

**Datenverarbeitung 2 – Projekt
Studienjahr 2011/2012**

**Projektaufgabe #003 –
„Vergleich: Diesel- gegen Elektromotor“**

Abgabedatum: 08.01.2013

Gruppe: #169

Verfasser: Jan Uhlig, jan.uhlig@tu-dortmund.de
Patrick Haße, patrick.hasse@tu-dortmund.de

Inhalt

1	Aufgabe	5
1.1	Gestellte Aufgabe, Zielstellung, Zweck	5
1.2	Analyse der Aufgabe	5
1.3	Randbedingungen	6
1.4	Annahmen zur Vereinfachung	7
2	Daten	8
2.1	Hauptspeicher	8
2.2	Initialisierung	10
2.3	Nebenbedingungen	10
2.4	Hintergrundspeicher	10
3	Dokumentation des Lösungsansatzes	11
3.1	Lösungsverfahren	11
3.2	Quellen	17
4	Elemente der Programmstruktur	18
5	Benutzungsschnittstelle und Bedienungsanleitung	20
5.1	Art der Benutzungsschnittstelle	20
5.2	Bedienungsanleitung	20
5.2.1	Hauptmenü	20
5.2.2	Simulation starten	20
5.2.3	automatischer Vergleich	20
5.2.4	Parameter aendern	20
5.2.5	Diagramm plotten	21
5.2.6	Sonstiges	22
6	Algorithmen	23
6.1	Strukturelemente	23
6.1.1	initialisieren	23
6.1.2	warten	23
6.1.3	titel	23
6.1.4	daten_ueberschreiben	23
6.1.5	daten_schreiben	24
6.1.6	modus_waehlen	24
6.1.7	leistung_waehlen	24
6.1.8	geschwindigkeit_waehlen	24

6.1.9	parameter_listen	25
6.1.10	parameter_aendern	25
6.1.11	plotmenu	25
6.1.12	diagramm_plotten	26
6.1.13	menu	26
6.2	Energieelemente	27
6.2.1	variablen_berechnen	27
6.2.2	ergebnis_ausgeben	27
6.2.3	vergleich	27
6.2.4	beschleunigen_bergauf	28
6.2.5	konstant_bergauf	29
6.2.6	bremsen_bergauf	29
6.2.7	beschleunigen_gerade	30
6.2.8	konstant_gerade	31
6.2.9	beschleunigen_bergab	31
6.2.10	konstant_bergab	33
6.2.11	bremsen	33
6.2.12	Energie_berechnen	34
6.3	Externe Elemente	35
6.3.1	Dieselplot.plt	35
6.3.2	Elektroplot.plt	35
6.3.3	Multiplot.plt	36
6.3.4	Batch-Skripte	36
7	Programm entwerfen und möglichst schon abschnittsweise testen	37
8	Testen: Testdaten entwerfen, Tests durchführen, dokumentieren	39
8.1	Programm	39
8.2	Physik	39
9	Anwendung des Programms	41
10	Literaturverzeichnis	44
11	Anhänge	45
11.1	Anhang: Energieverbrauchsdiagramme	45
11.2	Anhang: Programmcode	49
11.2.1	Antriebsvergleich.lpr	49
11.2.2	umain.pas	49
11.2.3	Dieselplot.plt	69

11.2.4	Elektroplot.plt.....	69
11.2.5	Multiplot.plt.....	70
11.3	Anhang: verkürzte Ausgabedatei Dieseldaten.dat (50 kW und 50 km/h).....	71
11.4	Anhang: Aufgabenstellung Projektaufgabe #003	72

1 Aufgabe

1.1 Gestellte Aufgabe, Zielstellung, Zweck

Die exakte Aufgabenstellung ist dem Anhang 11.4 zu entnehmen.

Ziel des Projektes ist es, den Energieverbrauch von zwei Autos mit unterschiedlichen Antriebskonzepten durch eine simulierte Fahrt über eine definierte Strecke mit vorgegebenem Geschwindigkeits-/Zeitprofil zu ermitteln und zu vergleichen.

Der Zweck dieser Simulation könnte darin bestehen, Entwicklungskosten gering zu halten, indem, statt Versuchswerte, die mit teuren Prototypen ermittelt wurden, zu vergleichen, am Computer simulierte Fahrzeug-/Streckenkonfigurationen zur Ergebnisfindung genutzt werden.

1.2 Analyse der Aufgabe

Das Programm soll grundsätzlich drei Funktionen anbieten:

- Energieverbrauch über die Teststrecke berechnen
- Simulationsparameter ändern
- Ergebnisse als Diagramm darstellen

Zusätzlich soll dem Anwender die Möglichkeit gegeben werden das Programm über eine Eingabe zu Beenden.

Um den Benutzer die abschließende Analyse und Interpretation der Ergebnisse zu vereinfachen, soll die Berechnung der Gesamtenergie in die Simulation eines einzelnen Fahrzeugs mit einem definierten Antriebskonzept und den automatischen Vergleich der beiden Antriebskonzepte mit ansonsten identischen Parametern unterteilt werden. Alle Berechnungen hierbei sollen vollständig im Hintergrund ausgeführt werden. Lediglich die Ergebnisse (Gesamtstrecke, Gesamtzeit, Gesamtenergie bzw. -energiedifferenz) sollen zum Schluss dem Benutzer ausgegeben werden.

Da in der Aufgabenstellung gefordert wird, den Energieverbrauch in Abhängigkeit von drei gegebenen Leistungsklassen sowie einer sinnvoll zu wählenden Geschwindigkeit, die jedoch zusätzlich um ± 20 Prozent variierbar sein soll, zu untersuchen, muss das Programm die Möglichkeit bieten, die Simulationsparameter zu ändern. Des Weiteren ergibt sich aus dem Vergleich der Antriebskonzepte die Notwendigkeit den Modus zwischen Diesel- und Elektroantrieb umzuschalten. Diese Funktionen sollen unter dem Menüpunkt „Parameter ändern“ implementiert und sequenziell abgefragt werden.

Um die Energie über die Strecke als Diagramm darzustellen, müssen die Ergebnisse jedes berechneten Zeitpunktes (Strecke(t) und Gesamtenergie(t)) in einer Datei gespeichert werden, die anschließend von einer externen Anwendung geplottet werden kann. Es ist dabei für den Plot hinreichend genau nur jeden k-ten Wert (k abhängig von gewähltem Δt ; erreicht werden soll ~ 1 Messwert pro Sekunde) in die Datensätze zu schreiben. Beim Schreiben der Dateien soll der Benutzer wählen können, ob eventuell schon vorhandene Dateien überschrieben oder der Schreibvorgang abgebrochen werden soll. Außerdem soll im Anschluss an den Schreibvorgang der Dateiname und Speicherort der soeben erstellten Datei angezeigt werden.

Um nun dem Nutzer die Diagramme auszugeben, soll ein weiterer Menüpunkt eingeführt werden. Es soll dabei wie schon bei der Energieberechnung zwischen der Ausgabe eines Einzeldiagramms und der eines Vergleichsdiagramms zur besseren Veranschaulichung der Unterschiede differenziert werden.

Eine Arbeitsaufteilung erfolgt insofern, als dass die Ausarbeitung der zu benutzenden Formeln und Gleichungen vollständig in der Gruppe geschieht. Die Einteilung der Gruppenmitglieder bezüglich der informationstechnischen Umsetzung des physikalischen Teils in Prozeduren kann der Tabelle unter Punkt 7 (S.37) entnommen werden. Des Weiteren wurde auch die Dokumentation zusammen erstellt, sodass hierbei die Eigenanteile etwa 50% eines jeden Punktes ausmachen.

1.3 Randbedingungen

Als Randbedingungen lassen sich die folgenden Größen direkt identifizieren:

- Masse des Fahrzeugs $m_F = 1500 \text{ kg}$ (Konstante "mf")
- Reibungskoeffizient $r = 0,015$ (Konstante "mue")
- Fahrzeugquerschnittsfläche $A = 2 \text{ m}^2$ (Konstante "A")
- Strömungswiderstandskoeffizient $c_W = 0,33$ (Konstante "Cw")
- Nennleistungsklassen $P_N = 30, 50, 60 \text{ kW}$ (Variable "p")
- Wirkungsgrad Dieselmotor $40 \% \Rightarrow \eta_{\text{Diesel1}} = 0,4$
- Mechanische Verluste Dieselmotor $20 \% \Rightarrow \eta_{\text{Diesel2}} = 0,8$
- Wirkungsgrad Elektromotor $85 \% \Rightarrow \eta_{\text{Elektro}} = 0,85$
- Wirkungsgrad Be-/Entladung Batterie $80 \% \Rightarrow \eta_{\text{Batterie}} = 0,8$

Des Weiteren wirkt die Erdbeschleunigung $g = 9,81 \text{ m/s}^2$ (Konstante "g").

Weiterhin ist das Streckenprofil vorgegeben, welches sich wie folgt zusammensetzt:

3 km ebene Strecke \rightarrow 2 km Steigung \rightarrow 2 km Gefälle \rightarrow 3 km ebene Strecke

Die Steigung bzw. das Gefälle beträgt jeweils $4 \% \Rightarrow$ Steigungswinkel $\alpha = 2,291^\circ$

Jeder einzelne Kilometer dieser Strecke wird außerdem nach folgendem Profil befahren:

300 m beschleunigen \rightarrow 500 m konstante Fahrt \rightarrow 200 m verzögern

Die Meter Angaben (300, 500, 200) stellen dabei die Abbruchbedingungen für die jeweiligen Schleifen dar.

Außerdem ist die Dichte der Luft (bei 20°C auf NN angenommen) $\rho_L = 1,1885^1 \text{ kg/m}^3$ (Konstante "Pl") implizit gegeben

¹ Verein Deutscher Ingenieure (2006): *VDI-Wärmeatlas*. Berlin, Heidelberg: Springer (10. Auflage) DbB 2 (Wert bei 1 bar Umgebungsdruck und 20°C)

Laut Aufgabenstellung soll die Sollgeschwindigkeit v_{SOLL} sinnvoll gewählt werden. Da es sich bei dem gegebenen Streckenprofil aufgrund der verhältnismäßig kurzen Stop-and-Go Abschnitte sehr wahrscheinlich um eine Stadtfahrt handelt, wird v_{SOLL} auf 50 km/h bzw. 13,89 m/s festgesetzt.

Zuletzt ist auch noch das Zeitinkrement Δt als 0,01 s vorgeschlagen. Es wird jedoch noch zu untersuchen sein, wie groß Δt zu wählen ist, um den Anforderungen an sinnvolle Ergebnisse hinreichend gerecht zu werden (siehe Punkt 8.2, S.40, "Test der Genauigkeit").

1.4 Annahmen zur Vereinfachung

In der Aufgabenstellung werden zur Vereinfachung die Schubabschaltung sowie auch der Wirkungsgrad des Getriebes vernachlässigt.

Weiterhin wird der Dieselmotor vereinfachend als ideale Wärmekraftmaschine mit einem Wirkungsgrad von 40 % sowie zusätzlich drehzahlunabhängiger, konstanter mechanischer Verluste in Höhe von $0,2 \cdot P_N$ betrachtet.

Auch die Beschleunigung bzw. Verzögerung auf den definierten Strecken ist als konstant anzunehmen und wird in Abhängigkeit von v_{SOLL} berechnet.

Zusätzlich vereinfacht sich die Energie beim Beschleunigen bergab für den Fall 80%iger Sollgeschwindigkeit ($0,8 \cdot v_{\text{SOLL}} = 40 \text{ km/h}$) sowie bei konstanter Fahrt bergab für jede wählbare Geschwindigkeit, beim Diesel zu $\sum_0^t \text{idle}$ bzw. beim Elektroantrieb zu 0 (Erläuterung siehe Punkt 3.1).

Diese Vereinfachung gilt ebenso für die Berechnung des Energieverbrauchs bei allen Bremsvorgängen mit Ausnahme der Bremsungen bergauf bei einer Geschwindigkeiten von 40 und 50 km/h (Erläuterung siehe Punkt 3.1).

2 Daten

2.1 Hauptspeicher

Zur Lösung der Aufgabe wird eine Reihe von Variablen eingeführt. Die wichtigsten davon können grob in zwei Kategorien unterteilt werden. Variablen, die Werte enthalten, die für Berechnungen benötigt werden (Kategorie I), und Variablen, die Werte enthalten, welche ausgegeben oder in Dateien geschrieben werden sollen bzw. zu diesem Prozess beitragen (Kategorie II). Der Vollständigkeit halber werden zum Schluss unter diesem Punkt auch noch alle eingeführten Konstanten, die nicht zu den Randbedingungen zählen, gelistet und falls nötig kurz erläutert.

Kategorie I:

eta (double) ist das Produkt der Wirkungsgrade $\eta_{\text{Diesel1}} * \eta_{\text{Diesel2}} = 0,32$ bzw. $\eta_{\text{Elektro}} * \eta_{\text{Batterie}} = 0,68$. Diese Variable wird benötigt um die zugeführte Gesamtenergie berechnen zu können.

Beschleunigung und Verzoegerung (double) werden in den Energieschleifen sehr häufig benötigt, um Strecke und Energie zu berechnen. Damit diese Variablen nicht bei jedem Schleifendurchlauf erneut berechnet werden, obwohl sie sich, solange v_{SOLL} konstant bleibt, nicht ändern, werden sie um der Effizienz Willen in einer eigenen Prozedur außerhalb der Energieschleifen berechnet.

Luftwiderstand, Rollreibung, Hangabtriebskraft (double) werden ebenso in den Energieschleifen häufig benötigt und sind innerhalb eines Simulationsdurchlaufes konstant. Bei der Variable Luftwiderstand handelt es sich jedoch lediglich um den konstanten Faktor $0.5 * A * c_w * \rho_L$ ohne die quadratische Geschwindigkeit.

Bei idle (double) handelt es sich um die Energie, die benötigt wird, um den Dieselmotor ohne Last am Laufen zu behalten, also die Leerlaufenergie. Diese stellt das Produkt $p * 0.2 * \text{deltat}$ dar und ist somit auch zeitweilig konstant.

p (double) ist die Nennleistung des Motors in Watt und kann, abhängig von der Wahl des Benutzers, die Werte 30000, 50000 oder 60000 annehmen.

v (double) ist die Sollgeschwindigkeit v_{SOLL} und hat die Einheit m/s. v kann die Werte 11.11, 13.89 oder 16.67 annehmen, die 80, 100 und 120 Prozent von v_{SOLL} entsprechen.

Die Variable vergleichswitch (boolean) wird aufgrund der Option des automatischen Vergleichs des Energieverbrauchs benötigt. Bei einer Einzelsimulation wird vergleichswitch der Wert false zugewiesen und durch eine if-Bedingung am Ende der Prozedur "Energie_berechnen" ins Hauptmenü zurückgekehrt. Steht vergleichswitch auf true, endet "Energie_berechnen" ohne das Menü aufzurufen und die übergeordnete Prozedur läuft weiter.

abort (boolean) sorgt dafür, dass die Prozedur „parameter_aendern“ vorzeitig verlassen werden kann, falls der Benutzer dies wünscht. Falls abort in einer Unterprozedur auf true gesetzt wird, wird diese verlassen und anschließend das Hauptmenü aufgerufen.

Kategorie II:

Gesamtenergie, Gesamtstrecke, Gesamtzeit (double) enthalten die während der Energieschleifendurchläufe inkrementell aufsummierten Werte für Gesamtenergie, Gesamtstrecke und Gesamtzeit. Diese Variablen werden am Anfang der Prozedur "Energie_berechnen" auf 0 initialisiert und zwischen den einzelnen Energieprozeduren wie z.B. "konstant_gerade" und "bremsen" übergeben. Gesamtzeit wird nicht direkt benötigt, um die Aufgaben zu lösen, erwies sich jedoch als praktisch bei Tests von veränderten Streckenprofilen oder einzelnen Abschnitten und bleibt deshalb im Programm enthalten.

x, y (Array (0..maxmess) of double) sowie i (integer) werden benötigt, um Paare von Gesamtstrecke und Gesamtenergie in plotbaren Datensätzen speichern zu können. Die Konstante maxmess initialisiert dabei die Arrays auf den Maximalwert von 3000. Dieser Wert ergibt sich aus der Formel

$$\text{maxmess} = [\text{Gesamtzeit}(80\% \text{ v}_{\text{SOLL}}) / (\Delta t * \text{ktewert})] + \text{Sicherheit}.$$

i ist die Indexvariable der Arrays und wird zu Beginn der Prozedur "Energie_berechnen" auf 0 initialisiert und jedes Mal, nachdem Daten in die Arrays geschrieben wurden, um 1 erhöht. Um im Diagramm auch wirklich bei Punkt (0,0) zu starten, wird in beide Arrays 0 als erster Wert geschrieben, noch bevor es in die Energieschleifen geht.

k (integer) bewegt sich zwischen 0 und ktewert - wird nach jedem Rechenschritt in den Energieprozeduren um 1 erhöht und, wenn ktewert erreicht wird, wieder auf 0 gesetzt - und sorgt dafür, dass nur Daten in die Arrays geschrieben werden, wenn $k = \text{ktewert}$ ist. Dadurch lässt sich die Arraygröße drastisch reduzieren, wobei die Genauigkeit für den Plot kaum leidet.

Die Variable filename (string) enthält, abhängig vom eingestellten Modus, entweder den string 'Dieseldaten.dat' oder 'Elektrodaten.dat' und gibt somit den korrespondierenden Dateien auf der Festplatte ihren Namen.

Zuletzt wird noch multiplot (boolean) benötigt, um durch eine Fallunterscheidung in der Prozedur "diagramm_plotten" ein anderes externes Skript aufzurufen, falls der Benutzer wünscht ein Vergleichsdiagramm zu plotten.

Konstanten:

deltat = 0.0001 s ist ein Zeitinkrement, das benötigt wird, um die Summation der Energien zu ermöglichen.

maxmess = 3000 (siehe Kategorie II, x, y)

ktewert = 5000 legt fest, jeder wievielte Wert in die Arrays übergeben wird. 5000 resultiert in einem Messwert jede halbe Sekunde, was für ein Diagramm genau genug ist.

eps = 0.00001 wird für problematische Vergleiche mit Real-Zahlen benötigt. Da die Genauigkeit bei der Rechnung mit ebendiesen begrenzt ist, kann es zu Problemen kommen, wenn das Programm prüft, ob zwei Real-Zahlen gleich groß sind. Um dies zu umgehen, prüft man besser, ob der Unterschied zweier Zahlen kleiner als ein kleiner Wert Epsilon ist.

2.2 Initialisierung

Beim Start des Programms werden alle Variablen initialisiert, die notwendig sind, um ein gültiges Antriebs-/Fahrszenario zu schaffen, sodass die Simulation gestartet werden kann ohne vorher Parameter von Hand einstellen zu müssen.

Initialisiert werden *eta* auf 0.32, *p* auf 50000, *v* auf 13.89, *filename* auf 'Dieseldaten.dat' und *modus* auf false. Dies sorgt für das Szenario Dieselmotor mit 50 kW Nennleistung bei einer Sollgeschwindigkeit von 50 km/h.

2.3 Nebenbedingungen

Es werden keine besonderen Bedingungen zur Erhaltung der Datenkonsistenz benötigt.

2.4 Hintergrundspeicher

Am Ende der Prozedur "Energie_berechnen" werden die gefüllten Arrays *x* und *y* durch Unterprozeduren in einer Textdatei mit dem Suffix *.dat im Programmordner auf der Festplatte abgelegt. Die Datei bekommt als Namen den String, den die Variable *filename* zum Zeitpunkt des Schreibvorgangs enthält. Existiert die Datei noch nicht, wird sie angelegt. Falls jedoch schon eine Datei mit demselben Namen im Programmverzeichnis liegt, hat der Benutzer die Wahl, ob er diese überschreiben möchte.

3 Dokumentation des Lösungsansatzes

3.1 Lösungsverfahren

Um eine erste Vorstellung von den physikalischen Zusammenhängen zwischen Energie, Zeit und zurückgelegter Strecke zu bekommen, haben wir uns zuerst an einem Beispiel orientiert. Dazu haben wir die Sollgeschwindigkeit v_{SOLL} auf 13,89 m/s (= 50 km/h) festgelegt und als Grundlage genommen, da sich insbesondere Zeit t und Beschleunigung a davon ableiten.

Da die Beschleunigung a als über einen Streckenabschnitt konstant angenommen werden konnte und das Streckenprofil und somit auch die Strecke s in der Aufgabenstellung gegeben war, konnten wir über die Gleichung

$$a = (v_e^2 - v_o^2)/(2*s),$$

welche sich aus

$$s(t) = 0,5*a*t^2 + v_0*t + s_0$$

und

$$v(t) = a*t + v_0 \text{ (hier: } v_0, s_0 = 0 \text{)}$$

zusammensetzt, die Beschleunigung berechnen, sodass das Fahrzeug einerseits die Sollgeschwindigkeit in der vorgegebenen Strecke erreicht und andererseits auch wieder entsprechend abgebremst wird.

