Jednostavan alat za određivanje potrošnje goriva i emisija motora za zadani radni ciklus

Babić, Boris

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Boris Babić

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Prof. dr. sc. Zoran Lulić, dipl. ing. stroj. Boris Babić, bacc. ing. mech.

Zagreb, 2020.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svojoj djevojci, sestri i roditeljima na pruženoj podršci tijekom cijelog studija.

Također, zahvaljujem se i svojem mentoru prof. dr. sc. Zoranu Luliću na susretljivosti, pomoći i savjetima pri izradi ovog rada.

Boris Babić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
procesno-energetski, konstrukcijski, brodostrojarski i inženjersko modeliranje i računalne simulacije

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 20 - 6 / 3	
Ur. broj: 15 - 1703 - 20 -	

DIPLOMSKI ZADATAK

Student: **Boris Babić** Mat. br.: 1191228064

Naslov rada na hrvatskom jeziku: **Jednostavan alat za određivanje potrošnje goriva i emisija motora za zadani radni ciklus**

Naslov rada na engleskom jeziku: **Simple tool for determining fuel consumption and engine emissions for a given duty cycle**

Opis zadatka:

Zadnjih godina iznimna se pažnja posvećuje emisijama štetnih tvari u okoliš iz motora s unutarnjim izgaranjem. Za određivanje emisija iz motora, uz konstrukcijske značajke koje se tijekom eksploatacije praktički ne mijenjaju, ključan utjecaj ima način uporabe motora. On se može odrediti snimanjem parametara rada motora u stvarnim uvjetima eksploatacije. Na temelju snimljenih podataka statističkom analizom određuju se standardni radni ciklusi. Kako je snimanje radnih parametara te mjerenje potrošnje goriva i emisija zahtjevno i skupo za različite uporabe i analize, koriste se različiti namjenski računalni simulacijski programi.

U okviru diplomskog rada treba izraditi računalni program koji će omogućiti praćenje rada vozila/radnog stroja, fazu po fazu, od pokretanja motora pa do ponovnog zaustavljanja motora. Režim rada motora se treba pratiti svaku sekundu tj. s frekvencijom 1 Hz.

Cilj je za neki opisani (zadani) režim rada motora, od pokretanja do zaustavljanja motora, izračunati brzinu vrtnje $n(t)$, snagu motora $P(t)$, moment motora $M(t)$, srednji efektivni tlak $p_e(t)$, pri čemu vrijeme t ide od $t = 0$ (pokretanje motora) do t_n (zaustavljanje motora), a na osnovu toga iz tablično definiranih podataka o specifičnoj efektivnoj potrošnji goriva $g_e(n, p_e)$, emisijskim faktorima $EF_i(n, p_e)$ odrediti trenutnu potrošnju goriva i trenutne emisije. Na kraju postupka (simulacije) potrebno je odrediti ukupnu potrošenu količinu goriva i ukupne emisije.

Za izradu vlastitog programa proučiti postojeće javno dostupne programske pakete otvorenog koda kao što su CO2MPAS i VECTO.

Funkcionalnost računalnog programa treba dokazati na nekoliko jednostavnih primjera radnog ciklusa i generičkih podataka za specifičnu potrošnju goriva i emisijske faktore. Za izračun potrošnje goriva i emisija koristiti pojednostavljene dijagrame specifične potrošnje goriva i emisijskih faktora.

Pri izradi se treba pridržavati uobičajenih pravila za izradu diplomskoga rada. U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datum obrane:
30. travnja 2020.	2. srpnja 2020.	6. – 10.7.2020.

Zadatak zadao: Predsjednica Povjerenstva:

Prof. dr. sc. Zoran Lulić Prof. dr. sc. Tanja Jurčević Lulić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
α	°	Kut nagiba ceste
η_m	-	Mehanički stupanj djelovanja prijenosa snage
ρ_z	kg/m^3	Gustoća zraka
A	m^2	Čeona površina vozila
a	m/s^2	Ubrzanje vozila
c_w	-	Faktor otpora zraka
$d_{kotača}$	m	Promjer kotača
f_k	-	Faktor otpora kotrljanja
F_a	N	Sila otpora ubrzanja
F_k	N	Sila otpora kotrljanja
F_z	N	Sila otpora zraka
i_{OR}	-	Osovinska redukcija
i_m	-	Prijenosni omjer stupnja u mjenjaču
i_{uk}	-	Ukupni prijenosni omjer
J_k	kgm^2	Moment inercije kotača
J_m	kgm^2	Moment inercije motora i njemu pridruženih dijelova
k_m	-	Faktor rotirajućih masa
M	Nm	Okretni moment motora
m_v	kg	Masa vozila
n_{mot}	okr/min	Brzina vrtnje motora
n_{kot}	okr/min	Brzina vrtnje kotača
P	W	Snaga
p_e	bar	Srednji efektivni tlak
r_d	m	Radijus kotača
t	s	Vrijeme
T	-	Broj taktova motora
v	m/s	Brzina vozila
V_m	m^3	Radni volumen motora

SAŽETAK

Emisije štetnih tvari iz vozila važan su faktor razvoja novih motora s unutarnjim izgaranjem. Zbog njihovog negativnog utjecaja zakonom su propisane njihove granične vrijednosti i metode mjerenja. U ovom radu opisane su vrste emisija iz vozila i ispitne procedure za mjerenje emisija u Europi. Prikazani su simulacijski alati CO_2MPAS i VECTO i njihov princip rada. Glavni razlozi njihova nastanka su problemi dugotrajnosti i veliki troškovi mjerenja emisija ugljikovog dioksida. Prema uzoru na princip rada dva opisana programa, napravljen je simulacijski alat za računanje nastalih emisija za vrijeme vožnje po proizvoljno zadanim ispitnim procedurama. Opisani su i objašnjeni računalni kôd i fizikalni principi prema kojima je alat modeliran. Prikazana je provjera rada programskog algoritma i smislenost njegovih rezultata.

Ključne riječi: emisije vozila, ispitna procedura, ugljikov dioksid, CO_2MPAS, VECTO, potrošnja goriva, otpori gibanja

SUMMARY

Emissions of harmful substances from vehicles are an important factor in the development of new internal combustion engines. Due to their negative impact, their limit values and measurement methods are prescribed by law. This paper describes the types of vehicle emissions and test procedures for measuring emissions in Europe. The simulation tools CO2MPAS and VECTO for measuring CO_2 emissions are described. Problems of longevity and the high cost of measuring carbon dioxide emissions are some of many reasons for making this kind of software. Simulation tool for calculating the generated emissions while driving according to arbitrarily given test procedures was made according to physical principles of these simulation tools. The computer code and used physical principles are described and explained. The functionality of algorithm and validity of program results are verified and presented.

Key words: vehicle emissions, test procedure, carbon dioxide, CO2MPAS, VECTO, fuel consumption, motion resistance

1. Uvod

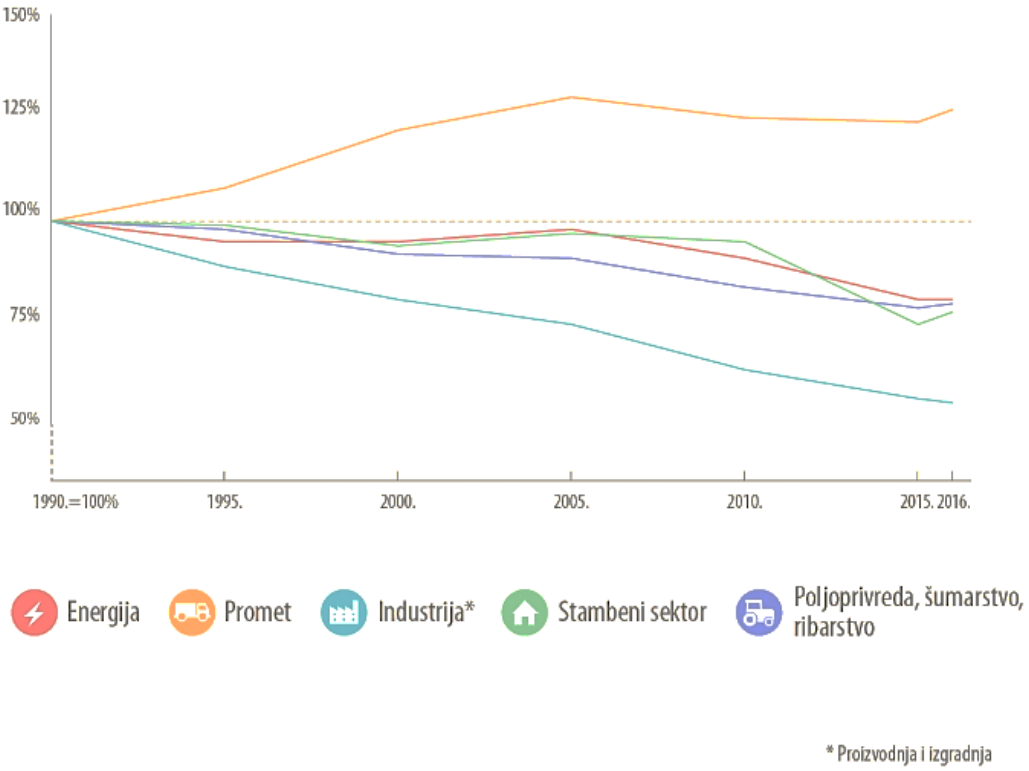
Cestovni prijevoz značajan je izvor stakleničkih plinova i onečišćujućih tvari u zraku. Emisije iz vozila dovode do visoke koncentracije onečišćujućih tvari u zraku, veće od zakonima propisanih graničnih vrijednosti EU u mnogim europskim gradovima. Iako su loša kvaliteta zraka i klimatske promjene vrlo različite pojave, oboje šteti ljudskom zdravlju i okolišu. Takvi štetni učinci uzrokovani onečišćenjem od cestovnog prometa uzrokuju stvarne ekonomske troškove društvu.

U proteklim godinama postignut je napredak u smanjenju ispušnih plinova mnogih onečišćivača iz cestovnog prometa. Ova dostignuća rezultat su kombinacije politika i raznih mjera kao što su postavljanje tehnoloških standarda za emisiju vozila i kvalitetu goriva i zakonodavstvo kojim se utvrđuju granične vrijednosti kvalitete zraka i upravljanja prometom. Unatoč tome, zbog sveukupnog povećanja prometa putnika i tereta, kao i nedovoljno ispunjenje određenih standarda vozila u stvarnim uvjetima vožnje, smanjenja emisija tijekom posljednjih godina nisu toliko velika koliko se planiralo.

1.1. Utjecaj emisija vozila na zdravlje ljudi i okoliš

1.1.1. Stakleнički plinovi

Iako su emisije stakleničkih plinova iz svih ostalih glavnih sektora gospodarstva posljednjih godina u padu, emisije iz prometa su se povećale [3]. Na slici 1. prikazan je trend promjene ukupnih emisija ugljikovog dioksida iz različitih gospodarskih sektora od 1990. do 2016. godine. Prema [4], udio koji cestovni promet čini ukupnim emisijama u EU povećan je sa 13 % udjela u 1990. na više od 25 % u 2020.



Slika 1. Emisije CO_2 u Europskoj uniji [3]

1.1.2. Onečišćenje zraka

Onečišćenje zraka može se definirati kao prisutnost onečišćujućih tvari u atmosferi na razinama koje štete ljudskom zdravlju, okolišu ili kulturnoj baštini (npr. oštećivanjem zgrada, spomenika i materijala). Kvaliteta zraka ovisi o mnogim čimbenicima, kao što su različiti izvora emisija na određenom području, lokalni krajolik i vremenski uvjeti, a svi oni mogu utjecati na stvaranje i kretanje onečišćujućih tvari.

Cestovni prijevoz ostaje važan izvor nekih od najštetnijih onečišćivača zraka. Konkretno, isti je odgovoran za značajan doprinos emisiji dušikovih oksida (NO_X) i krutih čestica. Onečišćenje koje stvaraju vozila osobito je važno jer se emisije obično javljaju u naseljenim područjima. Stoga, iako emisije iz cestovnog prometa nisu jedini izvor onečišćujućih tvari, izloženost stanovništva prema njima može biti veća nego izloženost izvorima poput elektrana ili velikih industrijskih postrojenja koja su često locirana u udaljenijim, manje naseljenim područjima.

Za razliku od emisija stakleničkih plinova, emisije glavnih onečišćivača zraka iz prometa posljednjih su godina uglavnom u padu. Međutim, najnovija procjena kvalitete zraka iz 2019. godine [4] koju je objavila Europska agencija za okoliš govori da je značajan dio europskog

urbanog stanovništva bio izložen razinama onečišćenja zraka koje premašuju EU standarde kvalitete zraka. Na primjer, Europska godišnja granična vrijednost za dušikov dioksid (NO_2) i dalje je uvelike premašena u Europi, uglavnom na cestovnim lokacijama. Slično tome, nekoliko država članica imalo je razine krutih čestica više od odgovarajućih EU standarda kvalitete zraka.

Kako bi se smanjili negativni učinci na kvalitetu zraka uzrokovanih emisijama iz cestovnog prometa, EU emisijski standardi postaju sve stroži tijekom posljednjih desetljeća i za osobna i gospodarska vozila. Proizvođači vozila pokušavaju ispunjavati zadana ograničenja uglavnom uvođenjem tehnoloških rješenja, posebno postupnom primjenom poboljšanih tehnologija upravljanja emisijama, poput oksidacijskih katalizatora, poslije katalizatora trostrukog djelovanja, filtera krutih čestica, filtera goriva, boljom kontrolom izgaranja...

1.1.3. Onečišćujuće tvari koje emitiraju vozila

Cestovna vozila emitiraju niz različitih onečišćujućih tvari i stakleničkih plinova. Te emisije se mogu podijeliti u dvije skupine, emisije koje su regulirane EU zakonodavstvom o cestovnom prometu i one koje trenutno nisu. Regulirani tvari uključuju:

- Ugljikov dioksid (CO_2) - glavni je proizvod izgaranja goriva u motorima vozila zajedno s vodenom parom. Iako ugljikov dioksid ne pripada skupini onečišćujućih tvari, najznačajniji je spoj stakleničkih plinova koji utječe na klimatske promjene i time utječe na ljudsko zdravlje i okoliš.
- Ugljikovodici (HC) – stvaraju se djelomičnim izgaranjem goriva. Opasni su po ljudsko zdravlje. Ugljikovodici, a posebno hlapljivi organski spojevi (engl. *volatile organic compounds*), doprinose stvaranju prizemnog ozona i fotokemijskog smoga u atmosferi. Posljedice ozona su iritacija očiju, oštećenje pluća i pogoršavanje respiratornih problema.
- Ugljikov monoksid (CO) - produkt je djelomičnog izgaranja goriva. Nastaje kada ugljik u gorivu samo djelomično oksidira, formirajući CO, a ne CO_2. Bezbojan je i bez mirisa, i vrlo otrovan. Izravno izlaganje smanjuje protok kisika u krvotoku i posebno je opasno za ljude koji imaju srčane bolesti. Kao i ugljikovodici, ugljikov monoksid također doprinosi stvaranju prizemnog ozona i smoga.
- Krute čestice (PM) - proizvod su nepotpunog izgaranja i složene su smjese primarnih i sekundarnih krutih čestica. Primarne krute čestice dio su krutih čestica koje se emitiraju izravno u atmosferu, dok se sekundarne krute čestice formiraju u atmosferi nakon oslobađanja drugih plinova (uglavnom sumpornog dioksida (SO_2), dušikovih oksida

(NO_X), amonijaka (NH_3) i drugih hlapljivih spojeva). Krute čestice su jedna od najopasnijih vrsta onečišćujućih tvari po ljudsko zdravlje jer prodiru u osjetljiva područja dišnog sustava te mogu uzrokovati ili pogoršati kardiovaskularne i respiratorne bolesti, te su kancerogene.

- Dušikovi oksidi (NO_X) - skupina različitih spojeva koji nastaju reakcijom dušika, najzastupljenijeg plina u zraku, s kisikom. Dušikovi oksidi mogu biti prisutni kao bezbojni dušični oksid (NO) i crvenkasto-smeđi, vrlo toksičan i reaktivan dušikov dioksid (NO_2). Emisije dušikovih oksida također dovode do naknadnog stvaranja sekundarnih krutih čestica i prizemnog ozona u atmosferi, te uzrokuju štetu okolišu tako što zakiseljuju vode i tla.

Onečišćujuće tvari koje ispuštaju cestovna vozila koje trenutno nisu regulirane propisima o dozvoljenim emisijama vozila u EU uključuju:

- određene kisele onečišćujuće tvari, poput amonijaka i sumpornog dioksida (iako se te emisije neizravno rješavaju zakonodavstvom o kvaliteti goriva, što ograničava količinu sumpora dopuštenog u gorivima)
- određene kancerogene i toksične organske spojeve, poput policikličkih aromatskih ugljikovodika, trajnih organskih onečišćivača; teških metala, kao što su olovo, arsen, kadmij, bakar, krom, živa, nikal, selen i cink.

