer

geboren am 16.11.1984 in Dresden

Betreuer: Dr. Martin Treiber

Dresden, den 31.08.2012

BIBLIOGRAPHISCHER NACHWEIS

Fabian Selzer

Studienarbeit, Technische Universität Dresden

Fakultät Verkehrswissenschaften, Institut für Wirtschaft und Verkehr

Dresden, 2012

48 Seiten, 13 Abbildungen, 2 Tabellen, 24 Quellen

AUTORENREFERAT

Die vorliegende Arbeit beschreibt die Erstellung eines Modells zur Ermittlung des Energieverbrauchs von elektrisch angetriebenen Fahrzeugen und die Umsetzung in dem mikroskopischen Verkehrssimulator MovSim.

Zunächst wird ein Physik-basiertes Modell zur Berechnung der benötigten Motorleistung vorgestellt und an Hand der Grundlagen elektrischer Maschinen die permanenterregte Drehstromsynchronmaschine für das Verbrauchsmodell ausgewählt, da sie den höchsten Wirkungsgrad bei kleinem Bauvolumen bietet und daher am weitesten verbreitet ist. Auf Basis der physikalischen und mathematischen Zusammenhänge in der permanenterregten Drehstromsynchronmaschine wird ein Rechenweg dargelegt, um den Wirkungsgrad der Maschine bei gegebener Fahrzeugsituation zu maximieren, indem mit dem Phasenversatz zwischen Spannung und Stromstärke die Blindleistung der Maschine minimiert wird. Die Implementation dieses Rechenwegs auf Basis der Wirkungsweise des Simulators wird beschrieben.

Schließlich werden konkrete Simulationen ausgewählt und entwickelt, mit denen das Modell getestet wird. Deren Ergebnisse, welche nah an den in der Realität auftretenden Werten liegen, werden ausgewertet und diskutiert.

Thesen zur Studienarbeit

1. Das entwickelte Modell stellt die physikalischen Zusammenhänge der permanenterregten Synchronmaschine stark vereinfacht dar. Dadurch ist es jedoch gelungen, die Anzahl der einzugebenden Werte klein zu halten.
2. Die erstellten Simulationen bieten eine gute Grundlage zur Weiterentwicklung. Es ist zu untersuchen, wie sich noch nicht betrachtete Einflussgrößen wie z.B. die Anzahl der Spuren oder die Höhe der Verkehrsdichte bzw. der Verkehrsstärke auf den Verbrauch auswirken.
3. Mit dem Optimierungsziel des maximalen Leistungsfaktors ist es gelungen, praxisnahe durchschnittliche Verbrauchswerte zu ermitteln.
4. Die Ausgabe der Daten am Ende der Simulation beschränkt die Informationen auf kumulierte Werte über alle Fahrzeuge. Für eine konkrete Diskussion der Auswirkungen des Fahrverhaltens einzelner Verkehrsteilnehmer auf ihren individuellen Verbrauch müssen die Werte einzelner Fahrzeuge betrachtet werden.
5. Da das Batteriemanagement nicht betrachtet wurde, lassen sich mit dem hier entwickelten Modell keine belastbaren Aussagen über die Reichweite von Elektroautos treffen. Der spätere Einbau in das Modell ist aber möglich.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Verkehr wird in einer globalisierten Welt immer wichtiger. Das Verständnis darüber, welche mathematischen Zusammenhänge ihm zu Grunde liegen, ist daher von wachsendem Wert.

Das Programm MovSim (**M**ulti-model **o**pen-source **v**ehicular-traffic **Sim**ulator) ist ein mikroskopischer Verkehrssimulator, der zu Untersuchung und Vergleich verschiedenster Verkehrsmodelle geeignet ist. Er bietet zudem die Möglichkeit, die Simulation mittels einer graphischen Oberfläche während ihres Ablaufs zu betrachten. Dadurch können mathematische Modelle direkt auf ihre Praxisnähe und ihre Auswirkungen überprüft werden[1].

Abbildung 1: Logo von MovSim

1.2 Problemstellung

Mit MovSim liegt ein leistungsfähiger Verkehrssimulator vor, dem allerdings noch ein Verbrauchsmodul fehlt. Es soll daher im Rahmen dieser Arbeit ein Modell entwickelt, implementiert und getestet werden, das auf Basis der vorhandenen Fahrzeugparameter und des aktuellen Straßen- und Fahrzeugzustandes den instantanen sowie den Gesamtverbrauch einzelner sowie aller Fahrzeuge ermittelt und ausgibt.

Da bei Auswahl und Bestätigung des Themas dieser Arbeit weder dem Autor noch dem Betreuer bekannt war, dass bereits ein Kraftstoffverbrauchsmodell in MovSim implementiert worden war, liegt in Absprache mit dem Betreuer das Hauptaugenmerk nun stattdessen auf einem Verbrauchsmodell für einen Elektromotor.

1.3 Organisation dieser Arbeit

Zunächst werden die Grundlagen der Verbrauchsmodelle und der elektrischen Maschinen betrachtet. Es folgen die für das Modell benötigten mathematischen und physikalischen Zusammenhänge der permanenterregten Synchronmaschine als momentan am weitesten verbreitete Traktionsart. Dann wird die Umsetzung der Problemstellung auf Basis von Aufbau und Wirkungsweise von MovSim beschrieben und schließlich werden die Implementation in MovSim und die Simulationen zum Testen des Modells sowie deren Resultate vorgestellt, bevor die Ergebnisse zusammengefasst und diskutiert werden.

2 Theoretische Grundlagen

2.1 Verbrauchsmodell

Grundlage der Berechnung des Verbrauchs ist das Verbrauchsmodell. Prinzipiell werden dabei mikroskopische und makroskopische Ansätze unterschieden [2]. Makroskopische Modelle betrachten die Gesamtheit der Fahrzeuge und liefern mittels empirisch ermittelter Durchschnittswerte Verbrauchs- und Emissionswerte für ein komplettes Streckennetz oder global für eine ganze Region, während mikroskopische Modelle jedes Fahrzeug für sich betrachten und dessen Verbrauch auf Basis des Beschleunigungs-, Brems- und Gangwahlverhaltens sowie des Zustands des Motors und äußerer Einflüsse ermitteln. Auf Grund der Wirkungsweise von MovSim, jedes Fahrzeug einzeln zu behandeln, liegt ein mikroskopischer Ansatz nahe. In den folgenden Unterkapiteln wird das physik-basierte Modell von TREIBER und KESTING vorgestellt, welches bereits für die Ermittlung des Kraftstoffverbrauchs in MovSim genutzt wird [1][2].

2.1.1 Fahrwiderstände

Die zu überwindenden Fahrwiderstände, welche die Hauptursache des Kraftstoffverbrauchs darstellen, ergeben die zur Aufrechterhaltung der aktuell gewählten Geschwindigkeit und Beschleunigung nötige Kraft:

$$F(v, \dot{v}) = m\dot{v} + (\mu + \beta)mg + \frac{1}{2} c_{\mathrm{w}} \rho A v^2. \tag{1}$$

Diese Formel beinhaltet die *Trägheitskraft* $m\dot{v}$, den *Rollreibungswiderstand* μmg, die *Hangabtriebskraft* $mg \sin\beta \approx mg\beta$ sowie den *Luftwiderstand* $\frac{1}{2} c_{\mathrm{w}} \rho A v^2$. Die Trägheitskraft und die Hangabtriebskraft nehmen bei Verzögerung bzw. bei Gefälle negative Werte an, was bei der Leistungsbetrachtung von Interesse ist.

2.1.2 Motorleistung

Die Motorleistung setzt sich aus der mechanischen Leistung $P_{\mathrm{dyn}} = F\,v$ und einem Mindestleistungsbedarf P_0 zusammen, welcher den Leistungsbedarf von Licht, Radio, Klimaanlage, Nebenaggregaten und Stellmotoren (für Fensterheber, Scheibenwischer etc.) beinhaltet, aber auch elektrische Verluste sowie die Reibungsverluste im Leerlaufbetrieb. In modernen Fahrzeugen ist die gesamte zur Berechnung des Kraftstoffverbrauchs benötigte Leistung demnach:

$$P(v, \dot{v}) = \max[P_0 + vF(v, \dot{v}), 0], \tag{2}$$

da im sogenannten Schleppbetrieb die Kraftstoffzufuhr abgeschaltet wird und P_0 auch von negativen Beiträgen von Fv bestritten werden kann. Hybrid- und Elektrofahrzeuge benötigen die Maximalbedingung aus Gl. (2) hingegen nicht, da sie die kinetische Energie beim Bremsen durch Rekuperation zurückgewinnen können. Für sie gilt:

$$P(v, \dot{v}) = P_0 + vF(v, \dot{v}). \tag{3}$$

Negative Beiträge von Fv können dann zum Aufladen der Energiespeicher genutzt werden.

2.1.3 Verbrauchsrate

Für Kraftstoffmotoren und Elektromotoren wird der Verbrauch nicht in derselben Einheit angegeben, da bei Ersterem meist nicht die tatsächlich aufgewandte Energie von Interesse ist, sondern der Verbrauch an Kraftstoff in Litern. Dadurch ergeben sich unterschiedliche momentane Verbrauchsraten. Bei Kraftstoffmotoren ergibt sich die momentane Verbrauchsrate $\dot{C}$ (Liter pro Zeiteinheit) als eine Funktion der Leistung, der Motordrehzahl f, des Wirkungsgrades γ und der volumenbezogenen kalorischen Energiedichte ω_{cal} als:

$$\dot{C} = \frac{\mathrm{d}C}{\mathrm{d}t} = \frac{P}{\gamma(P, f)\omega_{\mathrm{cal}}}, \tag{4}$$

wobei der Wirkungsgrad als Funktion der Drehzahl und der Leistung, des Drehmoments oder des effektiven Mitteldrucks aus einem sogenannten Motorkennfeld wie in Abbildung 2 abgelesen werden kann.

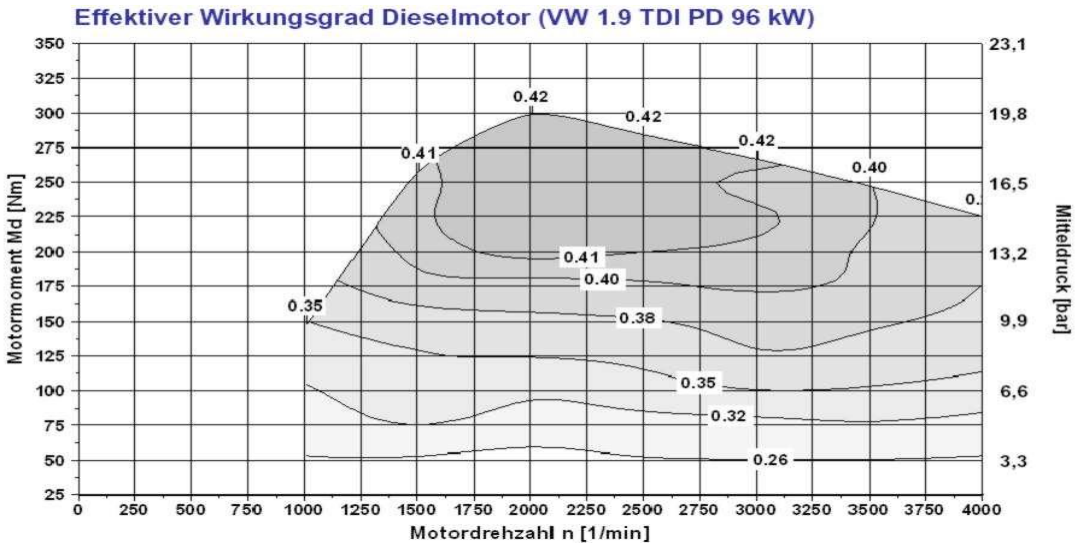


Abbildung 2: Motorkennfeld eines VW Passat Diesel

Bei Elektromotoren ist die momentane Verbrauchsrate (Kilowattstunden pro Zeiteinheit) hingegen gleich der momentanen elektrischen Leistung:

$$P_{\mathrm{el}} = U\, I, \tag{5}$$

da der Verbrauch an elektrischer Energie sich mittels $E_{\mathrm{el}} = P_{\mathrm{el}} t$ errechnet [3].