Mit der nun bekannten Beschleunigung konnte auch die Zeit t bestimmt werden, welche von dem Fahrzeug benötigt wird, um den dazugehörigen Streckenabschnitt zu durchfahren. Der Zusammenhang lautet folgendermaßen:

$$t = (v_e - v_o)/a.$$

Da bei der Beschleunigung $v_0 = 0$ gilt, kann für $v_e - v_0$ auch direkt v_{SOLL} eingesetzt werden. Durch Integration lassen sich weiterhin die Geschwindigkeitsfunktionen der einzelnen Streckenabschnitte berechnen.

Mit diesen berechneten Werten konnten zu einer ersten Veranschaulichung ein a - t -Diagramm, ein v - t -Diagramm und ein s - t -Diagramm erstellt werden. Letzteres konnte dann als eine Hilfestellung für die Interpretation des gewünschten E - s -Diagramms (Energie über Strecke), welches das Programm ausgeben können soll, genutzt werden.

Der Aufgabenstellung konnte weiterhin entnommen werden, dass sich der Energieaufwand und somit der Fahrwiderstand, aus drei Teilen zusammensetzt. Das bedeutet generell, dass

$$F_{\text{Fahr}} = \sum F_i.$$

Energie wird benötigt für das Beschleunigen, für die Überwindung der Rollreibung und für die Überwindung des Luftwiderstandes.

Der Luftwiderstand ist im Gegensatz zu den anderen Größen abhängig von der Geschwindigkeit und berechnet sich folgendermaßen:

$$F_L = 0.5*A*c_w*\rho_L*v^2.$$

Diese Größe wird im Folgenden und insbesondere im Programm als “Luftwiderstand* v^2 ” betrachtet, da es sich hierbei, bis auf v , um Konstanten handelt, sodass diese in “Luftwiderstand” zusammengefasst werden können.

Die Rollreibung setzt sich nur aus Normalkraft und dem Rollwiderstandskoeffizienten zusammen zu:

$$F_R = F_N \cdot r = m \cdot g \cdot \cos(\alpha) \cdot r.$$

Da $\alpha = 2,291^\circ$ und somit $\cos(\alpha) \approx 1$, vereinfacht sich die Formel zu

$$F_R = m \cdot g \cdot r = \text{“Rollreibung”}.$$

Die Beschleunigungskraft ergibt sich aus

$$F_a = m \cdot a = \text{“mf*Beschleunigung” bzw. “mf*Verzögerung”}.$$

Also ist allgemein

$$F_{\text{Fahr}} = \text{Luftwiderstand} \cdot v^2 + \text{Rollreibung} + \text{mf} \cdot \{\text{Beschl./Verzög.}\}$$

Bei vorhandener Steigung muss natürlich noch die Hangabtriebskraft F_{HA} berücksichtigt werden, welche wie folgt lautet:

$$F_{HA} = m \cdot g \cdot \sin(\alpha) = \text{“Hangabtriebskraft”}.$$

Da bei konstanter Fahrt $a=0$ gilt, folgt somit für diesen Fall auch $\text{mf} \cdot a = 0$.

Aus diesen Überlegungen ergeben sich vorläufig nun folgende Fahrwiderstände für die jeweiligen Streckenabschnitte:

Gerade Strecke:

- Beschleunigung ($v(t) = a \cdot t$)

$$F_{\text{gerade}}^{\text{beschleunigen}}(t) = \text{mf} \cdot \text{Beschleunigung} + \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung}$$

- konstante Fahrt ($v(t) = v_{\text{SOLL}}$)

$$F_{\text{gerade}}^{\text{konstant}}(t) = \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung}$$

- bremsen ($v(t) = -a \cdot t + v_{\text{SOLL}}$)

$$F_{\text{gerade}}^{\text{bremsen}}(t) = \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung}$$

Steigung/Gefälle:

- Beschleunigung ($v(t) = a \cdot t$)

$$F_{\text{bergauf/bergab}}^{\text{beschleunigen}}(t) = \text{mf} \cdot \text{Beschleunigung} + \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung} \pm \text{Hangabtriebskraft}$$

- konstante Fahrt ($v(t) = v_{\text{SOLL}}$)

$$F_{\text{bergauf/bergab}}^{\text{konstant}}(t) = \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung} \pm \text{Hangabtriebskraft}$$

- bremsen ($v(t) = -a \cdot t + v_{SOLL}$)

$$F_{\text{bergauf/bergab}}^{\text{bremsen}}(t) = \text{Luftwiderstand} \cdot v^2(t) + \text{Rollreibung} \pm \text{Hangabtriebskraft}$$

Aus den obigen Kräften konnten nun mit dem Zusammenhang

$$E = F \cdot s$$

mit

$$s = v(t) \cdot \Delta t$$

die eigentlichen Energiegleichungen aufgestellt werden. Da die Lösung per Summation und nicht über ein Integral erfolgen soll, musste an dieser Stelle das Zeitinkrement Δt eingebracht werden. Insbesondere ist hier zu beachten, dass s ungleich dem obigen $s(t)$, mit welchem parallel die eigentliche Strecke berechnet wird, ist!

Exemplarisch soll hier nun einmal die Gleichung für die Beschleunigung am Berg mit einem Dieselmotor aufgeführt werden:

$$E_{\text{bergauf}}^{\text{beschleunigen}}(t) = F_{\text{bergauf}}^{\text{beschleunigen}}(t) \cdot [(a \cdot t) \cdot \text{deltat}]$$

Diese Art von Gleichung liefert uns nun die zur Überwindung der Fahrwiderstände benötigte Energie. Da aber nun der Gesamtwirkungsgrad unserer Fahrzeuge $\neq 1$ ist, müssen wir noch die rechte Seite der Gleichung durch den zum Fahrzeug korrespondierenden kombinierten Wirkungsgrad η teilen, um auf die zugeführte (chemische) Energie zu kommen und somit der Definition des Wirkungsgrades

$$\eta = E_{ab} / E_{zu} \Leftrightarrow E_{zu} = E_{ab} / \eta$$

zu entsprechen. Dieser Wirkungsgrad beläuft sich im Falle des Diesels auf

$$\eta = \eta_{\text{Diesel1}} \cdot \eta_{\text{Diesel2}} \Leftrightarrow \underline{\eta} = 0,4 \cdot 0,8 = \underline{0,32}$$

und für das Elektroauto auf

$$\eta = \eta_{\text{Elektro}} \cdot \eta_{\text{Batterie}} \Leftrightarrow \underline{\eta} = 0,85 \cdot 0,8 = \underline{0,68}$$

Da der Dieselmotor generell laut Aufgabenstellung mechanische Verluste in Höhe von 20 % der Nennleistung zu bewältigen hat, wird schlussendlich noch eine Konstante

$$\text{idle} = 0,2 \cdot P_N \cdot \Delta t$$

zu der verbrauchten Energie hinzuaddiert. Diese Konstante stellt die Leerlaufenergie dar, welche es nur beim Dieselmotor gibt, solange dieser läuft jedoch immer!

Daraus folgt dann:

$$E_{\text{bergauf}}^{\text{beschleunigen}}(t) = [F_{\text{bergauf}}^{\text{beschleunigen}}(t) \cdot [(a \cdot t) \cdot \text{deltat}]] / \eta \quad (+ \text{idle} \{ \text{nur Diesel} \})$$

Analog dazu lassen sich die Energiegleichungen für die anderen Streckenabschnitte unter Berücksichtigung der obigen Fahrwiderstände und zugehörigen Geschwindigkeitsfunktionen bestimmen:

Gerade Strecke:

- Beschleunigung ($v(t) = a \cdot t$)

$$E_{\text{gerade}}^{\text{beschleunigen}}(t) = [F_{\text{gerade}}^{\text{beschleunigen}}(t) \cdot [(a \cdot t) \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})}$$

- konstante Fahrt ($v(t) = v_{\text{SOLL}}$)

$$E_{\text{gerade}}^{\text{konstant}}(t) = [F_{\text{gerade}}^{\text{konstant}}(t) \cdot [v \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})}$$

- bremsen ($v(t) = -a \cdot t + v_{\text{SOLL}}$)

$$E_{\text{gerade}}^{\text{bremsen}}(t) = [F_{\text{gerade}}^{\text{bremsen}}(t) \cdot [(-a \cdot t + v) \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})}$$

Steigung/Gefälle:

- Beschleunigung ($v(t) = a \cdot t$)

$$E_{\text{bergauf/bergab}}^{\text{beschleunigen}}(t) = [F_{\text{bergauf/bergab}}^{\text{beschleunigen}}(t) \cdot [(a \cdot t) \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})} \quad (*)$$

- konstante Fahrt ($v(t) = v_{\text{SOLL}}$)

$$E_{\text{bergauf/bergab}}^{\text{konstant}}(t) = [F_{\text{bergauf/bergab}}^{\text{konstant}}(t) \cdot [v \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})}$$

- bremsen ($v(t) = -a \cdot t + v_{\text{SOLL}}$) (vor Vereinfachungen!)

$$E_{\text{bergauf/bergab}}^{\text{bremsen}}(t) = [F_{\text{bergauf/bergab}}^{\text{bremsen}}(t) \cdot [(-a \cdot t + v) \cdot \text{deltat}]] / \eta \text{ (+ idle \{nur Diesel\})} \quad (**)$$

Im Zuge der Entwicklung dieser Gleichungen kam bei Betrachtung der einzelnen Leistungsklassen und Sollgeschwindigkeiten die Frage auf, inwiefern sich die Hangabtriebskraft bei ansteigender oder abfallender Strecke auf die Beschleunigung auswirkt.

Bei ersterem war also zu ermitteln, ob das Fahrzeug bei gegebener Leistung auch bergan die gewählte Geschwindigkeit erreichen kann. Überprüft haben wir dies für den kritischsten Fall (Dieselantrieb) mithilfe der Funktion

$$P_{\text{Benötigt}}(t) = F_{\text{bergauf}}^{\text{beschleunigen}}(t) \cdot (a \cdot t) + \text{idle},$$

welche sich aus der Definition der Leistung

$$P = F \cdot v$$

ergibt.

Mit den eingesetzten, fallspezifischen Werten für Beschleunigung und Nennleistung erhält man den Verlauf der benötigten Motorleistung über die Zeit für diesen Bereich.

Setzt man $P_{\text{Benötigt}}(t)$ nun mit der Nennleistung P_N gleich, kann man anhand der Lösung für t eine Aussage zu unserer Problemstellung treffen. Die Lösung $t = t^*$ gibt den Zeitpunkt an, ab dem die benötigte Leistung die zur Verfügung stehende Nennleistung des Motors übersteigt.

Da wir dank der Formel

$$t_{\text{Bereich}} = (v_e - v_0)/a$$

die Zeit kennen, die wir zu Bewältigung eines jeweiligen Fahrabschnittes benötigen, lässt sich nun einfach sagen, dass, falls

$$t_{\text{Bereich}} \geq t^*$$

erfüllt ist, das Auto mit den eingestellten Parametern die Strecke schafft.

Aus der Überprüfung aller möglichen Fälle ergibt sich folgende Tabelle:

$t_{\text{Bereich}} \text{ [s]}$		30 kW	50 kW	60 kW
54	40 km/h	$t^* = 92,5887 \checkmark$	$t^* = 136,334 \checkmark$	$t^* = 154,255 \checkmark$
43	50 km/h	$t^* = 53,088 \checkmark$	$t^* = 80,14 \checkmark$	$t^* = 91,51 \checkmark$
36	60 km/h	$t^* = 32,5 \text{ ⚡}$	$t^* = 50,32 \checkmark$	$t^* = 58,0193 \checkmark$

Tabelle 1: Schnittpunkte von Leistung über Zeit und Nennleistung

Wie man nun sieht, ist die Bedingung nur bei der Kombination 30 kW und 60 km/h nicht erfüllt und somit werden diese Parameter im Programm nicht zusammen wählbar sein.

Wirkt die Hangabtriebskraft nun in Fahrtrichtung, wird das Fahrzeug dadurch zusätzlich beschleunigt. Hier stellte sich die Frage, ob diese Kraft ausreicht, um die vorgegebene Sollgeschwindigkeit zu erreichen.

Mittels des zweiten Newtonschen Gesetzes

$$F = m \cdot a$$

wobei hier

$$F = F_{\text{Antrieb}}^{\text{bergab}}(v) = \text{Hangabtriebskraft} - \text{Rollreibung} - \text{Luftwiderstand} \cdot v^2$$

und

$$m = mf$$

gilt, lässt sich die allein aus der Hangabtriebskraft resultierende Beschleunigung zu $a_{\text{HA},40} = 0.213 \text{ m/s}^2$ bestimmen. Es wurde hierbei jedoch aufgrund der Abhängigkeit des Luftwiderstandes von der Geschwindigkeit, lediglich der für unsere Überlegung kritische Fall ($v = 40 \text{ km/h} \Rightarrow$ maximaler Luftwiderstand $\Rightarrow a_{\text{HA}}$ minimal) betrachtet. Dies bedeutet, dass die Antriebskraft und somit die resultierende Beschleunigung maximal wird. Da diese Beschleunigung $a_{\text{HA},40} = 0.213 \text{ m/s}^2$ nun größer als die bei $v_{\text{SOLL}} = 40 \text{ km/h}$ benötigte Beschleunigung $a_{40} = 0,206 \text{ m/s}^2$ ist, lässt sich sagen, dass die Hangabtriebskraft F_{HA} tatsächlich ausreicht, um in der vorgegebenen Zeit, bzw. Strecke die niedrigste Sollgeschwindigkeit (40 km/h) zu erreichen.

Also muss in diesem Fall beim Dieselmotor nur für die Kompensation mechanischer Verluste, d.h. für den Leerlauf, Energie aufgewendet werden. Das heißt, dass

$$E_{\text{bergab},40}^{\text{beschleunigen}}(t) = \text{idle}.$$

Der Elektromotor kann hier komplett abgeschaltet bleiben und verbraucht somit keine Energie.

Der Motor muss jedoch zusätzlich Energie aufwenden, um beim Beschleunigungsvorgang bei abfallender Strecke eine Sollgeschwindigkeit von 50 km/h oder 60 km/h zu erreichen und somit gilt für diese Fälle weiterhin Gleichung (*).

Auch für die bei Bremsvorgängen verbrauchte Energie lässt sich auf ähnliche Weise eine Vereinfachung herleiten.

Vergleicht man die aus der Hangabtriebskraft resultierende Verzögerung

$$a_{HA}(v) = F_{Antrieb}^{bergauf}(v) / mf$$

mit den vorgegebenen Verzögerungen

$$a(v) = v^2/400$$

erkennt man, dass im Fall von $v_{SOLL} = 60$ km/h

$$|a(60)| = 0.695 > 0.612 = |a_{HA}(60)|$$

gilt. Das bedeutet, dass unser Fahrzeug ohne zu bremsen erst nach über 200 Metern zum Stehen kommt und somit in diesem Fall vom Motor keine weitere Arbeit geleistet werden muss. Bei Bremsvorgängen in der Ebene oder bergab ist diese Annahme generell für jede wählbare Geschwindigkeit erfüllt.

Somit vereinfacht sich die Gleichung für die Bremsenergie beim Diesel in den meisten Fällen zu

$$E_{Diesel}^{bremsen}(t) = idle,$$

während das Elektroauto gar keine Energie verbraucht. Die Ausnahme bilden die Fälle bremsen bergauf bei 40 und 50 km/h, für die weiterhin Gleichung (***) gilt.

Mit Hilfe dieser physikalischen Ausarbeitung soll nun exemplarisch die informationstechnische Umsetzung der Annahmen und Gleichungen in Prozeduren gezeigt werden.

Es werden insgesamt neun verschiedene Energieprozeduren benötigt, die sich allesamt jedoch sehr ähnlich sind. An dieser Stelle beschränken wir uns daher auf die Beschleunigung bergauf. Zu Beginn der Prozedur werden alle Variablen, die nur für den momentan simulierten Bereich gelten, auf 0 initialisiert. Anschließend wird eine while-Schleife eingeführt, in der die einzelnen Berechnungen für jeden Zeitpunkt t getätigt und aufsummiert werden. Da die Bereichszeit sich bei verschiedenen Geschwindigkeiten ändert und die Bereichslänge (hier 300 m) als einzige Konstante bestehen bleibt, wird die Laufbedingung der Schleife zu $Strecke \leq 300$ gesetzt. Die Strecke wird über die Formel

$$Strecke = 0,5 * Beschleunigung * t^2$$

(bereichsabhängig) zusammen mit der Energie (hier $E_{bergauf}^{beschleunigen}(t)$) im Schleifenrumpf für jeden Zeitpunkt t errechnet.

t beginnt in jedem Bereich bei 0 und wird bei jedem Schleifendurchlauf um $deltat$ erhöht. Zusätzlich wird eine Hilfsvariable eingeführt, mit deren Hilfe die Energie aufsummiert wird. Am Ende des Schleifenrumpfes sollen noch die einzelnen Werte für die momentane Energie sowie die Strecke in Arrays übergeben werden. Mit Addition der Bereichsergebnisse für Energie, Strecke und Zeit auf die entsprechenden globalen Variablen endet diese Prozedur.

3.2 Quellen

Siehe Literaturverzeichnis (Punkt 10)

4 Elemente der Programmstruktur

Durch eine Bottom-Up Analyse der Aufgabenstellung lässt sich als unmittelbares Ziel die Ausgabe der verbrauchten Gesamtenergie sowie das Plotten eines Diagramms, das den Gesamtenergieverbrauch in Abhängigkeit der Gesamtstrecke zeigt, identifizieren. Aus diesen Problemen ergibt sich die Notwendigkeit folgender Prozeduren:

- Ermittlung der Gesamtenergie (Energie_berechnen)
- Bildschirmausgabe der Gesamtenergie (ergebnis_ausgeben)
- Daten auf Festplatte speichern (daten_schreiben)
- Diagramm plotten und anzeigen (diagramm_plotten)

Weiterhin muss die Ermittlung der Gesamtenergie aus mehreren aufeinanderfolgenden Prozeduren bestehen, um der Anforderung an das gegebene Streckenprofil gerecht zu werden. Dies soll realisiert werden, indem für jeden Fahrzustand (beschleunigen, konstant, bremsen) sowie für jede Streckengeometrie (bergauf, gerade, bergab) eine eigene Prozedur entworfen wird. Da diese sich ziemlich ähneln, werden sie im Folgenden nur noch als „Energieschleifen“ bezeichnet. Bevor es in die Energieschleifen geht, müssen jedoch noch die zeitweilig konstanten Variablen berechnet werden (variablen_berechnen).

Da die Berechnung der Energie und das Plotten des Diagramms jeweils auch allein und unabhängig voneinander funktionieren sollen, muss ein Hauptmenü (menu) eingeführt werden, von dem aus Prozeduren nach Wahl des Benutzers aufgerufen werden und nach Abarbeitung dieser wieder dorthin zurückgekehrt wird. Das Hauptmenü soll beim Start des Programmes aufgerufen werden, nachdem die notwendigen Variablen initialisiert wurden (variablen_initialisieren), um einen konsistenten Startpunkt zu schaffen.

Weiterhin ergibt sich aus der Aufgabenstellung die Notwendigkeit einer Prozedur, welche die Variablen ändert, die von dem gewünschten Antrieb und der Sollgeschwindigkeit abhängen.

Diese (parameter_aendern) kann weiter unterteilt werden in die Prozeduren

modus_waehlen → leistung_waehlen → geschwindigkeit_waehlen,
die sequenziell aufgerufen werden sollen.

Aufgrund der geforderten vergleichenden Analyse zweier Antriebskonzepte bietet es sich an, den Benutzer im Hauptmenü wählen zu lassen, ob er eine Einzel- oder Vergleichssimulation durchführen möchte. Für den zweiten Fall wird eine zusätzliche Prozedur (vergleich) eingeführt, die nichts anderes tut, als zweimal “Energie_berechnen” aufzurufen, während sie zwischen den beiden Aufrufen den Modus und alle dazugehörigen Variablen auf die andere Antriebsart umschaltet, um anschließend den Betrag der Differenz der benötigten Gesamtenergie auszugeben.

Da nun die Möglichkeit zu einem automatischen Vergleich besteht, ist es nur konsequent auch bei der Ausgabe des Diagramms durch einen Benutzerdialog (plotmenu) abzufragen, ob ein Einzeldiagramm oder Vergleichsdiagramm, welches sowohl die Diesel- als auch die Elektroenergiekurve in einem Koordinatensystem enthält, geplottet werden soll.