1.1.4. Vrste emisija iz cestovnih vozila

Emisije iz cestovnih vozila mogu se svrstati u tri skupine:

- Ispušni plinovi - emisije nastale prvenstveno izgaranjem različitih naftnih proizvoda kao što su benzin, dizelsko gorivo, prirodni plin i ukapljeni naftni plin. Ta goriva su mješavine različitih ugljikovodika, tj. spojeva koji sadrže vodikove i ugljikove atome. U teoriji u motoru, kisik iz zraka reagira u procesu izgaranja sa svim vodikom u gorivu te nastaje voda, te sa svim ugljikom u gorivu gdje nastaje ugljikov dioksid, a dušik u zraku bi ostao netaknut. U stvarnosti, nijedan proces izgaranja nije savršen. Motori vozila osim vodene pare i ugljikovog dioksida emitiraju i razne onečišćujuće tvari. Količina svake emitirane onečišćujuće tvari ovisi o vrsti goriva koja se koristi, o vrsti radnog ciklusa u motoru, te o načinu eksploatacije motora.
- Emisije abrazije - emisije nastale mehaničkom abrazijom i korozijom dijelova vozila. Abrazija je važna samo za emisije krutih čestica i emisije nekih teških metala. Značajne

razine emisija krutih čestica mogu se stvoriti mehaničkom abrazijom guma, kočnica i spojki vozila, istrošenošću cestovne površine ili korozijom šasije, karoserije i ostalih dijelova vozila.

- Emisije isparavanja - rezultat isparavanja iz sustava za gorivo u vozilu. Emisije isparavanja važne su za samo hlapljive spojeve. Pare benzina sadrže mnoštvo različitih ugljikovodika, koji se mogu ispuštati kad god se u spremniku nalazi gorivo, čak i kad je vozilo parkirano s isključenim motorom.

Tema ovog rada bazira se samo na emisijama iz ispušnih plinova stvorenih radom motora.

1.2. Emisije ispušnih plinova iz vozila

U konvencionalnom vozilu, prema [6], samo se 12 do 30 % energije iz goriva koristi za njegovo kretanje na cesti, ovisno o uvjetima vožnje. Ostatak energije gubi se zbog neučinkovitosti motora i pogonskog sklopa. Mali udio proizvedene energije koristi se za pogon dodatne opreme za vozila (npr. svjetla, radio, klima uređaj). Stoga je potencijal daljnjeg poboljšanja učinkovitosti goriva korištenjem naprednih tehnologija i dalje velik. Dok noviji dizelski motori ostaju ekonomičniji od benzinskih, njihov utjecaj na onečišćenje zraka je gori zbog viših razina dušikovih oksida i krutih čestica koje emitiraju.

1.2.1. Emisija ugljikovog dioksida

Europska unija zalaže se za smanjenje potrošnje goriva iz cestovnih vozila u nastojanju da smanji emisije stakleničkih plinova iz prometa i poboljša energetsku učinkovitost. U tu su svrhu u posljednje vrijeme uvedena dva važna propisa za nova osobna vozila i nova laka gospodarska vozila koja se prodaju u Europi. 2009. godine donesena je uredba (EZ) br. 443/2009 [7], kojom su utvrđeni obvezni godišnji ciljevi za prosječne emisije ugljikovog dioksida iz novih osobnih automobila prodanih u Europi. Novi automobili registrirani u EU morali su dostići prosječni cilj emisije od 130 grama CO_2 po kilometru (g CO_2 /km) do 2015. Također je utvrđen i daljnji cilj: do 2021., s početkom 2020., prosječna emisija koju trebaju postići svi novi automobili iznosi 95 g CO_2/km. Po primjeru zakonodavstva za automobile, dvije godine kasnije, donesena je posebna uredba (EU) br. 510/2011 [8], koja postavlja ciljeve za laka gospodarska vozila (kategorija N1). Nova laka gospodarska vozila registrirana u EU moraju zadovoljiti ciljane emisije od 175 g CO_2/km do 2017. Za 2020. cilj je 147 g CO_2/km.

Kao i kod mjerenja ugljikovog dioksida, sukladnost vozila s potrebnim ograničenjima provjerava se na temelju standardiziranih laboratorijskih mjerenja emisija. Prva direktiva Europskog vijeća koja je specificirala mjere protiv onečišćenja zraka iz motornih vozila donesena je 1970. (Direktiva 70/220/EEC [9]). Otprilike 20 godina kasnije (1992.), uvedene su Europske ekološke direktive za motorna vozila, počevši s Euro 1, a slijedile su je sve strože direktive (od Euro 2 do Euro 6). Sve strože granične vrijednosti emisija dovele su do uvođenja novih tehnologija pripreme smjese i obrade ispušnih plinova, a time je došlo i do značajnih smanjenja emisija vozila u Europi u posljednjih 50 godina. Kao primjer, najnovija tehnologija Euro 6 Dieselovih motora mora emitirati gotovo 97 % manje krutih čestica pri testiranju u odnosu na 25 godina starije vozilo Euro 1.

1.3. Mjerenje emisija

Za provjeru ispunjavaju li vozila zadane zahtjeve za ispušne plinove koriste se standardizirana mjerenja u laboratorijima. Službeni postupci koji su se donedavno koristili u Europi nisu bili reprezentativni za današnje stvarne uvjete vožnje. Taj je problem doveo do razvoja novih mjernih postupaka, kao i prijenosnih sustava za mjerenje emisija kako bi se dobile bolje informacije o emisijama tijekom stvarnih uvjeta vožnje.

1.3.1. Mjerenje emisija prema zakonodavstvu Europske unije

Prema europskim direktivama, prije prodaje, vozila se moraju ispitati kako bi se utvrdilo da su u skladu s traženim standardima zaštite okoliša, klime i sigurnosti. Kako nije praktično testirati svako pojedinačno vozilo, ispituje se jedno proizvodno vozilo, pa se to vozilo smatra reprezentativnim za određeni model, te ako se poštuju svi standardi, izdaje se dokumentacija o homologaciji. Ustanove za homologaciju odgovorne su za sve aspekte homologacije vozila. To uključuje izdavanje i povlačenje certifikata o homologaciji, kao i imenovanje službi tehničkog laboratorija koji provode ispitivanja i provjeravaju jesu li vozila u skladu s relevantnim europskim zakonima.

U sklopu testiranja sva laka vozila, bilo da su to osobna vozila, laka gospodarska vozila, mopedi ili motocikli, moraju se provesti testiranja na valjcima. Valjci su projektirani za upravljanje vozilom u zatvorenom prostoru na stacionarnoj platformi za simulaciju rada u stvarnom svijetu. Vozilo stoji na mjestu, a kotači se okreću na valjcima, prateći unaprijed definirani obrazac vožnje. Valjci simuliraju inerciju vozila kao i otpor zraka, tj. opterećenja na cesti. Razina otpora na dinamometru prilagođava se za svako određeno vozilo testirano kako bi simuliralo razinu otpora s kojim bi se vozilo susrelo kada bi upravljalo na cesti. Elementi koji su uzeti u obzir su:

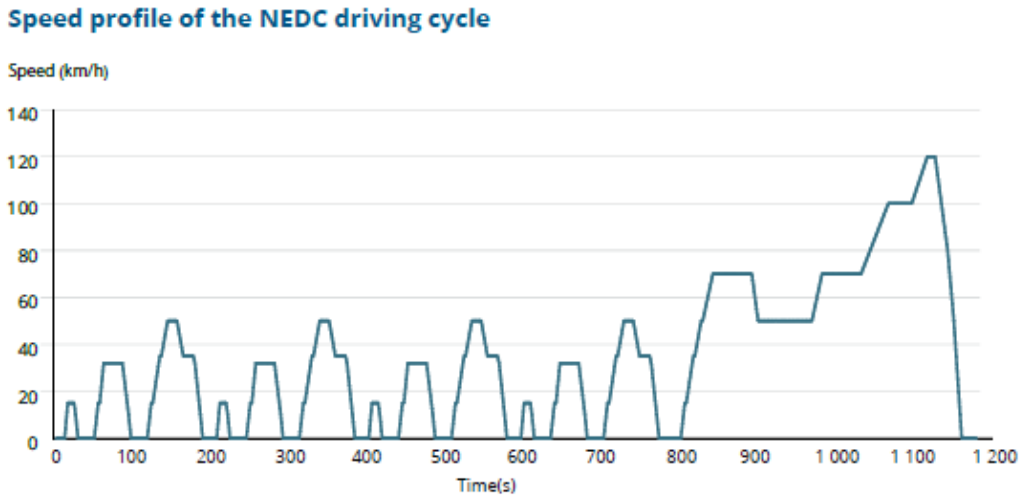
- Aerodinamički otpor vozila - faktor koji ovisi o veličini i obliku vozila. Određuje koliko zraka vozilo mora istisnuti s puta dok se kreće, što je otpor veći, to više energije treba potrošiti.
- Otpor kotrljanja kotača - faktor povezan s konstrukcijom kotača koji određuje koliko energije vozilo treba upotrijebiti za prevladavanje otpora uzrokovanog interakcijom gume (kotača) i ceste.

Da bi odredili emisiju i potrošnju goriva, svako vozilo slijedi unaprijed definirani ciklus vožnje na valjcima. Vozni ciklusi unaprijed su definirani ciklusi ubrzanja, promjena stupnjeva prijenosa, stalne brzine, usporavanja i rada u praznom hodu. Vozač mora pratiti određeni vozni ciklus unutar definiranih odstupanja.

Dok se vozilo vozi na valjcima, sve emisije iz auspuha sakupljaju se u zapečaćene vreće te se naknadno analiziraju. Razine emisija prvenstveno ovise o faktorima koji se odnose na vozila kao što su vrsta vozila, veličina, opterećenja na cesti, vrsta goriva i korištena tehnologija za rad vozila. Osim konfiguracije vozila, dinamika vožnje, uključujući brzinu vozila, ubrzanje, vrijeme rada u praznom hodu i odabir brzina, ima vrlo značajan utjecaj na emisije. Dakle, vrsta standardiziranog voznog ciklusa koja se koristi za testiranje važan je faktor u određivanju emisije vozila.

1.3.2. NEDC ciklus

NEDC ciklus (engl. *New European Driving Cycle*) koristio se u skladu sa zakonodavstvom EU za procjenu emisije i ekonomičnosti lakih gospodarskih i osobnih vozila tijekom homologacije. Prvi put je uveden 1970. godine kako bi predstavljao uobičajene uvjete vožnje u prometnim europskim gradovima. Ažuriran je 1990. godine u pokušaju boljeg predstavljanja zahtjevnijih načina vožnje većim brzinama. NEDC ciklus se sastoji od gradskog i vangradskog dijela vožnje. Izvorno je razvijen za lakša vozila s manjom snagom od današnjih. Iz tih razloga, test uključuje samo jednostavan obrazac brzina s malim ubrzanjima, konstantnom brzinom i više dijelova u praznom hodu motora koji obično slabo opterećuju moderne motore. Vrijednosti emisija i potrošnja goriva izmjerene u laboratoriju upotrebom NEDC ciklusa uvelike se razlikuju od stvarne razine izmjerene u stvarnim uvjetima vožnje. Zbog tih nedostataka NEDC ciklusa, u Europi i svijetu razvijeni su brojni drugi ciklusi vožnje.



Slika 2. Prikaz brzine vožnje u ovisnosti o vremenu tijekom NEDC ispitnog ciklusa [4]

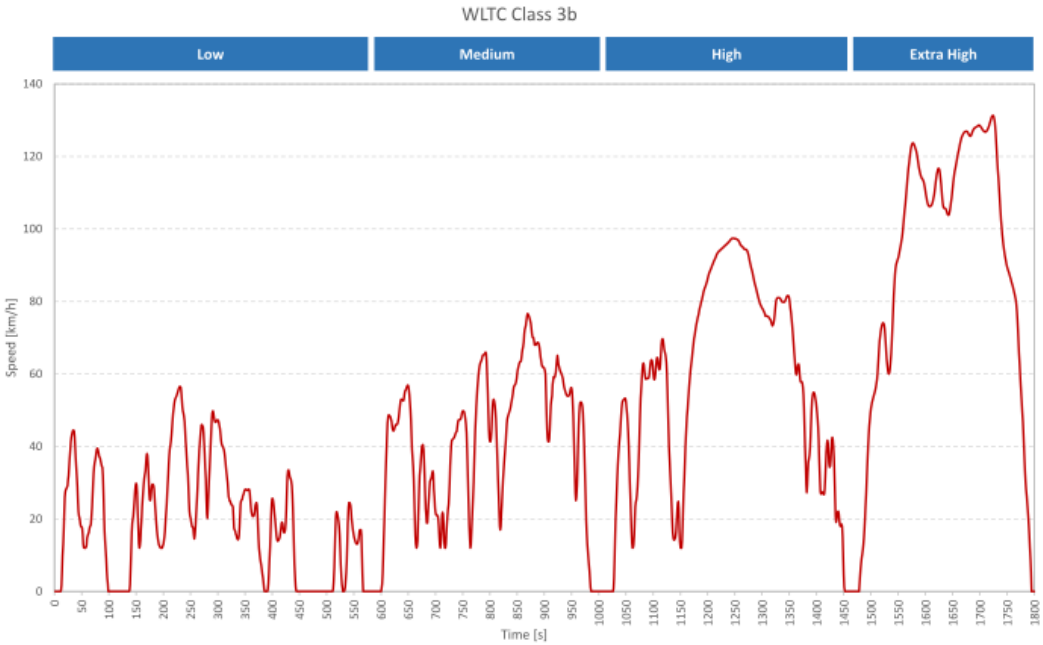
1.3.3. WLTP ciklus

WLTP ciklus (engl. *Worldwide Harmonised Light Vehicles Test Procedure*) europski je, usklađeni standard za određivanje razine emisije ugljikovog dioksida i potrošnje goriva kod vozila s motorima s unutarnjim izgaranjem i kod hibridnih vozila. Ovaj novi ciklus razvila je Gospodarska komisija Ujedinjenih naroda za Europu (UNECE) kako bi se zamijenio NEDC ciklus kao postupak homologacije vozila u Europi. Njegova konačna verzija objavljena je 2015. Jedan od glavnih ciljeva WLTP ciklusa je bolje usklađivanje laboratorijskih procjena potrošnje goriva i emisija sa stvarnim uvjetima na cesti.

Budući da ciljane vrijednosti za ugljikov dioksid postaju sve važnije za ekonomsku dobit proizvođača automobila u cijelom svijetu, uz pomoć WLTP ciklusa također se nastoji uskladiti ispitne postupke na međunarodnoj razini i uspostaviti jednake uvjete na globalnom tržištu. Pored zemalja EU, WLTP je standardni test potrošnje goriva i emisija također za Indiju, Južnu Koreju i Japan. Osim toga, WLTP ciklusom nastoji se osigurati da nova vozila ne emitiraju više ugljikovog dioksida od cilja koji je postavila Europska unija i koji je trenutno postavljen na 95 g CO_2 /km za 2021. godinu.

U usporedbi s NEDC ciklusom, WLTP ima:

- veću udaljenost ispitivanja (23,25 prema 11,0 km) i trajanje (1800 prema 1180 sekundi);
- veću prosječnu brzinu (46,5 prema 33,6 km/h);
- veću maksimalnu brzinu (131 prema 120 km/h);
- manje zaustavljanja (9 prema 14);
- manje vožnje stalnom brzinom (66 prema 475 sekundi);
- veća ubrzanja (789 prema 247 sekundi) i kočenja (719 prema 178 sekundi);
- manje vrijeme praznog hoda (226 prema 280 sekundi).



Slika 3. Prikaz brzine vožnje u ovisnosti o vremenu tijekom WLTP ispitnog ciklusa [4]

Najvažnije razlike između WLTP i NEDC ciklusa u pogledu utjecaja na emisiju ugljikovog dioksida mogu se grupirati u sljedeće kategorije:

- Veća dinamika vožnje - učestala ubrzanja i veće brzine WLTP ciklusa zahtijevaju veće količine energije, što rezultira većom potrošnjom goriva od NEDC ciklusa. Suprotno tome, veća učinkovitost motora pri većim opterećenjima smanjuje potrebnu količinu goriva. Kombinacija ova dva učinka dovest će do ukupno veće potrošnje goriva za WLTP u odnosu na NEDC ciklus. Osim toga, prednosti start-stop sustava (gašenje motora za vrijeme stajanja vozila što smanjuje emisije kod stajanja na nulu) bit će manje u WLTP-u zbog smanjenih faza rada u praznom hodu.
- Ispitna masa vozila - postoji jasnija definicija mase vozila u WLTP-u, koja uzima u obzir dodatnu opremu. Za postupak ispitivanja NEDC, masa najlakše verzije modela vozila može se koristiti za ispitivanje emisija ugljikovog dioksida. Dakle, različite verzije istog modela vozila imat će veće emisije iz WLTP-a od osnovnog modela bez dodatne opreme u NEDC testu.
- Pokretanje hladnog motora - općenito vožnja vozila s hladnim motorom povećava emisiju CO_2. Međutim, s obzirom da je WLTP duži od NEDC-a, dodani doprinos

emisija pokretanja hladnog motora distribuirat će se na veće udaljenosti i neće imati značajan utjecaj na ukupne emisije CO_2.