2.2 Elektrischer Antrieb

Elektrische Maschinen basieren auf der Umwandlung von elektrischer in mechanische Energie (motorischer Betrieb), wobei immer auch die umgekehrte Richtung möglich ist (generatorischer Betrieb). Stromdurchflossene Spulen im Ständer oder Stator induzieren eine Spannung im Läufer oder Rotor, wodurch dort wiederum ein Strom fließt. Die stromdurchflossenen Spulen des Rotors erfahren im Magnetfeld des Stators eine Kraft, die zur Drehung des Läufers führt.

Grundsätzlich wählt man hohe Spannungen, um bei geforderter Leistung niedrige Stromstärken und damit kleine Drahtquerschnitte zu ermöglichen. Durch hohe Frequenzen und ein Festgetriebe lässt sich außerdem bei gleicher Leistung das Drehmoment hochsetzen [4].

2.2.1 Arten elektrischer Antriebe

Prinzipiell unterscheidet man zwischen Gleichstrommaschinen und Wechselstrommaschinen, welche zumeist als Drehstromasynchronmaschinen und Drehstromsynchronmaschinen ausgeführt sind. Die außerdem vorkommenden Sonderbauformen spielen in der Elektrotraktion keine besondere Rolle [5].

Die *Gleichstrommaschine* benötigt zur Erzeugung einer Drehbewegung des Läufers einen mechanischen Stromwandlers (Kommutator), was sie länger als vergleichbare Drehstrommaschinen macht und wodurch sowohl ein hoher Verschleiß an den Kohlebürsten auftritt als auch die maximale Drehzahl auf 4000 bis 5000 Umdrehungen pro Minute begrenzt ist [4]. Aus diesen Gründen werden Gleichstrommotoren heutzutage kaum noch für Elektroautos im Straßenverkehr genutzt.

Die Drehstrommaschinen sind dagegen robust und wartungsarm. Bei der *Drehstromasynchronmaschine* erzeugt das Drehfeld im Stator ein Drehfeld im Rotor, welches sich immer mit einer etwas kleineren Frequenz dreht als das Statorfeld. Diese lastabhängige Differenz nennt sich Schlupf und ist funktionsbedingt, denn bei gleicher Drehzahl bliebe die Induktion aus und das Rotorfeld bräche zusammen. Bei der *Drehstromsynchronmaschine* drehen sich Stator- und Rotorfeld dagegen immer mit der gleichen Drehzahl, da der Läufer eine Gleichfelderregung besitzt [6].

Sowohl die Drehstromasynchron- als auch die Drehstromsynchronmaschine benötigen somit einen Teil der zugeführten elektrischen Leistung für die Erzeugung des Erregerfeldes im Rotor, was den Wirkungsgrad begrenzt. Eine Synchronmaschine, bei der sich im Läufer Permanentmagneten befinden, hat diesen Nachteil hingegen nicht; außerdem können mit Seltenerdmetallen hoher Energiedichte sehr kleine Bauvolumina realisiert werden. Diese sogenannten permanenterregten Synchronmaschinen haben sich als optimale Antriebskonzeptvariante für Elektro- und Hybridfahrzeuge herausgestellt und finden heutzutage in fast allen realisierten Fahrzeugen Anwendung [7]. Aus diesem Grund wird auch in dieser Arbeit das weitere Vorgehen auf dieser Traktionsart fußen.

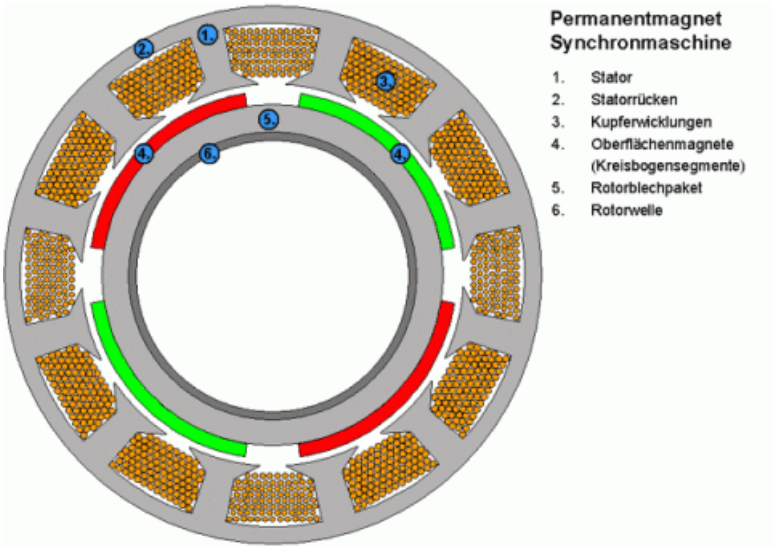


Abbildung 3: Aufbau einer permanterregten Synchronmaschine

Die physikalischen Zusammenhänge in der permanenterregten Synchronmaschine, wie von HOFER dargelegt, werden in den folgenden Unterkapiteln vorgestellt [4].

2.2.2 Ersatzschaltbild der permanenterregten Synchronmaschine

Da in allen drei Strängen des Stators dieselben Prozesse ablaufen, genügt ein einphasiges Ersatzschaltbild. Es enthält nur einen ohmsch-induktiven Widerstand und eine Spannungsquelle:

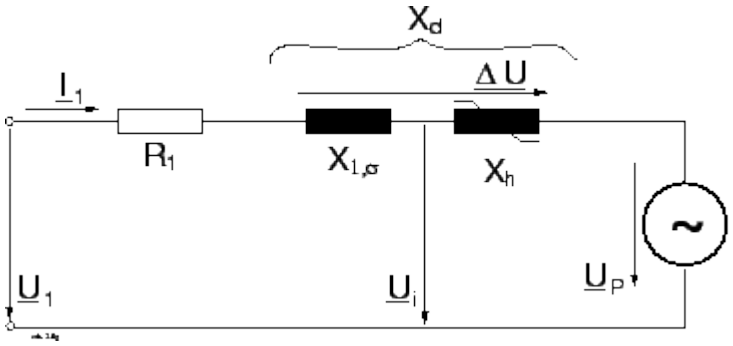


Abbildung 4: Ersatzschaltbild der permanenterregten Synchronmaschine

Die Ständerspannung $\underline{U}_1$ ist die Summe der Spannungsfälle am ohmschen Widerstand des Ständers R_1, der Ständerstreureaktanz $X_{1\sigma}$ und der Ständerhauptreaktanz X_h sowie der induzierten Polradspannung $\underline{U}_\mathrm{p}$:

$$\underline{U}_1 = R_1\underline{I}_1 + \mathrm{j}X_{1\sigma}\underline{I}_1 + \mathrm{j}X_\mathrm{h}\underline{I}_1 + \underline{U}_\mathrm{p}. \tag{6}$$

Die Polradspannung ist abhängig von der Drehfrequenz der Ständerspannung und dem Erregerfeld des Läufers, welches in permanenterregten Synchronmaschinen als konstant angenommen werden kann [8].

Zur weiteren Vereinfachung können die Streu- und die Hauptreaktanz zur sogenannten Synchronreaktanz X_d wie in Abbildung 4 zusammengefasst werden:

$$X_\mathrm{d} = X_{1\sigma} + X_\mathrm{h}, \tag{7}$$

womit sich Gl. (6) noch etwas verkürzt:

$$\underline{U}_1 = R_1\underline{I}_1 + \mathrm{j}X_\mathrm{d}\underline{I}_1 + \underline{U}_\mathrm{p}. \tag{8}$$

2.2.3 Stromortskurve der permanenterregten Synchronmaschine

Auf Grund der relativen großen Reaktanzen kann der Spannungsfall am ohmschen Widerstand des Ständers vernachlässigt werden ($R_1 I_1 = 0$). Stellt man Gl. (8) mit dieser Näherung nun nach dem Ständerstrom um, so erhält man:

$$\underline{I}_1 = -\mathrm{j}\frac{\underline{U}_1}{X_\mathrm{d}} + j\frac{\underline{U}_\mathrm{p}}{X_\mathrm{d}}, \tag{9}$$

Durch Fixierung der Ständerspannung auf der reellen Achse ($\underline{U}_1 = U_1$) ergibt sich der Ständerstrom als Funktion der Ständerspannung, der Polradspannung und des Polradwinkels ϑ als ein Kreis mit dem Mittelpunkt $-\mathrm{j}U_1/X_\mathrm{d}$ und dem Radius $U_\mathrm{p}/X_\mathrm{d}$, der sogenannten Stromortskurve:

$$\underline{I}_1 = -\mathrm{j}\frac{U_1}{X_\mathrm{d}} + j\frac{U_\mathrm{p}}{X_\mathrm{d}} e^{-\mathrm{j}\vartheta}. \tag{10}$$

Abbildung 5 zeigt die Stromortskurve während des motorischen Betriebs, da der Ständerstrom positiv ist:

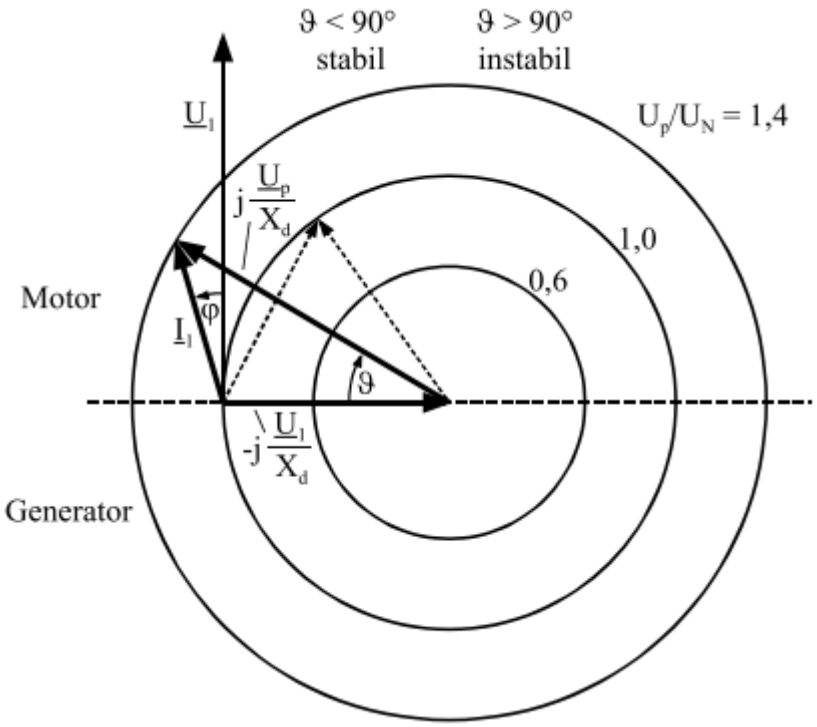


Abbildung 5: Stromortskurve der permanenterregten Synchronmaschine

Der Polradwinkel gibt an, wie stark die Polradspannung und damit das Rotorfeld der Ständerspannung und damit dem Ständerfeld im Motorbetrieb hinterherhängen bzw. im Generatorbetrieb vorauseilen. Er hängt von der mechanischen Leistung ab, die an der Welle anliegt: Erhöht sich bei gleicher Frequenz die Last, wandert der Stromzeiger auf dem Kreis nach oben und der Polradwinkel wird größer. Außerdem ändert sich auch der Phasenwinkel φ zwischen der Ständerspannung und dem Ständerstrom: Ist φ gerade 0, wird nur Wirkleistung übertragen, ansonsten gibt es einen Blindleistungsanteil.