Zum Schluss ergibt sich dieser vereinfachte Programmablaufplan:

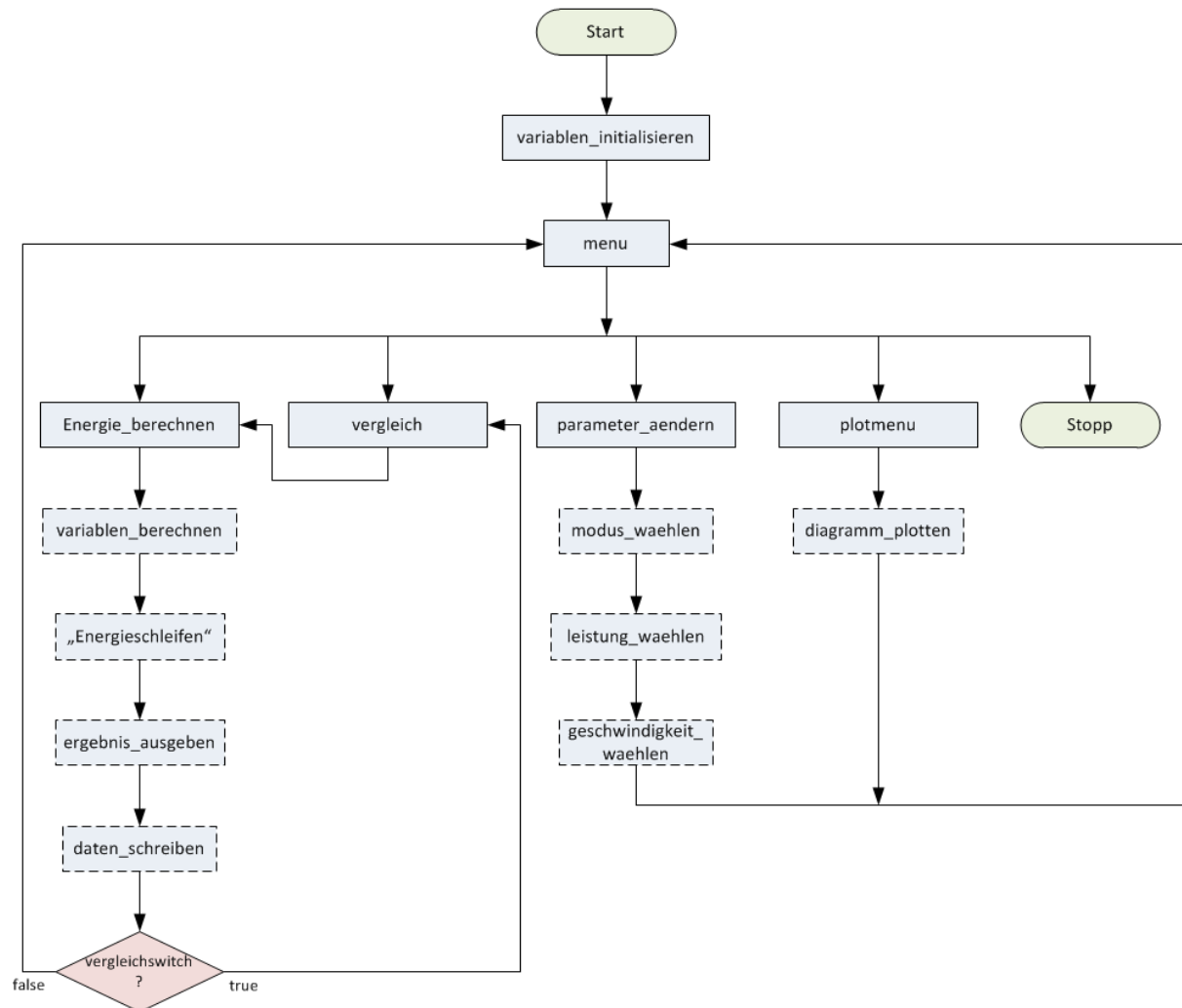


Abbildung 1: vereinfachter Programmablaufplan

5 Benutzungsschnittstelle und Bedienungsanleitung

5.1 Art der Benutzungsschnittstelle

Das Programm soll als traditionelles Kommandozeilenprogramm entstehen. Eine grafische Benutzeroberfläche ist nicht vorgesehen. Der Aufruf externer Programme und Skripte soll weitestgehend automatisiert und im Hintergrund bzw. ohne Interaktion mit dem Benutzer ablaufen

5.2 Bedienungsanleitung

5.2.1 Hauptmenü

Nach dem Start des Programms werden zunächst alle Variablen initialisiert, die benötigt werden, um einen definierten Ausgangspunkt zu schaffen. Anschließend wird der Programmtitel ausgegeben, die eingestellten Parameter aufgelistet und es folgt eine Eingabeaufforderung. Der Benutzer kann nun durch Eingabe einer Zahl wählen zwischen:

- 1: Simulation starten (führt eine Einzelsimulation durch)
- 2: automatischer Vergleich (führt eine Vergleichssimulation durch)
- 3: Parameter aendern (ändert die Randbedingungen wie z.B. Modus und Leistung)
- 4: Diagramm plotten (ruft das Plotmenü auf)
- 5: Programm beenden (schließt das Kommandozeilenfenster)

Dieses Menü wird solange erneut aufgerufen bis eine gültige Auswahl getroffen wurde.

5.2.2 Simulation starten

Diese Funktion berechnet die verbrauchte Gesamtenergie in Abhängigkeit der eingestellten Parameter und gibt anschließend die Ergebnisse (Gesamtenergie, Gesamtstrecke, Gesamtzeit) aus. Anschließend werden die Datensätze auf die Festplatte geschrieben. Dabei wird der Benutzer, falls eine Datei mit gleichem Namen bereits im Programmverzeichnis existiert, über einen Dialog gefragt, ob der sie überschreiben (Eingabe: 1) oder den Schreibvorgang abbrechen möchte (Eingabe: 2). Im Falle, dass überschrieben wurde, werden nun Speicherort und Dateiname angezeigt. Falls abgebrochen wurde, wird dies ebenfalls ausgegeben. Anschließend wird darauf gewartet, dass der Benutzer die Entertaste betätigt, um fortzufahren und ins Hauptmenü zurückzukehren.

5.2.3 automatischer Vergleich

Es geschieht genau dasselbe wie unter dem vorherigen Menüpunkt, mit der Ausnahme dass alle Benutzereingaben doppelt vorkommen, da "Simulation starten" praktisch zweimal hintereinander ausgeführt wird. Nachdem die zweite Datei geschrieben/abgebrochen und anschließend <Enter> betätigt wurde, werden dem Benutzer die Ergebnisse des Vergleichs (Energiedifferenz, Gesamtstrecke, Gesamtzeit) angezeigt und wieder auf <Enter> gewartet um, ins Hauptmenü zurückzukehren.

5.2.4 Parameter aendern

Unter diesem Menüpunkt hat der Benutzer die Möglichkeit die Randbedingungen der Simulation zu ändern. Es werden sequenziell folgende Eingabedialoge durchlaufen:

- Wahl des Modus durch Eingabe einer Zahl:

- 1: Dieselmodus (stellt *eta* etc. auf Dieselantriebswerte)
- 2: Elektromodus (stellt *eta* etc. auf Elektroantriebswerte)
- 3: zurueck zum Menue (kehrt ins Hauptmenü zurück)

- Wahl der Nennleistung durch Eingabe einer Zahl:
 - 1: 30 kW
 - 2: 50 kW
 - 3: 60 kW
 - 4: zurueck zum Menue (kehrt mit gerade eingestelltem Modus ins Hauptmenü zurück ohne Nennleistung und Geschwindigkeit zu ändern)
- Wahl der Sollgeschwindigkeit durch Eingabe einer Zahl:
 - 1: 40 km/h
 - 2: 50 km/h
 - 3: 60 km/h
 - 4: zurueck zum Menue (kehrt mit gerade eingestelltem Modus und Nennleistung ins Hauptmenü zurück, ohne die Geschwindigkeit zu ändern)

Anschließend ruft das Programm wieder das Hauptmenü auf.

5.2.5 Diagramm plotten

Das Programm erfragt über einen Dialog welche Art von Diagramm der Nutzer plotten möchte. Wie gewohnt kann durch Eingabe einer Zahl gewählt werden zwischen:

- 1: Einzeldiagramm plotten
(der aktuell eingestellte Modus wird unter dieser Auswahlmöglichkeit angezeigt, da dieser entscheidet, welches Diagramm (Diesel oder Elektro) geplottet wird)
- 2: Vergleichsdiagramm plotten
(hier werden beide Graphen in einem Koordinatensystem dargestellt)
- 3: zurueck zum Menue (kehrt ins Hauptmenü zurück)

Falls nur eine Datendatei existiert, wird auch im Vergleichsdiagramm nur ein Graph gezeichnet. Falls keine Datendatei existiert, wird dies dem Benutzer angezeigt und nach Bestätigung mit <Enter> ins Hauptmenü zurückgekehrt, ohne ein Diagramm zu plotten.

Alle hier geplotteten Dateien entstehen dadurch, dass eine portable Version von der externen Anwendung "Gnuplot" mit einem zum gewünschten Plot korrespondierendem Skript als Parameter gestartet wird. Anschließend wird die Bilddatei automatisch durch ein Batch-Skript mit dem Betriebssystemeigenen Bildbetrachter geöffnet. Alle verwendeten Skripte befinden sich im Ordner "/skripte" im Programmverzeichnis. Um einen problemlosen Ablauf zu gewährleisten, muss auch der Ordner "gnuplot" mit allen mitgelieferten Dateien im Programmverzeichnis liegen. Sollte dies nicht der Fall sein, kann Gnuplot unter folgender Adresse heruntergeladen werden:

<http://sourceforge.net/projects/gnuplot/files/gnuplot/4.6.0/gp460win32.zip/download>

(Die ordnungsgemäße Funktion der Gnuplot-Skripte kann nur für die Version 4.6.0 garantiert werden)
Die geplotteten Grafiken liegen anschließend als PNG-Datei im Ordner "/diagramme" im Programmverzeichnis. Auch dieser Ordner muss vor dem Plotvorgang bereits existieren, um eine ordnungsgemäße Funktion zu gewährleisten.

5.2.6 Sonstiges

Jedes Mal wenn der Benutzer aufgefordert wird die Entertaste zu betätigen (“<Enter> zum fortfahren...”), ist es auch möglich mit anderen Tasten wie z.B. <H> oder <9> fortzufahren. Dabei kann es jedoch, besonders bei Benutzung etwaiger “Sondertasten” wie z.B. den F- oder Pfeiltasten dazu kommen, dass Pufferrückstände bzw. Rückgabewerte bei der nächsten Eingabeaufforderung vom Programm selbständig eingegeben werden. Diese müssen dann vom Benutzer mit Hilfe der Rücktaste entfernt werden, um eine Fehleingabe zu vermeiden. Daher wird die Benutzung der Enter-, Return- oder aber Space-Taste empfohlen, die sich diesbezüglich alle völlig unauffällig verhalten.

6 Algorithmen

In diesem Kapitel werden die verwendeten Algorithmen in Pseudocode wiedergegeben.

Diese sind nach der Bottom-Up Methode geordnet, d.h. am Anfang stehen diejenigen Methoden, die selbst keine weiteren enthalten.

Eine weitere Aufteilung erfolgt insofern, als dass die Elemente, welche primär der Berechnung der Energie dienen, von denen abgegrenzt werden, die die Funktionalität des Programms gewährleisten.

Der letzte Abschnitt in diesem Kapitel behandelt die externen Elemente, auf welche während des Programmablaufs zugegriffen wird, beispielsweise um die Diagramme zu erstellen.

6.1 Strukturelemente

Im nachfolgenden Abschnitt werden diejenigen Algorithmen behandelt, welche der Struktur und der Funktionalität des Programms dienen.

6.1.1 initialisieren

Zweck: Initialisierung der für die Berechnung der Energie benötigten Parameter.

Definiere *eta* als 0,32, *p* als 50000 und *v* als 13,89.

Definiere *filename* als "Dieseldaten.dat".

Setze *modus* auf false.

6.1.2 warten

Zweck: Unterbrechung des Programms zur besseren Bedienbarkeit durch den Benutzer.

Schreibe Aufforderung <Enter> zu drücken.

Warte auf Eingabe.

6.1.3 titel

Zweck: Anzeigen des Programmtitels im Hauptmenü.

Schreibe Programmname.

6.1.4 daten_ueberschreiben

Zweck: Speichert Energieberechnung in Datei ab. Überschreibt vorhandene Datei, falls gewünscht.

Ändere *i* in 1 und *k* in *ktewert*.

Öffne *F*.

Erstelle, bzw. überschreibe *F*.

Solange $i < j$, dann

schreibe in *F* Wert von Array *x* an der Stelle *i*.

Schreibe in *F* Wert von Array *y* an der Stelle *i*.

Erhöhe *i* um 1.

Schließe *F*.

Schreibe, dass Datei *filename* im Programmverzeichnis gespeichert wurde.

Rufe Prozedur "warten" auf.

6.1.5 daten_schreiben

Zweck: Fragt den Benutzer, ob Datei überschrieben werden soll. Überschreibt Datei oder bricht den Vorgang ab.

Öffne *F*.

Wenn Datei mit Namen *filename* schon existiert, dann

schreibe Meldung, dass Datei schon existiert und frage, ob diese überschrieben werden soll.

Fordere zur Eingabe auf.

Lies *auswahl*.

Falls *auswahl*

= 1: Rufe Prozedur "daten_ueberschreiben" auf.

= 2: Schreibe Meldung, dass der Schreibvorgang abgebrochen wurde.

Rufe Prozedur "warten" auf.

Andernfalls

schreibe eine Fehlermeldung und fordere zur richtigen Eingabe auf.

Rufe Prozedur "warten" auf.

Rufe Prozedur "daten_schreiben" auf.

Ansonsten

rufe Prozedur "daten_ueberschreiben" auf.

6.1.6 modus_waehlen

Zweck: Auswahl zwischen Diesel- und Elektromotor

Setze *abort* auf false.

Schreibe Auswahloptionen für *auswahl* und fordere zur Eingabe auf.

Lies *auswahl*.

Falls *auswahl*

= 1: Setze *modus* auf false, ändere *eta* in 0,32 und *filename* in "Dieseldaten.dat",

= 2: Setze *modus* auf true, ändere *eta* in 0,68 und *filename* in "Elektrodaten.dat",

= 3: Setze *abort* auf true.

Andernfalls

gib Fehlermeldung wegen falscher Eingabe aus.

Rufe Prozedur "warten" auf.

Rufe Prozedur "modus_waehlen" erneut auf.

6.1.7 leistung_waehlen

Zweck: Wahl der Motorleistung. Auswahl zwischen 30, 50 und 60 kW.

Schreibe Auswahloptionen für *auswahl* und fordere zur Eingabe auf.

Lies *auswahl*.

Falls *auswahl*

= 1: Ändere *p* in 30000,

= 2: Ändere *p* in 50000,

= 3: Ändere *p* in 60000,

= 4: Setze *abort* auf true.

Andernfalls

gib Fehlermeldung wegen falscher Eingabe aus.

Rufe Prozedur "warten" auf.

Rufe Prozedur "leistung_waehlen" erneut auf.

6.1.8 geschwindigkeit_waehlen

Zweck: Wahl der Sollgeschwindigkeit v . Auswahl zwischen 40, 50 und 60 km/h.

Schreibe Auswahloptionen für *auswahl* und fordere zur Eingabe auf.

Lies *auswahl*.

Falls *auswahl*

= 1: Ändere v in 11,11,

= 2: Ändere v in 13,89,

= 3: Wenn $p = 30000$, dann

Schreibe Fehlermeldung wegen falsch eingestellter Parameter.

Rufe Prozedur "warten" auf.

Rufe Prozedur "geschwindigkeit_waehlen" erneut auf.

Ansonsten

ändere v in 16,67.

= 4: Beende die Prozedur ohne Auswahl einer Geschwindigkeit.

6.1.9 parameter_listen

Zweck: Voreingestellte, bzw. ausgewählte Parameter anzeigen.

Wenn *modus* = true, dann

schreibe, dass 'Elektromodus' gewählt ist.

Ansonsten

schreibe, dass 'Dieselmodus' gewählt ist.

Dividiere p durch 1000 (Umrechnung von Watt in Kilowatt).

Schreibe obiges Ergebnis als gewählte Leistung.

Multipliziere v mit 3.6 (Umrechnung von m/s in km/h).

Schreibe obiges Ergebnis als gewählte Geschwindigkeit.

6.1.10 parameter_aendern

Zweck: Einstellungen bezüglich Motor, Leistung und Geschwindigkeit wählen.

Rufe Prozedur "modus_waehlen" auf.

Wenn *abort* = true, dann rufe Prozedur "menu" auf.

Ansonsten

rufe Prozedur "leistung_waehlen" auf.

Wenn *abort* = true, dann rufe Prozedur "menu" auf.

Andernfalls

rufe Prozedur "geschwindigkeit_waehlen" auf.

Rufe Prozedur "menu" auf.

6.1.11 plotmenu

Zweck: Lässt den Benutzer zwischen Einzel- und Vergleichsdiagramm zum Plotten wählen. Leitet zu Prozedur "diagramm_plotten" weiter.

Wenn *modus* = true, dann ändere *mode* in '(Elektromodus)'.

Ansonsten ändere *mode* in '(Dieselmodus)'.

Schreibe Auswahlmöglichkeit für Plotfunktion und fordere zur Eingabe auf.

Lies *auswahl*.

Falls *auswahl*

= 1: Setze *multiplot* auf false.

(Einzelgraph)

Rufe Prozedur "diagramm_plotten" auf.

= 2: Setze *multiplot* auf true.

(Vergleichsgraph)

Rufe Prozedur "diagramm_plotten" auf.

= 3: Rufe Prozedur “menu” auf.
 Andernfalls
 Schreibe Fehlermeldung und fordere zur richtigen Eingabe auf.
 Rufe Prozedur “warten” auf.
 Rufe Prozedur “plotmenu” auf.

6.1.12 diagramm_plotten

Zweck: Verwertet Daten von Energieberechnung, indem diese in Diagramm geschrieben werden.

Wenn *multiplot* = false, dann
 wenn *modus* = true, dann
 wenn die Datei mit Namen *filename* existiert, dann
 ändere *diagramm* in ‘skripte/oeffneE.bat’.
 Führe Anwendung gnuplot mit Startparameter ‘skripte/Elektroplot.plt’ aus.
 Ansonsten
 schreibe, dass Datei *filename* nicht gefunden wurde.
 Schreibe, dass ins Hauptmenü zurückgekehrt wird.
 Rufe Prozedur “warten” auf.
 Rufe Prozedur “menu” auf.
 Ansonsten (*modus* auf false gesetzt)
 wenn die Datei mit Namen *filename* existiert, dann
 ändere *diagramm* in ‘skripte/oeffneD.bat’.
 Führe Anwendung gnuplot mit Startparameter ‘skripte/Dieselplot.plt’ aus.
 Ansonsten
 schreibe, dass Datei *filename* nicht gefunden wurde.
 Schreibe, dass ins Hauptmenü zurückgekehrt wird.
 Rufe Prozedur “warten” auf.
 Rufe Prozedur “menu” auf.
 Ansonsten
 wenn Datei “Dieseldaten.dat” und/oder Datei “Elektrodaten.dat” existiert, dann
 ändere *diagramm* in ‘skripte/oeffneM.bat’.
 Führe Anwendung gnuplot mit Startparameter ‘skripte/Multiplot.plt’ aus.
 Ansonsten
 schreibe, dass keine Datendatei gefunden wurde.
 Schreibe, dass ins Hauptmenü zurückgekehrt wird.
 Rufe Prozedur “warten” auf.
 Rufe Prozedur “menu” auf.
 Führe *diagramm* aus.
 Rufe Prozedur “menu” auf.

6.1.13 menu

Zweck: Hauptmenü. Der Benutzer kann von hier die gewünschten Funktionen des Programms starten.

Rufe Prozedur “titel” auf.
 Rufe Prozedur “parameter_listen” auf.
 Schreibe Auswahloptionen für Programmfunktionen und fordere zur Eingabe auf.
 Lies *auswahl*.
 Falls *auswahl*
 = 1: Setze *vergleichswitch* auf false.
 Rufe Prozedur “Energie_berechnen” auf.
 = 2: Setze *vergleichswitch* auf true.
 Rufe Prozedur “vergleich” auf.
 Rufe Prozedur “menu” auf.

- = 3: Rufe Prozedur "parameter_ändern" auf.
- = 4: Rufe Prozedur "plotmenu" auf
- = 5: Beende das Programm.

Andernfalls

schreibe Fehlermeldung und fordere zur richtigen Eingabe auf.
 Rufe Prozedur "warten" auf.
 Rufe Prozedur "menu" auf.

Antriebsvergleich

Hauptprogramm, nutzt Unit "uMain" mit hier aufgeführten Prozeduren.

Rufe Prozedur "initialisieren" auf.

Rufe Prozedur "menu" auf.

6.2 Energieelemente

Alle folgenden Elemente dienen der Berechnung oder Ausgabe der einzelnen Teilenergien, welche bei der Fahrt auf den verschiedenen Streckenabschnitten anfallen, sowie der dazu benötigten Variablen und letztendlich der Gesamtenergie.

6.2.1 variablen_berechnen

Zweck: Berechnet die Variablen für den jeweiligen Streckenabschnitt.

Definiere *Beschleunigung* als: $v^2/600$.

Definiere *Verzoegerung* als: $v^2/400$.

Definiere *Luftwiderstand* als: $0,5 * A * C_w * P_l$.

Definiere *Rollreibung* als: $m_f * m_f * g$.

Definiere *Hangabtriebskraft* als: $m_f * g * 0,04$.

Definiere *idle* als: $p * 0,2 * \text{deltat}$.

6.2.2 ergebnis_ausgeben

Zweck: Zeigt Modus, Gesamtenergie, -strecke und -zeit bzgl. der Parameter an.