- Okolna temperatura - temperatura ispitivanja u WLTP je 23 °C. Međutim, EU planira temperaturu sniziti na 14 ° C što je bliža vrijednost za europske prosječne temperature. To će rezultirati većom potrošnjom goriva zbog povećanog doprinosa emisijama pokretanja hladnog motora.

Prijelaz na WLTP ciklus odvijao se u fazama. Od rujna 2017. koristi se samo WLTP testiranje za homologacije novih vozila. Neki automobili su imali stare NEDC vrijednosti, dok su drugi već bili certificirani WLTP ciklusom. Tijekom prijelaznog razdoblja (do kraja 2018.) na oznakama i informacijama u prodavaonicama trebale su se koristiti samo NEDC vrijednosti kako bi se potrošačima omogućila usporedba različitih automobila. Državni porezni propisi i dalje su temeljeni na vrijednostima NEDC-a. Od rujna 2018. svi novi automobili morali su biti certificirani prema WLTP testnom postupku. Od siječnja 2019. svi automobili u trgovinama trebali su imati vrijednosti samo WLTP-CO_2 bi se izbjegla zabuna među potrošačima. Nacionalne vlade trebale su prilagoditi oporezivanje vozila i fiskalne poticaje vrijednostima [12].

2. CO_2MPAS

Europska komisija morala je izmijeniti europske propise zbog uvođenja WLTP testnog ciklusa u homologaciju lakih gospodarskih i osobnih vozila. Novi postupak ispitivanja, zamijenio je stari postupak homologacije, temeljen na zastarjelom NEDC testnog ciklusu. NEDC ciklus je zastario zbog svoje manje sposobnosti da pruži pouzdane procjene potrošnje goriva i stvaranja emisija plinova novih vozila, pokazujući postojanje rastuće razlike između vrijednosti homologacije i vrijednosti pri svakodnevnoj uporabi.

Kako su NEDC i WLTP ciklusi različiti, izmjerene vrijednosti razlikovale su se jedna od druge. Problem je nastao zbog toga što se iznos emisija ugljikovog dioksida iz tih testova koristi u mnogim zemljama za utvrđivanje troškova trošarina na nove automobile. Sa zakonodavne strane, postizanje tog cilja bilo je vrlo problematično. Proizvođačima vozila zadane su ciljane vrijednosti koje predstavljaju najveću dopuštenu prosječnu specifičnu emisiju CO_2 (u g/km) za vozila registrirana u Europi. Proizvođači koji ne bi uspjeli postići te vrijednosti morali bi platiti premije za svaki gram za svako vozilo kojim bi se ta vrijednost premašila. Ciljane vrijednosti bile su zadane za 2015. i 2021. godinu, a iznosile su 130 g CO_2/km, odnosno 95 g CO_2/km.

Razdoblje prijelaza testiranja s NEDC-a na WLTP ciklus započelo je 2017. godine. Proizvođači automobila morali su od 1. rujna 2017. godine, zbog sporog zakonskog prijelaza, imati deklarirane vrijednosti i WLTP i NEDC testnog postupka za svako novo vozilo. Zbog toga što je testiranje skupo i dugotrajno, te zbog toga što se u WLTP testnoj proceduri moraju uzeti u obzir različite konfiguracije vozila, nove vrijednosti emisija CO_2 izmjerene WLTP postupkom, pretvarale su se u vrijednosti NEDC postupka. Za pretvaranje tih vrijednosti, u JRC-u (*Joint Research Centre*) napravljen je CO_2MPAS (*CO_2 Model for PAssenger and commercial vehicles Simulation*) alat.

CO_2MPAS je software alat otvorenog kôda razvijen u Python programskom jeziku. Radi ili pomoću terminala ili kao aplikacija s grafičkim sučeljem. Koristi svoje unaprijed napravljene Excel datoteke unos ulaznih podataka i ispis izračunatih rezultata.

Da bi postigao dovoljnu točnost za regulatornu primjenu, program bi trebao imati razne podatke o cijelom vozilu. Međutim, iz praktičnih razloga, budući da je program namijenjen za upotrebu prilikom homologacije vozila, ne koriste se detaljne informacije koje se obično zahtijevaju u modelima kada se rade simulacije za istraživanje i razvoj. Stoga, iako dijeli iste fizikalne principe, u smislu uzdužne dinamike vozila, s drugim simulacijskim modelima, u CO_2MPAS-u su usvojeni različiti pristupi kako bi se riješio nedostatak velikog složenog skupa informacija. Oni uključuju izdvajanje podataka iz službenog WLTP testa, empirijske modele koji se odnose

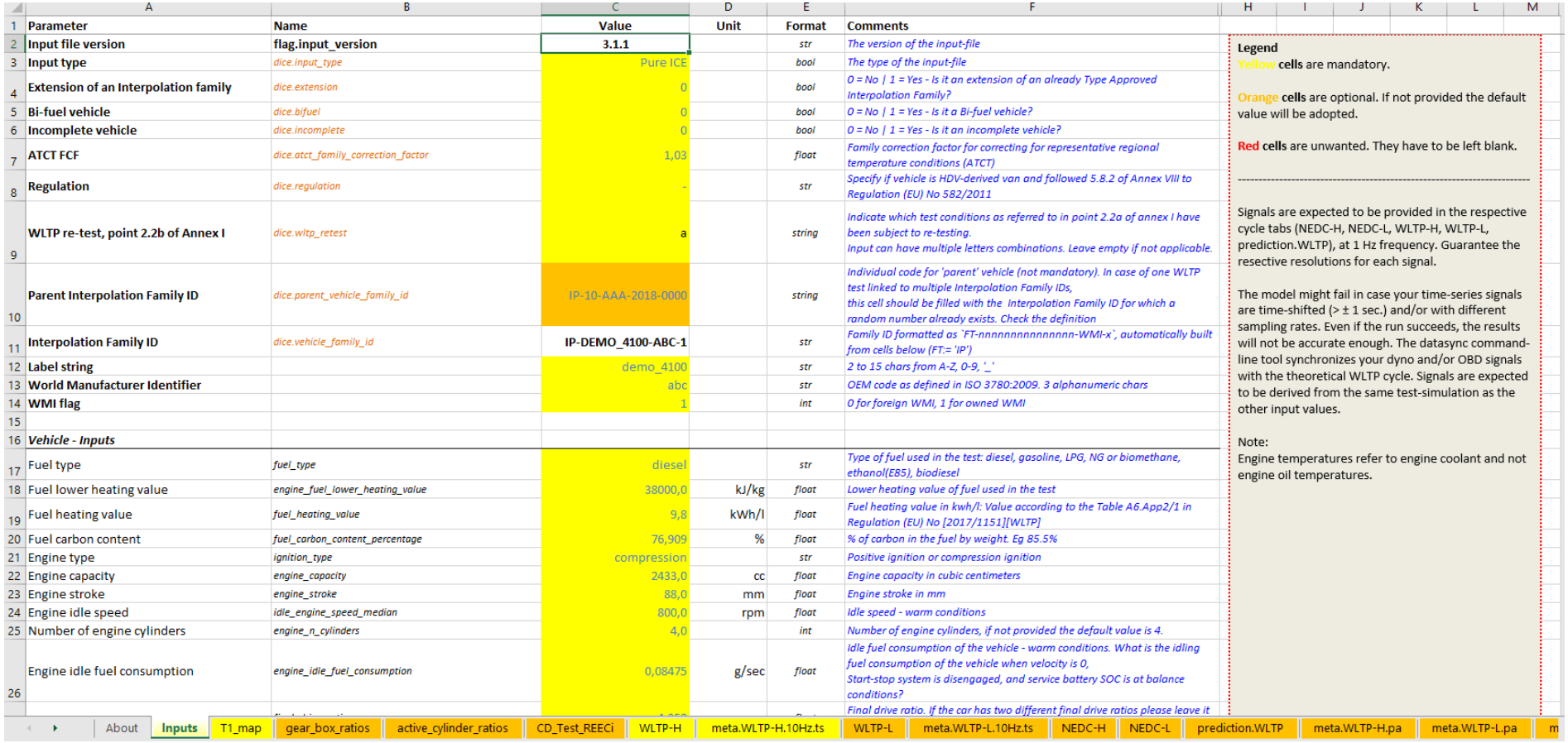
na rad različitih komponenti dobivenih iz već postojeće baze podataka vozila i dijelova i algoritme strojnog učenja. Sve ove značajke čine CO_2MPAS sličnijim modelima koji se koriste u simulacijama za optimizaciju. Ovaj pristup omogućio je da se na razini vozila zabilježe performanse potrošnje goriva, odnosno energije s točnošću od 2,0 % u usporedbi sa stvarnim ispitivanjima na valjcima, uz ograničene informacije dostupne tijekom homologacije vozila (uključujući rezultate WLTP testa). CO_2MPAS-ov dio programa za sinkronizaciju uz pomoć podataka izmjerenih iz certifikacijskih testova je glavna razlika CO_2MPAS-a u odnosu na već postojeće i složenije modele simulacije vozila koji se koriste u potrebe istraživanja i razvoja.

2.1. Princip rada

Program CO_2MPAS simulira vrijednost NEDC emisije CO_2 za dano vozilo u potpunosti usklađenu s WLTP-NEDC uredbom o korelaciji. Program radi po principu simulacije fizičkog modela koji se temelji na standardnoj uzdužnoj dinamici vozila i simulaciji potrošnje energije. Najvažniji čimbenici koji utječu na izračun emisije ugljikovog dioksida su:

- točan izračun potrebne snage;
- ponašanje u vožnji (mijenjanje brzina, obrasci ubrzanja);
- pogon i učinkovitost pogonskog sklopa;
- pokretanje hladnog motora i temperaturni uvjeti;
- upravljanje sekundarnim sustavima, tj. izvorima napajanja i trošilima.

Podaci o vozilu i rezultati WLTP testiranja se unose u već pripremljenu Excel datoteku prikazanu na slici 4.



	A	B	C	D	E	F
1	**Parameter**	**Name**	**Value**	**Unit**	**Format**	**Comments**
2	**Input file version**	**flag.input_version**	**3.1.1**		*str*	*The version of the input-file*
3	**Input type**	*dice.input_type*	Pure ICE		*bool*	*The type of the input-file*
4	**Extension of an Interpolation family**	*dice.extension*	0		*bool*	*0 = No | 1 = Yes - Is it an extension of an already Type Approved Interpolation Family?*
5	**Bi-fuel vehicle**	*dice.bifuel*	0		*bool*	*0 = No | 1 = Yes - Is it a Bi-fuel vehicle?*
6	**Incomplete vehicle**	*dice.incomplete*	0		*bool*	*0 = No | 1 = Yes - Is it an incomplete vehicle?*
7	**ATCT FCF**	*dice.atct_family_correction_factor*	1,03		*float*	*Family correction factor for correcting for representative regional temperature conditions (ATCT)*
8	**Regulation**	*dice.regulation*	-		*str*	*Specify if vehicle is HDV-derived van and followed 5.8.2 of Annex VIII to Regulation (EU) No 582/2011*
9	**WLTP re-test, point 2.2b of Annex I**	*dice.wltp_retest*	a		*string*	*Indicate which test conditions as referred to in point 2.2a of annex I have been subject to re-testing.* *Input can have multiple letters combinations. Leave empty if not applicable.*
10	**Parent Interpolation Family ID**	*dice.parent_vehicle_family_id*	IP-10-AAA-2018-0000		*string*	*Individual code for 'parent' vehicle (not mandatory). In case of one WLTP test linked to multiple Interpolation Family IDs,* *this cell should be filled with the Interpolation Family ID for which a random number already exists. Check the definition*
11	**Interpolation Family ID**	*dice.vehicle_family_id*	**IP-DEMO_4100-ABC-1**		*str*	*Family ID formatted as `FT-nnnnnnnnnnnnnnn-WMI-x`, automatically built from cells below (FT:= 'IP')*
12	**Label string**		demo_4100		*str*	*2 to 15 chars from A-Z, 0-9, '_'*
13	**World Manufacturer Identifier**		abc		*str*	*OEM code as defined in ISO 3780:2009. 3 alphanumeric chars*
14	**WMI flag**		1		*int*	*0 for foreign WMI, 1 for owned WMI*
15						
16	***Vehicle - Inputs***					
17	Fuel type	*fuel_type*	diesel		*str*	*Type of fuel used in the test: diesel, gasoline, LPG, NG or biomethane, ethanol(E85), biodiesel*
18	Fuel lower heating value	*engine_fuel_lower_heating_value*	38000,0	kJ/kg	*float*	*Lower heating value of fuel used in the test*
19	Fuel heating value	*fuel_heating_value*	9,8	kWh/l	*float*	*Fuel heating value in kwh/l: Value according to the Table A6.App2/1 in Regulation (EU) No [2017/1151][WLTP]*
20	Fuel carbon content	*fuel_carbon_content_percentage*	76,909	%	*float*	*% of carbon in the fuel by weight. Eg 85.5%*
21	Engine type	*ignition_type*	compression		*str*	*Positive ignition or compression ignition*
22	Engine capacity	*engine_capacity*	2433,0	cc	*float*	*Engine capacity in cubic centimeters*
23	Engine stroke	*engine_stroke*	88,0	mm	*float*	*Engine stroke in mm*
24	Engine idle speed	*idle_engine_speed_median*	800,0	rpm	*float*	*Idle speed - warm conditions*
25	Number of engine cylinders	*engine_n_cylinders*	4,0		*int*	*Number of engine cylinders, if not provided the default value is 4.*
26	Engine idle fuel consumption	*engine_idle_fuel_consumption*	0,08475	g/sec	*float*	*Idle fuel consumption of the vehicle - warm conditions. What is the idling fuel consumption of the vehicle when velocity is 0,* *Start-stop system is disengaged, and service battery SOC is at balance conditions?*

Legend
Yellow **cells** are mandatory.

Orange **cells** are optional. If not provided the default value will be adopted.

Red **cells** are unwanted. They have to be left blank.

Signals are expected to be provided in the respective cycle tabs (NEDC-H, NEDC-L, WLTP-H, WLTP-L, prediction.WLTP), at 1 Hz frequency. Guarantee the resective resolutions for each signal.

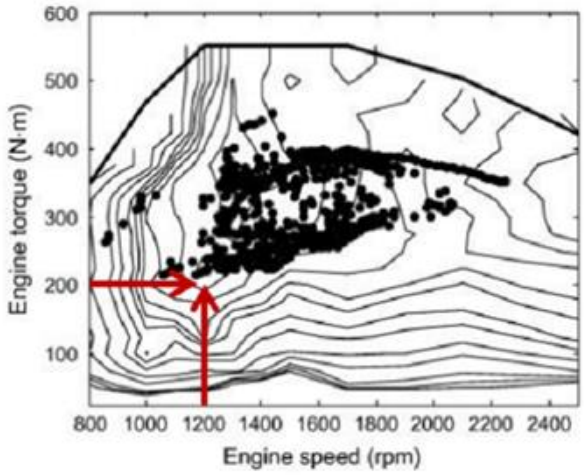
The model might fail in case your time-series signals are time-shifted (> ± 1 sec.) and/or with different sampling rates. Even if the run succeeds, the results will not be accurate enough. The datasync command-line tool synchronizes your dyno and/or OBD signals with the theoretical WLTP cycle. Signals are expected to be derived from the same test-simulation as the other input values.

Note:
Engine temperatures refer to engine coolant and not engine oil temperatures.

About | Inputs | T1_map | gear_box_ratios | active_cylinder_ratios | CD_Test_REECi | WLTP-H | meta.WLTP-H.10Hz.ts | WLTP-L | meta.WLTP-L.10Hz.ts | NEDC-H | NEDC-L | prediction.WLTP | meta.WLTP-H.pa | meta.WLTP-L.pa | m

Slika 4. Excel datoteka s ulaznim podacima

U slučaju viška ili manjka potrebnih informacija, program daje upozorenje i prekida simulaciju. Osim podataka o vozilu, potrebni su i rezultati WLTP testa. Potrebni su podaci izmjereni na valjcima za testiranje i podaci iz računala vozila, kao što su vrijeme, brzina kretanja s valjaka i iz vozila, brzina vrtnje motora, temperatura rashladne tekućine, jakost potrebne struje i opterećenje motora. Zbog različitih izvora tih podataka, potrebno ih je uskladiti, tj. sinkronizirati. Sinkronizacija podataka ključna je za valjane rezultate. CO_2MPAS-ov alat za sinkronizaciju koristi zajednički podatak kao referencu. Alat za sinkronizaciju, u skladu s referentnim podacima, mijenja i usklađuje ostale podatke.



Moment / Brzina vrtnje motora	10	50	100	150	200
1000	1	3	4	5	6
1200	2	4	6	8	10
1400	5	7	9	11	13
1600	6	8	10	12	14
1800	8	10	12	14	16

Slika 5. Prikaz radne točke motora u mapi motora i tablici

Mapa emisija motora može se gledati kao tablica koja sadrži vrijednosti potrošnje goriva ili emisija plinova za moguće kombinacije brzine vrtnje motora i npr. momenta. Uz pomoć toga, vozilu je moguće odrediti količinu proizvedenih emisija ili potrošenog goriva na nekom zadanom koraku (npr. za svaku sekundu). Time se eliminira potreba fizičkog motora za daljnje testiranje vozila ukoliko se vozilu promijeni neka od utjecajnih komponenti.