2.2.4 Leistung und der permanenterregten Synchronmaschine

In Abbildung 5 zeigt sich die stationäre Abhängigkeit des Drehmoments vom Polradwinkel über den Wirkanteil des Ständerstroms. Bei Vernachlässigung der Ständerverluste ($R_1 = 0$) ist der Wirkungsgrad der Maschine 1, d.h. es gilt die allgemeine Leistungsformel der Drehstrommaschinen:

$$P = \sqrt{3}\, U\, I \cos\varphi \tag{11}$$

Da die verkettete Ständerspannung noch um den Faktor $\sqrt{3}$ größer ist als die Strangspannung, gilt somit für die abgegebene Wirkleistung:

$$P = 3U_1 I_1 \cos\varphi = 3U_1 I_{1W} = 3\frac{U_1 U_\mathrm{p}}{X_\mathrm{d}} \sin\vartheta \tag{12}$$

Bei einem Polradwinkel von 90° gibt die Synchronmaschine ihre maximale Leistung und ihr maximales Drehmoment ab, danach „kippt“ sie und bleibt stehen, daher ist diese Grenze das sogenannte Kippmoment.

3 Umsetzung

3.1 Vorgehensweise

Die Umsetzung erfolgt durch die Implementation des Modells in MovSim. Da das Verbrauchsmodell für Kraftstoff bereits implementiert ist, kann in Analogie zum dort verwendeten Verfahren vorgegangen werden.

Aus Gl. (5) wird die benötigte mechanische Leistung ermittelt und mit der abgegebenen mechanischen Leistung aus Gl. (12) gleichgesetzt. Das Ziel ist nun die Minimierung des Phasenwinkels φ, damit keine Blindleistung anfällt und die elektrische Leistung nach Gl. (5) ebenfalls minimal wird. Die einzige Stellgröße dafür ist die Ständerspannung, da die Polradspannung wegen der festen Erregung durch die Frequenz vorgegeben ist [8]. Dabei sind die Betriebsgrenzen des Motors einzuhalten.

Falls der Anteil der mechanischen Leistung an Gl. (3) negativ ist, soll ein Teil dieser Leistung mittels Rekuperation zurückgewonnen werden. In der Praxis ist dafür ein komplexes Batteriemanagement nötig, welches diese Möglichkeit stark begrenzt [9]; da aber das Interesse der hiesigen Betrachtung nicht zuvorderst in der Reichweite, sondern im Verbrauch liegt und auch das bereits implementierte Modell des Kraftstoffverbrauchs keine Tankkapazitäten oder –füllstände betrachtet, wird hier darauf verzichtet und ein sehr einfaches Modell gewählt.

In der Praxis beinhaltet die Betrieb einer permanenterregten Synchronmaschine noch die sogenannte feldorientierte oder Vektorregelung, bei der die Strangspannungen jeweils so gewählt werden, dass der Stromzeiger immer nur eine das Drehmoment bildende und keine die Magnetisierung bildende Komponente hat [4]. Weil dies aber nur das Verhältnis der Spannungen der drei Phasen des Drehstroms untereinander, nicht aber die Gesamtspannung betrifft, wurde für dieses Modell die Annahme getroffen, dass diese Regelung bereits vorhanden ist.

3.2 Aufbau und Wirkungsweise von MovSim

Zur Umsetzung ist das Verständnis des Aufbaus und der Wirkungsweise von MovSim nötig. Der Ablauf einer Simulation geschieht dort in drei Phasen:

Beim *Initialisieren* wird eine xml-Datei mit den Parametern der Simulation eingelesen, danach folgt der eigentliche *Ablauf* in kleinen Zeitschritten, wonach jeweils der Zustand aller Simulationsobjekte aktualisiert wird, und schließlich erfolgt am Ende der Simulation (evtl. auch schon davor) die *Ausgabe* der Daten. Auf diese drei Phasen wird nun näher eingegangen.

3.2.1 Initialisieren der Simulation

Beim Aufruf des Programms muss eine xml-Datei übergeben werden, die dem Format entspricht, welches in der Datei `multiModelTrafficSimulatorInput.dtd` definiert ist. Die übergebene Datei enthält die Parameter der Simulation, die momentan aus drei Blöcken bestehen, welchen drei Objekte entsprechen: `SimulationInput` enthält das Straßennetz (welches seinerseits alle Lichtsignalanlagen, Tempolimits, Verkehrsquellen etc. enthält), die Flottenzusammensetzung und den Output (z.B. von Floating-Car-Data), `VehiclesInput` enthält die Definitionen der Fahrzeugklassen und ihres Fahrverhaltens und `FuelConsumptionInput` enthält die Motor- und Fahrzeugkennwerte zur Berechnung des Kraftstoffverbrauchs. Die Daten aus diesen drei Objekten werden schließlich genutzt, um die Objekte zu instanziieren (`Vehicle`, `FuelConsumption`, `TrafficLight`, `TrafficSink` etc.), mit denen die Simulation letztlich arbeitet.

3.2.2 Ablauf der Simulation

Der Ablauf einer Simulation erfolgt nach dem Top-Down-Prinzip. Das oberste Objekt ist ein `Simulator`, von dem es nur eine Instanz gibt. Solange die vorgegebene Dauer der Simulation noch nicht erreicht ist und noch Objekte zu bearbeiten sind, wird immer ein kurzer Zeitschritt (ein sogenannter Timestep) weitergezählt und dann alle untergeordneten Objekte angestoßen, ihrerseits nun diesen Timestep durchzuführen; für `Simulator` sind das einmal `RoadNetwork` und einmal `SimulationOutput`. `RoadNetwork` stößt `RoadSegment` an, darunter folgt `LaneSegment` und schließlich `Vehicle`, welches dann auf Basis seines aktuellen Zustands (also entweder den eingelesenen Anfangsdaten oder dem Ergebnis des letzten Timesteps) seine neue Werte für Position, Geschwindigkeit, Beschleunigung, Spurwechselentscheidungen etc. berechnet. Auf diese Weise erhalten nach jedem Timestep alle Objekte der Simulation einmal neue Werte.

3.2.3 Ausgabe

`SimulationOutput` gibt schon während der Simulation Daten aus, die Zusammenfassung der ermittelten Werte erfolgt jedoch erst am Ende des Ablaufs in einem Dialogfenster:

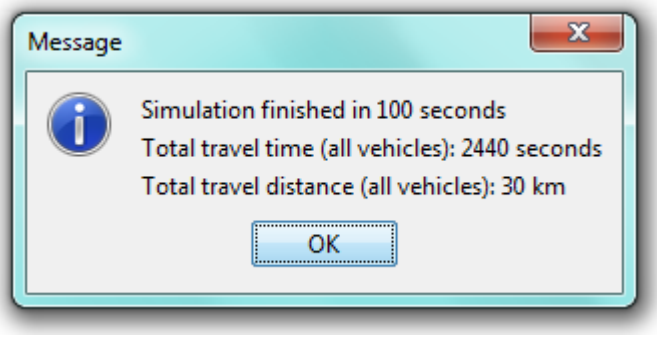


Abbildung 6: Dialogfenster am Ende einer Simulation

Dazu wird wiederum die Hierarchie der Objekte durchlaufen: `RoadNetwork` summiert die gefahrenen Strecken und Verbräuche aller seiner Objekte vom Typ `RoadSegment`, diese wiederum die aller ihrer Objekte vom Typ `LaneSegment` und diese schließlich die aller ihrer Objekte vom Typ `Vehicle`.

3.3 Implementierung

Auf Basis der zusammengetragenen Erkenntnisse erfolgte die Implementierung des Verbrauchsmodells für Elektrofahrzeuge, indem für jede der drei Phasen des Programms die notwendigen Änderungen bzw. Erweiterungen ermittelt und durchgeführt wurden.

All die hier erwähnten Implementierungen fanden in der Programmiersprache Java statt oder wurden mittels der Auszeichnungssprache XML umgesetzt.

3.3.1 Initialisieren der Simulation

Um Dopplungen mit dem bereits implementierten Kraftstoffverbrauchsmodell zu vermeiden und eine Unterscheidung zur Laufzeit zu ermöglichen, wurde ein neues Feld „type“ in `FuelConsumptionInput` und `FuelConsumption` eingeführt, welches momentan die Werte „fuel“ für Kraftstoffverbrauch oder „electric“ für elektrischen Energieverbrauch annehmen kann. Auf die Schaffung neuer Klassen oder einer eigenen Klassenstruktur wurde verzichtet;

es wird vielmehr angeregt, eine Superklasse für den Verbrauch zu schaffen und alle spezifischen Antriebsformen (Kraftstoffmotor, Elektromotor, Hybridantrieb etc.) davon abzuleiten.

Des Weiteren wurden die eingelesenen Daten erweitert. Zunächst wurde das oben erwähnte Attribut „type“ hinzugefügt. Das Kraftstoffverbrauchsmodell bringt außerdem bereits die Werte für die Fahrzeugmasse m, die Fahrzeugquerschnittsfläche A, den cw-Wert c_w, die Mindestleistung P_0, den Reibungskoeffizienten μ und den Reifenradius r_dyn mit, die im Objekt `CarModel` gebündelt werden und zur Berechnung der geforderten Motorleistung benötigt werden. Die mittlere Dichte der Luft ρ und die Fallbeschleunigung g sind ebenfalls bereits vorhanden. Für das hier entwickelte Modell sind noch die Nennleistung P_N, das Nennmoment M_N und die Nennspannung U_N nötig, welche den sogenannten Typenpunkt der Synchronmaschine definieren; weiterhin der maximale Ständerstrom I_max, die maximale Drehzahl n_max, die maximal zulässige Zeit für die Überlast t_max und der Überlastfaktor s, welche die Betriebsgrenzen definieren; schließlich das Leerlauf-Kurzschlussverhältnis k_K und die feste Übersetzung ü zur Ermittlung der Synchronreaktanz und der Motordrehzahl.