Schreibe 'Modus: '.

Wenn *modus* = true, dann

schreibe 'Elektromodus'.

Ansonsten

schreibe 'Dieselmodus'.

Schreibe Ergebnis für *Gesamtenergie*. (in Kilojoule)

Schreibe Ergebnis für *Gesamtstrecke*. (in Metern)

Schreibe Ergebnis für *Gesamtzeit*. (in Sekunden)

6.2.3 vergleich

Zweck: Berechnet die Energie für beide Modi und berechnet danach die Energiedifferenz.

Definiere *GesamtenergieAlt* und *GesamtenergieDiff* als 0.

Rufe Prozedur "Energie_berechnen" auf.

Ändere *GesamtenergieAlt* in *Gesamtenergie*.

Wenn *modus* = true, dann

setze *modus* auf false.

Ändere *eta* in 0,32.

Ändere *filename* in "Dieseldaten.dat".

Weise *MAlt* 'Energieverbrauch Elektro: ' und *MNeu* 'Energieverbrauch Diesel: ' zu.

Ansonsten

setze *modus* auf true.

Ändere *eta* in 0,68.

Ändere *filename* in "Elektrodaten.dat".

Weise *MAlt* 'Energieverbrauch Diesel: ' und *MNeu* 'Energieverbrauch Elektro: ' zu.

Rufe Prozedur "Energie berechnen" auf.

Ändere *GesamtenergieDiff* in Absolutbetrag von: *GesamtenergieAlt* - *Gesamtenergie*.

Gib *MAlt* mit dazugehöriger Energie *GesamtenergieAlt* (in Kilojoule) aus.

Gib *MNeu* mit dazugehöriger Energie *Gesamtenergie* (in Kilojoule) aus.

Schreibe Ergebnis für *GesamtenergieDiff*. (in Kilojoule)

Schreibe Ergebnis für *Gesamtstrecke*. (in Metern)

Schreibe Ergebnis für *Gesamtzeit*. (in Sekunden)

Rufe Prozedur "warten" auf.

6.2.4 beschleunigen_bergauf

Zweck: Berechnung der Energie bei beschleunigter Fahrt bergauf.

Definiere *E_Bereich*, *Strecke*, *Energie* und *t* als 0.

Wenn *modus* = false, dann

Solange *Strecke* ≤ 300, dann

berechne *Strecke* mit der Formel: $0,5 * \text{Beschleunigung} * t^2$.

Berechne *Energie* mit der Formel:

$$[(m_f * \text{Beschleunigung} + \text{Luftwiderstand} * (\text{Beschleunigung} * t)^2 + \text{Rollreibung} + \text{Hangabtriebskraft}) * (\text{Beschleunigung} * t * \text{deltat})] / \text{eta} + \text{idle}.$$

Erhöhe *t* um *deltat*.

Erhöhe *E_Bereich* um *Energie*.

Wenn *k* = *ktewert*, dann

ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.

Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.

Erhöhe *j* um 1.

Ändere *k* in 0.

Erhöhe *k* um 1.

Ansonsten

solange *Strecke* ≤ 300, dann

berechne *Strecke* mit der Formel: $0,5 * \text{Beschleunigung} * t^2$.

Berechne *Energie* mit der Formel:

$$[(m_f * \text{Beschleunigung} + \text{Luftwiderstand} * (\text{Beschleunigung} * t)^2 + \text{Rollreibung} + \text{Hangabtriebskraft}) * (\text{Beschleunigung} * t * \text{deltat})] / \text{eta}.$$

Erhöhe *t* um *deltat*.

Erhöhe *E_Bereich* um *Energie*.

Wenn *k* = *ktewert*, dann

ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.

Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.

Erhöhe *j* um 1.

Ändere *k* in 0.

Erhöhe *k* um 1.

Erhöhe *Gesamtstrecke* um *Strecke*.

Erhöhe *Gesamtzeit* um *t*.

Erhöhe *Gesamtenergie* um *E_Bereich*.

6.2.5 konstant_bergauf

Zweck: Berechnung der Energie bei konstanter Fahrt bergauf.

Definiere $E_Bereich$, $Strecke$, $Energie$ und t als 0.

Wenn $modus = false$, dann

Solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Berechne $Energie$ mit der Formel:

$[(Luftwiderstand * v^2 + Rollreibung + Hangabtriebskraft) * (v * \Delta t)] / \eta + idle$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $Energie$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

Solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Berechne $Energie$ mit der Formel:

$[(Luftwiderstand * v^2 + Rollreibung + Hangabtriebskraft) * (v * \Delta t)] / \eta$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $Energie$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Erhöhe $Gesamtstrecke$ um $Strecke$.

Erhöhe $Gesamtzeit$ um t .

Erhöhe $Gesamtenergie$ um $E_Bereich$.

6.2.6 bremsen_bergauf

Zweck: Berechnung der Energie bei Bremsvorgängen bergauf mit $v_soll < 60$ km/h.

Definiere $E_Bereich$, $Strecke$, $Energie$ und t als 0.

Wenn $modus = false$, dann

Solange $|Strecke - 200| \geq eps$, dann

berechne $Strecke$ mit der Formel: $v * t - 0,5 * Verzögerung * t^2$.

Berechne $Energie$ mit der Formel:

$[(mf * Verzögerung + Luftwiderstand * (v - Verzögerung * t)^2 + Rollreibung + Hangabtriebskraft) * ((v - Verzögerung * t) * \Delta t)] / \eta + idle$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $Energie$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

Solange $|Strecke - 200| \geq eps$, dann
 berechne *Strecke* mit der Formel: $v * t - 0,5 * Verzögerung * t^2$.
 Berechne *Energie* mit der Formel:

$$[(mf * Verzögerung + Luftwiderstand * (v - Verzögerung * t)^2 + Rollreibung + \text{Hangabtriebskraft}) * ((v - Verzögerung * t) * \text{deltat})] / \eta$$

 Erhöhe *t* um *deltat*.
 Erhöhe *E_Bereich* um *Energie*.
 Wenn $k = ktewert$, dann
 ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.
 Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.
 Erhöhe *j* um 1.
 Ändere *k* in 0.
 Erhöhe *k* um 1
 Erhöhe *Gesamtstrecke* um *Strecke*.
 Erhöhe *Gesamtzeit* um *t*.
 Erhöhe *Gesamtenergie* um *E_Bereich*.

6.2.7 beschleunigen_gerade

Zweck: Berechnung der Energie bei beschleunigter Fahrt auf gerader Strecke.

Definiere *E_Bereich*, *Strecke*, *Energie* und *t* als 0.

Wenn *modus* = false, dann

Solange $Strecke \leq 300$, dann

berechne *Strecke* mit der Formel: $0,5 * Beschleunigung * t^2$.

Berechne *Energie* mit der Formel:

$$[(mf * Beschleunigung + Luftwiderstand * (Beschleunigung * t)^2 + Rollreibung) * (Beschleunigung * t * \text{deltat})] / \eta + idle$$

Erhöhe *t* um *deltat*.

Erhöhe *E_Bereich* um *Energie*.

Wenn $k = ktewert$, dann

ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.

Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.

Erhöhe *j* um 1.

Ändere *k* in 0.

Erhöhe *k* um 1.

Ansonsten

Solange $Strecke \leq 300$, dann

berechne *Strecke* mit der Formel: $0,5 * Beschleunigung * t^2$.

Berechne *Energie* mit der Formel:

$$[(mf * Beschleunigung + Luftwiderstand * (Beschleunigung * t)^2 + Rollreibung) * (Beschleunigung * t * \text{deltat})] / \eta$$

Erhöhe *t* um *deltat*.

Erhöhe *E_Bereich* um *Energie*.

Wenn $k = ktewert$, dann

ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.

Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.

Erhöhe *j* um 1.

Ändere *k* in 0.

Erhöhe *k* um 1.

Erhöhe *Gesamtstrecke* um *Strecke*.

Erhöhe *Gesamtzeit* um *t*.

Erhöhe *Gesamtenergie* um *E_Bereich*.

6.2.8 konstant_gerade

Zweck: Berechnung der Energie bei konstanter Fahrt auf gerader Strecke.

Definiere $E_Bereich$, $Strecke$, $Energie$ und t als 0.

Wenn $modus = false$, dann

Solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Berechne $Energie$ mit der Formel:

$[(Luftwiderstand * v^2 + Rollreibung) * (v * \Delta t)] / \eta + idle$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $Energie$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

Solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Berechne $Energie$ mit der Formel:

$[(Luftwiderstand * v^2 + Rollreibung) * (v * \Delta t)] / \eta$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $Energie$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Erhöhe $Gesamtstrecke$ um $Strecke$.

Erhöhe $Gesamtzeit$ um t .

Erhöhe $Gesamtenergie$ um $E_Bereich$.

6.2.9 beschleunigen_bergab

Zweck: Berechnung der Energie bei beschleunigter Fahrt bergab.

Definiere $E_Bereich$, $E_Bereich2$, $Strecke$, $Energie$ und t als 0.

Wenn $modus = false$, dann

solange $Strecke \leq 300$, dann

wenn $v = 11,11$, dann

berechne $Strecke$ mit der Formel: $0,5 * Beschleunigung * t^2$.

Erhöhe t um Δt .

Erhöhe $E_Bereich$ um $idle$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

berechne *Strecke* mit der Formel: $0,5 * \text{Beschleunigung} * t^2$.
 Berechne *Energie* mit der Formel:

$$[(m_f * \text{Beschleunigung} + \text{Luftwiderstand} * (\text{Beschleunigung} * t)^2 + \text{Rollreibung} - \text{Hangabtriebskraft}) * (\text{Beschleunigung} * t * \text{deltat})] / \text{eta}.$$

 Erhöhe *t* um *deltat*.
 Erhöhe *E_Bereich* um *Energie*.
 Erhöhe *E_Bereich2* um *idle*.
 Wenn $k = \text{ktewert}$, dann
 ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.
 Wenn $E_Bereich > 0$, dann
 Ändere den Wert von Array *y* an der Stelle *j* zu:
 Gesamtenergie + *E_Bereich*.
 Ansonsten
 Ändere den Wert von Array *y* an der Stelle *j* zu:
 Gesamtenergie + *E_Bereich2*.
 Erhöhe *j* um 1.
 Ändere *k* in 0.
 Erhöhe *k* um 1.
 Ansonsten
 solange $\text{Strecke} \leq 300$, dann
 wenn $v = 11,11$, dann
 berechne *Strecke* mit der Formel: $0,5 * \text{Beschleunigung} * t^2$.
 Erhöhe *t* um *deltat*.
 Wenn $k = \text{ktewert}$, dann
 ändere den Wert von Array *x* an der Stelle *j* zu: *Gesamtstrecke* + *Strecke*.
 Ändere den Wert von Array *y* an der Stelle *j* zu: *Gesamtenergie* + *E_Bereich*.
 Erhöhe *j* um 1.
 Ändere *k* in 0.
 Erhöhe *k* um 1.
 Ansonsten
 berechne *Strecke* mit der Formel: $0,5 * \text{Beschleunigung} * t^2$.
 Berechne *Energie* mit der Formel:

$$[(m_f * \text{Beschleunigung} + \text{Luftwiderstand} * (\text{Beschleunigung} * t)^2 + \text{Rollreibung} - \text{Hangabtriebskraft}) * (\text{Beschleunigung} * t * \text{deltat})] / \text{eta}.$$

 Erhöhe *t* um *deltat*.
 Erhöhe *E_Bereich* um *Energie*.
 Wenn $k = \text{ktewert}$, dann
 ändere den Wert von Array *x* an der Stelle *j* zu:
 Gesamtstrecke + *Strecke*.
 Wenn $E_Bereich > 0$, dann
 ändere den Wert von Array *y* an der Stelle *j* zu:
 Gesamtenergie + *E_Bereich*.
 Ansonsten
 ändere den Wert von Array *y* an der Stelle *j* zu:
 Gesamtenergie.
 Erhöhe *j* um 1.
 Ändere *k* in 0.
 Erhöhe *k* um 1.
 Erhöhe *Gesamtstrecke* um *Strecke*.
 Erhöhe *Gesamtzeit* um *t*.
 Wenn $E_Bereich > 0$, dann
 erhöhe *Gesamtenergie* um *E_Bereich*.
 Ansonsten

erhöhe *Gesamtenergie* um $E_Bereich2$.

6.2.10 konstant_bergab

Zweck: Berechnung der Energie bei konstanter Fahrt bergab.

Definiere $E_Bereich$, $Strecke$ und t als 0.

Wenn $modus = false$, dann

solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Erhöhe t um $deltat$.

Erhöhe $E_Bereich$ um $idle$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

solange $Strecke \leq 500$, dann

berechne $Strecke$ mit der Formel: $v * t$.

Erhöhe t um $deltat$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Erhöhe $Gesamtstrecke$ um $Strecke$.

Erhöhe $Gesamtzeit$ um t .

Erhöhe $Gesamtenergie$ um $E_Bereich$.

6.2.11 bremsen

Zweck: Berechnung der Energie bei den meisten Bremsvorgängen.

Definiere $E_Bereich$, $Strecke$ und t als 0.

Wenn $modus = false$, dann

solange $|Strecke - 200| \geq eps$, dann

berechne $Strecke$ mit der Formel: $v * t - 0,5 * Verzögerung * t^2$.

Erhöhe t um $deltat$.

Erhöhe $E_Bereich$ um $idle$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: $Gesamtenergie + E_Bereich$.

Erhöhe j um 1.

Ändere k in 0.

Erhöhe k um 1.

Ansonsten

solange $|Strecke - 200| \geq eps$, dann

berechne $Strecke$ mit der Formel: $v * t - 0,5 * Verzögerung * t^2$.

Erhöhe t um $deltat$.

Wenn $k = ktewert$, dann

ändere den Wert von Array x an der Stelle j zu: $Gesamtstrecke + Strecke$.

Ändere den Wert von Array y an der Stelle j zu: *Gesamtenergie*.
 Erhöhe j um 1.
 Ändere k in 0.

Erhöhe k um 1.

Erhöhe *Gesamtstrecke* um *Strecke*.

Erhöhe *Gesamtzeit* um t .

Erhöhe *Gesamtenergie* um $E_Bereich$.

6.2.12 Energie_berechnen

Zweck: Berechnet die gesamte, benötigte Energie in Abhängigkeit der eingestellten Parameter.

Definiere *Gesamtenergie*, *Gesamtzeit* und *Gesamtstrecke* als 0.

Definiere k als *ktewert*.

Definiere j als 0.

Definiere den Wert von x und y jeweils an der Stelle j als 0.

Erhöhe j um 1.

Rufe Prozedur "variablen_berechnen" auf.

Rufe Prozedur "beschleunigen_gerade" auf.

Rufe Prozedur "konstant_gerade" auf.

Rufe Prozedur "bremsen" auf.

(1 km gefahren)

Rufe Prozedur "beschleunigen_gerade" auf.

Rufe Prozedur "konstant_gerade" auf.

Rufe Prozedur "bremsen" auf.

(2 km gefahren)

Rufe Prozedur "beschleunigen_gerade" auf.

Rufe Prozedur "konstant_gerade" auf.

Rufe Prozedur "bremsen" auf.

(3 km gefahren)

Rufe Prozedur "beschleunigen_bergauf" auf.

Rufe Prozedur "konstant_bergauf" auf.

Wenn $|v-16.67| < eps$ dann

 rufe Prozedur "bremsen" auf.

Ansonsten

 rufe Prozedur "bremsen_bergauf" auf.

(4 km gefahren)

Rufe Prozedur "beschleunigen_bergauf" auf.

Rufe Prozedur "konstant_bergauf" auf.

Wenn $|v-16.67| < eps$ dann

 rufe Prozedur "bremsen" auf.

Ansonsten

 rufe Prozedur "bremsen_bergauf" auf.

(5 km gefahren)

Rufe Prozedur "beschleunigen_bergab" auf.

Rufe Prozedur "konstant_bergab" auf.

Rufe Prozedur "bremsen" auf.

(6 km gefahren)

Rufe Prozedur "beschleunigen_bergab" auf.

Rufe Prozedur "konstant_bergab" auf.

Rufe Prozedur "bremsen" auf.

(7 km gefahren)

Rufe Prozedur "beschleunigen_gerade" auf.

Rufe Prozedur "konstant_gerade" auf.

Rufe Prozedur “bremsen” auf.
 (8 km gefahren)

Rufe Prozedur “beschleunigen_gerade” auf.
 Rufe Prozedur “konstant_gerade” auf.
 Rufe Prozedur “bremsen” auf.
 (9 km gefahren)

Rufe Prozedur “beschleunigen_gerade” auf.
 Rufe Prozedur “konstant_gerade” auf.
 Rufe Prozedur “bremsen” auf.
 (10 km gefahren)

Rufe Prozedur “ergebnis_ausgeben” auf.
 Rufe Prozedur “daten_schreiben” auf.
 Wenn *vergleichswitch* auf false gestellt, dann
 rufe Prozedur “menu” auf.

6.3 Externe Elemente

Unter diesem Punkt werden die externen Elemente aufgeführt.

6.3.1 Dieselplot.plt

Zweck: Plottet ein Diagramm für den Energieverbrauch des Dieselmotors.

Plote Diagramm aus “Dieseldaten.dat”.

Definiere *y_{max}* als *GPVAL_DATA_Y_MAX*.
 {Diese read-only Variable enthält den größten y-Wert des geplotteten Datensatzes.}

Stelle Ausgabeterminal auf pngcairo mit Bildgröße 1280x1024 Pixel und Schriftart Times New Roman in 20pt.

Setze Pfad/Name der Ausgabedatei zu ‘diagramme/Dieseldiagramm.png’.

Betitele Diagramm mit ‘Energieverbrauch Dieselmotor’.

Benenne x-Achse mit ‘Strecke [m]’.

Benenne y-Achse mit ‘Energie [kJ]’.

Hinterlege das Diagramm mit einem Gitter.

Schreibe die Legende in die obere linke Ecke.

Lege x-Achsenbereich auf 0..10000 fest.

Lege y-Achsenbereich auf 0..(*y_{max}* + 100) fest.

Plote Diagramm (und erstelle Datei) aus “Dieseldaten.dat” im Linienstil Stärke 3 mit Namen ‘Dieselantrieb’.

6.3.2 Elektroplot.plt

Zweck: Plottet ein Diagramm für den Energieverbrauch des Elektromotors.

Plote Diagramm aus “Elektrodaten.dat”.

Definiere *y_{max}* als *GPVAL_DATA_Y_MAX*.

Stelle Ausgabeterminal auf pngcairo mit Bildgröße 1280x1024 Pixel und Schriftart Times New Roman in 20pt.

Setze Pfad/Name der Ausgabedatei zu ‘diagramme/Elektrodiagramm.png’.

Betitele Diagramm mit ‘Energieverbrauch Elektromotor’.

Benenne x-Achse mit 'Strecke [m]'.
 Benenne y-Achse mit 'Energie [kJ]'.
 Hinterlege das Diagramm mit einem Gitter.
 Schreibe die Legende in die obere linke Ecke.
 Lege x-Achsenbereich auf 0..10000 fest.
 Lege y-Achsenbereich auf 0..($y_{max} + 100$) fest.
 Plote Diagramm (und erstelle Datei) aus "Elektrodaten.dat" im Linienstil Stärke 3 mit Namen "Elektroantrieb".

6.3.3 Multiplot.plt

Zweck: Plottet den Energieverbrauch von sowohl dem Elektro- als auch dem Dieselmotor in einem Diagramm.

Stelle Ausgabeterminal auf pngcairo mit Bildgröße 1280x1024 Pixel und Schriftart Times New Roman in 20pt.
 Setze Pfad/Name der Ausgabedatei zu 'diagramme/Vergleichsdiagramm.png'.
 Betitele Diagramm mit 'Energieverbrauch Diesel-/Elektromotor'.
 Benenne x-Achse mit 'Strecke [m]'.
 Benenne y-Achse mit 'Energie [kJ]'.
 Hinterlege das Diagramm mit einem Gitter.
 Schreibe die Legende in die obere linke Ecke.
 Lege x-Achsenbereich auf 0..10000 fest.
 Lege y-Achsenbereich auf 0..30000 fest.
 Plote Diagramm (und erstelle Datei) aus "Dieseldaten.dat" im Linienstil Stärke 3 mit Namen 'Dieselantrieb' und "Elektrodaten.dat" im Linienstil Stärke 3 mit Namen "Elektroantrieb".

6.3.4 Batch-Skripte

Zweck: Öffnet das entsprechende Energie-Strecken-Diagramm im betriebssystemeigenen Bildbetrachter.

Schalte die Bildschirmausgabe aller folgenden Befehlszeilen aus.
 Öffne das Diesel-/Elektro-/Vergleichsdiagramm.

7 Programm entwerfen und möglichst schon abschnittsweise testen

Das Programm wurde nach folgender, chronologisch geordneter, Tabelle entworfen. Bei den Zeitangaben wurde jeweils nur der Zeitpunkt der ersten Implementation der jeweiligen Funktion beachtet. Spätere Änderungen der Prozeduren wurden nicht berücksichtigt.