Potrebna snaga za savladavanje ukupnog opterećenja, izračunava se na temelju pogonskih otpora, gubitaka snage u pogonskom sustavu i potrošnje energije pomoćnih jedinica vozila prema formuli:

$$P = P_{\text{kotrljanje}} + P_{\text{zrak}} + P_{\text{akceleracija}} + P_{\text{uspon}} + P_{\text{transmisija}} + P_{\text{ostalo}} \qquad (1)$$

Brzina vrtnje motora *n*, može se izračunati na temelju poznate brzine vozila i prijenosnih omjera transmisije i podataka o kotaču:

$$n = \frac{v * 60 * i_{\mathrm{OR}} * i_{\mathrm{m},i}}{d_{\text{kotača}} * \pi} \qquad (2)$$

gdje su:

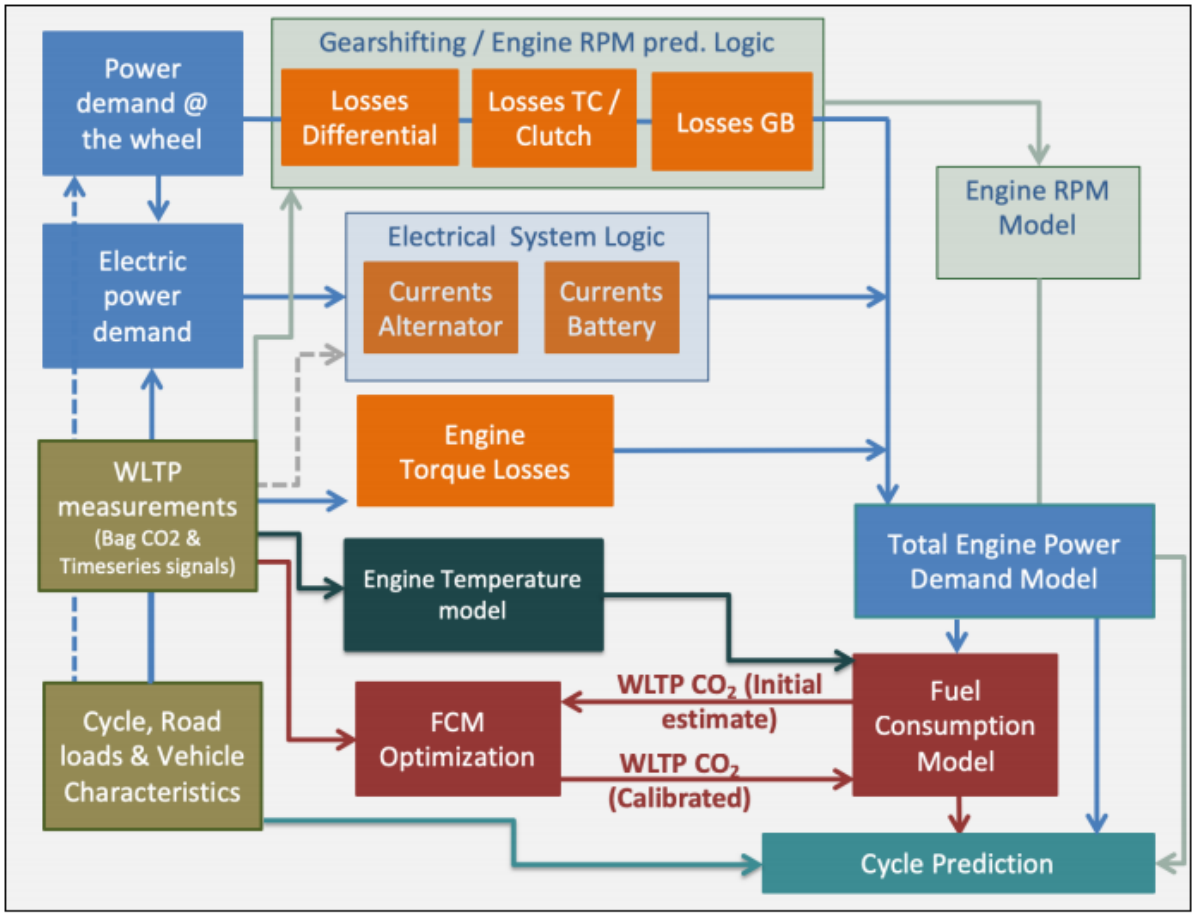
v - brzina vozila, km/h,

i_{OR} – osovinska redukcija, -,

$i_{\mathrm{m},i}$ – prijenosni omjer stupnja u mjenjaču, -,

$d_{\text{kotača}}$ – promjer gume (kotača), m.

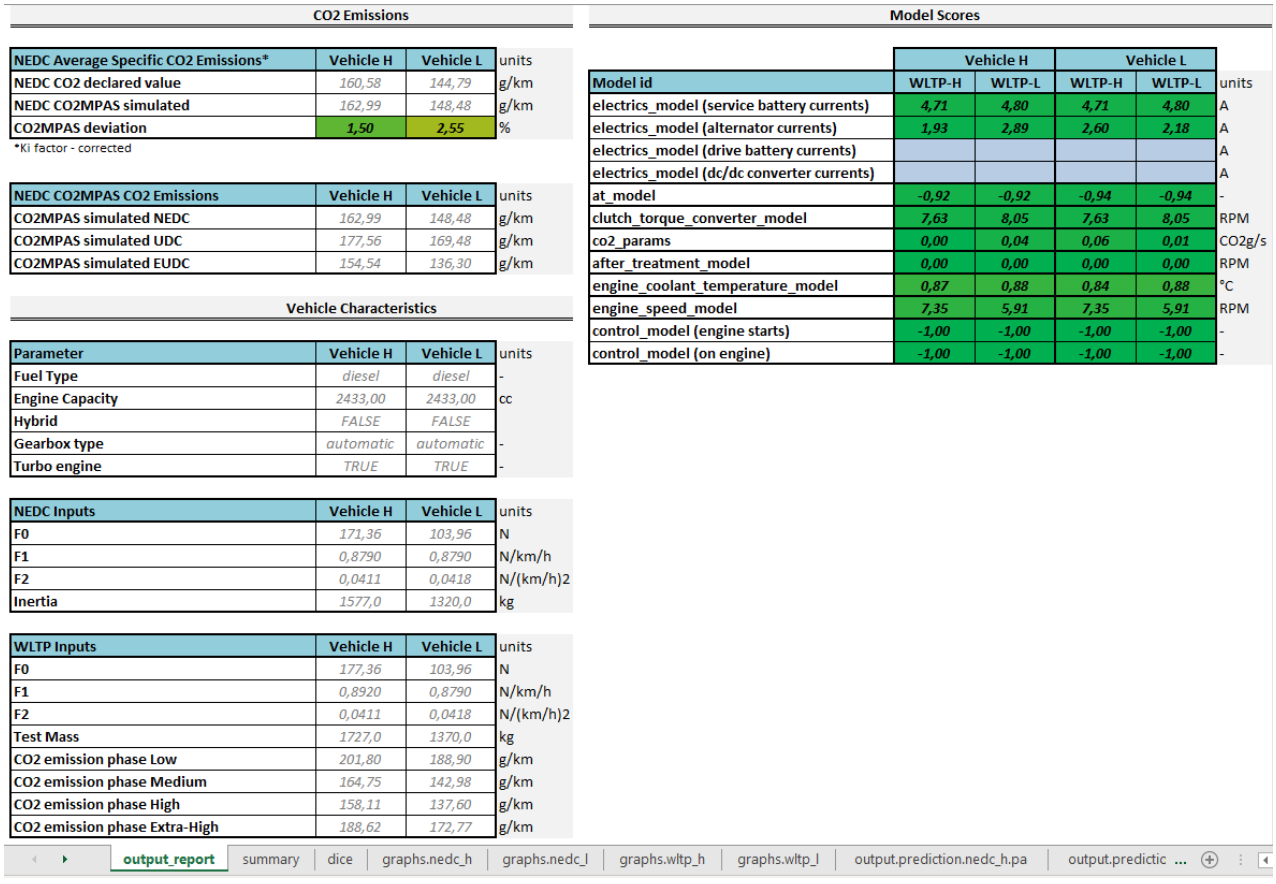
CO_2MPAS alat sastoji se od nekoliko potprograma (modela) o kojima ovisi izračun potrebne snage. Oni su prikazani na slici 6.



Slika 6. Shematski prikaz toka podataka u potprogramima [15]

Program iterira vrijednosti, te primjenjuje vrijednosti u modelima koji daju najbolje rezultate. Ti modeli su:

- Model automatskog mjenjača (promjena stupnja prijenosa)
- Model spojke i pretvarača okretnog momenta
- Model brzine vrtnje pokretanja hladnog motora
- Model brzine vrtnje motora
- Start-stop model
- Model alternatora
- Model temperature rashladne tekućine u motoru
- Model potrošnje goriva i emisija CO_2.



CO2 Emissions

NEDC Average Specific CO2 Emissions*	Vehicle H	Vehicle L	units
NEDC CO2 declared value	160,58	144,79	g/km
NEDC CO2MPAS simulated	162,99	148,48	g/km
CO2MPAS deviation	1,50	2,55	%

*Ki factor - corrected

NEDC CO2MPAS CO2 Emissions	Vehicle H	Vehicle L	units
CO2MPAS simulated NEDC	162,99	148,48	g/km
CO2MPAS simulated UDC	177,56	169,48	g/km
CO2MPAS simulated EUDC	154,54	136,30	g/km

Vehicle Characteristics

Parameter	Vehicle H	Vehicle L	units
Fuel Type	diesel	diesel	-
Engine Capacity	2433,00	2433,00	cc
Hybrid	FALSE	FALSE	
Gearbox type	automatic	automatic	-
Turbo engine	TRUE	TRUE	-

NEDC Inputs	Vehicle H	Vehicle L	units
F0	171,36	103,96	N
F1	0,8790	0,8790	N/km/h
F2	0,0411	0,0418	N/(km/h)2
Inertia	1577,0	1320,0	kg

WLTP Inputs	Vehicle H	Vehicle L	units
F0	177,36	103,96	N
F1	0,8920	0,8790	N/km/h
F2	0,0411	0,0418	N/(km/h)2
Test Mass	1727,0	1370,0	kg
CO2 emission phase Low	201,80	188,90	g/km
CO2 emission phase Medium	164,75	142,98	g/km
CO2 emission phase High	158,11	137,60	g/km
CO2 emission phase Extra-High	188,62	172,77	g/km

Model Scores

	Vehicle H		Vehicle L		
Model id	WLTP-H	WLTP-L	WLTP-H	WLTP-L	units
electrics_model (service battery currents)	4,71	4,80	4,71	4,80	A
electrics_model (alternator currents)	1,93	2,89	2,60	2,18	A
electrics_model (drive battery currents)					A
electrics_model (dc/dc converter currents)					A
at_model	-0,92	-0,92	-0,94	-0,94	-
clutch_torque_converter_model	7,63	8,05	7,63	8,05	RPM
co2_params	0,00	0,04	0,06	0,01	CO2g/s
after_treatment_model	0,00	0,00	0,00	0,00	RPM
engine_coolant_temperature_model	0,87	0,88	0,84	0,88	°C
engine_speed_model	7,35	5,91	7,35	5,91	RPM
control_model (engine starts)	-1,00	-1,00	-1,00	-1,00	-
control_model (on engine)	-1,00	-1,00	-1,00	-1,00	-

output_report | summary | dice | graphs.nedc_h | graphs.nedc_l | graphs.wltp_h | graphs.wltp_l | output.prediction.nedc_h.pa | output.predictic ...

Slika 7. Prikaz izlaznih rezultata CO_2MPAS programa

3. VECTO

S ciljem smanjenja emisije CO_2 iz cestovnog prometa, Europska komisija napravila je novi postupak za certificiranje emisije CO_2 iz teških gospodarskih vozila. Osnovni pristup tog postupka certificiranja baziran je na ispitivanjima pojedinih dijelova vozila i naknadnoj simulaciji potrošnje goriva i emisije CO_2 na cijeloj kompoziciji teškog gospodarskog vozila. Takav pristup nudi mogućnost točnog opisa vrlo različitih vozila različitih namjena i njihovog utjecaja na potrošnju goriva i emisiju CO_2, bez povećanja složenosti ispitivanja i troškova za certificiranje. Konfiguracija vozila predstavlja najveći problem certificiranja takvih vozila. Zbog vrlo specifičnih primjena, takva vozila se proizvode prema željama kupaca, obično u manjem broju primjeraka. Zbog tog razloga, iznimno je nepraktično, dugotrajno i skupo testirati svako vozilo zasebno. Još jedan od razloga razvoja takvog postupka bio je i pojednostavljivanje cijele metode certificiranja CO_2 za teška gospodarska vozila prema zakonodavnom postupku.
Cijeli postupak certificiranja vozila zasniva se na razvoju softvera koji eliminira fizičko ispitivanje vozila. Program VECTO (engl. *Vehicle Energy Consumption calculation Tool*, Alat za proračun potrošnje energije u vozilu) zamišljen je na principu da računa emisiju CO_2 i potrošnju goriva na temelju simulacije uzdužne dinamike vozila.
Da bi se dobili valjani postupci ispitivanja i prateći program, moralo se uzeti u obzir:

- Razvoj odgovarajućih metoda za testiranje pojedinih komponenata, te za provjere valjanosti i simulaciju
- Razvoj softvera potrebnog za ispitivanje vozila
- Izrada zadanih ulaznih podataka i generičkih vrijednosti potrebnih za taj softver

Rad na tome je proveden u uskoj suradnji s industrijom kako bi se zajedničkim istraživanjem osiguralo učinkovito korištenje resursa i jamčilo prihvaćanje takvog postupka certificiranja.

3.1. Princip rada

Program VECTO zasnovan je na „modeliranju unatrag", pri čemu se brzina i akceleracija vozila daju kao ulaz u proračun uzdužne dinamike vozila, a kao rezultat se izračunavaju opterećenja i okretni moment u pogonskom sustavu vozila. Brzina i ubrzanje vozila definirani su simuliranim ciklusom vožnje (ciljana brzina, nagib ceste) i modelom vozača. Potrebna snaga motora s unutarnjim izgaranjem izračunava se na temelju pogonskih otpora, gubitaka snage u pogonskom sustavu i potrošnje energije ostalih sustava vozila. Brzina vrtnje motora određuje se na temelju modela promjene stupnja prijenosa, prijenosnih omjera i promjera kotača.

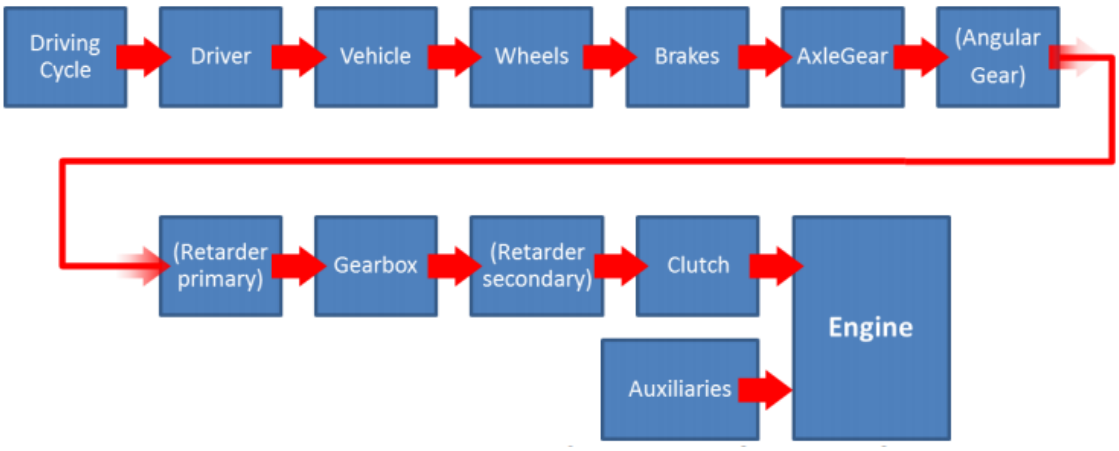
Rezultati se potom u izlaznom modulu koji na kraju generira izvješća koja sadrže opis vozila, parametre koji ukazuju na vozne performanse vozila u različitim ciklusima vožnje i potrošnju goriva, odnosno emisiju CO_2.

3.1.1. Dijelovi vozila i parametri modela

Svaka komponenta relevantna za potrošnju energije pogonskog sklopa vozila modelirana je kao zasebna komponenta u alatu za simulaciju. Za svaku su komponentu namijenjena zasebna sučelja u programu. To omogućava sastavljanje različitih konfiguracija pogonskog sklopa sve dok su povezane komponente međusobno kompatibilne.