Aus Gl. (12) und Abbildung 5 ist ersichtlich, dass die Synchronreaktanz X_d und die Polradspannung U_p zur Ermittlung der optimalen Ständerspannung für einen gegebenen Betriebspunkt benötigt werden. Da die Reaktanz nach $X = \omega L$ von der Frequenz des Ständerstroms abhängt, muss vor dem Ablauf der Simulation noch die Induktivität L der Ständerwicklungen ermittelt werden. In der Praxis geschieht dies durch dreisträngiges Kurzschließen des Motors aus dem Leerlauf heraus, wodurch sich nach kurzer Zeit der Kurzschlussstrom I_K0 einstellt, der allein durch die Ständerimpedanzen begrenzt wird [10]:

$$I_\mathrm{K0} = \frac{U_\mathrm{1N}}{X_\mathrm{d}}. \tag{13}$$

Das Leerlauf-Kurzschlussverhältnis gibt die Stärke des Leerlaufstroms zum Bemessungs- oder Nennstrom an:

$$k_\mathrm{K} = \frac{I_\mathrm{K0}}{I_\mathrm{1N}} \tag{14}$$

und ist ein Maß für die zulässige Überbelastbarkeit der Maschine; in der Praxis liegen die Werte zwischen 0,4 und 0,65 [10]. Da die Strangspannung um den Verkettungsfaktor $\sqrt{3}$ kleiner als die Nennspannung ist:

$$U_{1\mathrm{N}} = \frac{1}{\sqrt{3}} U_{\mathrm{N}} \tag{15}$$

und die Kreisfrequenz $\omega = 2\pi n$ bekannt ist durch:

$$M = \frac{P}{\omega}, \tag{16}$$

kann unter der Annahme, dass im Typenpunkt der Leistungsfaktor $\cos\varphi = 1$ erreicht wird, aus Gl. (12) der Nennstrom und damit aus Gl. (13-16) die Induktivität der Ständerwicklungen berechnet werden [11]:

$$L = \frac{U_{1\mathrm{N}}}{\omega_{\mathrm{N}} I_{1\mathrm{N}} k_{\mathrm{K}}}. \tag{17}$$

Die Polradspannung berechnet sich allgemein aus:

$$U_{\mathrm{p}} = k_0 I_{\mathrm{f}} \omega = k\omega, \tag{18}$$

wobei bei Permanenterregung I_{f} als konstant angenommen [8] und mit der Konstante k_0 zur Konstanten k zusammengefasst werden kann. Nach Abbildung 5 und der erneuten Annahme, dass im Typenpunkt der Leistungsfaktor $\cos\varphi = 1$ erreicht wird, gilt:

$$U_{\mathrm{P}}{}^2 = U_1{}^2 + (X_{\mathrm{d}} I_1)^2 \tag{19}$$

und somit für die Konstante k:

$$k = \frac{\sqrt{U_{1\mathrm{N}}{}^2 + (X_{\mathrm{d}} I_{1\mathrm{N}})^2}}{\omega_{\mathrm{N}}}. \tag{20}$$

Diese Berechnungen erfolgen im Konstruktor der Klasse `FuelConsumption` und somit nur einmal zu Beginn der Simulation. An derselben Stelle wird die Variable t_{over} mit dem Wert 0 initialisiert, die im Verlauf der Simulation die Überlastzeit enthält.

Die genannten neu eingeführten Parameter entsprechen denjenigen, die für das Kraftstoffmodell bereits implementiert und im Objekt `EngineModel` gebündelt sind (maximale Motorleistung, Getriebeübersetzungen, maximaler Motorinnendruck etc.). Da aber keine eigene Klasse für einen Elektromotor geschrieben wurde, wurde auch auf das Anpassen dieses Objekts verzichtet. Dadurch können die Parameter der Synchronmaschine momentan nicht auf dem regulären Weg des Einlesens aus der übergebenen xml-Datei verändert werden, sondern nur im Quelltext der Klasse `FuelConsumption`, welche im Kraftstoffmodell den Verbrauch berechnet und nun auch zur Berechnung des elektrischen Energieverbrauchs genutzt wird.

3.3.2 Ablauf der Simulation

Nach jedem Timestep wird im implementierten Kraftstoffverbrauchsmodell für jedes Fahrzeug der aktuelle Verbrauch berechnet und zum bisherigen addiert. Analog dazu wurde in der Klasse `Vehicle` ein neues Feld „totalElectricEnergyConsumption“ eingeführt, welches den Gesamtverbrauch des Fahrzeugs an elektrischer Energie enthält, und an derselben Stelle die Berechnung der elektrischen Leistung angesetzt. Dabei ist eine Fallunterscheidung anhand der mechanischen Leistung P_{mech} nötig, die sich aus Gl. (3) ohne den Mindestleistungsbedarf ergibt, da Elektroautos meist keine Lichtmaschine besitzen, sondern P_0 direkt aus dem Energiespeicher decken [12].

Falls $P_{\mathrm{mech}} > 0$ zutrifft, muss eine Ständerspannung angelegt werden werden, die einen Betrieb innerhalb der Betriebsgrenzen ermöglicht. Nach Gl. (12) darf der Polradwinkel einen sicheren Bereich nicht nach oben verlassen und wurde daher begrenzt mit:

$$\sin\vartheta \leq 0{,}9, \tag{21}$$

was einem Maximalwert von rund 64° entspricht. Dadurch ergibt sich eine Mindestspannung von:

$$U_{1_{\min}} = \frac{P_{\mathrm{mech}} L}{2{,}7k}. \tag{22}$$

Die Maximalspannung $U_{1_{max}}$ entspricht der Nennspannung.

Die Drehzahl n ergibt sich aus der Geschwindigkeit und den Werten des Antriebsstranges:

$$n = \ddot{\mathrm{u}} \frac{v}{2\pi r_{\mathrm{dyn}}}; \tag{23}$$

auch ihr Maximalwert darf nicht überschritten werden.

Die Maximalwerte für das Drehmoment M, die mechanische Leistung und den Ständerstrom I_1 können kurzzeitig um den Überlastfaktor s überschritten werden [7], dann muss aber t_{over} hochgezählt werden; ist t_{max} erreicht, müssen die Maximalwerte strikt eingehalten werden, da sonst eine Überhitzung des Motors droht.

Da das Ziel die Maximierung des Wirkungsfaktors $\cos\varphi$ ist, wird zunächst versucht, mit $\cos\varphi = 1$ zu fahren. Nach Gl. (19) ergibt sich eine Gleichung vierter Ordnung:

$$0 = {U_1}^4 - {U_{\mathrm{P}}}^2 {U_1}^2 + \left(\frac{X_{\mathrm{d}} P_{\mathrm{mech}}}{3}\right)^2, \tag{24}$$

die aber durch die Substition $x = {U_1}^2$ in eine gewöhnliche quadratische Gleichung überführt werden kann:

$$0 = x^2 - {U_{\mathrm{p}}}^2 x + \left(\frac{X_{\mathrm{d}} P_{\mathrm{mech}}}{3}\right)^2. \tag{25}$$

Nach der allgemeinen Lösungsformel für quadratische Gleichungen ergeben sich zwei Lösungen:

$$x_{1/2} = \frac{{U_{\mathrm{p}}}^2}{2} \pm \sqrt{\frac{{U_{\mathrm{p}}}^4}{2} - \left(\frac{X_{\mathrm{d}} P_{\mathrm{mech}}}{3}\right)^2}, \tag{26}$$

die nur dann reelle Werte annehmen, wenn für die Leistung gilt:

$$P_{\mathrm{mech}} \leq \frac{3}{2} \frac{{U_{\mathrm{P}}}^2}{X_{\mathrm{d}}} \tag{27}$$

Wenn dies nicht der Fall ist, ist $\cos\varphi = 1$ nicht erreichbar; dann muss die Spannung gefunden werden, für die $\cos\varphi$ maximal wird. Dazu wird die Hilfsgröße a eingeführt, sodass nach Abbildung 5 gilt:

$$\tan\varphi = \frac{a}{I_1 \cos\varphi}, \tag{28}$$

wobei für a gilt:

$$a = \sqrt{\left(\frac{U_\mathrm{P}}{X_\mathrm{d}}\right)^2 - (I_1 \cos\varphi)^2} - \frac{U_1}{X_\mathrm{d}} \tag{29}$$

Durch Verbindung mit Gl. (12) erhält man somit den Leistungsfaktor nur noch als Funktion bekannter Größen sowie der variablen Ständerspannung:

$$\cos\varphi = \arctan\left[\frac{3U_1}{P_\mathrm{mech}}\left(\sqrt{\left(\frac{U_\mathrm{P}}{X_\mathrm{d}}\right)^2 - \left(\frac{P_\mathrm{mech}}{3U_1}\right)^2} - \frac{U_1}{X_\mathrm{d}}\right)\right] \tag{30}$$

Da eine analytische Lösung dieses Optimierungsproblems unrealistisch erschien, wurde ein iterativer Weg gewählt: mittels eines Suchlaufs in ausreichend kleinen Schritten von der maximal bis zur minimal möglichen Spannung wird der beste Leistungsfaktor gesucht, für den die Betriebsgrenzen erfüllt sind.

Falls hingegen $\cos\varphi = 1$ erreichbar ist, weil Gl. (27) erfüllt ist, erhält man durch Rücksubstitution mit $U_1 = \sqrt{x}$ zwei Spannungen U_{1_1} und U_{1_2}, für die $U_{1_1} \geq U_{1_2}$ gilt. Da größere Spannungen geringere Stromstärken und damit besseres Leistungsverhalten bewirken, wurde folgende Logik zur Bestimmung der optimalen Spannung gewählt:

Ist U_{1_1} im erlaubten Bereich, dann setze $U_1 = U_{1_1}$ und berechne I_1 nach Gl. (12).

Wenn nicht: Ist U_{1_2} im erlaubten Bereich, dann setze $U_1 = U_{1_2}$ und berechne I_1 nach Gl. (12).

Wenn nicht: Sind beide Spannungen zu hoch, dann setze $U_1 = U_{1\mathrm{max}}$ und berechne I_1 nach Gl. (30) und Gl. (12).

Wenn nicht: Sind beide Spannungen zu niedrig, dann setze $U_1 = U_{1_{\min}}$ *und berechne* I_1 *nach Gl. (30) und Gl. (12).*

Wenn nicht: Suche zwischen $U_{1_{\max}}$ *und* $U_{1_{\min}}$ *das* U_1*, für das innerhalb der Betriebsgrenzen* $\cos\varphi$ *maximal wird.*

Auf den letzten Punkt muss auch dann zurückgegriffen werden, wenn der zuvor ermittelte Wert für I_1 jenseits des erlaubten Maximums liegt.