Funktion / Prozedur	wann	Anmerkung
“Energieschleifen” Prozeduren	bis KW 38	[1]
Energie_berechnen	KW 39	[2]
menu		
variablen_berechnen		
parameter_aendern	KW 40	[3]
modus_waehlen		
leistung_waehlen		
geschwindigkeit_ waehlen		
initialisieren		
parameter_listen		
daten_schreiben	KW 44	[4]
daten_ueberschreiben		
titel	KW 45	[5]
diagramm_plotten		
“Skripte”		
vergleich	KW 46	[6]
ergebnis_ausgeben		
plotmenu		
warten	KW 47	[7]

Tabelle 2: Entstehung des Programms

Anmerkungen:

[1]: Erste Version. Die einzelnen Energieprozeduren (noch ziemlich ineffizient) konnten nur einzeln aufgerufen werden, um sie unabhängig voneinander zu testen.

- [2]: Um das geforderte Streckenprofil simulieren zu können, wurde “Energie_berechnen” eingeführt. Mit Hinblick auf weitere Funktionen wurde ein Menü erstellt. Außerdem wurde die Berechnung der Variablen zur Steigerung der Effizienz in die Prozedur “variablen_berechnen” ausgelagert.

- [3]: “parameter_aendern” inklusive der zugehörigen Unterprozeduren wurde eingeführt. Durch die nun variablen Parameter trat das Problem auf, dass nun beim starten des Programms die Randbedingungen undefiniert waren. Um Abhilfe zu schaffen, wurde “initialisieren” erstellt. Des Weiteren mussten die aktuelle Parameterkonstellation dem Benutzer angezeigt werden, was durch “parameter_listen” erreicht wurde.

- [4]: Um eine plotbare Datei zu bekommen, wurden “daten_schreiben” und “daten_ueberschreiben” eingeführt. Die Prozedur wurde zuerst so entworfen, dass nur eine Datei, unabhängig von der gewählten Antriebsart, erstellt wurde, was den geforderten Vergleich sehr umständlich machte (für Lösung siehe [6]). Außerdem wurden die Arrays, die benötigt werden, um die Messwerte zwischen zu speichern, sehr groß (ca. 13.5 Millionen Werte), da erst in der Prozedur “daten_ueberschreiben” nur jeder k-te Wert aus den Arrays geschrieben wurde. Dies wurde gelöst, indem bereits in den Energieschleifen nur jeder k-te Wert in die Arrays übergeben wird. So konnte die Größe der Arrays drastisch (auf unter 3000 Werte) reduziert werden.

- [5]: Die Prozedur “diagramm_plotten” rief in erster Version ein Batch-Skript über TProcess auf, welches anschließend Gnuplot samt Userskript startete und schließlich die ausgegebene Bilddatei öffnete. Dabei kam es teilweise zu dem Problem, dass Gnuplot sich nicht schloss oder sogar komplett abstürzte. Wir vermuteten das Problem darin, dass die Batchdatei nicht auf Gnuplot wartete und fügten ihr eine Zwangspause hinzu. Dies beseitigte jedoch nur das Problem der Abstürze von Gnuplot. Erst als der Aufruf von Gnuplot und das Öffnen der Bilddatei, in eigenen Prozessen und unabhängig voneinander abliefen, funktionierte alles wie gewollt. Zuletzt wurde die Prozedur jedoch noch auf die Benutzung von SysUtils.ExecuteProcess umgestellt, da uns diese Methode übersichtlicher und eleganter erschien.

- [6]: “vergleich” wurde zur Arbeitserleichterung eingeführt. Im Zuge dessen wurde auch “daten_ueberschreiben” so abgeändert, dass es nun zwei Dateien für unterschiedliche Antriebsarten geben kann. Auch “diagramm_plotten” musste diesem zwei-Datei-System angepasst werden. Weiterhin wurde ein zusätzliches Menü “plotmenu” eingeführt, in dem zwischen einem Einzel- und einem Vergleichsplot gewählt werden kann.

- [7]: Bei dieser Prozedur, welche das Programm auf eine Tastatureingabe warten lässt, um dem Benutzer Zeit zu geben etwaige Bildschirmausgaben zu lesen, kam es anfänglich zu leichten Problemen. In erster Version wurde die Prozedur readln verwendet und der Nutzer anschließend aufgefordert die Eingabetaste zu betätigen. Dabei kam es dazu, dass der readln-Aufruf einfach übersprungen wurde. Da dieses Verhalten von uns nicht behoben werden konnte, entschlossen wir uns, die readln-Zeile mit “repeat until keypressed” zu ersetzen. Dieses aktive Warten wurde später noch mit “repeat delay(500) until keypressed” entschärft und funktionierte nun eigentlich wie es sollte. Jedoch gab es aufgrund der wiederholten 500 ms langen Pause eine Verzögerung beim Tastendruck, die von uns als störend empfunden wurde. Erst ein erneutes Umstellen der Prozedur warten auf einen Aufruf der Funktion readkey brachte, bei Beachtung der Hinweise unter Punkt 5.2.6, ein zufriedenstellendes Ergebnis.

8 Testen: Testdaten entwerfen, Tests durchführen, dokumentieren

An dieser Stelle soll, durch Tests, die ordnungsgemäße Funktion des Programmes sichergestellt und dokumentiert werden. Zu diesem Zweck werden die Tests in zwei Kategorien unterteilt. Die erste Kategorie "Programm" enthält Tests, die sich direkt mit dem Programm befassen (Codesyntax, Dialoge etc.), wobei die zweite Kategorie "Physik" sich eher damit befasst, die Richtigkeit der vom Programm getätigten Berechnungen zu bestätigen.

8.1 Programm

- Test des Leerzustandes

Verhält sich das Programm korrekt wenn noch keine von Programm erzeugten Daten vorhanden sind? :

Funktion ohne Diesel-/Elektrodaten.dat im Programmverzeichnis (Datei erstellen und Diagramm plotten ohne vorhandene Datei)? (Ergebnis: OK)

Funktion ohne Diagramm im Verzeichnis "diagramm", Diagramm plotten (Ergebnis: OK)

- Test des automatischen Vergleiches

Funktioniert die Simulation ausgehend von beiden Modi (Diesel/Elektro)? (Ergebnis: OK)

- Test des Schreibvorganges

Werden die Datendateien, falls gewünscht, überschrieben? (Ergebnis: OK)

Sind die Datendateien wie gewünscht formatiert? (Ergebnis: OK, Datendatei enthält am Ende eine Leerzeile, was aber nicht weiter kritisch ist ⇒ bleibt so)

- Test der Diagramme

Werden Diagramme ohne Fehler erstellt und überschrieben? (Ergebnis: OK)

Entspricht das Aussehen der Diagramme den in den Skripten eingestellten Vorgaben? (Ergebnis: OK)

- Test der Benutzerdialoge

Werden alle Fehleingaben (in allen Menüs) geeignet abgefangen? (Ergebnis: OK)

Wird dem Benutzer der Fehler korrekt angezeigt? (Ergebnis: OK, aber erst nachdem die Prozedur warten auf readkey umgestellt wurde! Readlns wurden teilweise unberechenbar übersprungen, was die Ausgabe von Meldungen unmöglich machte.)

8.2 Physik

- Test der Berechnungen

An dieser Stelle wird geprüft, ob sich die Ergebnisse der einzeln betrachteten Energieprozeduren mit den "von Hand" erzielten decken. Die folgenden Beispielergebnisse beziehen sich auf die Parameter Dieselmotor mit 50 kW bei 50 km/h. Getestet wurden alle Energieprozeduren.

Wird die Zeit korrekt berechnet? Bsp.: "beschleunigen_bergauf"

- Programm:

43,1967 s

- von Hand:

$$t^* = \frac{v}{a} = 43,1965 \text{ s (Ergebnis: OK)}$$

Stimmt die berechnete Energie mit der von Hand überein? Bsp.: “beschleunigen_bergauf”

- Programm:

1678,37 kJ

- von Hand:

$$\int_{t=0}^{t^*} \frac{E_{\text{bergauf}}^{\text{beschleunigen}}(t)}{\text{deltat}} dt = 1675,31 \text{ kJ (Ergebnis: OK)}$$

- Test der Genauigkeit

Ist Δt mit 0,01 klein genug für hinreichend genaue Ergebnisse?

Parameter: Dieselmotor, 50 kW, 50 km/h

deltat [s]	0,01	0,001	0,0001	0,00001
Energieverbrauch [kJ]	24357,3159	24348,7406	24347,6968	24347,5936

Tabelle 3: Werte für verschiedene Δt s

(Ergebnis: Erst ab $\text{deltat} = 0,0001$ ändern sich nur noch die Nachkommastellen. deltat wird auf 0,0001 festgesetzt. Daraus ergeben sich weiterhin die Werte der Konstanten maxmess (nach der in Punkt 3.1 beschriebenen Formel) und ktewert .)

- Test der Realitätsnähe der Ergebnisse

An dieser Stelle sollen die Ergebnisse der Simulation realen Verbräuchen gegenübergestellt werden.

Bei Simulationsparameter von 60 kW und 50 km/h verbraucht das Fahrzeug mit Dieselantrieb 26507,405 kJ/10km \Rightarrow 265074,05 kJ/100km. Aus dem Heizwert 42400² kJ/kg und der Dichte von Diesel 826² kg/m³ von Diesel ergibt sich ein Energiegehalt von 35041,32 kJ/l. Nun kann über die Formel

$$\text{Verbrauch [l/100km]} = \text{Energieverbrauch [kJ/100km]} / 35041,32 \text{ [l/kJ]}$$

der Dieserverbrauch berechnet werden. Das Ergebnis beträgt 7,6 l/100 km. Obwohl dieser Verbrauch den eines modernen Dieselfahrzeugs übersteigt, ist er, besonders im Vergleich mit älteren Modellen und im Hinblick auf die relativ hohe Fahrzeugmasse von 1500 kg sowie die nicht vorhandene Schubabschaltung, als realistisch anzusehen.

Bei dem Elektrofahrzeug gehen wir nun etwas anders vor. Das Ergebnis der Simulation mit denselben Parametern wie oben lautet nun 6376,074 kJ/10 km \Rightarrow 17,71 kWh/100 km. Bei realen, sich in Serienproduktion befindenden, Elektroautos ist eine relativ große Divergenz in den Verbräuchen zu beobachten. Diese liegen ungefähr im Bereich von 13 bis 21 kWh/100 km (z.B.: Ford Focus Electric 18,85 kWh/100 km). Somit liegt auch in diesem Fall der simulierte Verbrauch absolut im Bereich der realen Messwerte.

² Aral: *Dieselmotorkraftstoff. Anforderungen, Qualität, Perspektiven*. URL

http://www.aral.de/liveassets/bp_internet/aral/aral_de/STAGING/local_assets/downloads_pdfs/d/Dieselmotorkraftstoff_Broschuere.pdf Stand: 04.12.2012, S.31

9 Anwendung des Programms

Wie bereits in Abschnitt 1.1 erwähnt, soll an dieser Stelle der Energieverbrauch zweier ansonsten identischer Fahrzeuge mit unterschiedlichen Antriebskonzepten verglichen und analysiert werden.

Ausgehend von den Parametern 50 kW Nennleistung bei 100% Sollgeschwindigkeit (50 km/h) ergibt sich auf der Teststrecke beim Dieselantrieb ein Gesamtenergieverbrauch von 24347,697 kJ sowie 6376,074 kJ beim Elektroantrieb. Die Energiedifferenz beträgt in diesem Fall 17971,623 kJ und übersteigt den Energieverbrauch des Elektromotors somit um mehr als das Doppelte. Aus diesem Szenario erhält man weiterhin folgendes Diagramm:

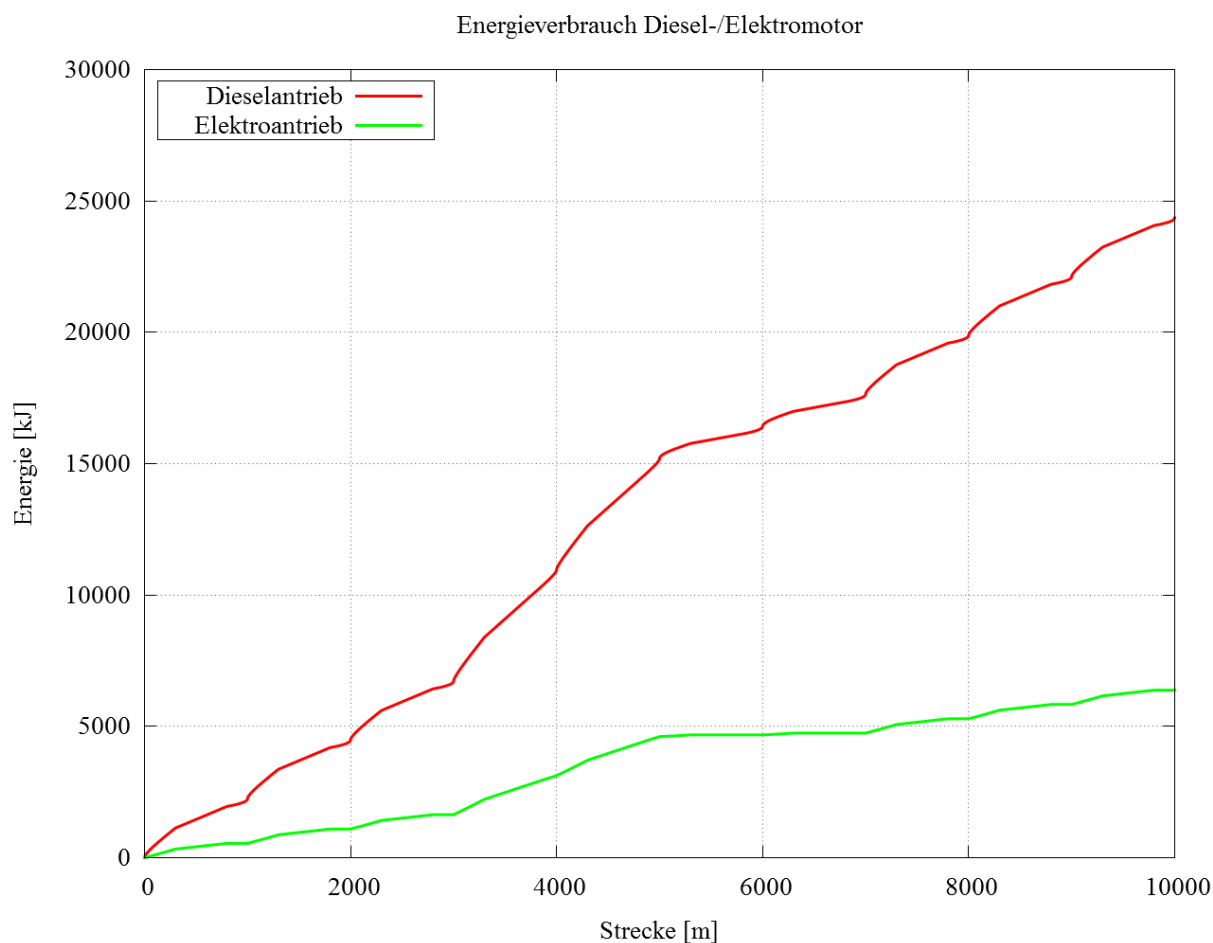


Abbildung 2: Energieverbrauch bei 50 kW und 50 km/h

Es lässt sich erkennen, dass der Energieverbrauch des Dieselfahrzeugs auf jedem Streckenabschnitt deutlich höher ist als der des Elektroautos. Dies ist hauptsächlich auf den wesentlich schlechteren Gesamtwirkungsgrad des Dieselfahrzeugs zurückzuführen. Da während der Fahrt bergauf bei 50 km/h ständig Arbeit geleistet werden muss und die auftretenden Fahrwiderstände in dieser Phase am größten sind, steigt die Energiekurve hier auch am stärksten. Hinzu kommt, dass der Elektroantrieb bei konstanter Fahrt bergab und in den meisten Bremsphasen im Gegensatz zum Dieselmotor, welcher auch ohne Last mechanische Verluste kompensieren muss, überhaupt keine Energie verbraucht und somit auf der Teststrecke wesentlich effizienter ist.

Vergleicht man nun die anderen Leistungsklassen miteinander (Abbildung 6 und Abbildung 7, Anhang 11.1) so stellt man fest, dass sich der Verbrauch des Elektromotors nicht ändert. Unter Berücksichtigung, dass der Elektromotor in Phasen ohne Last keine Energie verbraucht, lässt sich

daraus schließen, dass die Nennleistung beim Elektroantrieb im Rahmen der Aufgabenstellung keine Auswirkung auf den Gesamtverbrauch hat.

Die Energiedifferenz beträgt bei 30 kW und 50 km/h 13652,207 kJ sowie 20131,331 kJ bei 60 kW und gleicher Geschwindigkeit. Das entspricht einem Energieänderungsquotienten von $\Delta E/\Delta P = 2159,708$ kJ/10 kW, welcher sich allein aus der Leerlaufenergie des Diesels ergibt. Die Quotienten für die übrigen Geschwindigkeiten liegen auch in etwa in diesem Bereich.

Variiert man nun einmal bei konstanten 50 kW die Sollgeschwindigkeit um $\pm 20\%$ erhält man Energiedifferenzen von 19523,304 kJ bei 80% (40 km/h) und 16967,553 kJ bei 120% (60 km/h) gegenüber den 17971,623 kJ bei 100% Sollgeschwindigkeit (50 km/h). Während sich der Elektroantrieb so verhält wie man es erwarten würde – je größer v_{SOLL} , desto höher der Energieverbrauch ($\sim -15\%$ bzw. $+11\%$) –, tritt beim Dieselmotor mit 50 ($\sim +2,4\%$ und $-1,3\%$ Energieverbrauch) und 60 kW ($\sim +4,3\%$ und $-2,5\%$ Energieverbrauch) genau der umgekehrte Fall ein. Diese Beobachtung lässt sich erneut anhand der Leerlaufenergie erklären. Während man bei 40 km/h 1350 Sekunden für die Teststrecke benötigt, sind es bei 50 km/h nur noch 1080 Sekunden. Da die Streckenabschnitte ja immer gleich lang sind, erhöht sich die Zeit, die in den einzelnen Bereichen verbracht wird. Somit läuft der Dieselmotor auch länger und kann dadurch mehr lastunabhängige Energie (idle) verbrauchen. Beim 30 kW Diesel ist die Leerlaufenergie zu gering, um sich aufgrund der Zeit so stark auszuwirken, dass es zu einer Umkehrung der erwarteten Energiezunahme kommt, welche wir bei den Dieselmotoren mit höherer Leistung finden.


Geschwindigkeit		30 kW	50 kW	60kW
40 km/h	Diesel	19535,062	24935,302	27635,442
	Elektro	5411,997	5411,997	5411,997
	Differenz	14123,064	19523,304	22223,424
50 km/h	Diesel	20028,281	24347,697	26507,405
	Elektro	6376,074	6376,074	6376,074
	Differenz	13652,207	17971,623	20131,331
60 km/h	Diesel		24051,850	25851,394
	Elektro		7084,296	7084,296
	Differenz		16967,553	18767,097

Tabelle 4: Messwerte der verbrauchten Gesamtenergie in kJ

Zusammenfassend und mit Blick auf obige Tabelle lässt sich nun sagen, dass das Fahrzeug mit Elektroantrieb dem mit Dieselmotor bei jeder wählbaren Parameterkombination auf der vorgegebenen Teststrecke hinsichtlich des Energieverbrauchs deutlich überlegen ist. Dieses Resultat wird verstärkt, je höher die Nennleistung und je niedriger die Geschwindigkeit gewählt wird. Der geringste Unterschied wird bei 30 kW und 50 km/h erreicht, ist aber immer noch so groß, dass das Elektrofahrzeug deutlich vorzuziehen ist.

Auch bei Betrachtung des im Rahmen dieser Simulation nicht beachteten kumulierten Energieaufwandes, ändert sich das Ergebnis nicht. Während der Well-to-Tank Wirkungsgrad bei

Dieselmotoren sehr gute 0,9³ beträgt, lässt er sich für elektrische Energie im Strommix für Deutschland zu 0,5⁴ approximieren. Daraus folgt bei Betrachtung des Falles mit der geringsten Energiedifferenz ein Primärenergieverbrauch von $E_{\text{Diesel}}^{\text{Primär}} = 22253,65 \text{ kJ}$ und $E_{\text{Elektro}}^{\text{Primär}} = 12752,148 \text{ kJ}$ und somit eine Primärenergiedifferenz von $E_{\text{Differenz}}^{\text{Primär}} = 9501,502 \text{ kJ}$ gegenüber der Differenz der verbrauchten Energie $E_{\text{Differenz}}^{\text{verbraucht}} = 13652,207 \text{ kJ}$.