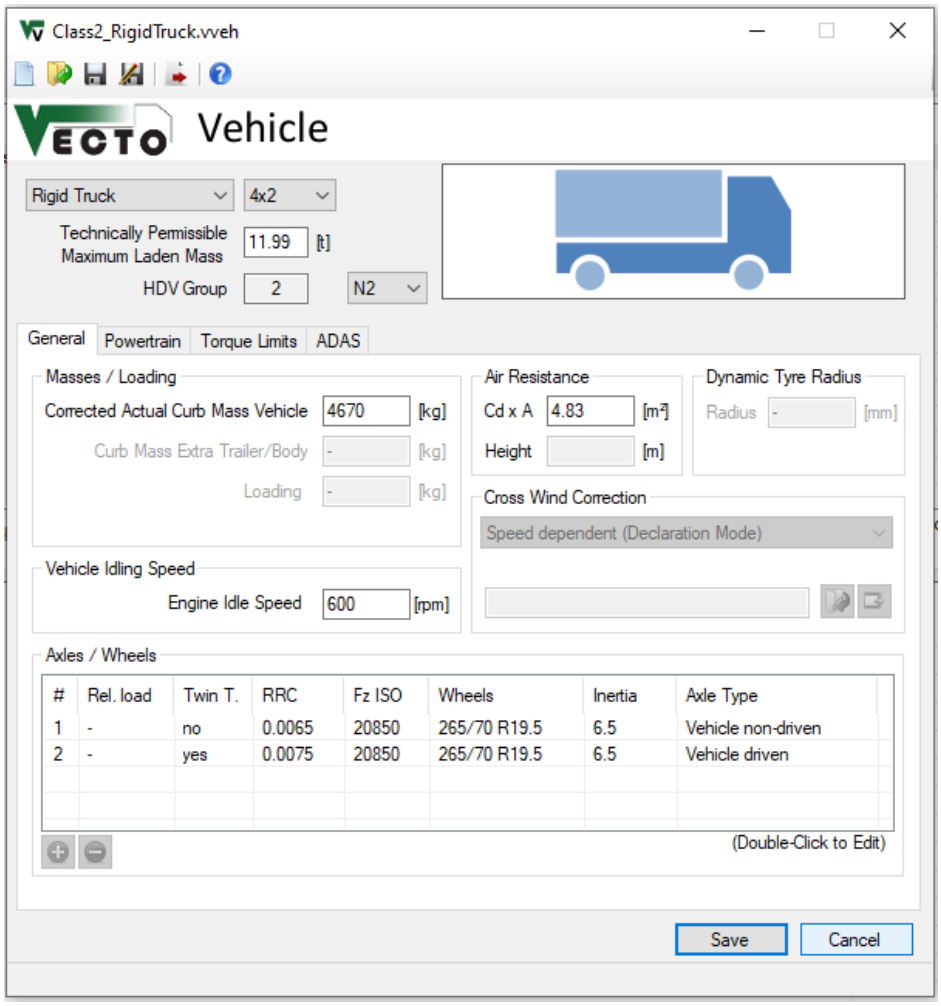
Minimalna konfiguracija prikazana je na slici 8., a sadrži sljedeće komponente:

- vozni ciklus,
- vozač,
- vozilo,
- kotači,
- kočnice,
- osovinski prijenos,
- usporivač (engl. *retarder*),
- mjenjač,
- spojka,
- motor s unutarnjim izgaranjem.



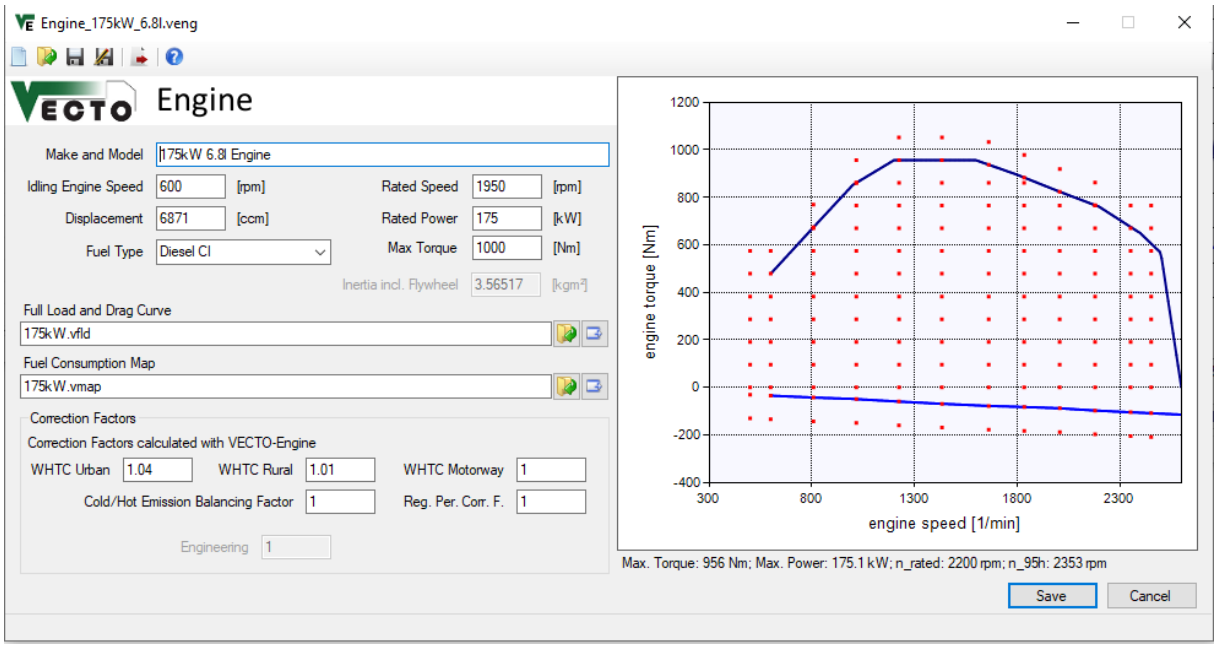
Slika 8. Elementi minimalne konfiguracija vozila [22]

Na slici 9. prikazano je sučelje programa u kojem se definiraju podaci vozilu.



Slika 9. Konfiguracija vozila u programu VECTO

Svaka komponenta sadrži podatke s vlastitim parametrima modela. Parametri modela su npr. mape gubitaka, mape potrošnje goriva ili generičke vrijednosti uobičajene za zadanu komponentu. Ovaj komponentni prikaz pogonskog sklopa omogućuje sastavljanje modularnog simulacijskog modela. Na slici 10. prikazano je sučelje programa u kojem se definiraju podaci o motoru.



Slika 10. Konfiguracija motora u programu VECTO

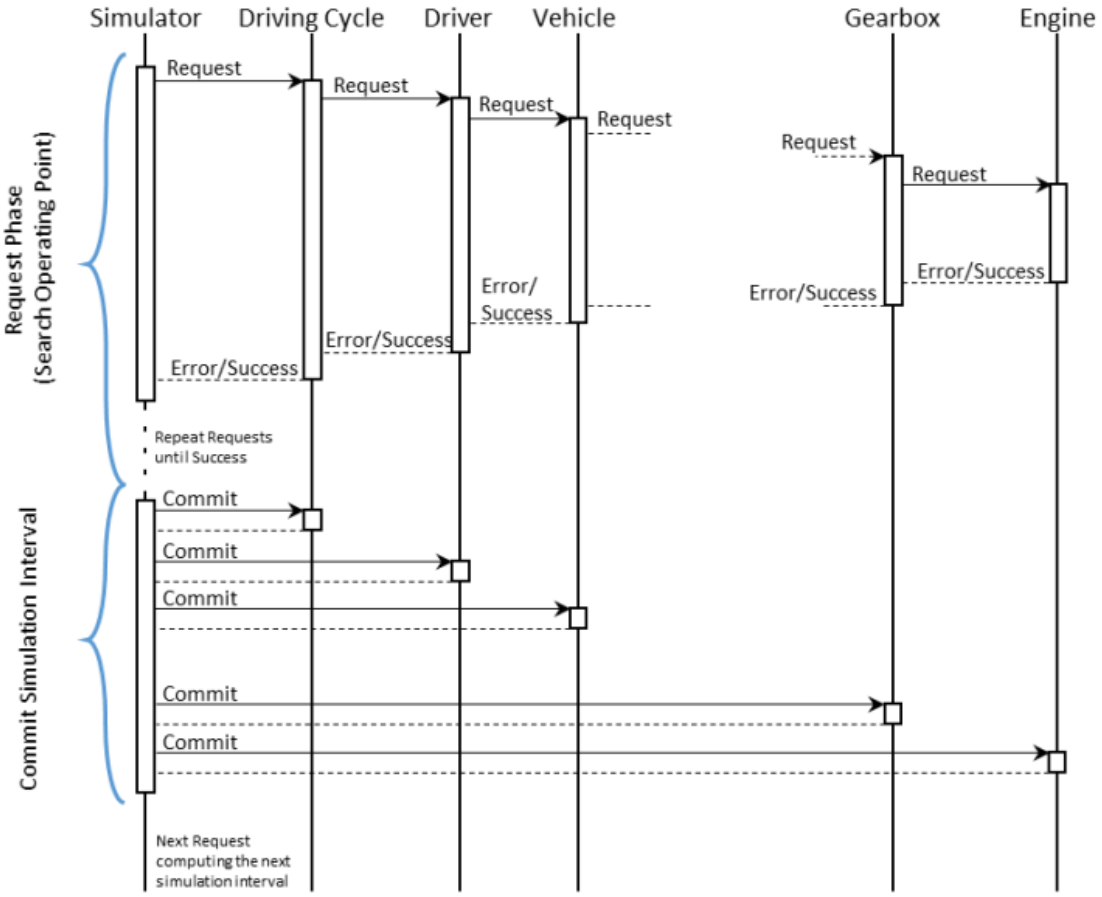
3.1.2. Izračun vrijednosti za zadani ciklus vožnje

Vozni ciklus definira se kao ciljna brzina na nekoj udaljenosti. U simulaciji vozilo mora pratiti ciljnu brzinu što je moguće bolje, a važno je da se svako vozilo simulira na identičnom prijeđenom putu. Dakle, ciklus vožnje podijeljen je na vremenske korake. Duljina koraka simulacije prilagođava se ovisno o trenutnoj brzini vozila tako da korak simulacije pokriva približno 0,5 sekundi. U komponenti vozača preračunava se iz prostorne domene u vremensku domenu, pa svaki korak simulacije simulira male udaljenosti. U pogonskim komponentama se računa u vremenskoj domeni. Modul vozača određuje ubrzanje vozila za zadanu simulacijsku udaljenost. Uz trenutnu brzinu vozila i ubrzanje za trenutni korak simulacije modul vozača može izračunati vrijeme potrebno za prelazak zadane udaljenosti simulacije.

Svaki korak simulacije podijeljen je u dvije faze. U prvoj fazi mora se pronaći realna radna točka za sve komponente pogonskog sklopa. To u osnovi znači da se ubrzanje vozila mora prilagoditi tako da za radnu točku postoji realna brzina vrtnje motora i okretni moment. Druga faza je dovršiti trenutni korak simulacije, tj. zapisati trenutni korak simulacije u rezultate, ažurirati stanje svake komponente pogonskog sklopa i prijeći na sljedeći korak simulacije.

Traženje radne točke pogonskog pogona vrši se davanjem zahtjeva za potencijalnu radnu točku koji započinje u voznom ciklusu radi simuliranja određene zadane udaljenosti. Ovaj zahtjev se zatim prosljeđuje na sve komponente pogonskog sklopa. Svaka komponenta preračunava fizičku veličinu. Na primjer, zahtjev upućen komponenti kotača sadrži brzinu i silu koja se pretvara u kutnu brzinu i okreni moment za sljedeću komponentu. Svaka komponenta obično

dodaje svoje gubitke i zahtjev prosljeđuje na sljedeću komponentu. U idealnom slučaju, zahtjev rezultira valjanom radnom točkom za sve komponente, a komponenta motora potvrđuje radnu točku. U slučaju da je u zahtjevu traženi okretni moment prevelike ili premale vrijednosti ili bilo koja druga komponenta ne može ispuniti traženu vrijednost za radnu točku, postupak se ponavlja ovisno o tom zahtjevu. Na slici 11. prikazan je shematski prikaz traženja realne radne točke na temelju slanja zahtjeva svakom elementu konfiguracije vozila.



Slika 11. Shematski prikaz potvrđivanja ili odbijanja zahtjeva za neku radnu točku [22]

Ovisno o zahtjevu, modul vozača (točnije zadana strategija vožnje) mora odlučiti kako se može pronaći valjana radna točka. Ako zahtjev ima preveliki traženi okretni moment, ubrzanje se može smanjiti ili ako je okretni moment premali, potrebno je koristiti kočnice. U oba slučaja potrebno je pronaći snagu ubrzanja ili kočnice tako da se rezultirajuća radna točka motora nalazi na ili ispod crte punog opterećenja.

Za simulaciju koraka $[s_i, s_{i+1}]$ (odnosno $[t_i, t_{i+1}]$) stanje svake komponente na početku simulacije je poznato. Na početku simulacije vozilo obično miruje, a motor radi u praznom hodu. Iz ovog

stanja izračunava se sljedeće stanje s_{i+1}. Opća pretpostavka za proračun nekog koraka simulacije je konstantno ubrzanje. To znači da su brzina vozila, kutna brzina duž pogonskog sklopa linearne funkcije, a okretni moment je konstantan. Međutim, ova pretpostavka ne vrijedi za svaku komponentu. Otpor zraka ovisi o kvadratu brzine vozila i budući da je brzina linearna funkcija, sila otpora zraka općenito nije konstantna unutar simulacijskog intervala. Za takve komponente VECTO primjenjuje prosječnu silu za ekvivalentnu energiju. To znači da VECTO integrira gubitak snage zbog otpora zraka tijekom cijelog simulacijskog koraka i izračunava prosječnu silu otpora zraka koristeći prosječnu brzinu vozila u tom koraku simulacije. To se može analitički izračunati jer su poznati odnosi tih veličina. Za ostale nelinearne komponente poput pretvarača okretnog momenta upotrebljava se slična metoda. Ako se odnos ne može analitički opisati, koristi se linearna aproksimacija. Gubici nastali na svakoj komponenti prilikom slanja zahtjeva izračunavaju se ili pregledavaju u mapi gubitaka s prosječnom potrošnjom energije u trenutnom koraku simulacije.

4. Jednostavan alat za određivanje potrošnje goriva i emisija motora za zadani radni ciklus

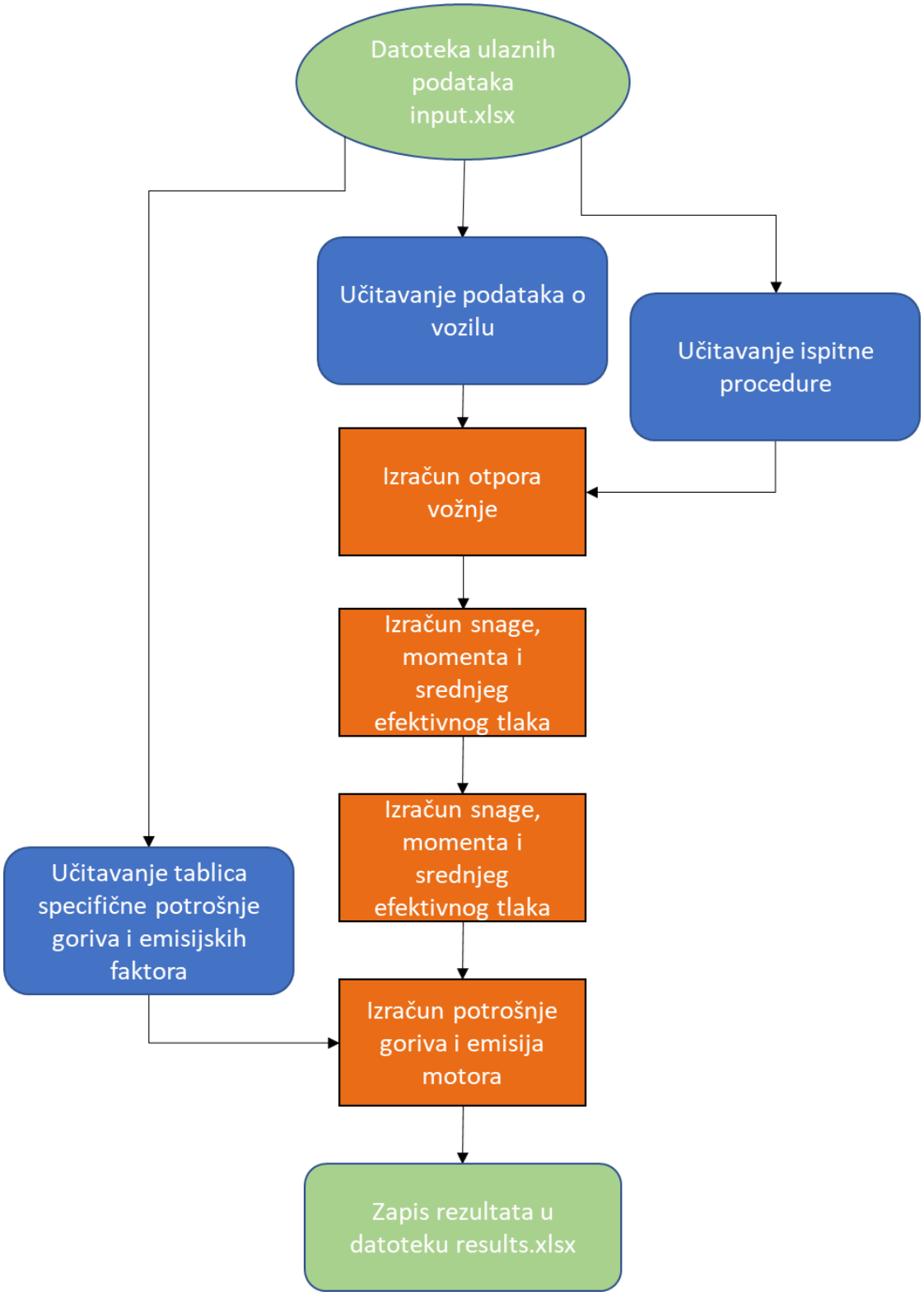
Pri razvoju motora s unutarnjim izgaranjem velika pažnja se posvećuje emisijama koje se stvaraju pri njihovom radu. Za vrijeme rada motora, konstrukcijske značajke ostaju nepromijenjene. Način uporabe motora jedina je promjenjiva stavka koja utječe na količinu nastalih emisija. Način uporabe motora može se odrediti snimanjem radnih parametara motora i mjerenjem potrošnje goriva za vrijeme eksploatacije, no zbog dugotrajnosti i skupoće u tu svrhu se danas koriste razni simulacijski programi.