Die elektrische Leistung, die vom Energiespeicher aufgebracht werden muss, um die mechanische Leistung an der Maschine abzugeben, errechnet sich nach Gl. (3), (5) und (12) schließlich zu:

$$P_{\text{el}} = 3U_1 I_1 + P_0 \tag{31}$$

Falls $P_{\text{mech}} < 0$ zutrifft, kann ein Teil der Leistung wieder zum Aufladen der Energiespeicher genutzt werden. Wie bereits erwähnt, wurde ein sehr einfaches Modell gewählt, da das Fehlen der Betrachtung der Batteriekapazität ein realistisches Batteriemanagement ausschließt. Stattdessen wurde ein fixer Wirkungsgrad η eingeführt, sodass gilt:

$$P_{\text{el}} = \eta_{\text{Rek}} P_{\text{mech}} + P_0, \tag{32}$$

und dieser Wert mit 0,7 festgelegt [13][14].

Für den Verbrauch an elektrischer Energie muss die elektrische Leistung nun noch mit der Dauer t eines Timesteps multipliziert werden. Dieses Ergebnis kann auch negativ sein, sodass der kumulierte Verbrauch des Fahrzeugs dann sinkt.

3.3.3 Ausgabe

Schon während der Simulation ist es möglich, über das sogenannte Logging Nachrichten auszugeben. Da dieses zur Fehlersuche im Programm und für Informationen über den Fortschritt des Programmablaufs gedacht ist, wurde es jedoch nicht für die Ausgabe am Ende der Simulation genutzt, wenn dem User die Informationen über den Verbrauch zur Verfügung gestellt werden. Dazu wurde das oben genannte Dialogfenster um den Gesamtverbrauch aller

Fahrzeuge erweitert; dies erforderte das Hinzufügen einer Summierungshierarchie für den elektrischen Verbrauch analog zum Kraftstoffverbrauch (s. Kapitel 3.2.3). Da auch die insgesamt zurückgelegte Strecke aller Fahrzeuge eingeblendet ist, wurde auf die zusätzliche Angabe des durchschnittlichen Verbrauchs auf 100 Kilometer verzichtet.

Die Implementation der Highscore-Tabelle für die Simulationen mit Spielcharakter erforderte das Speichern der relevanten Daten in persistenten Dateien und nicht in Variablen in MovSim, damit sie beim Beenden des Programmes nicht verloren gehen. Dazu wurde das Dateiformat `.txt` gewählt, da es eine einfache Bearbeitung erlaubt und keine komplexen Daten verarbeitet werden müssen, sondern nur die Ergebnisse der Simulation (Zahlenwerte) und die Namen der Spieler (alphanumerisch). Als Kriterium für die Platzierung war die Simulationsdauer vorgegeben, außerdem sollten nur die Spieler zur Eingabe ihres Namens aufgefordert werden, die eine noch zu wählende Mindestplatzierung erreicht haben.

Gespeichert wird die Highscore-Tabelle momentan nun in einer Datei `[Name der Simulation]_highscore.txt`, damit die Unterscheidung der einzelnen Simulationen möglich ist. Die einzelnen Ergebnisse liegen in Zeilen des Formates „Simulationsdauer;Verbrauch;Benutzername“, wobei der dritte Wert fehlen kann. Nach Beendigung einer Simulation wird der Rang des Spielers ermittelt, evtl. sein Name abgefragt und immer die Highscore-Tabelle angezeigt, sodass auch Spieler mit schlechteren Ergebnissen ihre Zeit einordnen können. Außerdem wurde dem oben genannten Dialogfenster noch eine Nachricht über die relative Platzierung des Spielers hinzugefügt.

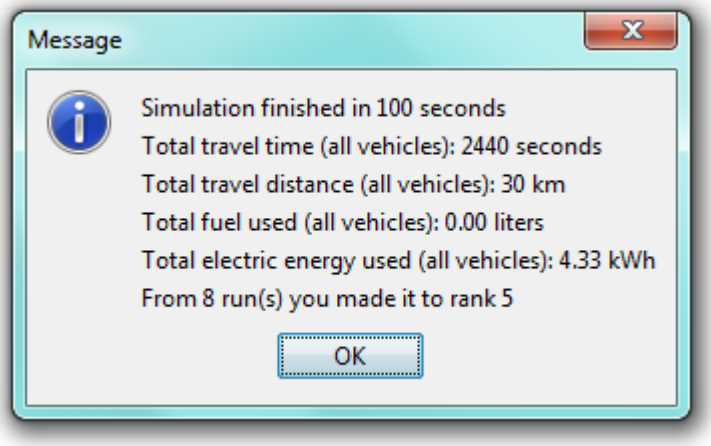
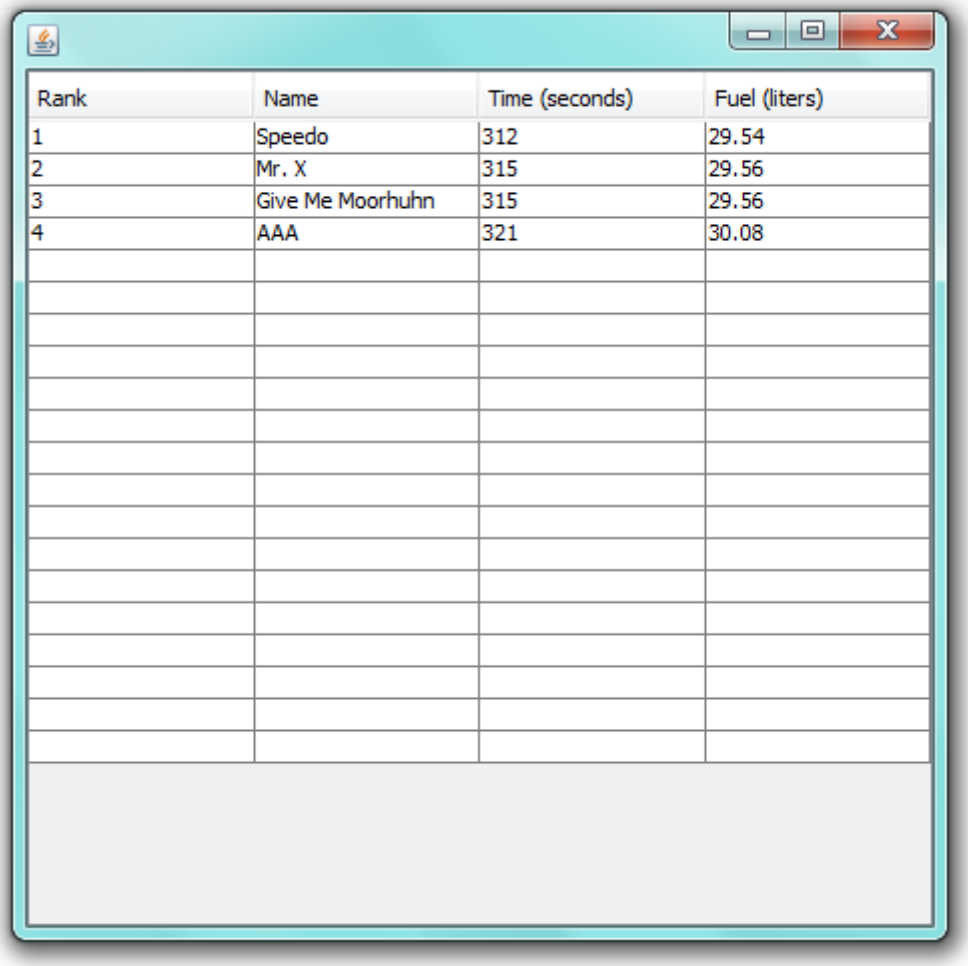


Abbildung 7: erweiterter Dialog am Ende einer Simulation

Abbildung 8 zeigt eine beispielhafte Highscore-Tabelle für eine Simulation mit Kraftstoffverbrauch.



Rank	Name	Time (seconds)	Fuel (liters)
1	Speedo	312	29.54
2	Mr. X	315	29.56
3	Give Me Moorhuhn	315	29.56
4	AAA	321	30.08

Abbildung 8: Highscore-Tabelle mit Kraftstoffverbrauch

4 Tests

4.1 Zielsetzung

Das Testen des hier vorgestellten Modells erfolgte an Hand schon vorhandener und neu erstellter Simulationen. Dabei lag das Hauptaugenmerk auf zwei Fragen: Wie realistisch sind die Verbrauchswerte und entsprechen die Auswirkungen auf gezielte Änderungen einzelner Parameter den Erwartungen?

Als Grundlage dienten die typischen Werte für die Berechnung des Treibstoffverbrauchs nach TREIBER und KESTING [2]. Als Referenz für ein Fahrzeug mit Elektroantrieb wurde ein Toyota Prius der 3. Generation gewählt, der zwar ein Hybridfahrzeug ist, aber auch komplett elektrisch fahren kann [7]. In Tabelle 1 sind die Ausgangswerte für die Simulationen zusammengefasst.

Abbildung 9: Toyota Prius

Größe	Symbol	Wert
Mindestleistung	P_0	3 kW
Fahrzeugmasse	m	1500 kg
Reibungskoeffizient	μ	0,015
cw-Wert	c_w	0,32
Querschnittsfläche	A	2,2 m²
Dynamischer Reifenradius	r_dyn	0,31 m
Dichte der Luft	ρ	1,29 kg/m³
Fallbeschleunigung	g	9,81 m/s²
Nennleistung	P_N	60 kW
Nennmoment	M_N	207 Nm
Nennspannung	U_N	650 V
maximaler Strom	I_max	170 A
maximale Drehzahl	n_max	13500 U/min
maximale Überlastzeit	t_max	10 s
Überlastfaktor	s	2
Leerlauf-Kurzschlussverhältnis	k_K	0,6
feste Übersetzung	ü	2,636

Tabelle 1: Ausgangswerte der Simulationen

4.2 vorhandene Simulationen

Die beiden vorhandenen Simulationen mit Spielcharakter – „Ramp Metering“ und „Routing“ – wurden zum Testen ausgewählt, da der Spieler die Möglichkeit hat, den Fahrweg einzelner Fahrzeuge zu beeinflussen.

4.2.1 Ramp Metering

„Ramp Metering“ simuliert den Einfluss einer Zuflussregelungsanlage. Der Spieler kann entscheiden, ob die Lichtsignalanlage Grün oder Rot anzeigt und damit den Zufluss auf die Schnellstraße regeln. Zu lange Grünphasen bremsen den Verkehrsfluss auf der Schnellstraße, während zu lange Rotphasen einen Rückstau auf die Nebenstraße bewirken.