³ Frischknecht, Rolf; Tuchschnid, Matthias (2008): *Primärenergiefaktoren von Energiesystemen* (Version 1.4). URL <http://www.esu-services.ch/fileadmin/download/frischknecht-2008-Energiesysteme.pdf> Stand: 15.12.2012, Tab. 1.1, S.6

⁴ Klell, Manfred; Cona, Patrick (2009): *Wirkungsgrade und CO₂-Emissionen verschiedener Energieketten*. URL <http://www.hycenta.tugraz.at/Image/Report%20Hy8-2009%20HyCentA%20Research%20GmbH.pdf> Stand: 15.12.2012

10 Literaturverzeichnis

(1)

Verein Deutscher Ingenieure (2006): *VDI-Wärmeatlas*. Berlin, Heidelberg: Springer (10. Auflage)

(2)

Aral: *Dieselmotoren. Anforderungen, Qualität, Perspektiven*. URL

http://www.aral.de/liveassets/bp_internet/aral/aral_de/STAGING/local_assets/downloads_pdfs/d/Dieselmotoren_Broschuere.pdf Stand: 04.12.2012, S.31

(3)

Frischknecht, Rolf; Tuchschnid, Matthias (2008): *Primärenergiefaktoren von Energiesystemen*

(Version 1.4). URL [http://www.esu-services.ch/fileadmin/download/frischknecht-2008-](http://www.esu-services.ch/fileadmin/download/frischknecht-2008-Energiesysteme.pdf)

[Energiesysteme.pdf](http://www.esu-services.ch/fileadmin/download/frischknecht-2008-Energiesysteme.pdf) Stand: 15.12.2012, Tab. 1.1, S.6

(4)

Klell, Manfred; Cona, Patrick (2009): *Wirkungsgrade und CO₂-Emissionen verschiedener*

Energieketten. URL [http://www.hycenat.tugraz.at/Image/Report%20Hy8-2009%20HyCentA%20](http://www.hycenat.tugraz.at/Image/Report%20Hy8-2009%20HyCentA%20Research%20GmbH.pdf)

[Research%20GmbH.pdf](http://www.hycenat.tugraz.at/Image/Report%20Hy8-2009%20HyCentA%20Research%20GmbH.pdf) Stand: 15.12.2012

Erbrecht, Rüdiger; König, Hubert; Martin, Karlheinz; Pfeil, Wolfgang; Wörstenfeld, Willi (2002): *Das große Tafelwerk. Mathematik, Physik, Chemie, Astronomie, Informatik, Biologie*. Berlin: Cornelsen, S. 86-90

Lindner, Helmut; Siebke, Wolfgang (2006): *Physik für Ingenieure* (Band 10). München, Wien: Hanser Verlag (17. Auflage)

11 Anhänge

11.1 Anhang: Energieverbrauchsdiagramme



Abbildung 3: Energieverbrauch bei 30 kW und 40 km/h

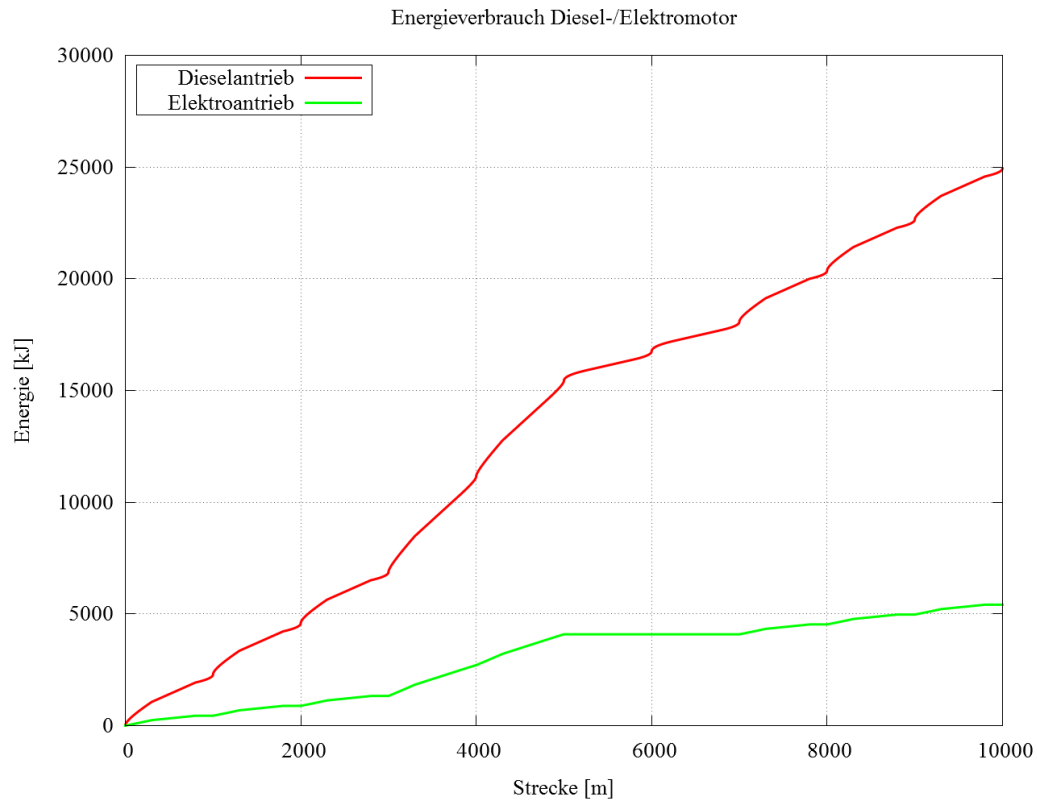


Abbildung 4: Energieverbrauch bei 50 kW und 40 km/h

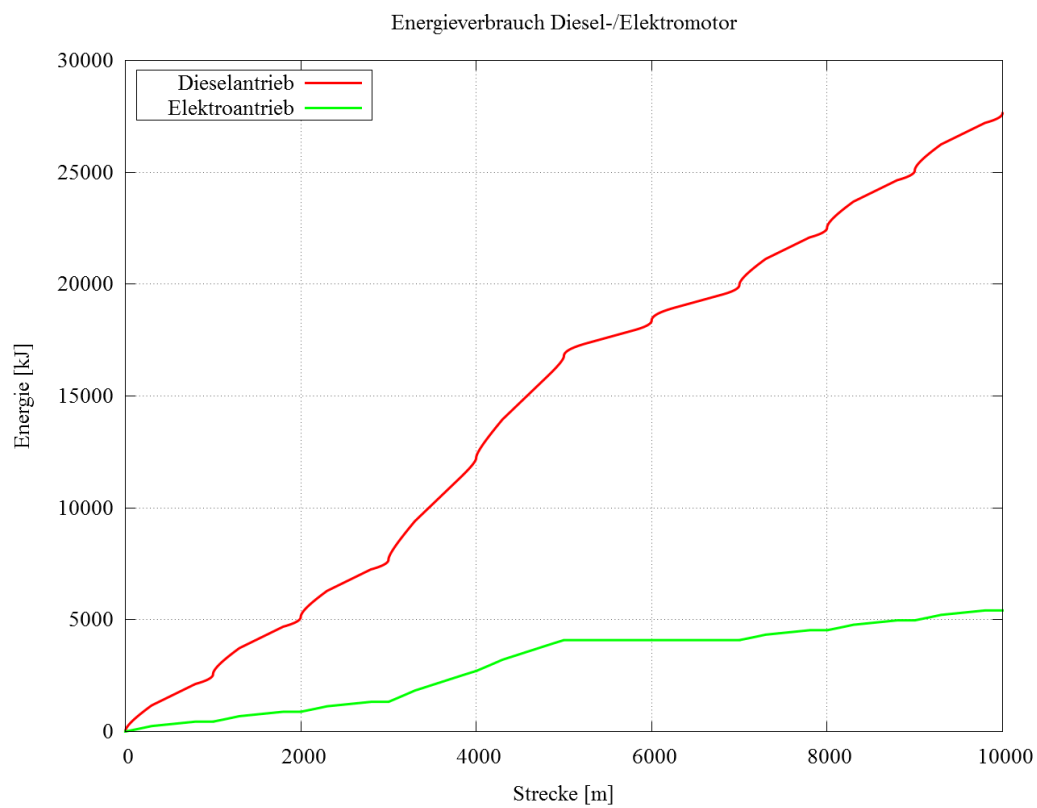


Abbildung 5: Energieverbrauch bei 60 kW und 40 km/h

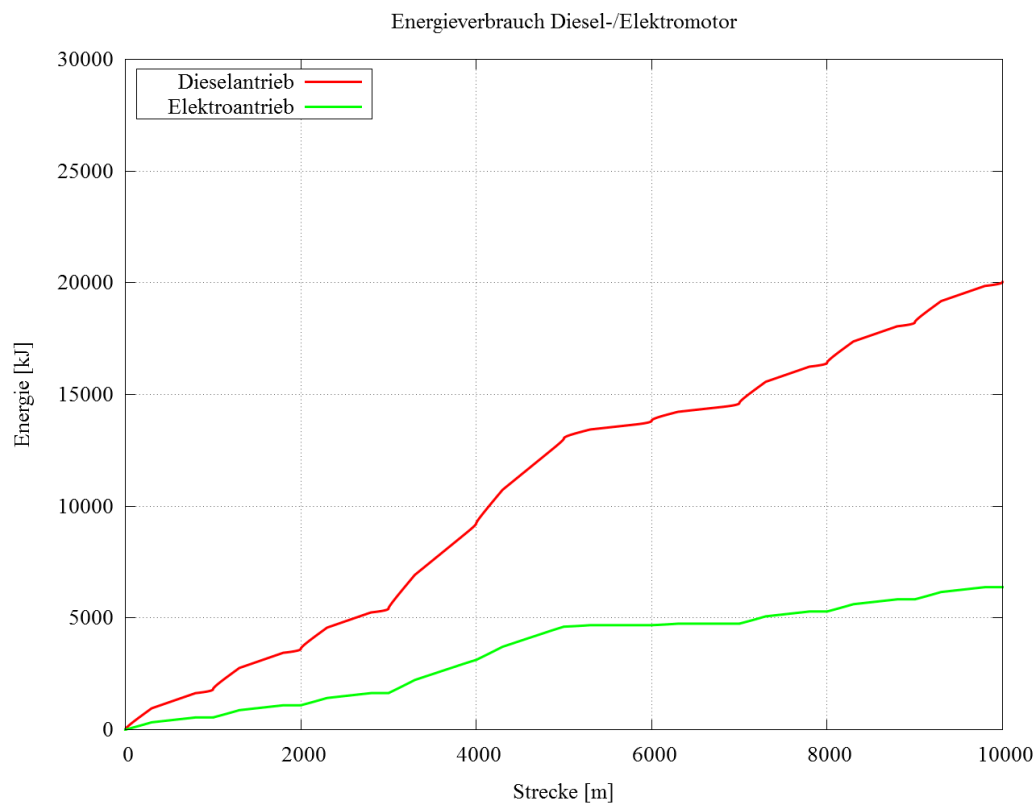


Abbildung 6: Energieverbrauch bei 30 kW und 50 km/h



Abbildung 7: Energieverbrauch bei 60 kW und 50 km/h

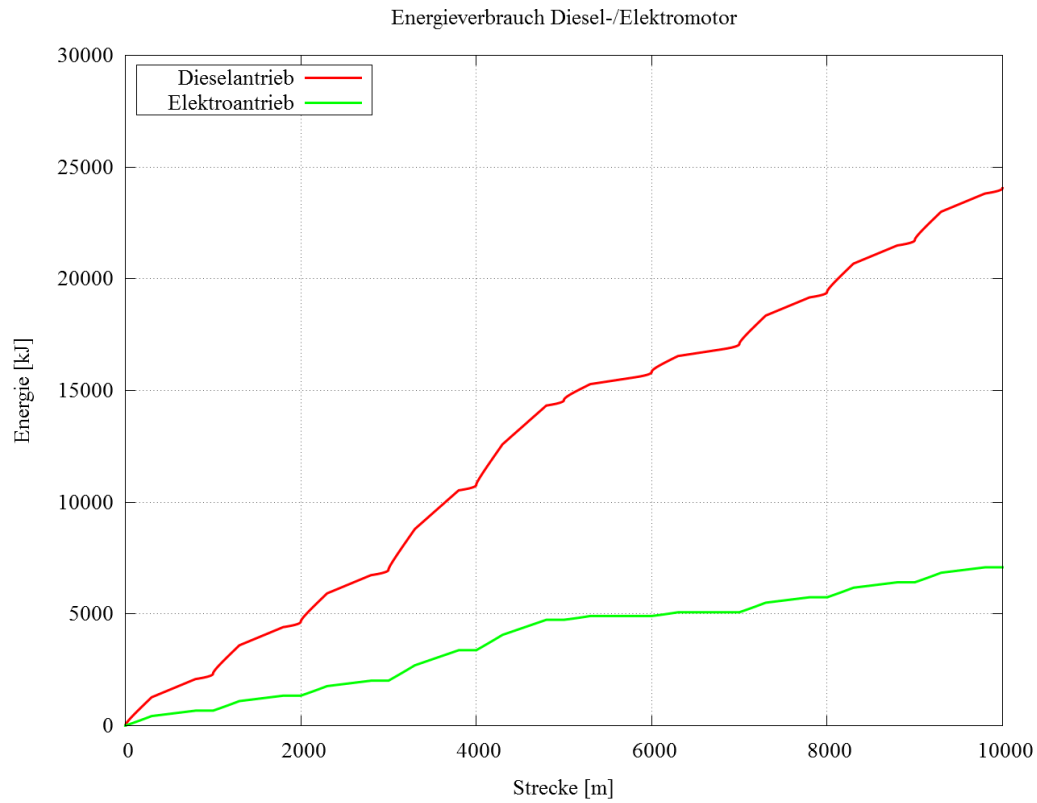


Abbildung 8: Energieverbrauch bei 50 kW und 60 km/h

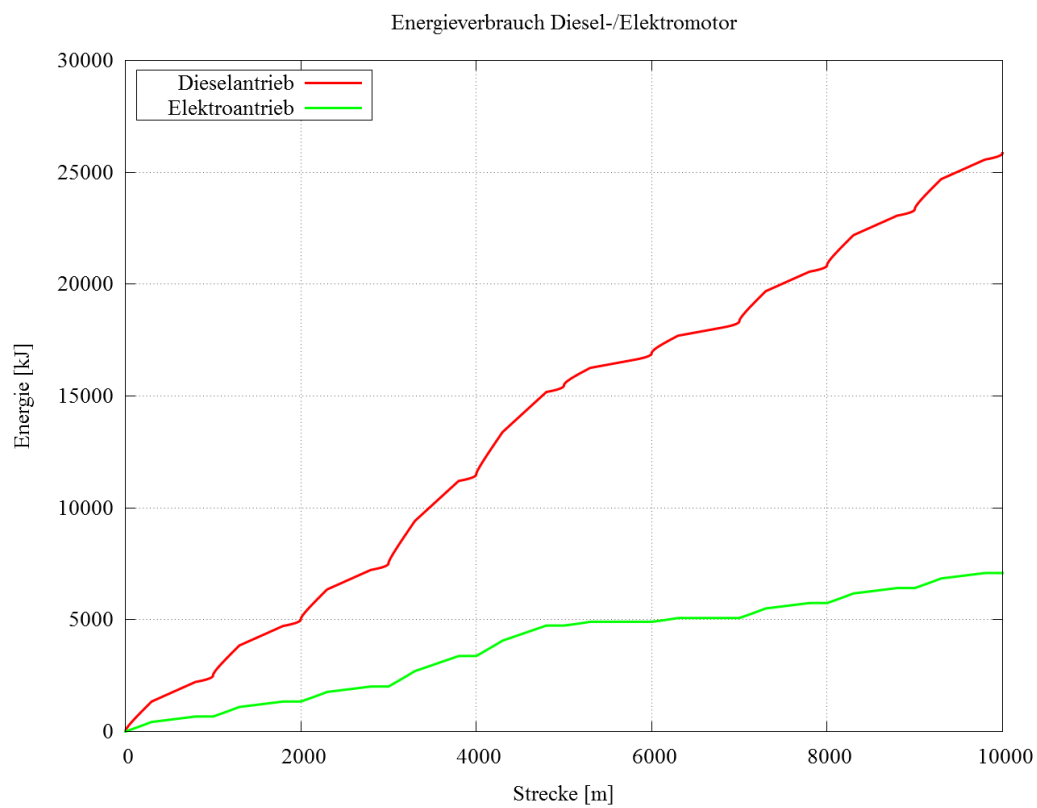


Abbildung 9: Energieverbrauch bei 60 kW und 60 km/h

11.2 Anhang: Programmcode

11.2.1 Antriebsvergleich.lpr

```

program Antriebsvergleich;

{$mode objfpc}{$H+}

uses {$IFDEF UNIX} {$IFDEF UseCThreads}
  cthreads, {$ENDIF} {$ENDIF}
  Classes,
  Crt,
  Sysutils,
  uMain { you can add units after this };

{$R *.res}

begin
  initialisieren(eta,p,v,filename,modus);
  menu(vergleichswitch);
end. {Antriebsvergleich}

```

11.2.2 uMain.pas

```

unit uMain;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Crt;

const
  A = 2;           {Fahrzeugquerschnitt [m^2]}
  cw = 0.33;       {cw Wert}
  Pl = 1.1885;     {spezifische Luftmasse bei 1bar und 20°C [kg/m^3]}
  mue = 0.015;     {Reibungskoeffizient}
  mf = 1500;       {Fahrzeugmasse [kg]}
  g = 9.81;        {Erdbeschleunigung [m/s^2]}
  deltat = 0.0001; {Zeitinkrement}
  maxmess = 3000;  {obere Array Grenze mit Sicherheit; gilt für:
deltat>=0.0001      v>=40km/h und ktewert>=5000}
  ktewert = 5000;  {nur jeder "ktewert" wird in die Datei geschrieben
                    10000=messwert/sek, 5000=messwert/0.5sek}
  eps = 0.00001;  {Epsilon für Real Genauigkeit}

  {'oeffentliche' Variablen}
var
  eta, p, v: double;
  modus,vergleichswitch:boolean;
  filename:string;

procedure initialisieren(var eta,p,v:double;var filename:string;
                        var modus:boolean);

```

```

procedure warten;
procedure titel;
procedure menu(var vergleichswitch:boolean);
procedure parameter_listen;
procedure parameter_aendern;
procedure modus_waehlen(var abort:boolean);
procedure leistung_waehlen(var p:double; var abort:boolean);
procedure geschwindigkeit_waehlen(var v:double);
procedure daten_ueberschreiben(var x,y:array of double; var j:integer;
                             filename:string);
procedure daten_schreiben(var filename:string);
procedure variablen_berechnen(var Beschleunigung,Verzoegerung,
                             Luftwiderstand,Rollreibung, Hangabtriebskraft, idle:double);
procedure plotmenu(var multiplot:boolean);
procedure diagramm_plotten;
procedure ergebnis_ausgeben(Gesamtenergie,Gesamtzeit,Gesamtstrecke:double;
                             modus:boolean);
procedure vergleich(var modus:boolean; var eta:double;
                    var filename:string);
procedure Energie_berechnen(var Gesamtenergie,Gesamtzeit,Gesamtstrecke:
                             double; var x,y:Array of double;
                             var j,k:integer;
                             var vergleichswitch:boolean);
procedure beschleunigen_bergauf(var Gesamtenergie,Gesamtzeit,Gesamtstrecke:
                                double; var x,y:Array of double;
                                var j,k:integer);
procedure konstant_bergauf(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
                             double; var x,y:Array of double;
                             var j,k:integer);
procedure bremsen_bergauf(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
                             double;var x,y:Array of double; var j,k:integer);
procedure beschleunigen_gerade(var Gesamtenergie, Gesamtzeit,
                                Gesamtstrecke:double; var x,y:Array of double;
                                var j,k:integer);
procedure konstant_gerade(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
                             double; var x,y:Array of double; var j,k:integer);
procedure beschleunigen_bergab(var Gesamtenergie, Gesamtzeit,
                                Gesamtstrecke:double; var x,y:Array of double;
                                var j,k:integer);
procedure konstant_bergab(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
                             double; var x,y:Array of double; var j,k:integer);
procedure bremsen(var Gesamtenergie, Gesamtzeit, Gesamtstrecke: double;
                    var x,y:Array of double; var j,k:integer);

```

implementation

```

{'private' Variablen}
var
    Beschleunigung,Verzoegerung,Luftwiderstand,Rollreibung,Hangabtriebskraft,
    Gesamtenergie, Gesamtzeit, Gesamtstrecke, idle:double;
    abort, multiplot:boolean;
    j,k: integer;                                {modus: false=Diesel/true=Elektro}
    x,y: array [0..maxmess] of double;

{#####}
{
    Initialisierung
}
{#####}

procedure initialisieren(var eta,p,v:double;var filename:string;
                        var modus:boolean);