U okviru ovog diplomskog rada, napravljen je računalni program koji izračunava nastale emisije iz vozila i računa ukupnu potrošnju goriva na zadanom radnom ciklusu. Rad vozila se prati pomoću unesenih podataka o brzini vozila i vremenu. Neovisno o unesenim podacima, režim rada se prati u intervalima od jedne sekunde (frekvencijom 1 Hz).

Program je namijenjen za izračun potrošnje goriva i nastalih emisija sa određenim ograničenjima. Za točan izračun pretpostavlja se rad zagrijanog motora pri preporučenoj uporabi, što podrazumijeva rad motora u poznatim radnim točkama. Program se može podijeliti u četiri veća dijela. Prvi dio sastoji se od učitavanja ulaznih podataka. Osim podataka o ispitnoj proceduri i režimu rada, potrebni su i određeni podaci o vozilu. Drugi dio programa računa opterećenje motora s obzirom na unesene podatke o vozilu i željenom načinu korištenja vozila. Željeni način korištenja motora određuje se unošenjem podataka o željenoj brzini vrtnje motora za prebacivanje u viši, odnosno niži stupanj prijenosa. Opterećenje motora izračunava se na temelju otpora koje vozilo treba svladati. Pomoću njih se određuju radne značajke motora i vozila, kao što su potrebna snaga motora u danom trenutku, moment motora i srednji efektivni tlak. Treći dio programa izračunava vrijednosti stvorenih emisija motora uz pomoć tablica specifične potrošnje goriva i emisijskih faktora. Četvrti i zadnji dio je spremanje svih izračunatih vrijednosti. Na slici 12. prikazana je shema rada programa.

Program je napisan u programskom jeziku Python, a za unos podataka odabrana je i napravljena tablična .xlsx datoteka. Za spremanje rezultata također se koristi .xlsx datoteka.

U narednom dijelu rada biti će prikazan i objašnjen dio kôda programa i metode izračuna traženih vrijednosti.



Slika 12. Shema rada programa

4.1. Unos podataka

Za unos podataka namijenjena je datoteka *input.xlsx*. Iz nje program učitava sve unesene vrijednosti kao varijable. Tražene vrijednosti potrebno je unijeti na za to predviđene ćelije zbog toga što .xlsx datoteke unesene vrijednosti zapisuju po koordinatama ćelija.

Ulazna datoteka podijeljena je na osam listova (engl. *sheet*). Prvi list, prikazan na slici 13., nazvan je *Vehicle info* i u njega se upisuju tražene opće karakteristike vozila. Osim podataka o vozilu, potrebno je unijeti i željeni način korištenja vozila, tj. željene brzine vrtnje kod promjena stupnja prijenosa u mjenjaču.

	A	B	C	D	E	F	G	H	I
1	Vehicle input data								
2	Alfa Romeo Mito								
3	2.0 TDI								
4									
5	Vehicle mass	m	kg	1300		Axle reduction	i_{axle}	2,64829	
6	Front surface of vehicle	A	m^2	1,96		Gearbox reduction ratios:			
7	Air resistance coefficient	c_w	-	0,29		1. gear reduction ratio	$i_{m,1}$	3,81	
8	Air density	ρ_z	kg/m^3	1,204		2. gear reduction ratio	$i_{m,2}$	2,16	
9	Gravity	g	m/s^2	9,81		3. gear reduction ratio	$i_{m,3}$	1,48	
10	Mechanica gearboxl efficiency coefficient	η_m	-	0,9		4. gear reduction ratio	$i_{m,4}$	1,07	
11	Engine volume	V	m^3	0,001968		5. gear reduction ratio	$i_{m,5}$	0,88	
12	Strokes	T	-	4		6. gear reduction ratio	$i_{m,6}$	0,74	
13	Max. Engine power	$P_{e,max}$	kW	103		7. gear reduction ratio	$i_{m,7}$		
14	Tyre dimension	example: 205/45		R17		8. gear reduction ratio	$i_{m,8}$		
15		205	45	17		9. gear reduction ratio	$i_{m,9}$		
16						...	...		
17									
18									
19	Upshift engine speed	min^{-1}	3000						
20	Downshift engine speed	min^{-1}	1000						
21									
22	Idle engine speed	min^{-1}	800						
23									
24									

Slika 13. Unos podataka o vozilu

Navedene željene promjene stupnjeva prijenosa su gornja i donja granica pri kojima bi program trebao simulirati promjene prijenosnih omjera kod ubrzavanja i usporavanja vozila.

Polja za unos svih traženih veličina označena su žutom bojom.

Drugi list nazvan je *Test procedure* na kojemu je potrebno unijeti traženu ispitnu proceduru koja se sastoji od vremena i tražene brzine vozila. Ostali listovi su listovi s tablicama potrošnje goriva ili emisijskih faktora pojedinih onečišćujućih tvari.

4.2. Korištene biblioteke programskih rutina

U programu su korištene četiri biblioteke, NumPy, Xlrd, XlsxXWriter i sys. Korištene su zbog svojih funkcija koje sam Python programski jezik ne sadrži u svojoj osnovnoj verziji.

NumPy

NumPy (Numerical Python) je biblioteka za obradu nizova (engl. *array*) opće namjene. Omogućuje brz rad sa višedimenzionalnim nizovima i alate i funkcije za operacije sa njima. Jedna je od temeljnih biblioteki za rad na znanstvenim područjima računanja s Pythonom. Sadrži različite značajke, uključujući:

- funkcionalne n-dimenzionalne nizove (matrice),
- sofisticirane funkcije za lakšu manipulaciju i računanje s nizovima (matricama),
- alate za integraciju C/C++ i Fortran kôdova,
- linearnu algebra, Fourierovu transformaciju i mogućnosti izbora slučajnih brojeva.

Uz znanstvenu uporabu, NumPy se može koristiti i kao učinkovit višedimenzionalni spremnik generičkih podataka. U NumPyu mogu se učitati različite vrste podataka zbog čega se jednostavno i brzo može integrirati sa širokim rasponom baza podataka.

Xlrd

Xlrd je biblioteka za učitavanje podataka i oblikovanje podataka iz Excel datoteka, bilo kojeg formata (.xls ili .xlsx).

XlsxWriter

XlsxWriter je biblioteka koja se može koristiti za pisanje teksta, brojeva i formula na više radnih listova u .xlsx datotekama. Podržava mnogo značajki kao što su:

- 100 % kompatibilnost s .xlsx datotekama,
- Formatiranje elemenata proračunske tablice,
- definiranje imena,
- izrada grafova,
- provjera podataka i padajućih popisa,
- rad s PNG/JPEG/BMP/WMF/EMF slikama,
- komentari u ćelijama,
- integracija s Pandas i NumPy bibliotekama,
- optimizacija memorije za pisanje velikih datoteka.

4.3. Kôd programa

Na početku svakog kôda programa potrebno je napisati, odnosno uključiti korištene biblioteke kako bi program mogao interpretirati njihove posebne funkcije i njihov zapis. U ovom programu korištene su već spomenute četiri biblioteke; NumPy za zapis i računske operacije s nizovima, Xlrd za učitavanje podataka iz ulazne datoteke u .xlsx formatu, XlsxWriter za zapis izračunatih rezultata u novu .xlsx datoteku i sys iz koje se poziva samo funkcija *exit*.

```
14
15  import numpy as np
16  import xlrd
17  import xlsxwriter
18  from sys import exit
19
20
```

Slika 14. Pozivanje korištenih biblioteka

Od 22. do 33. linije kôda, učitana je *input.xlsx* datoteka te njeni radni listovi pomoću njihovih imena. Ukoliko se ime ili datoteke ili nekog od listova promijeni, program neće moći učitati vrijednosti iz njih.

```
21
22  book_input = xlrd.open_workbook('input.xlsx')
23
24  sheet_vt = book_input.sheet_by_name('Test procedure')
25
26  sheet_vehicle_info = book_input.sheet_by_name('Vehicle info')
27
28  sheet_fuel_consumption = book_input.sheet_by_name('fuel_consumption')
29  sheet_ef_co2 = book_input.sheet_by_name('ef_CO2')
30  sheet_ef_co = book_input.sheet_by_name('ef_CO')
31  sheet_ef_hc = book_input.sheet_by_name('ef_HC')
32  sheet_ef_nox = book_input.sheet_by_name('ef_NOx')
33  sheet_ef_pm = book_input.sheet_by_name('ef_PM')
```

Slika 15. Učitavanje datoteke i radnih listova sa ulaznim podacima

4.3.1. Učitavanje podataka o vozilu

U linijama prikazanima na slici 16. definirane su globalne varijable kojima su pridodane vrijednosti iz radnog lista *Vehicle info*.

```
35
36  # loading vehicle data
37
38  # general data
39  vehicle_mass = sheet_vehicle_info.cell_value (4, 3)
40  vehicle_surface = sheet_vehicle_info.cell_value (5, 3)
41  air_resistance_coefficient = sheet_vehicle_info.cell_value (6, 3)
42  air_density = sheet_vehicle_info.cell_value (7, 3)
43  g = sheet_vehicle_info.cell_value (8, 3)
44  transmission_ef_coef = sheet_vehicle_info.cell_value (9, 3)
45  engine_volume = sheet_vehicle_info.cell_value (10, 3)
46  stroke = sheet_vehicle_info.cell_value (11, 3)
47  power_max = sheet_vehicle_info.cell_value (12, 3)
48
49  # tyre dimensions
50  tyre_HR_1 = sheet_vehicle_info.cell_value (14, 1) / 1000
51  tyre_HR_2 = sheet_vehicle_info.cell_value (14, 2) / 1000
52  tyre_R = sheet_vehicle_info.cell_value (14, 3) * 0.0254 / 2
53  tyre_HR = tyre_HR_1 * tyre_HR_2
54
55  tyre_r_dyn = 0.97 * (tyre_R + tyre_HR)
56
57
58  # changing gears data
59  engine_speed_upshift = sheet_vehicle_info.cell_value (18, 2)
60  engine_speed_downshift = sheet_vehicle_info.cell_value (19, 2)
61  engine_idle_speed = sheet_vehicle_info.cell_value (21, 2)
62
```

Slika 16. Učitavanje ulaznih podataka kao varijable

U linijama 50-53 učitane su vrijednosti veličine pneumatika, te preračunate u metre. U liniji 55 određen je dinamički radijus kotača. Dinamički radijus je udaljenost osovine kotača od tla kada je kotač opterećen težinom vozila pri određenoj brzini kretanja. Može se izračunati formulom:

$$r_{\mathrm{d}} = f(v) = \frac{v}{2 * \pi * n_{\mathrm{kot}}} * 60 \qquad (3)$$

gdje je:

v – brzina, km/h,

n_{kot} – brzina vrtnje kotača, okr/min.

Može se izračunati i formulom za konstantnu vrijednost:

$$r_{\mathrm{d}} = 0{,}97 * r \qquad (4)$$

gdje je:

r- radijus kotača.

U tom slučaju stvara se maksimalna razlika od 1 % pri najvećim brzinama kretanja. Zbog relativno male pogreške, program računa dinamički radijus kao konstantnu vrijednost prema formuli (4).

4.3.2. Učitavanje podataka o mjenjačkoj kutiji

```
65  # axle reduction
66  i_axle = sheet_vehicle_info.cell_value (4, 7)
67
68
69  # loading gearbox ratios
70
71  gear_ratio = np.array (sheet_vehicle_info.cell_value (6, 7))
72  n=7
73  m=0
74  while m == 0:
75      x = sheet_vehicle_info.cell_value (n, 7)
76      if x !="":
77          gear_ratio = np.append(gear_ratio, x)
78          n += 1
79      else:
80          m=1
81
```

Slika 17. Učitavanje podataka o prijenosnim omjerima mjenjača

U linijama 66-80 učitavaju se prijenosni omjeri osovinske redukcije i nepoznatog broja brzina u mjenjačkoj kutiji vozila prema slici 17. U liniji 71, napravljena je varijabla `gear_ratio` u obliku niza od jednog člana. U linijama 72 i 73 napravljene su varijable `n` i `m`. Varijabla `n` služi za davanje informacije *while* petlji koji red bi se pri svakoj iteraciji trebao učitat. Varijabla `m` služi samo kao varijabla za prekid iteriranja, odnosno izlazak iz petlje. *While* petlja je petlja koja prima jedan uvjet i ponavlja sve naredbe pod sobom dok je taj uvjet ispunjen. U ovom slučaju, *while* petlja će se ponavljati dokle god je prethodno zadana varijabla `m` jednaka 0. U liniji 76 definirana je lokalna varijabla `x`. Lokalne varijable su varijable koje postoje samo na mjestu gdje su definirane, u ovom slučaju u *while* petlji. Ona ima iznos učitane ćelije sa lista *Vehicle info* sa koordinatom (`n`,7). Nakon toga postavljeno je *if* grananje koje će za svaku lokalnu varijablu `x`, različitu od praznog učitanog polja, nizu `gear_ratio` dodati na kraj vrijednost `x`, te potom povećati varijablu `n` za 1. Ukoliko je lokalna varijabla `x` jednaka učitanom praznom polju, vrijednost varijable `m` se mijenja u 1, te će se kod sljedeće iteracije *while* petlja prekinuti. Time se zapisuje niz `gear_ratio` sa brojem članova jednakim brojem prijenosnih omjera u vozilu.

4.3.3. Ispitna procedura

Podaci potrebni za ispitnu proceduru su vrijeme i brzina kojom se vozilo kreće u tom trenutku. Oni se unose u radni list pod imenom *Test procedure* u dva stupca. Prvi podatak bi uvijek trebao biti 0, odnosno ispitna procedura počinje iz stanja mirovanja vozila. Podatke nije potrebno unositi za svaku sekundu već samo ciljne brzine u određenom vremenu, npr. ukoliko vozilo ubrzava od 20. sekunde do 40. s brzine 40 km/h na brzinu 90 km/h, dovoljno je unijeti podatke za ta dva stanja (20 s, 40 km/h i 40 s, 90 km/h) ili ukoliko je brzina konstantna u nekom periodu, npr. ukoliko se od 20. do 40. sekunde vozi istom brzinom (20 s, 40 km/h i 40 s, 40 km/h).

	A	B
1	Vrijeme	Brzina
2	0	0
3	8	40
4	10	40
5	15	60
6	20	60
7	25	80
8	30	90
9		
10		
11		

Slika 18. Primjer ispitne procedure

Za učitavanje podataka, prvo se definiraju dva niza kao varijable nazvane `time` i `velocity`. Nizovi su definirani prema veličini od jednog retka s brojem članova jednakim broju unesenih vremenskih koraka. U početku su sve vrijednosti u nizovima postavljene na vrijednost 0.

```
84   # loading test procedure (time and velocity)
85
86   time = np.zeros(shape=(sheet_vt.nrows-1), dtype=int)
87   velocity = np.zeros(shape=(sheet_vt.nrows-1), dtype=float)
88
89   for i in range(sheet_vt.nrows-1):
90       time [i]=sheet_vt.cell_value (i+1,0)
91       velocity [i]=sheet_vt.cell_value (i+1,1)
92
93
94   for i in range (1, time.size):
95       if time[i-1] > time[i]:
96           workbook_results = xlsxwriter.Workbook ("results.xlsx")
97           sheet_warning = workbook_results.add_worksheet("warning")
98           cell_format_red = workbook_results.add_format({'bold': True, 'font_color': 'red'})
99           sheet_warning.write(1,1, "Warning: time in test procedure is not valid. Please check.", cell_format_red)
100          workbook_results.close()
101          exit()
102
```

Slika 19. Učitavanje ispitne procedure i njena provjera

U liniji 89 definirana je *for* petlja. *For* petlja u ovom slučaju prima dvije vrijednosti prema kojima ponavlja sve zadane naredbe. Prva vrijednost je iterator, u ovom slučaju varijabla `i`. Druga vrijednost označava broj ponavljanja *for* petlje. Iterator `i` poprima vrijednosti od 0 do vrijednosti varijable `time.size` koja je jednaka članova u nizu time. U linijama 90 i 91 nizovi `time` i `velocity` na i-tom mjestu poprimaju učitane vrijednosti iz radnog lista *Test procedure*.