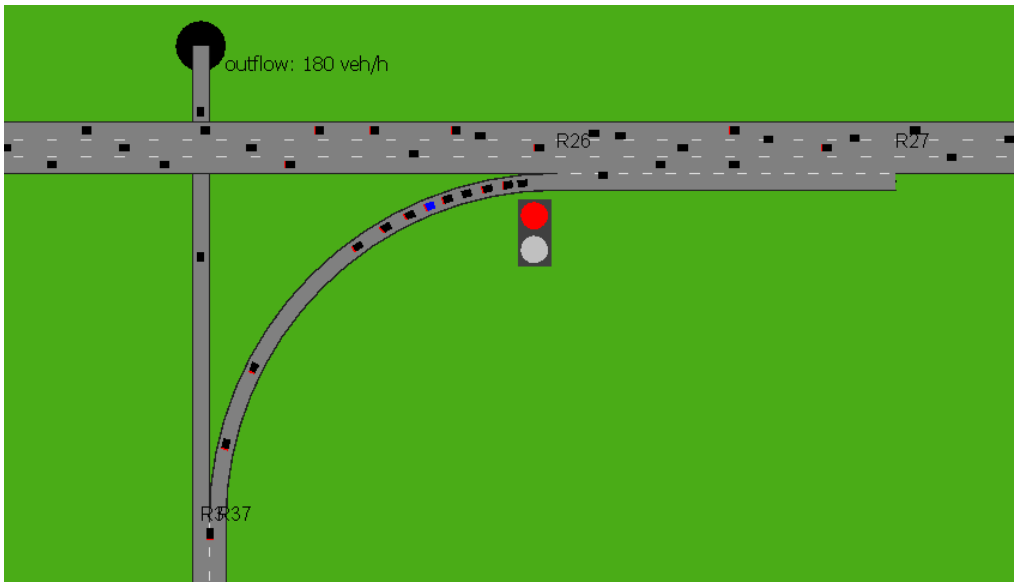


Abbildung 10: Simulation "Ramp Metering"

4.2.2 Routing

„Routing“ simuliert den Einfluss eines Wechselverkehrszeichens. Der Spieler kann entscheiden, ob alle Fahrzeuge die Hauptstraße nutzen, welche eine Engstelle besitzt, oder ob einige Fahrzeuge die Engstelle auf einer Nebenstraße umfahren, wo sie allerdings auf eine Geschwindigkeitsbegrenzung stoßen. Zu viel Verkehr auf der Hauptstraße führt zum Stau an der Engstelle, währen zu viel Verkehr auf der Nebenstraße einen Rückstau auf die Hauptstraße zur Folge hat.

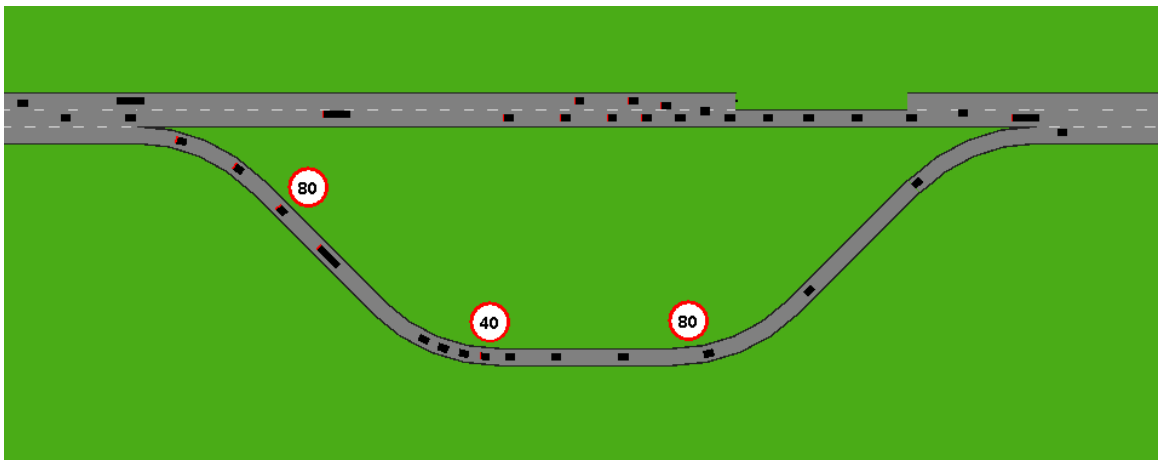


Abbildung 11: Simulation "Routing"

4.3 neue Simulationen

Die Basis für die neuen Simulationen boten ebenfalls TREIBER und KESTING [2]. Sie vergleichen die theoretischen Kraftstoffverbräuche in Zonen mit einer Höchstgeschwindigkeit von 30 km/h und 50 km/h sowie die von 130 km/h und 150 km/h und kommen zu dem Ergebnis, dass der Verbrauch in Tempo-30-Zonen höher ist als in Tempo-50-Zonen und dass der Verbrauch bei Tempo 130 geringer ist als bei Tempo 150. Des Weiteren ermitteln sie den Einfluss von Stoppschildern an Kreuzungen und kommen zu dem Ergebnis, dass der Verbrauch mit dem Anhalten und Beschleunigen deutlich über dem des ungehinderten Durchfahrens liegt.

Diese Betrachtungen für Kraftstoffverbräuche wurden nun in Simulationen für elektrische Energieverbräuche überführt. Dafür war das Format OpenDRIVE® zu nutzen, um die Geometrie des gewünschten Straßennetzes sowie seinen logischen Aufbau zu beschreiben [15][1].

4.3.1 Tempo 30 / 50 / 130 / 150

MovSim variiert die Höchstgeschwindigkeit der Fahrzeuge leicht, um dem in der Praxis auftretenden unterschiedlichen Fahrverhalten der Verkehrsteilnehmer Rechnung zu tragen. Daher wurde eine gerade, zweispurige Straße gewählt, um Überholvorgänge zu ermöglichen. Die Länge spielt nur eine untergeordnete Rolle und wurde willkürlich auf 5000 Meter festgesetzt. Direkt am Anfang der Strecke wurde eine Geschwindigkeitsbegrenzung auf den jeweiligen Wert platziert, womit sich insgesamt vier Simulationen ergaben.

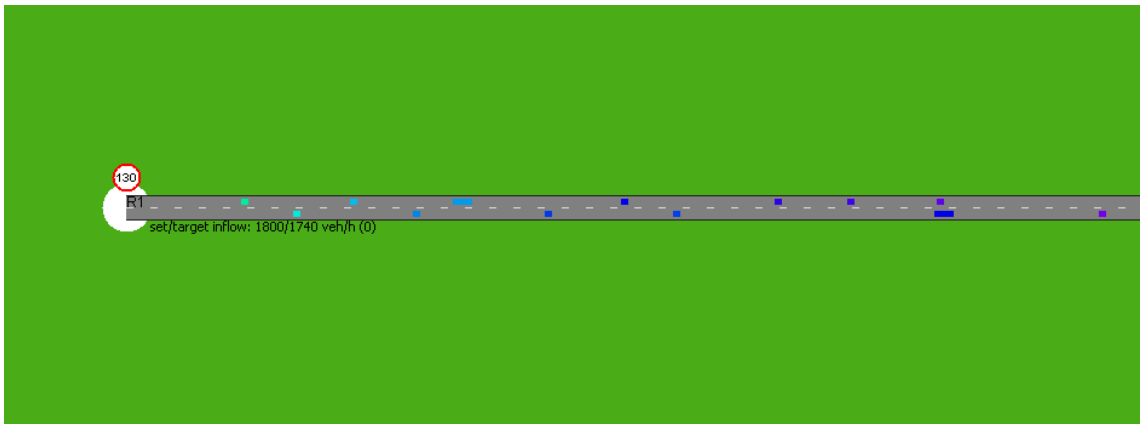


Abbildung 12: Simulation "Tempo 130"

4.3.2 Rote Welle / Grüne Welle

Die Darstellung eines zwingenden Halts bzw. einer freien Durchfahrt an einer Kreuzung wurde durch Lichtsignalanlagen erreicht. Dafür wurde auf einer ebenfalls geraden, zweispurigen Strecke nach 300 Metern eine Lichtsignalanlage platziert, um eine Pulkbildung zu erreichen. Nach weiteren 500 Metern wurde eine zweite Lichtsignalanlage platziert, welche dem ankommenden Pulk entweder gerade Rot oder Grün zeigt. Die Länge der Strecke wurde auf 1000 Meter begrenzt, um den Einfluss anderer Faktoren auf den Verbrauch zu minimieren.

Mit einer Grünphase von 30 Sekunden, einer Gelbphase von 3 Sekunden, einer Rotphase von 40 Sekunden und einer Rotgelbphase von 1 Sekunde ergibt sich eine Umlaufzeit von 74 Sekunden. Bei einer Richtgeschwindigkeit von 50 km/h brauchen die Fahrzeuge 36 Sekunden für die 500 Meter zwischen den zwei Lichtsignalanlagen zuzüglich der Zeit fürs Beschleunigen. Bei zeitlichem Parallellauf der beiden Lichtsignalanlagen trifft der Pulk also gerade zum Beginn einer Rotphase ein, während bei einem Zeitversatz der zweiten Anlage um 40 Sekunden nach vorn gerade der Beginn einer Grünphase vorliegt. Diese Werte wurden schließlich für die zwei entstandenen Simulationen genutzt.

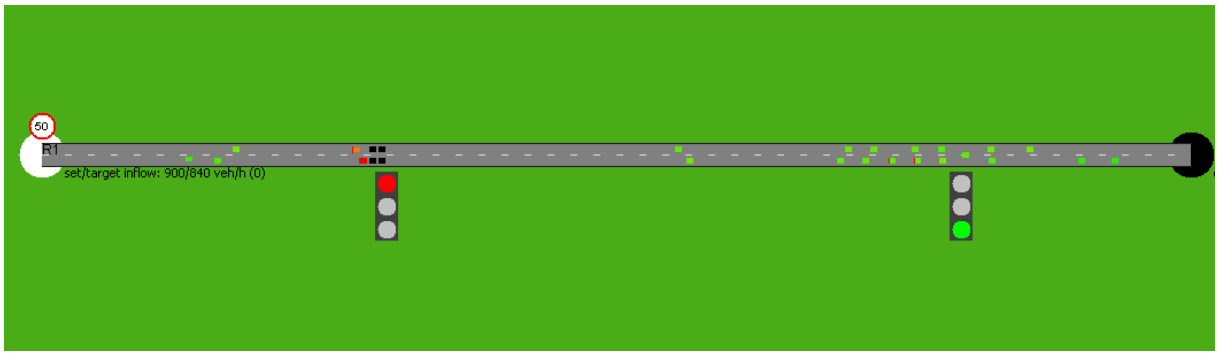


Abbildung 13: Simulation "Grüne Welle"

4.4 Ergebnisse

Mehrmaliges Durchführen einer Simulation mit denselben Eingangswerten führte jedes Mal zum selben Ergebnis, daher wurden die Simulationen ohne Interaktion nur einmal durchgeführt. Für die Simulationen mit Spielcharakter wurden jeweils 10 Testläufe

durchgeführt. Tabelle 2 zeigt die Ergebnisse der Simulationen unter Nutzung der Werte aus Tabelle 1.

Die durchschnittlichen Verbräuche auf 100 Kilometer liegen also bei den Simulationen mit Geschwindigkeiten bis 50 km/h (wie sie im Stadtverkehr üblich sind) unter 20 kWh, allein bei der Roten Welle knapp darüber, woran der negative Einfluss von Stoppschildern auf den Verbrauch ablesbar ist. Auch der höhere Verbrauch bei Tempo 30 im Vergleich zu Tempo 50 und der niedrigere Verbrauch von Tempo 130 im Vergleich zu Tempo 150 lassen sich deutlich erkennen.