```

```

begin
    eta:=0.32; p:=50000; v:=13.89;
    filename:='Dieseldata.dat'; modus:=false;
end; {initialisieren}

{#####}
{
    warten
}
{#####}

procedure warten;

begin
    writeln;
    write('<Enter> zum fortfahren...');
    readkey; // da readlns nicht beachtet wurden
end; {warten}

{#####}
{
    Titel
}
{#####}

procedure titel;

begin
    writeln('*****
    *',
        '*****');
    writeln('*           Antriebsvergleich: Diesel- gegen
    Elektroantrieb',
        '           *');

    writeln('*****
    *',
        '*****');
end; {titel}

{#####}
{
    Menue
}
{#####}

procedure menu(var vergleichswitch:boolean);

var
    auswahl:byte;

begin
    try
        clrscr;
        titel;
        parameter_listen;
        writeln('Was moechten Sie tun?');
        writeln('1: Simulation starten      2: automatischer Vergleich');
        writeln('3: Parameter aendern      4: Diagramm plotten');
        writeln('5: Programm beenden');
        writeln();
        write('Eingabe: ');
        read(auswahl);
    
```

```

case auswahl of
  1: begin
    vergleichswitch:=false; {-> Rueckkehr ins Menue nach
Energie_berechnen}
    Energie_berechnen(Gesamtenergie,Gesamtzeit,Gesamtstrecke,x,y,j,k,
                      vergleichswitch);

    end; {1}
  2: begin
    vergleichswitch:=true; {-> Energie_berechnen laeuft vollstaendig
durch}
    vergleich(modus,eta,filename);
    menu(vergleichswitch);
    end; {2}
  3: parameter_aendern;
  4: plotmenu(multiplot);
  5: exit;
else
  begin
    writeln('Ungueltige Wahl! Bitte Zahl zwischen 1 und 5 eingeben.');
```

warten;

```
    menu(vergleichswitch);
    end; {else}
end; {case}
except
  on e: exception do
    begin
      writeln('Ungueltige Wahl! Bitte Zahl zwischen 1 und 5 eingeben.');
```

writeln('Error (menu): ' + e.message);

```
      warten;
      initialisieren(eta,p,v,filename,modus);
      menu(vergleichswitch);
      end; {e: exception}
    end; {try except}
end; {menu}

{#####}
{                                     Parameter listen                                     }
{#####}

procedure parameter_listen;

begin
  writeln('Die aktuell eingestellten Parameter lauten: ');
  write('Modus: ');
  if modus = true then
    write('Elektromodus')
  else write('Dieselmodus');
  writeln('   Leistungsklasse: ',(p/1000):0:0,'kW',
          '   Sollgeschwindigkeit: ',(v*3.6):0:0,'km/h');
  writeln;
end; {parameter_listen}

{#####}
{                                     Parameter aendern                                     }
{#####}

procedure parameter_aendern;

begin
  modus_waehlen(abort);
```

```

if abort then menu(vergleichswitch)
else
begin
leistung_waehlen(p,abort);
if abort then menu(vergleichswitch)
else
begin
geschwindigkeit_waehlen(v);
menu(vergleichswitch);
end; {inneres abort}
end; {aeusseres abort}
end; {parameter_aendern}

{#####}
{##### Modus waehlen #####}
{#####}

procedure modus_waehlen(var abort:boolean);

var
auswahl:byte;

begin
try
abort:=false;
clrscr;
writeln('1: Dieselmodus 2: Elektromodus');
writeln('3: zurueck zum Menue');
writeln();
write('Bitte Modus waehlen: ');
read(auswahl);
case auswahl of
1: begin modus:=false; eta:=0.32; filename:='Dieselraten.dat' end;
2: begin modus:=true; eta:=0.68; filename:='Elektrodaten.dat' end;
3: abort:=true
else
begin
writeln;
writeln('Unguelte Wahl! Bitte 1, 2 oder 3 eingeben. ');
warten;
modus_waehlen(abort);
end; {else}
end; {case}
except
on e: exception do
begin
writeln;
writeln('Unguelte Wahl! Bitte 1, 2 oder 3 eingeben. ');
writeln('Error (modus_waehlen): ' + e.message);
warten;
modus_waehlen(abort);
end; {e: exception}
end; {try except}
end; {modus_waehlen}

{#####}
{##### Leistung waehlen #####}
{#####}

procedure leistung_waehlen(var p:double; var abort:boolean);

```

```

var
    auswahl:byte;

begin
    try
        clrscr;
        writeln('1: 30kW   2: 50kW   3: 60kW');
        writeln('4: zurueck zum Menue');
        writeln();
        Write('Bitte Leistungsklasse waehlen: ');
        read(auswahl);
        case auswahl of
            1: p:=30000;
            2: p:=50000;
            3: p:=60000;
            4: abort:=true
        else
            begin
                writeln;
                writeln('Unguelte Wahl! Bitte Zahl zwischen 1 und 4 eingeben. ');
                warten;
                leistung_waehlen(p,abort);
            end; {else}
        end; {case}
    except
        on e: exception do
            begin
                writeln;
                writeln('Unguelte Wahl! Bitte Zahl zwischen 1 und 4 eingeben. ');
                Writeln('Error (leistung_waehlen): ' + e.message);
                warten;
                leistung_waehlen(p,abort);
            end; {e: exception}
        end; {try except}
    end; {leistung_waehlen}

{#####}
{                               Geschwindigkeit waehlen                               }
{#####}

procedure geschwindigkeit_waehlen(var v:double);

var
    auswahl:byte;

begin
    try
        clrscr;
        writeln('1: 40km/h   2: 50km/h   3: 60km/h');
        writeln('4: zurueck zum Menue');
        writeln();
        write('Bitte Sollgeschwindigkeit waehlen: ');
        read(auswahl);
        case auswahl of
            1:
                v:=11.11;
            2:
                v:=13.89;
            3:
                if p=30000 then

```

```

begin
  clrscr;
  writeln('Die gewaehlte Motorleistung (', (p/1000):0:0, 'kW) ist fuer',
    ' diese Geschwindigkeit zu gering. ');
  writeln('Bitte waehlen sie 40km/h oder 50km/h');
  warten;
  geschwindigkeit_waehlen(v);
end
else
  v:=16.67;
4:
  writeln();
else
  begin
    writeln;
    writeln('Unguelte Wahl! Bitte Zahl zwischen 1 und 4 eingeben. ');
    warten;
    geschwindigkeit_waehlen(v);
  end; {else}
end; {case}
except
  on e: exception do
    begin
      writeln;
      writeln('Unguelte Wahl! Bitte Zahl zwischen 1 und 4 eingeben. ');
      writeln('Error (geschwindigkeit_waehlen): ' + e.message);
      warten;
      geschwindigkeit_waehlen(v);
    end; {e: exception}
  end; {try except}
end; {geschwindigkeit_waehlen}

{#####}
{                                     Daten ueberschreiben                                     }
{#####}

procedure daten_ueberschreiben(var x,y:array of double; var j:integer;
                               filename:string);

var
  i:integer;
  F:Textfile;

begin
  try
    i:=1; k:=ktewert;
    AssignFile(F, filename);
    Rewrite(F);
    while i<j do
      begin
        write(F, x[i], ' ');
        writeln(F, (y[i]/1000)); //Umrechnung von J in kJ
        i:=i+1;
      end;
    CloseFile(F);
    writeln();
    writeln('Datei wurde als "', filename, '" im Programmverzeichnis',
      ' gespeichert. ');
    warten;
  except
    on e: exception do

```

```

begin
    writeln('Error (daten_ueberschreiben): ' + e.message);
    warten;
    daten_ueberschreiben(x,y,j,filename);
end; {e: exception}
end; {try except}
end; {daten_ueberschreiben}

{#####}
{                                     Daten schreiben                                     }
{#####}

procedure daten_schreiben(var filename:string);

var
    F:Textfile;
    auswahl:integer;

begin
    try
        AssignFile(F, filename);
        if FileExists(filename) then
            begin
                writeln();
                writeln('Datei existiert bereits. Ueberschreiben?  1: Ja   2: Nein');
                write('Eingabe: ');
                readln(auswahl);
                case auswahl of
                    1:
                        daten_ueberschreiben(x,y,j,filename);
                    2:
                        begin
                            writeln();
                            writeln('Schreibvorgang abgebrochen');
                            warten;
                        end; {2}
                    else
                        begin
                            writeln();
                            writeln('Ungueltige Wahl! Bitte 1 fuer Ja oder 2 fuer Nein
                                eingeben. ');
                            warten;
                            writeln();
                            daten_schreiben(filename);
                        end; {else}
                end; {case}
            end {if FileExists(filename)}
        else
            daten_ueberschreiben(x,y,j,filename);
        except
            on e: exception do
                begin
                    writeln('Ungueltige Wahl! Bitte 1 fuer Ja oder 2 fuer Nein
                        eingeben. ');
                    writeln('Error (daten_schreiben): ' + e.message);
                    warten;
                    daten_schreiben(filename);
                end; {e: exception}
            end; {try except}
        end; {daten_schreiben}
    end;
end;

```



```

{#####}
{
    Variablen berechnen
}
{#####}

```

```

procedure variablen_berechnen(var Beschleunigung,
Verzoegerung, Luftwiderstand,
                                Rollreibung, Hangabtriebskraft,
idle:double);

begin
    Beschleunigung := (v*v)/600;
    Verzoegerung   := (v*v)/400;
    Luftwiderstand := 0.5*A*Cw*Pl; //nur der konstante Faktor, ohne v^2 !
    Rollreibung    := mue*mf*g; //cos(a)=0.9992≈1, deshalb vernachlaessigt
    Hangabtriebskraft := mf*g*(0.04);
    idle := p*0.2*deltat; //Leerlaufenergie
end; {variablen_berechnen}

```

```

{#####}
{
    Plotmenue
}
{#####}

```

```

procedure plotmenu(var multiplot:boolean);

```

```

var
    auswahl:byte;
    mode:string;

begin
    try
        clrscr;
        if modus then mode:='(Elektromodus)';
        else mode:='(Dieselmodus)';
        writeln('1: Einzeldiagramm plotten  2: Vergleichsdiagramm plotten  ',
                '3: zurueck zum Menue');
        writeln(' ',mode);
        writeln();
        write('Bitte Plotmodus waehlen: ');
        readln(auswahl);
        case auswahl of
            1:
                begin
                    multiplot:=false;
                    diagramm_plotten;
                end; {1}
            2:
                begin
                    multiplot:=true;
                    diagramm_plotten;
                end; {2}
            3: menu(vergleichswitch)
        else
            begin
                writeln('Ungueltige Wahl! Bitte 1, 2 oder 3 eingeben. ');
                warten;
                plotmenu(multiplot);
            end; {else}
        end; {case}
    except
        on e: exception do

```

```

begin
    writeln('Ungueltige Wahl! Bitte 1, 2 oder 3 eingeben.');
```

```

    writeln('Error (plotmenu): ' + e.message);
    warten;
    plotmenu(multiplot);
end; {e: exception}
end; {try except}
end; {plotmenu}

{#####}
{                                Diagramm plotten                                }
{#####}

procedure diagramm_plotten;

var
    diagramm:string;

begin
    try
        if multiplot=false then
            begin
                if modus = true then
                    if fileexists(filename) then
                        begin
                            diagramm := 'skripte/oeffneE.bat';
                            SysUtils.ExecuteProcess('gnuplot/bin/wgnuplot.exe',
                                                    ['skripte/Elektroplot.plt']);
                        end {fileexists(filename)}
                    else
                        begin
                            writeln('Die benoetigte Datendatei "',filename,'" konnte nicht',
                                    ' gefunden werden!');
                            writeln('Kehre ins Hauptmenue zurueck...');
                            warten;
                            menu(vergleichswitch);
                        end {else}
                    else {modus = false}
                        if fileexists(filename) then
                            begin
                                diagramm := 'skripte/oeffneD.bat';
                                SysUtils.ExecuteProcess('gnuplot/bin/wgnuplot.exe',
                                                        ['skripte/Dieselplot.plt']);
                            end {fileexists(filename)}
                        else
                            begin
                                writeln('Die benoetigte Datendatei "',filename,'" konnte nicht',
                                        ' gefunden werden!');
                                writeln('Kehre ins Hauptmenue zurueck...');
                                warten;
                                menu(vergleichswitch);
                            end; {else}
                        end {multiplot=false}
                    else {multiplot=true}
                        if fileexists('Dieselraten.dat') or fileexists('Elektrodaten.dat') then
                            begin
                                diagramm := 'skripte/oeffneM.bat';
                                SysUtils.ExecuteProcess('gnuplot/bin/wgnuplot.exe',
                                                        ['skripte/Multiplot.plt']);
                            end {fileexists('Dieselraten.dat') or fileexists('Elektrodaten.dat')}
                        else

```

```

begin
    writeln('Es wurde keine Datendatei gefunden!');
    writeln('Kehre ins Hauptmenue zurueck...');
    warten;
    menu(vergleichswitch);
end; {else}
SysUtils.ExecuteProcess(diagramm, []);
menu(vergleichswitch);
except
    on e: exception do
        begin
            writeln('Error (diagramm_plotten): ' + e.message);
            warten;
            plotmenu(multiplot);
        end; {e: exception}
    end; {try except}
end; {diagramm_plotten}

{#####}
{
    Ergebnis ausgeben
}
{#####}

procedure ergebnis_ausgeben(Gesamtenergie, Gesamtzeit, Gesamtstrecke:double;
                             modus:boolean);

begin
    writeln();
    writeln('-----');
    write('Modus: ':15);
    if modus = true then writeln('Elektromodus':12) else
writeln('Dieselmodus':12);
    writeln('Gesamtenergie: ', (Gesamtenergie / 1000):9:3, ' kJ':3);
    writeln('Gesamtstrecke: ', Gesamtstrecke:9:0, ' m':3);
    writeln('Gesamtzeit: ':15, Gesamtzeit:9:0, ' s':3);
    writeln('-----');
end; {ergebnis_ausgeben}

{#####}
{
    Vergleich
}
{#####}

procedure vergleich(var modus:boolean; var eta:double; var
filename:string);

var
    GesamtenergieAlt, GesamtenergieDiff:double;
    MAlt, MNeu:string;

begin
    GesamtenergieAlt:=0; GesamtenergieDiff:=0;
    Energie_berechnen(Gesamtenergie, Gesamtzeit, Gesamtstrecke, x, y, j, k,
                      vergleichswitch);
    GesamtenergieAlt:=Gesamtenergie; //letzte Werte merken
    if modus = true then
        begin modus:=false; eta:=0.32; filename:='Dieseldata.dat';
            MAlt:='Energieverbrauch Elektro: '; MNeu:='Energieverbrauch Diesel: ';
        end
    else {modus = false} //Modus wechseln
        begin modus:=true; eta:=0.68; filename:='Elektrodata.dat';

```

```

MAlt:='Energieverbrauch Diesel: '; MNeu:='Energieverbrauch Elektro: ';
end;
Energie_berechnen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k,
    vergleichswitch);
GesamtenergieDiff:= abs(GesamtenergieAlt-Gesamtenergie);

writeln();
writeln('-----');
writeln(MAlt:26, (GesamtenergieAlt/1000):9:3, ' kJ':3);
writeln(MNeu:26, (Gesamtenergie/1000):9:3, ' kJ':3);
writeln('-----':38);
writeln('GesamtenergieDifferenz: ':26,(GesamtenergieDiff/1000):9:3,'
kJ':3);
writeln;
writeln('Gesamtstrecke: ':26, Gesamtstrecke:9:0, ' m':3);
writeln('Gesamtzeit: ':26, Gesamtzeit:9:0, ' s':3);
writeln('-----');
warten;
end; {vergleich}

{#####}
{
    Energie berechnen
}
{#####}

procedure Energie_berechnen(var
Gesamtenergie,Gesamtzeit,Gesamtstrecke:double;
    var x,y:Array of double; var j,k:integer;
    var vergleichswitch:boolean);

begin
    Gesamtenergie:=0; Gesamtzeit:=0; Gesamtstrecke:=0; k:=ktewert;
    j:=0; x[j]:=0; y[j]:=0; j:=j+1;
    variablen_berechnen(Beschleunigung, Verzoeigerung, Luftwiderstand,
Rollreibung,
        Hangabtriebskraft, idle);
    beschleunigen_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 1km
    beschleunigen_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 2km
    beschleunigen_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_gerade(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 3km
    beschleunigen_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    if abs(v-16.67)<eps then {real Genauigkeit macht sonst Probleme (eig.
        v=16.67)}
        bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k)
    else
        bremsen_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 4km
    beschleunigen_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    if abs(v-16.67)<eps then
        bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k)
    else
        bremsen_bergauf(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 5km
    beschleunigen_bergab(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    konstant_bergab(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);
    bremsen(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k); // 6km
    beschleunigen_bergab(Gesamtenergie, Gesamtzeit, Gesamtstrecke,x,y,j,k);

```



```

    if k=ktewert then
    begin
        x[j]:=Gesamtstrecke+Strecke;
        y[j]:=Gesamtenergie+E_Bereich;
        j:=j+1;
        k:=0;
    end; {k=ktewert}
    k:=k+1;
end; {end if modus=true}
Gesamtstrecke := Gesamtstrecke + Strecke;
Gesamtzeit := Gesamtzeit + t;
Gesamtenergie := Gesamtenergie + E_Bereich;
end; {beschleunigen_bergauf}

{#####}
{               konstante Fahrt bergauf               }
{#####}

procedure konstant_bergauf(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
double;
                           var x,y:Array of double; var j,k:integer);

var
    Strecke, t, E_Bereich, Energie: double;

begin
    E_Bereich := 0;
    Strecke := 0;
    Energie := 0;
    t := 0;
    if modus=false then
        while Strecke <= 500 do
            begin
                Strecke := v * t;
                Energie := (((Luftwiderstand*v*v+Rollreibung+Hangabtriebskraft)*
                            (v*deltat))/eta)+idle;
                t := t + deltat;
                E_Bereich := E_Bereich + Energie;
                if k=ktewert then
                    begin
                        x[j]:=Gesamtstrecke+Strecke;
                        y[j]:=Gesamtenergie+E_Bereich;
                        j:=j+1;
                        k:=0;
                    end; {k=ktewert}
                k:=k+1;
            end {end if modus=false}
        else {modus=true}
            while Strecke <= 500 do
                begin
                    Strecke := v * t;
                    Energie := (((Luftwiderstand*v*v+Rollreibung+Hangabtriebskraft)*
                                (v*deltat))/eta);
                    t := t + deltat;
                    E_Bereich := E_Bereich + Energie;
                    if k=ktewert then
                        begin
                            x[j]:=Gesamtstrecke+Strecke;
                            y[j]:=Gesamtenergie+E_Bereich;
                            j:=j+1;
                            k:=0;

```

```

        end; {k=ktewert}
        k:=k+1;
    end; {end if modus=true}
    Gesamtstrecke := Gesamtstrecke + Strecke;
    Gesamtzeit := Gesamtzeit + t;
    Gesamtenergie := Gesamtenergie + E_Bereich;
end; {konstant_bergauf}

#####
{ bremsen bergauf }
#####

procedure bremsen_bergauf(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
double;
                        var x,y:Array of double; var
j,k:integer);
var
    Strecke, t, E_Bereich, Energie: double;
begin
    E_Bereich := 0;
    Strecke := 0;
    Energie := 0;
    t := 0;
    if modus=false then
        while abs(Strecke-200)>=eps do {hier wird eig. Strecke <= 199.99999
                                        geprüft}

            begin
                Strecke := -0.5*Verzoegerung*t*t + v*t;
                Energie := ((Luftwiderstand*(v-Verzoegerung*t)*(v-Verzoegerung*t)
                            +Rollreibung+Hangabtriebskraft)*(v-Verzoegerung*t)
                            *deltat))/eta)+idle;
                t := t + deltat;
                E_Bereich := E_Bereich + Energie;
                if k=ktewert then
                    begin
                        x[j] := Gesamtstrecke+Strecke;
                        y[j] := Gesamtenergie+E_Bereich;
                        j:=j+1;
                        k:=0;
                    end; {k=ktewert}
                    k:=k+1;
                end {end if modus=false}
            else {modus=true}
                while abs(Strecke-200)>=eps do {hier wird eig. Strecke <= 199.99999
                                                geprüft}

                    begin
                        Strecke := -0.5*Verzoegerung*t*t + v*t;
                        Energie := ((Luftwiderstand*(v-Verzoegerung*t)*(v-Verzoegerung*t)
                                    +Rollreibung+Hangabtriebskraft)*(v-Verzoegerung*t)
                                    *deltat))/eta);
                        t := t + deltat;
                        E_Bereich := E_Bereich + Energie;
                        if k=ktewert then
                            begin
                                x[j] := Gesamtstrecke+Strecke;
                                y[j] := Gesamtenergie+E_Bereich;
                                j:=j+1;
                                k:=0;
                            end; {k=ktewert}
                            k:=k+1;
                        end; {end if modus=true}
                    end
                end
            end
        end
    end
end