Sljedeći dio kôda provjerava je li niz time pravilno unesen, tj. je li svaka naredna vrijednost vremena veća od prethodne. Provjerava se *for* petljom koja iterira od vrijednosti 1 do ukupnog broja članova niza `time`. U liniji 95 postavljen je *if* uvjet koji provjerava te vrijednosti, te ukoliko je niz krivo unesen, stvara datoteku result.xlsx i radni list te ispisuje upozorenje i izlazi iz programa uz pomoć funkcije *exit*.

```
106  # adding time and velocity values for 1 Hz intervals
107
108  for i in range (time[-1]):
109    if i !=time [i] and i!=0:
110        time = np.insert(time, i, i)
111
112        v_interpolation = velocity[i-1]+((time[i]-time[i-1])*(velocity[i]-velocity[i-1])/(time[i+1]-time[i-1]))
113        velocity = np.insert(velocity, i, v_interpolation)
114
115  velocity = velocity / 3.6
116
```

Slika 20. Nadopunjavanje ispitne procedure

Program zatim zapisuje nizove vremena i brzine u intervalima od jedne sekunde. Za to je napisana *for* petlja koja iterira od vrijednosti 0 do iznosa zadnje vrijednosti u nizu `time`. Zadnja vrijednost se definira kao vrijednost iz niza na poziciji -1, odnosno prva vrijednost od kraja. U tijelu *for* petlje je *if* uvjet koji za svaku vrijednost iteratora `i` različitog od i-te vrijednosti iz niza `time` osim za vrijeme 0 ubacuje u niz nove vrijednosti. To znači da koliko nije unesen podatak za svaku sekundu, program će ga sam dodati. U liniji 110, u niz time se ubacuje vrijednost vremena `i`, a u liniji 112 se prema formuli za linearnu interpolaciju:

$$y = y_0 + (x - x_0) * \frac{y_1 - y_0}{x_1 - x_0} \qquad (5)$$

izračunava vrijednost brzine za to vrijeme, te se u sljedećoj liniji zapisuje u niz sa vrijednostima brzina. Nakon izlaza iz petlje cijeli niz `velocity` se preračunava iz km/h u m/s.

4.3.4. Ubrzanje

```
121  #   acceleration calculation
122
123  acceleration = np.zeros(shape=(time.size), dtype=float)
124
125  for i in range (1, time.size):
126      acc = (velocity [i] - velocity [i-1])
127      acceleration [i]= acc
128
```

Slika 21. Izračun ubrzanja

Ubrzanje se računa prema formuli:

$$a = \frac{v_2 - v_1}{\Delta t} \tag{6}$$

gdje je:

v_1 – brzina u prethodnom vremenskom koraku, m/s,

v_2 – brzina u trenutnom vremenskom koraku, m/s,

Δt – vremenska razlika između dva vremenska koraka, s.

Zbog toga što su nizovi vremena i brzine zapisani u intervalima od jedne sekunde, vrijednost akceleracije se može izračunati samo razlikom brzina.

Za zapis u programu, akceleracija je definirana kao niz s imenom `acceleration` veličine kao i niz `time`, sa svim vrijednostima inicijalno postavljenim na 0. U *for* petlji se izračunava vrijednost akceleracije za svaku sekundu, te se zapisuje na svoje mjesto.

4.3.5. Određivanje brzine vrtnje motora

Brzina vrtnje motora pri kretanju vozila računa se prema formuli:

$$n_{\mathrm{mot}} = \frac{60 * v * i_{\mathrm{m},i} * i_{\mathrm{OR}}}{2 * \pi * r_{\mathrm{d}}} \qquad (7)$$

gdje je:

$i_{\mathrm{m},i}$ – prijenosni omjer i-tog stupnja prijenosa, -,

i_{OR} – osovinska redukcija, -.

Zbog mogućeg unosa željene brzine vrtnje motora za promjenu stupnja prijenosa u mjenjaču, napisan je kôd prikazan na slici 22. Taj dio programa računa i zapisuje u dva niza brzinu vrtnje motora i prijenosni omjer u mjenjaču za svaki vremenski korak uz pomoć *for* petlje.

```
132  # engine speed calculation
133
134  engine_speed = np.zeros(shape=(time.size), dtype=float)
135  gear = np.zeros(shape=(time.size), dtype=int)
136
137
138  j=0
139
140  for i in range (time.size):
141      eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)
142
143      if velocity [i] == 0:
144          eng_spd = engine_idle_speed
145          engine_speed [i] = eng_spd
146      else:
147          if eng_spd > engine_speed_upshift and acceleration[i]>0:
148              if j == gear_ratio.size-1:
149                  engine_speed [i] = eng_spd
150              else:
151                  j +=1
152                  eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)
153                  engine_speed [i] = eng_spd
154          elif eng_spd < engine_speed_downshift and acceleration[i]<0:
155              if j == 0:
156                  eng_spd = engine_speed_downshift
157                  engine_speed [i] = eng_spd
158              else:
159                  j -= 1
160                  eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)
161                  engine_speed [i] = eng_spd
162          elif eng_spd<engine_idle_speed:
163              engine_speed [i]=engine_speed_downshift
164          else:
165              engine_speed [i] = eng_spd
166
167      gear [i] = j
168
```

Slika 22. Izračun brzine vrtnje motora za svaki vremenski korak

Unutar *for* petlje, prvo se izračunava potencijalna brzina vrtnje motora prema formuli (7), te zapisuje kao lokalna varijabla `eng_spd`. Nakon toga se provjerava stoji li vozilo na mjestu. Ukoliko vozilo stoji, varijabla `eng_spd` se mijenja u brzinu vrtnje praznog hoda motora te se kao takva zapisuje u niz `engine_speed` za taj vremenski korak. Ukoliko vozilo ne stoji na

mjestu, tj. brzina mu je različita od 0, postoje tri moguća slučaja. Oni su napisani kao *if* grananje u kojemu se ispituje je li brzina vrtnje veća, manja ili unutar postavljenih granica za željenu promjenu stupnja prijenosa u mjenjaču. Ukoliko je veća, program provjerava može li prebaciti u višu brzinu mjenjača, tj. nalazi li se u najvišem stupnju prijenosa. Ukoliko se nalazi u najvišem stupnju, brzina vrtnje ostaje ista, a ako može, brzina vrtnje se ponovno računa s drugim prijenosnim omjerom mjenjača. Isti slučaj se ispituje ukoliko je brzina vrtnje manja od tražene granice. Treći slučaj je da je brzina vrtnje unutar granica te se kao takva zapisuje u niz.

4.3.6. Sile otpora u vožnji

Kretanju vozila suprotstavljaju se određeni otpori koje pogon vozila mora savladati. Istovremeno, performanse vozila ograničene su performansama motora. Pogonske sile moraju biti jednake ili veće svim otporima vožnje koji djeluju na vozilo. Na performanse ne utječe samo najveći moment ili najveća snaga motora već i ponašanje u uvjetima djelomičnog opterećenja pri različitim režimima rada.

U ovom programu u obzir su uzete sile otpora kotrljanja, otpora zraka i otpora ubrzanja. Otpor nagiba ceste nije uzet u obzir zbog različitih i promjenjivih pravca kretanja. Bez dodatnog podatka o nagibu prometnice za svaku njegovu promjenu u vremenskoj domeni, ne bi bilo moguće odrediti smislen rezultat.

4.3.6.1. Izračun otpora kotrljanja

Otpor kotrljanja je posljedica gubitaka energije koji nastaju uslijed deformiranja gume kotača prilikom kotrljanja po podlozi. Na iznos otpora kotrljanja utječu:

- karakteristike gume,
- masa vozila,
- stanje površine kolnika,
- brzina vožnje,
- geometrija ovjesa,
- stanje kočnica, trenje u ležajevima.

Otpor kotrljanja računa se prema formuli:

$$F_k = f_k * m_v * g * \cos\ \alpha \qquad (8)$$

gdje su:

m_v – ukupna masa vozila, kg,

$\alpha = 0°$, kut nagiba ceste.

Faktor otpora kotrljanja f_k računa se prema izrazu:

$$f_k = f_{k,1} + f_{k,2} * \left(\frac{v}{100}\right) + f_{k,3} * \left(\frac{v}{100}\right)^4 \qquad (9)$$

Koeficijenti $f_{k,i}$ su konstantnog iznosa, $f_{k,1}$ = 0,0090; $f_{k,2}$ = 0,0020; $f_{k,3}$ = 0,0003; te kao takvi vrijede za radijalne gume pri brzinama do 210 km/h.

```
173  # rolling resistance calculation
174
175  force_rolling = np.zeros(shape=(time.size), dtype=float)
176  rolling_resistance_coefficient = np.zeros(shape=(time.size), dtype=float)
177
178  for i in range (1, time.size):
179      v = velocity [i] * 3.6
180      rolling_resistance_coefficient [i] = 0.0090 + 0.0020*(v/100) + 0.0003*((v/100)**4)
181
182      if velocity[i]==0:
183          force_rolling [i] = 0
184      else:
185          force_rolling [i] = vehicle_mass * rolling_resistance_coefficient [i] * g
186
```

Slika 23. Izračun otpora kotrljanja

U programu se otpor kotrljanja zapisuje u zaseban niz, kao i faktor otpora kotrljanja. Računaju se prema formulama (8) i (9) te zapisuju za svaki vremenski korak.

4.3.6.2. Izračun otpora zraka

Otpor zraka posljedica je gubitaka energije koji nastaju uslijed gibanja vozila kroz zrak. Prilikom vožnje, vozilo pomiče zrak oko sebe što uzrokuje vrtloženje i stvara razliku tlakova ispred i iza vozila. Dio zraka klizi po površini vozila i dio prolazi kroz vozilo radi hlađenja motora, kočnica i dovođenja zraka u motor. Svi ti zbrojeni utjecaji stvaraju otpor zraka. Stoga se može zaključiti da na iznos otpora zraka utječu:

- oblik vozila,
- brzina vožnje.

Otpor zraka računa se prema formuli:

$$F_z = \rho_z * \frac{(v + v_0)^2}{2} * c_w * A \qquad (10)$$

gdje su:

ρ_z – gustoća zraka, kg/m^3,

$v_0 = 0$, brzina protuvjetra, m/s,

c_w – faktor otpora zraka, -,

A – čeona površina vozila, m^2.

```
187  # air resistance calculation
188
189  force_airdrag = np.zeros(shape=(time.size), dtype=float)
190
191  for i in range (time.size):
192      force_airdrag [i] = 0.5 * (velocity[i])**2 * vehicle_surface * air_resistance_coefficient * air_density
193
```

Slika 24. Izračun otpora zraka

U programu iznos otpora zraka se zapisuje za svaki vremenski korak prema formuli (10) kao varijabla `force_airdrag`.

4.3.6.3. Izračun otpora ubrzanja

Otpor ubrzanja nastaje u nestacionarnim uvjetima rada kao posljedica inercijskih sila kod ubrzavanja. Prilikom ubrzavanja pogon vozila mora translatorno ubrzati masu vozila i rotacijski sve rotirajuće pogonske dijelove. Računa se prema formuli:

$$F_{\mathrm{a}} = \left[m_{\mathrm{v}} + \frac{1}{r_{\mathrm{d}}^2} * \left(J_{\mathrm{k}} + J_{\mathrm{m}} * i_{\mathrm{uk}}^2 \right) \right] * a \qquad (11)$$

gdje su:

J_{k} - moment inercije kotača, kgm^2,

J_{m} – moment inercije motora i njemu pridruženih dijelova, kgm^2,

i_{uk} – ukupni prijenosni omjer, -,

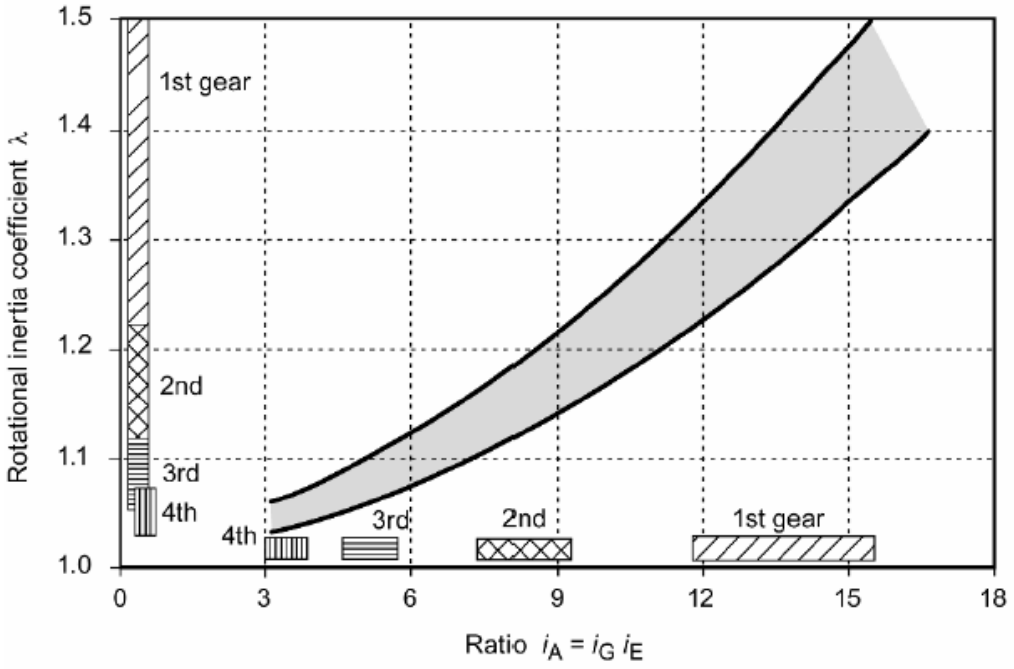
a – ubrzanje vozila, m/s^2.

Također, otpor ubrzanja može se izračunati i prema pojednostavljenoj formuli:

$$F_{\mathrm{a}} = m_{\mathrm{v}} * k_{\mathrm{m}} * a \qquad (12)$$

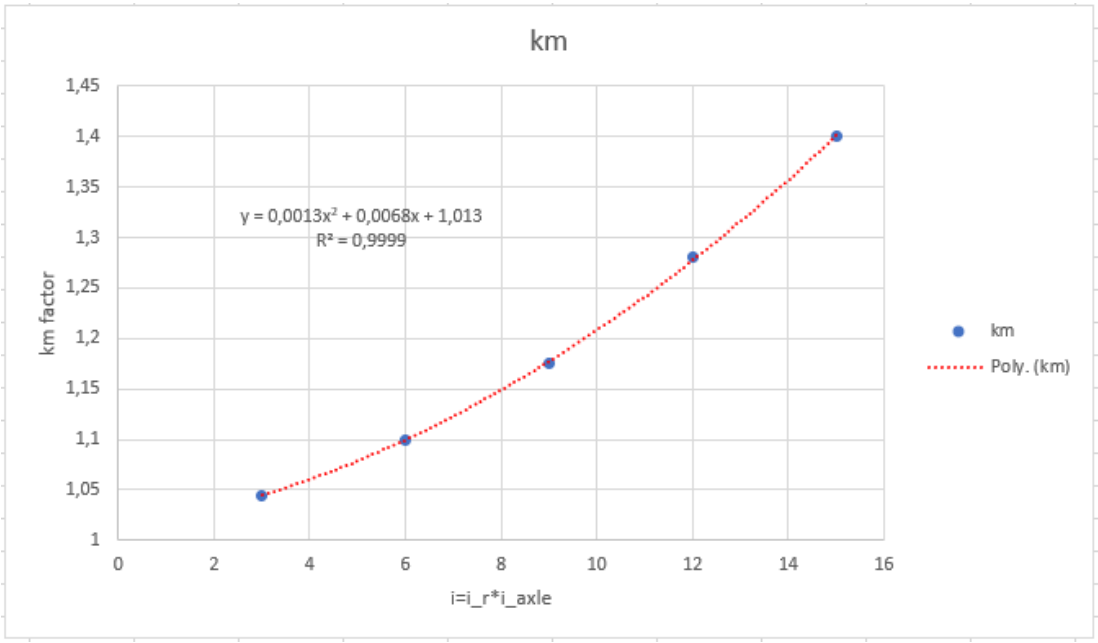
gdje je k_{m} faktor rotirajućih masa., -.