Simulation	Simulationsdauer [s]	Gesamtstrecke aller Fahrzeuge [km]	Gesamtverbrauch aller Fahrzeuge [kWh]	Durchschnittlicher Gesamtverbrauch [kWh/100km]
Ramp Metering	260	340	77,97	22,9
	267	340	86,89	25,6
	273	340	80,30	23,6
	280	339	88,50	26,1
	281	339	85,18	25,1
	289	340	80,45	23,7
	294	340	82,29	24,2
	294	339	80,33	23,7
	299	339	82,12	24,2
	307	339	81,49	24,0
Routing	274	137	34,75	25,4
	286	137	34,84	25,4
	289	137	34,84	25,4
	291	137	35,51	25,9
	296	137	34,02	24,8
	314	137	34,12	24,9
	327	137	36,72	26,8
	346	137	37,26	27,2
	426	140	35,79	25,6
	457	134	44,35	33,1
Tempo 30	100	18	3,16	17,6
	500	453	82,41	18,2
	1000	1595	295,60	18,5
Tempo 50	100	30	4,33	14,4
	500	717	106,41	14,8
	1000	1942	290,99	15,0
Tempo 130	100	77	24,50	31,8
	500	1036	273,41	26,4
	1000	2268	591,68	26,1
Tempo 150	100	83	30,65	36,9
	500	1052	321,63	30,6
	1000	2286	689,35	30,2
Rote Welle	300	52	11,51	22,1
Grüne Welle	300	54	10,21	18,9

Tabelle 2: Ergebnisse der Simulationen

5 Zusammenfassung und Ausblick

5.1 Fazit

Das im Rahmen dieser Arbeit entwickelte Modell zur Verbrauchsberechnung von Elektroantrieben basiert auf dem Antriebskonzept der permanenterregten Synchronmaschine als am weitesten verbreitete Traktionsart. Nach der Wahl eines Modells zur Ermittlung der benötigten Leistung und der Erstellung eines Regelungskonzeptes für die Maschine wurde es analog zum bereits vorliegenden Kraftstoffverbrauchsmodell in MovSim implementiert und an schon vorhandenen sowie neu entwickelten und erstellten Simulationen getestet und die Ergebnisse mit realen Verbrauchswerten verglichen.

5.2 Diskussion

Mit der Maximierung des Leistungsfaktors als primärem Optimierungsziel ist es gelungen, Verbrauchswerte zu ermitteln, die im Mittel nah an den im wahren Leben auftretenden Zahlen liegen. Ein durchschnittlicher Verbrauch von 20 kWh auf 100 bis 150 km – „eine Stunde Fahrt bei einer mäßigen Leistung von 20 [kW]“ [5] – ist realistisch. Die Datenblätter des Toyota Prius geben jedoch keine Auskunft über den Verbrauch bei rein elektrischer Fahrt [16], weshalb zur Überprüfung auf andere Elektroautos zurückgegriffen werden muss. Der Renault Kangoo Z.E. verbraucht im Stadtverkehr rund 22 kWh auf 100 km [17], der Renault Fluence Z.E. 25,7 kWh auf 100 km und der Volvo C30 electric 28,3 kWh auf 100 km [18]. Der Tesla Roadster, ein vollständig elektrisch angetriebener Sportwagen, braucht bei „strammer Fahrt“ 26,5 kWh auf 100 km [19], sodass die hohen Verbräuche bei Tempo 130 und 150 durchaus im Bereich des Möglichen liegen.

Die sehr simple Methode der Rekuperationsberechnung ist noch verbesserungswürdig, was aber auf Grund des fehlenden Batteriemanagementsystems nicht geschehen ist. Außerdem sind noch viele Testmöglichkeiten offen geblieben, z.B. durch Variation der Fahrzeugmassen, des Verhaltens der Fahrer, anderer Maschinenwerte etc. Diese Tests werden zeigen, ob das Modell auch dann noch praxisnahe Zahlen liefert.

Die Durchdringung des Aufbaus und der Wirkungsweise von MovSim kann als Erfolg gewertet werden; die gewählten Eingriffspunkte im Quelltext haben die erwünschten

Resultate gebracht, ohne ihn zu sehr zu entstellen. Nichtsdestotrotz bleibt als Mängel neben dem Fehlen einer eigenen Klassenstruktur das Vorgeben der Maschinenwerte im Quelltext statt auf dem regulären Weg mittels der einzulesenden xml-Datei, was jedoch allein aus Zeitgründen unterblieben musste, da der prinzipielle Mechanismus verstanden und mit dem Feld „type“ in den Klassen `FuelConsumptionInput` und `FuelConsumption` bereits erfolgreich angewandt wurde.

Die Vernachlässigung des Ständerwiderstandes, der Eisen-, Kupfer- und Hystereseverluste sowie des Einflusses der Sättigung des Rotors und der Reibung des Antriebsstranges geschah mit Blick auf die sonst drohende Komplexität des Modells. Bei ersten Tests fiel jedoch auf, dass auch sehr kleine Iterationsschritte ($< 0{,}25$) während des Suchlaufs nach dem optimalen Leistungsfaktor die Simulation durch die schiere Masse an Rechenoperationen arg verlangsamten. Daher ist bei allen weiteren Entwicklungen im Rahmen von MovSim immer auf den benötigten Rechenaufwand zu achten.

Die Objekte der Klasse `Vehicle` werden während des Ablaufs einer Simulation von Objekten der Klasse `TrafficSink` aufgefangen, wenn sie vor Erreichen der Simulationsdauer bereits das Straßennetz verlassen haben. Da nur zu diesem Zeitpunkt der Gesamt- und der durchschnittliche Verbrauch bezogen auf die Fahrstrecke der einzelnen Fahrzeuge bekannt ist, bevor diese Werte in der Kumulation aller Fahrzeuge aufgehen, ist an dieser Stelle im Programm noch die Ausgabe der Werte vorzunehmen.

5.3 Ausblick

Ein offensichtlicher Ansatzpunkt für die weitere Beschäftigung mit dem Thema sind die CO_2-Bilanzen der unterschiedlichen Antriebsarten. Da die Energiespeicher die große Schwäche der Elektrofahrzeuge waren und sind [20][21], ist außerdem zu untersuchen, inwiefern Verbesserungen dort den Verbrauch von Elektromobilen noch mindern können.

Schlussendlich liegt das weite Feld der Hybridfahrzeuge voraus. Da nun sowohl ein Kraftstoff- als auch ein Elektroenergieverbrauchsmodell in MovSim implementiert sind, wäre der nächste Schritt die Verbindung beider Modelle, wobei die vorhandenen Hybridtechnologien eine Vielfalt von Möglichkeiten bieten.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

bzw.	beziehungsweise
d. h.	das heißt
etc.	et cetera
MovSim	Multi-model open-source vehicular-traffic Simulator
s.	siehe
u. a.	unter anderem
z. B.	zum Beispiel

Symbolverzeichnis

1	Index für Stranggrößen
A	Querschnittsfläche
β	Steigung
c_{w}	Luftwiderstandskoeffizient („cw-Wert“)
C	Kraftstoffverbrauch
γ	spezifischer Wirkungsgrad des Verbrennungsmotors
E	Energie
η_{Rek}	Wirkungsgrad der Rekuperation
f	Drehzahl, Frequenz
F	Kraft
φ	Phasenverschiebungswinkel
g	Fallbeschleunigung
I	Stromstärke
I_{f}	Erregerstrom
I_{K0}	Kurzschlussstrom
j	imaginäre Einheit
k	Polradspannungskonstante
k_{K}	Leerlauf-Kurzschlussverhältnis
L	Induktivität
m	Masse
max	Index für zulässige Maximalwerte
min	Index für zulässige Minimalwerte
M	Drehmoment
μ	Reibungskoeffizient

N	Index für Nenngrößen
n	Frequenz, Drehzahl
ω	Kreisfrequenz
ω_{cal}	volumenbezogene kalorische Energiedichte
P	Leistung
P_0	Mindestleistung
P_{dyn}	dynamische Motorleistung
P_{el}	elektrische Leistung
P_{mech}	mechanische Leistung
R	ohmscher Widerstand
ρ	Dichte der Luft
s	Überlastfaktor
t	Zeit
t_{over}	Dauer der Überlast
ϑ	Polradwinkel
U	Spannung
U_{p}	Polradspannung
ü	feste Getriebeübersetzung
v	Geschwindigkeit
W	Index für Wirkgrößen
X	Reaktanz
$X_{1\sigma}$	Ständerstreureaktanz
X_{h}	Hauptreaktanz
X_{d}	Synchronreaktanz

Literaturverzeichnis

[1] www.movsim.org *(MovSim)*, Stand: 29.08.2012

[2] Martin Treiber, Arne Kesting: *Verkehrsdynamik und -simulation*, Berlin Heidelberg 2010

[3] Gerhard Babiel: *Elektrische Antriebe in der Fahrzeugtechnik*, Wiesbaden 2007

[4] Klaus Hofer: *Elektrotraktion*, Berlin 2006

[5] Cornel Stan: *Alternative Antriebe für Automobile*, Berlin Heidelberg 2012

[6] Heinz Gottschalk, Max Lemberg: *Elektrotechnik Elektronik*, Berlin 1972

[7] Peter Hofmann: *Hybridfahrzeuge*, Wien 2010

[8] http://www.betz-simon.homepage.t-online.de/ba/SM.pdf (*Synchronmaschine*), Stand: 24.07.2012

[9] Dietrich Naunin et.al.: *Hybrid-, Batterie- und Brennstoffzellen-Elektrofahrzeuge*, Renningen 2004

[10] http://www.fh-oow.de/fbi/we/el/ema/DOWNLOAD/EMA2/Kapitel3.PDF (*Die Synchronmaschine)*, Stand: 24.07.2012

[11] http://www.eit.uni-kl.de/uploads/media/Folien_Uebung5_01.pdf (*Elektrische Antriebstechnik II, 5. Übung*), Stand: 19.07.2012

[12] Reiner Korthauer: *Handbuch Elektromobilität*, Frankfurt am Main 2011

[13] Bernhard Gerl: *Innovative Automobilantriebe*, Landsberg/Lech 2002

[14] http://www.heinzmann.com/de/elektrische-und-hybridantriebe/download-elektrische-und-hybrid-antriebe/doc_download/1632-pgs-156-technische-daten (*PGS 156 - Permanenterregter Synchrongenerator*), Stand: 24.07.2012

[15] http://www.opendrive.org/links.htm (*OpenDRIVE*), Stand: 21.08.2012

[16] http://www.toyota.de/cars/new_cars/prius/specs.aspx *(Toyoto Deutschland)*, Stand: 21.08.2012

[17] http://www.renault.de/renault-welt/produkt/pressespiegel/pressespiegel-kangoo-ze-auto-bild-1107/ (*Renault Deutschland*), Stand: 21.08.2012

[18] http://www.tagesspiegel.de/wirtschaft/emobility/elektroautos-weniger-gruen-hoher-verbrauch-daempft-die-freude/6335510.html (*Der Tagesspiegel*), Stand: 21.08.2012

[19] http://www.auto-motor-und-sport.de/einzeltests/tesla-roadster-sport-im-test-sportwagen-mit-elektromotor-1853449.html?paging_current=2 (*Auto Motor und Sport*), Stand: 21.08.2012

[20] O.A.Stavrov: *Elektromobile*, Berlin 1988

[21] Konrad Reif: *Konventioneller Antriebsstrang und Hybridantriebe*, Wiesbaden 2010

Jean-Régis Hadji-Minaglou: *Antriebskonzepte mit permanenterregten Synchronmotoren für den Einsatz im Elektrofahrzeug*, Aachen 1994

http://www.iew.uni-stuttgart.de/dateien/gp/Versuch_008_Synchronmaschine.pdf (*Praktikum Erneuerbare Energien, Versuch 3*), Stand: 19.07.2012

http://www.ew.tu-darmstadt.de/media/ew/vorlesungen_4/ema/folie_ema_9.pdf (*Elektrisch und permanentmagnetisch erregte Synchronmaschinen*), Stand: 27.07.2012

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage eingereichte Studienarbeit selbständig verfasst und andere als die angegebenen Hilfsmittel nicht benutzt habe.