```

```

Gesamtstrecke := Gesamtstrecke + Strecke;
Gesamtzeit := Gesamtzeit + t;
Gesamtenergie := Gesamtenergie + E_Bereich;
end; {bremsen_bergauf}

{#####}
{      beschleunigen gerade Strecke      }
{#####}

procedure beschleunigen_gerade(var Gesamtenergie,Gesamtzeit,Gesamtstrecke:
                                double;var x,y:Array of double; var
j,k:integer);

var
    Strecke, t, E_Bereich, Energie: double;

begin
    E_Bereich := 0;
    Strecke := 0;
    Energie := 0;
    t := 0;
    if modus=false then
        while Strecke <= 300 do
            begin
                Strecke := 0.5 * Beschleunigung * t * t;
                Energie := ((mf*Beschleunigung+Luftwiderstand*(Beschleunigung*t)*
                    (Beschleunigung*t)+Rollreibung)*((Beschleunigung*t)
                    *deltat))/eta+idle;
                t := t + deltat;
                E_Bereich := E_Bereich + Energie;
                if k=ktewert then
                    begin
                        x[j]:=Gesamtstrecke+Strecke;
                        y[j]:=Gesamtenergie+E_Bereich;
                        j:=j+1;
                        k:=0;
                    end; {k=ktewert}
                    k:=k+1;
                end {end if modus=false}
            else {modus=true}
                while Strecke <= 300 do
                    begin
                        Strecke := 0.5 * Beschleunigung * t * t;
                        Energie := ((mf*Beschleunigung+Luftwiderstand*(Beschleunigung*t)*
                            (Beschleunigung*t)+Rollreibung)*((Beschleunigung*t)
                            *deltat))/eta;
                        t := t + deltat;
                        E_Bereich := E_Bereich + Energie;
                        if k=ktewert then
                            begin
                                x[j]:=Gesamtstrecke+Strecke;
                                y[j]:=Gesamtenergie+E_Bereich;
                                j:=j+1;
                                k:=0;
                            end; {k=ktewert}
                                k:=k+1;
                        end; {end if modus=true}
                    Gesamtstrecke := Gesamtstrecke + Strecke;
                    Gesamtzeit := Gesamtzeit + t;
                    Gesamtenergie := Gesamtenergie + E_Bereich;

```



```

end; {beschleunigen_gerade}

#####
{
    konstante Fahrt gerade Strecke
}
#####

procedure konstant_gerade(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
double;
                        var x,y:Array of double; var j,k:integer);

var
    Strecke, t, E_Bereich, Energie: double;

begin
    E_Bereich := 0;
    Strecke := 0;
    Energie := 0;
    t := 0;
    if modus=false then
        while Strecke <= 500 do
            begin
                Strecke := v * t;
                Energie := (((Luftwiderstand*v*v+Rollreibung)*(v*deltat))/eta)+idle;
                t := t + deltat;
                E_Bereich := E_Bereich + Energie;
                if k=ktewert then
                    begin
                        x[j]:=Gesamtstrecke+Strecke;
                        y[j]:=Gesamtenergie+E_Bereich;
                        j:=j+1;
                        k:=0;
                    end; {k=ktewert}
                    k:=k+1;
                end {end if modus=false}
            else {modus=true}
                while Strecke <= 500 do
                    begin
                        Strecke := v * t;
                        Energie := (((Luftwiderstand*v*v+Rollreibung)*(v*deltat))/eta);
                        t := t + deltat;
                        E_Bereich := E_Bereich + Energie;
                        if k=ktewert then
                            begin
                                x[j]:=Gesamtstrecke+Strecke;
                                y[j]:=Gesamtenergie+E_Bereich;
                                j:=j+1;
                                k:=0;
                            end; {k=ktewert}
                            k:=k+1;
                        end; {end if modus=true}
                    Gesamtstrecke := Gesamtstrecke + Strecke;
                    Gesamtzeit := Gesamtzeit + t;
                    Gesamtenergie := Gesamtenergie + E_Bereich;
                end; {konstant_gerade}

#####
{
    beschleunigen bergab
}
#####

```

```

procedure beschleunigen_bergab(var Gesamtenergie, Gesamtzeit,
Gesamtstrecke:
                                double; var x,y:array of double; var
j,k:integer);

var
    Strecke, t, E_Bereich, E_Bereich2, Energie: double;

begin {Ebene 0 auf}
    E_Bereich := 0;
    E_Bereich2 := 0;
    Strecke := 0;
    Energie := 0;
    t := 0;
    if modus=false then {Ebene 1 auf}
    begin
        while Strecke <= 300 do {Ebene 2 auf}
        begin
            if v=11.11 then {Ebene 3 auf}
            begin
                Strecke := 0.5 * Beschleunigung * t * t;
                t := t + deltat;
                E_Bereich := E_Bereich + idle;
                if k=ktewert then {Ebene 4 auf}
                begin
                    x[j]:=Gesamtstrecke+Strecke;
                    y[j]:=Gesamtenergie+E_Bereich;
                    j:=j+1;
                    k:=0;
                end; {if k=ktewert, Ebene 4 zu}
                k:=k+1;
            end {if v=11.11, Ebene 3 zu}
            else {else if v=11.11, Ebene 3 auf}
            begin
                Strecke := 0.5 * Beschleunigung * t * t;
                Energie := ((mf*Beschleunigung+Luftwiderstand*(Beschleunigung*t)*
                    (Beschleunigung*t)+Rollreibung-Hangabtriebskraft)*
                    ((Beschleunigung*t)*deltat))/eta)+idle;
                t := t + deltat;
                E_Bereich := E_Bereich + Energie;
                E_Bereich2 := E_Bereich2 + idle;
                if k=ktewert then {Ebene 4 auf}
                begin
                    x[j] := Gesamtstrecke+Strecke;
                    if E_Bereich>0 then {Ebene 5 auf}
                    y[j] := Gesamtenergie+E_Bereich
                else
                    y[j] := Gesamtenergie+E_Bereich2; {Ebene 5 zu}
                j:=j+1;
                k:=0;
            end; {Ebene 4 zu}
                k:=k+1;
            end; {else if v=11.11, Ebene 3 zu}
        end; {Strecke <= 300}
    end {modus=false}
    else { modus=true, Ebene 1 auf}
    begin
        while Strecke <= 300 do {Ebene 2 auf}
        begin
            if v=11.11 then //Energie bleibt konstant {Ebene 3 auf}
            begin
                Strecke := 0.5 * Beschleunigung * t * t;

```

```

t := t + deltat;
if k=ktewert then {Ebene 4 auf}
begin
  x[j]:=Gesamtstrecke+Strecke;
  y[j]:=Gesamtenergie;
  j:=j+1;
  k:=0;
end; {Ebene 4 zu}
k:=k+1;
end {if v=11.11, Ebene 3 zu}
else {else if v=11.11, Ebene 3 auf}
begin
  Strecke := 0.5 * Beschleunigung * t * t;
  Energie := ((mf*Beschleunigung+Luftwiderstand*(Beschleunigung*t)*
              (Beschleunigung*t)+Rollreibung-Hangabtriebskraft)*
              ((Beschleunigung*t)*deltat))/eta);
  t := t + deltat;
  E_Bereich := E_Bereich + Energie;
  //E_Bereich2 steht auf 0 und aendert sich hier nicht
  if k=ktewert then {Ebene 4 auf}
  begin
    x[j] := Gesamtstrecke+Strecke;
    if E_Bereich>0 then {Ebene 5 auf}
      y[j] := Gesamtenergie+E_Bereich
    else
      y[j] := Gesamtenergie; {Ebene 5 zu}
      j:=j+1;
      k:=0;
    end; {if k=ktewert, Ebene 4 zu}
    k:=k+1;
  end; {else if v=11.11, Ebene 3 zu}
end; {while Strecke<=300, Ebene 2 zu}
end; {end if modus=true, Ebene 1 auf}
Gesamtstrecke := Gesamtstrecke + Strecke;
Gesamtzeit := Gesamtzeit + t;
if E_Bereich>0 then
  Gesamtenergie := Gesamtenergie + E_Bereich
else
  Gesamtenergie := Gesamtenergie + E_Bereich2;
end; {beschleunigen_bergab, Ebene 0 zu}

#####
{
  konstante Fahrt bergab
}
#####

procedure konstant_bergab(var Gesamtenergie, Gesamtzeit, Gesamtstrecke:
double;
                        var x,y:array of double; var j,k:integer);

var
  Strecke, t, E_Bereich: double;

begin
  E_Bereich := 0;
  Strecke := 0;
  t := 0;
  if modus=false then
    while Strecke <= 500 do
      begin
        Strecke := v * t;
        t := t + deltat;

```

```

    E_Bereich := E_Bereich + idle;
    if k=ktewert then
        begin
            x[j]:=Gesamtstrecke+Strecke;
            y[j]:=Gesamtenergie+E_Bereich;
            j:=j+1;
            k:=0;
        end; {k=ktewert}
        k:=k+1;
    end {end if modus=false}
else {modus=true}
    while Strecke <= 500 do
        begin
            Strecke := v * t;
            t := t + deltat;
            if k=ktewert then
                begin
                    x[j]:=Gesamtstrecke+Strecke;
                    y[j]:=Gesamtenergie;
                    j:=j+1;
                    k:=0;
                end; {k=ktewert}
                k:=k+1;
            end; {end if modus=true}
            Gesamtstrecke := Gesamtstrecke + Strecke;
            Gesamtzeit := Gesamtzeit + t;
            Gesamtenergie := Gesamtenergie + E_Bereich
        end; {konstant_bergab}

{#####}
{                               bremsen                               }
{#####}

procedure bremsen(var Gesamtenergie, Gesamtzeit, Gesamtstrecke: double;
                  var x,y:Array of double; var j,k:integer);

var
    Strecke, t, E_Bereich: double;

begin
    E_Bereich := 0;
    Strecke := 0;
    t := 0;
    if modus=false then
        while abs(Strecke-200)>=eps do {hier wird eig. Strecke <= 199.99999
geprüft}
            begin
                Strecke := v*t - 0.5*Verzoegerung*t*t;
                t := t + deltat;
                E_Bereich := E_Bereich + idle;
                if k=ktewert then
                    begin
                        x[j]:=Gesamtstrecke+Strecke;
                        y[j]:=Gesamtenergie+E_Bereich;
                        j:=j+1;
                        k:=0;
                    end; {k=ktewert}
                    k:=k+1;
                end {modus=false}
            else {modus=true}
                while abs(Strecke-200)>=eps do

```

```

begin
  Strecke := v*t - 0.5*Verzoegerung*t*t;
  t := t + deltat;
  if k=ktewert then
    begin
      x[j]:=Gesamtstrecke+Strecke;
      y[j]:=Gesamtenergie;
      j:=j+1;
      k:=0;
    end; {k=ktewert}
    k:=k+1;
  end; {modus=true}
  Gesamtstrecke := Gesamtstrecke + Strecke;
  Gesamtzeit := Gesamtzeit + t;
  Gesamtenergie := Gesamtenergie + E_Bereich
end; {bremsen}

end.{uMain}

```

11.2.3 Dieselplot.plt

```

# Dieses Gnuplot Script plottet die Daten aus der Datei "Dieselraten.dat"
# Diese Datei hei "Dieselplot.plt"
plot 'Dieselraten.dat'
ymax = GPVAL_DATA_Y_MAX
set terminal pngcairo size 1280,1024 font "Times New Roman,20"
set output 'diagramme/Dieseldiagramm.png'
set title "Energieverbrauch Dieselmotor"
set xlabel "Strecke [m]"
set ylabel "Energie [kJ]"
set grid
set key t l
set xrange [0:10000]
set yrange [0:ymax+100]
plot 'Dieselraten.dat' w lines t "Dieselantrieb"

```

11.2.4 Elektroplot.plt

```

# Dieses Gnuplot Script plottet die Daten aus der Datei "Elektrodaten.dat"
# Diese Datei hei "Elektroplot.plt"
plot 'Elektrodaten.dat'
ymax = GPVAL_DATA_Y_MAX
set terminal pngcairo size 1280,1024 font "Times New Roman,20"
set output 'diagramme/Elektrodiagramm.png'
set title "Energieverbrauch Elektromotor"
set xlabel "Strecke [m]"
set ylabel "Energie [kJ]"
set grid
set key t l
set xrange [0:10000]
set yrange [0:ymax+100]
plot 'Elektrodaten.dat' w lines t "Elektroantrieb"

```

11.2.5 Multiplot.plt

```
# Dieses Gnuplot Script plottet die Daten in den Dateien "Diesel- und
Elektrodaten.dat"
# Diese Datei hei "Multiplot.plt"
set terminal pngcairo size 1280,1024 font "Times New Roman,20"
set output 'diagramme/Vergleichsdiagramm.png'
set title "Energieverbrauch Diesel-/Elektromotor"
set xlabel "Strecke [m]"
set ylabel "Energie [kJ]"
set grid
set key box t l
set xrange [0:10000]
set yrange [0:30000]
plot 'Dieselraten.dat' w lines t "Dieselantrieb" , 'Elektrodaten.dat' w
lines t "Elektroantrieb"
```

11.3 Anhang: verkürzte Ausgabedatei Dieseldaten.dat (50 kW und 50 km/h)

```

0.000000000000000E+000  1.000000000000000E-003
4.01941874999938E-002  5.08932684411733E+000
1.60776749999970E-001  1.03542796946115E+001
3.61747687499928E-001  1.57958814746353E+001
6.43106999999869E-001  2.14141703884248E+001
1.00485468750068E+000  2.72091999212991E+001
1.44699075000184E+000  3.31810388396605E+001
1.96951518750333E+000  3.93297711909949E+001
2.57242800000517E+000  4.56554963038703E+001
3.25572918750413E+000  5.21583287879384E+001
4.01941875000271E+000  5.88383985339332E+001
4.86349668750092E+000  6.56958507136728E+001
5.78796299999876E+000  7.27308457800587E+001
6.79281768749622E+000  7.99435594670744E+001
7.87806074999330E+000  8.73341827897873E+001
9.04369218749001E+000  9.49029220443488E+001
1.02897119999864E+001  1.02649998807992E+002
1.16161201874823E+001  1.10575649939033E+002
1.30229167499779E+001  1.18680127576873E+002
...
...
...
9.98316260282109E+003  2.42641578273744E+004
9.98511852341417E+003  2.42691578273744E+004
9.98695386144475E+003  2.42741578273744E+004
9.98866861691283E+003  2.42791578273744E+004
9.99026278981841E+003  2.42841578273744E+004
9.99173638016149E+003  2.42891578273744E+004
9.99308938794207E+003  2.42941578273744E+004
9.99432181316016E+003  2.42991578273744E+004
9.99543365581574E+003  2.43041578273744E+004
9.99642491590882E+003  2.43091578273744E+004
9.99729559343941E+003  2.43141578273744E+004
9.99804568840749E+003  2.43191578273744E+004
9.99867520081308E+003  2.43241578273744E+004
9.99918413065616E+003  2.43291578273744E+004
9.99957247793675E+003  2.43341578273744E+004
9.99984024265483E+003  2.43391578273744E+004
9.99998742481042E+003  2.43441578273744E+004

```

11.4 Anhang: Aufgabenstellung Projektaufgabe #003

Datenverarbeitung 2

Studienjahr 2011/12

– Projektaufgabe #003 – „Vergleich: Diesel gegen Elektromotor“

Aufgabenstellung

Dies Projekt soll den nötigen Energieaufwand bei der Fahrt eines Automobils über eine definierte Strecke mit definiertem Zeit-Geschwindigkeitsprofil ermitteln, und dabei einen Dieselantrieb mit einem elektrischen Antrieb vergleichen. Der Einsatz von Energie ist für das Beschleunigen, die Überwindung der Rollreibung sowie die Überwindung des Luftwiderstandes notwendig.

Annahmen zum Fahrzeug: Das Fahrzeug hat eine Masse m_F . Die Rollreibung des Fahrzeuges ist unabhängig von der Geschwindigkeit und wird beschrieben durch einen Reibkoeffizienten r . Der Luftwiderstand ist von der Geschwindigkeit quadratisch abhängig: Die zur Überwindung des Luftwiderstandes nötige Kraft F_L berechnet sich aus dem Fahrzeugquerschnitt A , dem Windwiderstandsbeiwert c_w und der aktuellen Geschwindigkeit v nach $F_L = 0.5 \cdot A \cdot c_w \cdot \rho_L \cdot v^2$, wobei ρ_L die spezifische Luftmasse ist. Die heute übliche Schubabschaltung (d. h. soweit möglich Antrieb des Motors durch abbremsendes Fahrzeug) kann zur Vereinfachung vernachlässigt werden. Der Wirkungsgrad des Getriebes kann vernachlässigt werden.

Annahmen zum Dieselantrieb: Der Motor sei für die Nennleistung P_N konstruiert und mit einem automatischen Getriebe ausgerüstet, das den Motor stets im günstigsten Arbeitsbereich hält. Vereinfacht sei anzunehmen, dass sich der Motor als eine ideale Wärmekraftmaschine mit einem sehr guten Wirkungsgrad von 40% und drehzahlunabhängigen mechanischen Verlusten in der Höhe von 20% der Nennleistung beschreiben lässt. Bei Bremsvorgängen läuft die Maschine weiter und muss dann nur die eigenen mechanischen Verluste kompensieren.

Annahmen zum Elektroantrieb: Der Elektromotor mit der Nennleistung P_N habe einen konstanten Wirkungsgrad von 85%. Der Lade-Entladevorgang der Batterie hat einen Wirkungsgrad von 80%. Beim Bremsen kann der Motor ausgekoppelt werden und benötigt keine Energie.

Annahmen zum Fahrzyklus: Die Teststrecke ist 10 km lang und hat folgende Charakteristik:

- 3 km ebene Strecke,
- 2 km Strecke mit 4% Steigung,
- 2 km Strecke mit 4% Gefälle und nochmals
- 3 km ebene Strecke.

Jeder einzelne Kilometer dieser Strecke wird nach folgendem Profil befahren:

- 300 m Beschleunigung von 0 auf v_{SOLL} ,
- 500 m konstante Fahrt mit v_{SOLL} und
- 200 m Abbremsung auf 0.

Vereinfachend darf angenommen werden, dass die Beschleunigung a beim Beschleunigen und Bremsen jeweils konstant ist.

Analyse: Für Fahrzeuge mit $m_F = 1500\text{kg}$, $r = 0,015$, $A = 2\text{m}^2$, $c_w = 0,33$ sind die Antriebe auf der Teststrecke zu vergleichen und der Gesamtenergieverbrauch zu bestimmen. Die Vergleiche sind für drei Fahrzeugklassen mit $P_N = 30, 50, 60\text{ kW}$ durchzuführen. Die Geschwindigkeit v_{SOLL} ist sinnvoll zu wählen und weiter zu untersuchen, wie sich eine Veränderung der Sollgeschwindigkeit um $\pm 20\%$ auf den Energiebedarf auswirkt.

Die Ergebnisse sind weiter als Diagramme des Energieverbrauchs über dem zurückgelegten Weg darzustellen. Hierfür können beispielsweise Ausgabedateien erstellt werden, die in einem geeigneten Grafik- oder Tabellenkalkulationsprogramm eingelesen werden.

Schwierigkeitsgrad der Aufgabe

Der Schwierigkeitsgrad dieser Aufgabe ist: *sehr gering* – *gering* – *mittel* – *hoch* – *sehr hoch*¹

Die Schwierigkeit kann erhöht werden, indem die heute übliche Energierückgewinnung mit untersucht wird (in diesem Fall beide Fälle – mit und ohne Rückgewinnung – betrachten!). Der Dieselmotor kann abgeschaltet bleiben, solange die durch Bremsen zur Verfügung stehende Energie nach Kompensation von Rollreibung und Luftwiderstand ausreicht, um die mechanischen Verluste des Motors zu decken (Schubabschaltung). Das elektrisch angetriebene Fahrzeug kann beim Bremsen die Energie in die Batterie zurückspeisen, mit denselben Wirkungsgraden wie bei angetriebener Fahrt.

Weiter kann der Schwierigkeitsgrad erhöht werden, indem die grafische Darstellung direkt in das Programm eingebunden wird (grafische Bibliothek, Ansteuerung von MS Excel o.ä.).

Generelle Hinweise

Bei der Durchführung der Projektaufgabe sind zwingend die „Prüfungsbedingungen Modul Maschinenbauinformatik“ sowie die „Bewertungsrichtlinien Modul Maschinenbauinformatik“ zu beachten, die jeweils von der Webseite der Veranstaltung heruntergeladen werden können. Insbesondere ist zu beachten, dass für die Durchführung der Projektaufgabe eine vorherige Anmeldung erforderlich ist.

Wenn unter „Besondere Anforderungen“ nichts anderes angegeben ist, sind neben allgemeinen ingenieurwissenschaftlichen Kenntnissen (einschließlich der Fähigkeit, sich eigenständig neue Inhalte zu erschließen), nur die Kenntnisse aus der Vorlesung „Datenverarbeitung 1“ erforderlich.

Besondere Anforderungen

(keine)

Hinweise zur Durchführung

Es ist empfehlenswert, relativ kleine Zeitschritte zu verwenden; ein guter Wert stellt $t = 0,01\text{ s}$ dar. Danach kann durch Variation von t untersucht werden, wie klein das Intervall tatsächlich für sinnvolle Ergebnisse sein muss.

Für die Darstellung der Energiekurve sollten um der Effizienz willen größere Abständen gewählt werden (z. B. 1 s), d. h. nur jeder k -te Wert wird an die Grafik übergeben.

¹ Aufgaben der Schwierigkeitsgrad „gering“ und „sehr gering“ werden im Modul Maschinenbauinformatik nicht gestellt. Die Notengebung ist auf den Schwierigkeitsgrad bezogen, d.h. Projekte mit höherer Schwierigkeit werden bei gleich sorgfältiger Umsetzung besser bewertet. Grundsätzlich kann auch bei Aufgaben mittleren Schwierigkeitsgrades bei sehr guter Umsetzung die Note „sehr gut“ erworben werden.