Dresden, 31.08.2012

Fabian Selzer

Anhang: Quelltext der Berechnung des Energieverbrauchs

Die vom Autor programmierten Zeilen sind grau unterlegt. Um die Übersichtlichkeit zu bewahren, wurden nur die für die tatsächliche Berechnung des Energieverbrauchs zuständigen Teile des Quelltextes aufgeführt, nicht jedoch die unterstützenden Teile wie z.B. die Summierungshierarchie oder sonstige kleinere Änderungen. Der komplette Code des Simulators ist unter http://www.movsim.org zu finden.

`org.movsim.simulator.vehicles.consumption.FuelConsumption:`

```
/*
 * Copyright (C) 2010, 2011, 2012 by Arne Kesting, Martin Treiber, Ralph Germ, Martin Budden
 *                                   <movsim.org@gmail.com>
 * -----------------------------------------------------------------------------------------
 *
 * This file is part of
 *
 * MovSim - the multi-model open-source vehicular-traffic simulator.
 *
 * MovSim is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * MovSim is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with MovSim. If not, see <http://www.gnu.org/licenses/>
 * or <http://www.movsim.org>.
 *
 * -----------------------------------------------------------------------------------------
 */

[...]

public class FuelConsumption {

[...]

    private final CarModel carModel;

    private final EngineModel engineModel;

    private final String type;

//energy consumption

    // values to be read from xml BEGIN
    private final double fixedGearRatio;
    private final double UN;
    private final double PN;
    private final double MN;
    private final double Imax;
    private final double nmax;
    private final double kK;
    private final double tmax;
    private final double s;
```

```
// values to be read from xml END
    private final double k;
    private final double L;
    private double t;


///energy consumption

    public FuelConsumption(String keyLabel, ConsumptionModelInput input) {
        carModel = new CarModel(input.getCarData());
        engineModel = new EngineModel(input.getEngineData(), carModel);
        type = input.getType();

//energy consumption

        //values to be read from xml BEGIN
        fixedGearRatio = 2.636;
        UN = 650;
        PN = 60 * 1000;
        MN = 207;
        Imax = 170;
        nmax = 13500 / 60;
        kK = 0.6;
        tmax = 10;
        s = 2;
        //values to be read from xml END

        t = 0;

        final double U1N = UN / Math.sqrt(3);
        final double I1N = PN / (UN * Math.sqrt(3));
        final double omegaN = PN / MN;
        L = U1N / (I1N * kK * omegaN);

        final double UpN = Math.sqrt(U1N * U1N + Math.pow(omegaN * L * I1N, 2));
        k = UpN / omegaN;

        logger.info(String.format("L=%f, k=%f", L, k));

///energy consumption

        if (input.isOutput()) {
            writeOutput(keyLabel);
        }
    }

[…]

    public String getType() {
        return type;
    }

//energy consumption

    public double getElectricPower(double v, double a, double dt) {
        double Pel = 0;
        final double P_ERROR = PN;

        double U1I1cosphi[] = {0, 0, 0};

        final double n = fixedGearRatio * v / carModel.getDynamicWheelCircumfence();
        final double omega = 2 * Math.PI * n;
        final double Pmech = v * carModel.getForceMech(v, a); // can be <0
        final double M = EngineModel.getMoment(Pmech, n);
        final double Up = k * omega;
        final double Xd = omega * L;

        final double U1max = UN / Math.sqrt(3);
        final boolean tover = (t >= tmax);
        final double I1max = tover ? Imax : Imax * s;
        final double Pmechmax = tover ? PN : PN * s;
        final double Mmax = tover ? MN : MN * s;

        if (n > nmax)               logger.error(String.format("motor frequency %d/min is too
high", n * 60));
```

```
        if (Pmech > Pmechmax)    logger.error(String.format("power demand %fkW is too high",
Pmechmax / 1000));
        if (M > Mmax)            logger.error(String.format("moment demand %fNm is too high",
M));

        if (Pmech > 0) { // energy must be taken from battery

            final double U1min = Pmech * L / (2.7 * k); // theta<=64°

            if (Pmech <= 1.5 * Up * k / L) { // cos(phi)=1 is possible

                final double D = 0.25 * Math.pow(Up, 4) - Pmech * Pmech * Xd * Xd / 9;
                final double x1 = Up * Up / 2 + Math.sqrt(D);
                final double x2 = Math.max(Up * Up / 2 - Math.sqrt(D), 0);
                final double U11 = Math.sqrt(x1);
                final double U12 = Math.sqrt(x2);
                if (U11 >= U1min) {
                    if (U11 <= U1max) {
                        U1I1cosphi[0] = U11;
                        U1I1cosphi[1] = Pmech / (3 * U11);
                        logger.debug("U1=U11");
                    } else if (U12 >= U1min) {
                        if (U12 <= U1max) {
                            U1I1cosphi[0] = U12;
                            U1I1cosphi[1] = Pmech / (3 * U12);
                            logger.debug("U1=U12");
                        } else {
                            U1I1cosphi[0] = U1max;
                            U1I1cosphi[1] = getI1(U1max, Pmech, omega);
                            logger.debug("U1=U1max");
                        }
                    } else {
                        U1I1cosphi = getU1I1phiBest(U1min, Pmech, omega, I1max);
                        logger.debug("U1=?");
                    }
                } else {
                    U1I1cosphi[0] = U1min;
                    U1I1cosphi[1] = getI1(U1min, Pmech, omega);
                    logger.debug("U1=U1min");
                }

                if (U1I1cosphi[1] > I1max) {
                    U1I1cosphi = getU1I1phiBest(U1min, Pmech, omega, I1max);
                    logger.debug("I1>I1max");
                }

            } else {
                logger.debug(String.format("cos(phi)=1 not possible for %fkW at %fkm/h", Pmech
/ 1000, v * 3.6));

                U1I1cosphi = getU1I1phiBest(U1min, Pmech, omega, I1max);

            }

            if (U1I1cosphi[0] == 0) {
                logger.error(String.format("No possible voltage found between %fV and %fV for
power demand %fkW at motor frequency %f/min (t_over = %fs)", U1min, UN / Math.sqrt(3), Pmech /
1000, n * 60, t));
                return P_ERROR;
            }

            //is engine working beyond design limits?
            if ((U1I1cosphi[1] > Imax) || (Pmech > PN) || (M > MN)) {
                t += dt;
                logger.debug(String.format("engine working beyond design limits - t_over =
%.2f seconds", t));
            } else {
                t = 0;
            }

            Pel = 3 * U1I1cosphi[0] * U1I1cosphi[1];

        } else { // energy can be recouped

            if (v > 1 / 3.6) {
```

```
                Pel = 0.7 * Pmech;

                logger.debug(String.format("recouping  %fkW  at  %.0fkm/h,  decelerating  with
%.2fm/s²", Pel / (-1000), v * 3.6, -a));

                // TODO

            }

        }

        final double Pconst = carModel.getElectricPower();

        return Pel + Pconst; //TODO calculate switching losses (n), Ohm losses (I1)
    }

    private double getI1(double U1, double Pmech, double omega) {
        double x = Math.sqrt((k * k / (L * L)) - Pmech * Pmech / (9 * U1 * U1));
        double phi = Math.atan((x - U1 / (omega * L)) * 3 * U1 / Pmech);
        return Pmech / (3 * U1 * Math.cos(phi));
    }

    private double[] getU1I1phiBest(double U1min, double Pmech, double omega, double I1max) {
        double U1I1cosphi[] = {0, 0, 0};
        for (double U1 = UN / Math.sqrt(3); U1 >= U1min; U1 -= 0.5 ) {
            double I1 = getI1(U1, Pmech, omega);
            double cosphi = Pmech / (3 * U1 * I1);
            if ((cosphi > U1I1cosphi[2]) && (I1 <= I1max)) {
                U1I1cosphi[0] = U1;
                U1I1cosphi[1] = I1;
                U1I1cosphi[2] = cosphi;
            }
        }
        logger.debug(String.format("best U1 / I1 / cos(phi) for %.2fkW at omega=%.2fHz: %.3fV
/ %.3fA / %f", Pmech / 1000, omega, U1I1cosphi[0], U1I1cosphi[1], U1I1cosphi[2]));
        return U1I1cosphi;
    }

///energy consumption

}
```

org.movsim.simulator.vehicles.Vehicle:

```
/*
 * Copyright (C) 2010, 2011, 2012 by Arne Kesting, Martin Treiber, Ralph Germ, Martin Budden
 *                                   <movsim.org@gmail.com>
 * -----------------------------------------------------------------------------------------
 *
 * This file is part of
 *
 * MovSim - the multi-model open-source vehicular-traffic simulator.
 *
 * MovSim is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * MovSim is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with MovSim. If not, see <http://www.gnu.org/licenses/>
 * or <http://www.movsim.org>.
 *
 * -----------------------------------------------------------------------------------------
 */

[...]
```

```
public class Vehicle {

[...]

    private double totalTravelDistance;
    private double totalTravelTime;
    private double totalFuelUsedLiters;
    private double totalElectricEnergyUsed;

[...]

    /**
     * Update position and speed. Case distinction between cellular automata, Newell and
continuos models/iterated maps
     *
     * @param dt
     *            delta-t, simulation time interval, seconds
     */
    public void updatePositionAndSpeed(double dt) {

[...]

        if (fuelModel != null) {
            if (fuelModel.getType().equals("fuel")) {
                totalFuelUsedLiters += fuelModel.getFuelFlowInLiterPerS(speed, acc) *dt;
            }
            if (fuelModel.getType().equals("electric")) {
                totalElectricEnergyUsed += fuelModel.getElectricPower(speed, acc, dt) * dt;
            }
        }
    }

[...]

    /**
     * Returns the total distance this vehicle has traveled.
     * @return total travel distance
     */
    public final double totalTravelDistance() {
        return totalTravelDistance;
    }

    /**
     * Returns the total time this vehicle has been on the road network.
     * @return total travel time
     *
     * @return
     */
    public final double totalTravelTime() {
        return totalTravelTime;
    }

    /**
     * Returns the total fuel used by this vehicle.
     * @return total fuel used
     */
    public final double totalFuelUsedLiters() {
        return totalFuelUsedLiters;
    }

    public final double totalElectricEnergyUsed() {
        return totalElectricEnergyUsed;
    }

[...]

}
```