Zbog toga što su podaci o momentima inercije kotača (J_{k}) i inercije motora i ostalih pogonskih dijelova (J_{m}) teško dostupni, program računa otpor ubrzanja prema pojednostavljenoj formuli. Faktor rotirajućih masa k_{m} može se očitati iz dijagrama sa slike 25., te prema [24] ovisi o kvadratu ukupnog prijenosnog omjera i_{uk}.



Slika 25. Dijagram faktora rotirajućih masa k_m *[24]*

Radi automatizacije postupka, iz grafa je očitano nekoliko točaka te je u programu Excel, uz pomoć *trendline* funkcije, izračunata kvadratna formula za faktor rotirajućih masa. Očitane su vrijednosti unutar granica maksimalnog i minimalnog koeficijenta kod kojih postoji maksimalna razlika od ±5 % od granične (1,475) i očitane (1,4) vrijednosti, odnosno očitane i minimalne (1,33).



Slika 26. Jednadžba faktora rotirajućih masa k_m

```
195  # acceleration resistance calculation
196
197  force_acc = np.zeros(shape=(time.size), dtype=float)
198  km_factor = np.zeros(shape=(gear_ratio.size), dtype=float)
199
200  for i in range (km_factor.size):
201      red = i_axle * gear_ratio [i]
202
203      km_factor [i] = 0.0013*red**2 + 0.0068*red +1.013
204
205  for i in range (time.size):
206      if velocity [i]==0:
207          force_acc[i]=0
208      else:
209          force_acc [i] = vehicle_mass * acceleration [i] * km_factor [gear[i]]
210
```

Slika 27. Izračun faktora k_m za svaki stupanj prijenosa i otpora ubrzanja

U programu se prvo deklariraju nizovi za iznose sila otpora ubrzanja i faktora rotirajućih masa. Prvo se u posebnoj for petlji računa faktor k_m za svaki prijenosni omjer u mjenjaču prema formuli sa slike 26. Nakon toga se prema formuli (12) računa iznos otpora zraka za svaki vremenski korak ispitne procedure osim u slučaju da vozilo stoji na mjestu.

4.3.6.4. Ukupan iznos otpora u vožnji

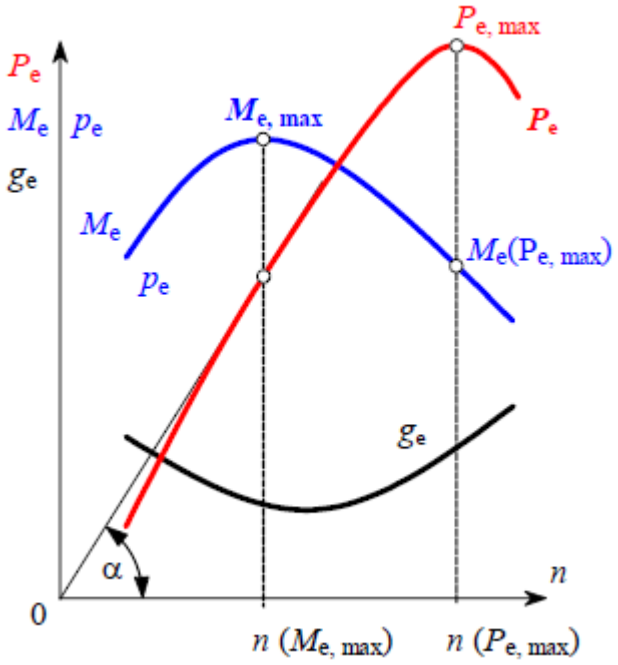
```
212  # forces sum
213
214  force_all = force_airdrag + force_rolling + force_acc
215
```

Slika 28. Izračun ukupnog otpora

Prethodno izračunata tri otpora (otpor kotrljanja, otpor zraka i otpor ubrzanja) jednostavno se zbrajaju kao cijeli nizovi. Biblioteka NumPy svaki niz sa jednakom veličinom, odnosno brojem članova, zbraja prema poziciji svakog člana, tj. i-ti član niza će se zbrojiti sa i-tim članom drugog niza itd.

4.3.7. Značajke rada motora

Performanse motora ograničene su maksimalnim opterećenjem koje je prikazano kao krivulja punog opterećenja. Motor daje potrebnu snagu kako bi vozilo moglo savladati sva opterećenja. Ti otpori ne moraju biti jednaki punom opterećenju motora, već se mogu nalazi na bilo kojoj točki u dijagramu ispod nje. Osim pozitivnih vrijednosti snage i momenta, one mogu biti i negativne u slučaju kočenja motorom. Radne točke motora određene su brzinom vrtnje i snagom, odnosno momentom motora.



Slika 29. Prikaz značajki motora pod punim opterećenjem [23]

4.3.7.1. Potrebna snaga motora

Snaga koju motor treba kako bi vozilo savladalo sve otpore vožnje računa se prema formuli:

$$P = \frac{F * v}{\eta_m} \qquad (13)$$

gdje su:

F – ukupan otpor vozila, N,

v – brzina vozila, m/s,

η_m – mehanički stupanj djelovanja prijenosa snage, -.

```
219  # required power calculation
220
221  power = np.zeros(shape=(time.size), dtype=float)
222
223  limit_power = False
224
225  for i in range (time.size):
226      power [i] = force_all [i] * velocity [i] / (1000 * transmission_ef_coef)
227
228      if power [i] > power_max:
229          limit_power = True
230
```

Slika 30. Izračun potrebne snage

Program računa snagu motora prema formuli (13). Uz računanje potrebne snage za svaki korak ispitne procedure postavljena je i provjera u 228. i 229. liniji kôda. Provjera je u obliku *if* uvjeta koji provjerava za svaki korak je li izračunata potrebna snaga veća od unesene maksimalne snage motora. Provjera ne prekida program već samo daje informaciju o prekoračenju maksimalne snage.

4.3.7.2. Potreban moment

Moment koji motor proizvodi povezan je sa snagom motora prema izrazu:

$$M = \frac{2 * \pi * n_{\mathrm{m}}}{P} \tag{14}$$

U programu se koristi formula s ovisnosti momenta o opterećenjima:

$$M = \frac{F * r_{\mathrm{d}}}{i_{\mathrm{m},i} * i_{\mathrm{OR}} * \eta_{\mathrm{m}}} \tag{15}$$

```
243  # moment calculation
244
245  moment = np.zeros(shape=(time.size), dtype=float)
246
247
248  for i in range (time.size):
249      moment [i] = (force_all [i] * tyre_r_dyn) / (gear_ratio[gear[i]] * i_axle * transmission_ef_coef)
250
```

Slika 31. Izračun momenta

4.3.7.3. Srednji efektivni tlak

Srednji efektivni tlak računa se prema izrazu:

$$p_e = \frac{P}{V_m * \frac{2 * n}{T}} \tag{16}$$

gdje su:

V_m – radni volumen motora,

T – broj taktova motora.

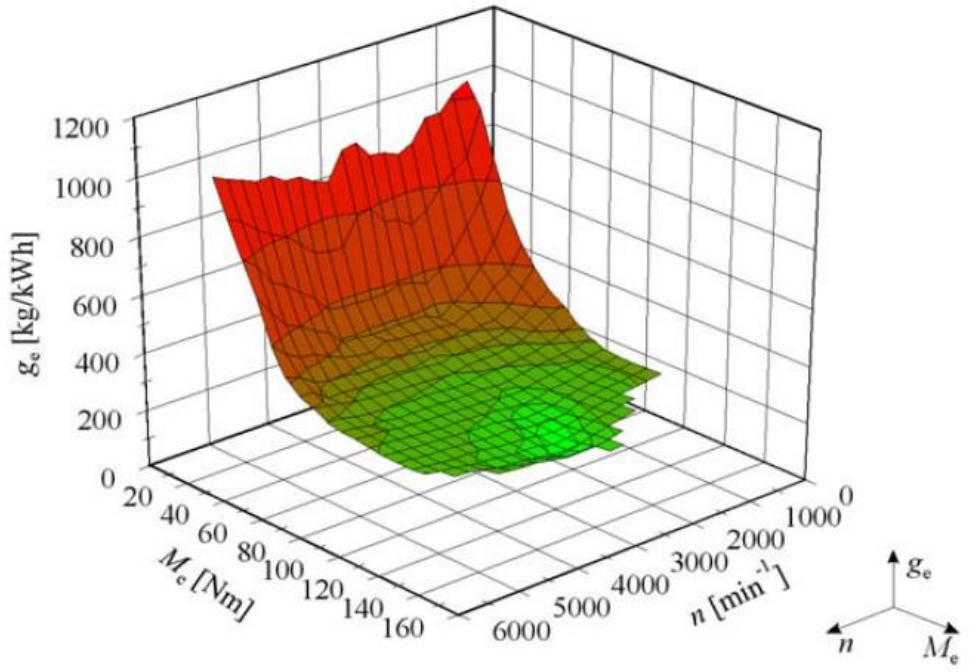
```
233  # mean effective pressure calculation
234
235  pressure_ef = np.zeros(shape=(time.size), dtype=float)
236
237  for i in range (time.size):
238      pressure_ef [i] = power [i] / (engine_volume * (engine_speed [i] / (30 * stroke)) * 100)
239
```

Slika 32. Izračun srednjeg efektivnog tlaka

U programu izraz je modificiran kako bi se iz postojećih podataka izračunao rezultat u barima.

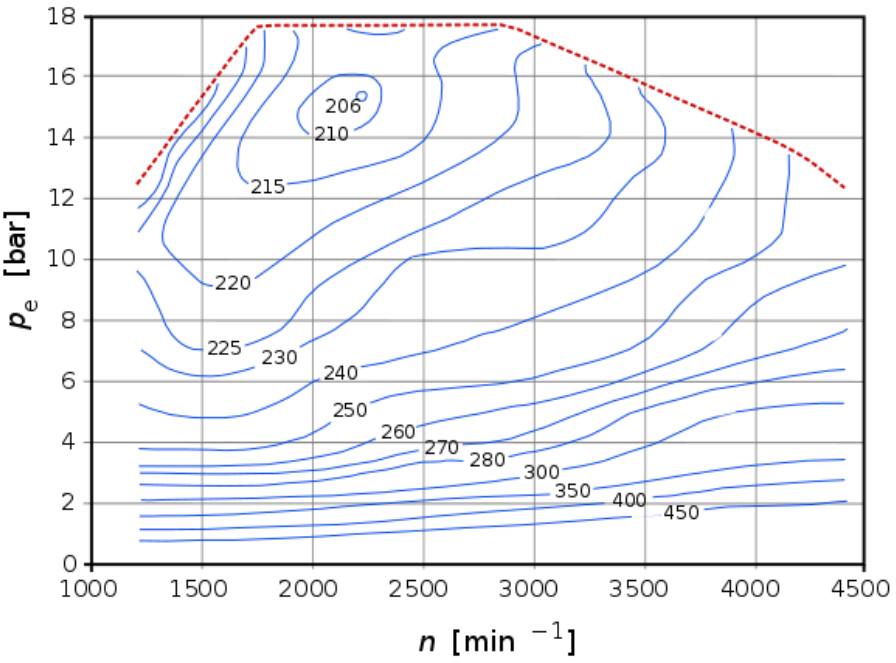
4.3.8. Učitavanje tablica specifične potrošnje i emisijskih faktora

Specifična efektivna potrošnja goriva kod motora ovisi o brzini vrtnje motora i o trenutnom opterećenju motora. Zbog takve ovisnosti, najlakše ju je prikazati kao površinu u tri dimenzije. Primjer takve površine dan je na slici 33.



Slika 33. 3D prikaz dijagrama specifične potrošnje goriva [23]

U praksi ovakav dijagram je teško koristiti te se češće koristi kao topografski dijagram u dvije dimenzije. Određene vrijednosti za specifičnu efektivnu potrošnju prikazane su kao krivulje konstantnih vrijednosti u ovisnosti o brzini vrtnje motora i momenta ili srednjeg efektivnog tlaka. Te vrijednosti ograničene su krivuljom najvećeg srednjeg efektivnog tlaka ili momenta. Primjer takvog dijagrama prikazan je na slici 34.



Slika 34. Topografski prikaz specifične potrošnje goriva [23]

Iz takvog dijagrama moguće je očitati vrijednosti za točke ovisne o tlaku i broju vrtnje, odnosno moguće je iz dijagrama napraviti tablicu. Taj princip primijenjen je u ovom programu za računanje potrošnje goriva i emisija onečišćujućih tvari.

U input datoteci napravljeni su radni listovi namijenjeni za tablice za specifičnu potrošnju i emisijske faktore.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	table fuel consumption												
2	engine speed →	0	0	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
3	pressure↓	0	11	12	13	14	15	16	17	18	19	20	21
4		2	21	22	23	24	25	26	27	28	29	30	31
5		4	31	32	33	34	35	36	37	38	39	40	41
6		6	41	42	43	44	45	46	47	48	49	50	51
7		8	51	52	53	54	55	56	57	58	59	60	61
8		10	61	62	63	64	65	66	67	68	69	70	71
9		12	71	72	73	74	75	76	77	78	79	80	81
10		14	81	82	83	84	85	86	87	88	89	90	91
11		16	91	92	93	94	95	96	97	98	99	100	101
12		18	101	102	103	104	105	106	107	108	109	110	111
13		20	111	112	113	114	115	116	117	118	119	120	121
14		40	120	120	120	120	120	120	120	120	120	120	120
15		60	150	150	150	150	150	150	150	150	150	150	150
16													

Slika 35. Primjer tablice specifične potrošnje goriva

Program te tablice učitava kao matrice napravljene od proizvoljno zapisanog broja redova i stupaca. Program učitava cijelu tablicu neovisno o njenoj veličini. Valja napomenuti, što je tablica veća, tj. očitani broj radnih točaka je veći, rezultati će biti precizniji.

```
255  # function for loading table of emission factors
256
257  def table_loading (table_name, sheet_name):
258      for c in range(sheet_name.ncols-1):
259          for r in range(sheet_name.nrows-1):
260              table_name [r][c] = sheet_name.cell_value(r+1, c+1)
261
262
263  table_fuel_consumption = np.zeros(shape=(sheet_fuel_consumption.nrows-1, sheet_fuel_consumption.ncols-1))
264
265  table_loading (table_fuel_consumption, sheet_fuel_consumption)
266
```

Slika 36. Kôd za učitavanje tablice specifične potrošnje goriva ili emisijskih faktora

Za učitavanje tablice definirana je funkcija s imenom `table_loading`. Funkcija mora primiti ime tablice i ime radnog lista kao ulazne podatke kako bi mogla pravilno očitati i zapisati vrijednosti. Napravljena je od dvije *for* petlje koje se iteriraju po broju redova odnosno stupaca od kojih je tablica napravljena, te tako za svaku koordinatu zapisuju vrijednost. Tom metodom prvi stupac u matrici sadrži vrijednosti specifičnog efektivnog tlaka, a prvi redak vrijednosti brzine vrtnje motora. Koordinata (0,0) ima vrijednost 0 zbog toga što program ne može učitati praznu vrijednost. Na slici 36. prikazan je dio kôda za definiranje matrice za pohranu podataka iz tablice za specifičnu potrošnju i pozivanje funkcije koja prepisuje njene vrijednosti iz ulaznih podataka u matricu u programu.

4.3.9. Izračun potrošnje goriva i stvorenih emisija

Algoritam za računanje potrošnje goriva i stvorenih emisija napravljen je pomoću vlastite funkcije `emission_calculation`. Pomoću već izračunatih vrijednosti srednjeg efektivnog tlaka i brzine vrtnje motora mogu se interpolacijom odrediti vrijednosti emisijskih faktora iz učitanih tablica. Funkcija kao ulazne podatke mora primit ime tablice i radnog lista na kojem je tablica učitana, te dva niza za zapis rezultata o trenutnoj i ukupnoj emisiji.

```
297  # calculation of emission or fuel consumption in 1 Hz intervals
298
299
300  def emission_calculation (ef_table, sheet_name, emission_result, emission_result_cumulative):
301
302      for i in range (1, time.size):
303          a= pressure_ef [i]
304          b= engine_speed [i]
305
306          i1=0
307          i2=0
308
309          # finding values of engine speed for interpolation
310
311          for ia in range (1, sheet_name.ncols-2):
312              x1=ef_table[0][ia]
313              x2=ef_table[0][ia+1]
314              if b >= x1 and b < x2:
315                  i1=ia
316
317              elif b >= ef_table[0][sheet_name.ncols-2]:
318                  i1=-1
319
320          # finding values of mean effective pressure for interpolation
321
322          for ib in range (1, sheet_name.nrows-2):
323              x1=ef_table[ib][0]
324              x2=ef_table[ib+1][0]
325              if a >= x1 and a < x2:
326                  i2=ib
327
328              elif a < 0:
329                  i2=0
330
331              elif a >= ef_table[sheet_name.nrows-2][0]:
332                  i2=-1
333
```

Slika 37. Kôd funkcije za izračun potrošnje goriva i proizvedenih emisija (1/2)

Algoritam se *for* petljom u 302. liniji ponavlja za svaki vremenski interval. U lokalne varijable `a` i `b` se zapisuju podaci o tlaku i brzini vrtnje, a varijable `i1` i `i2` se postavljaju na 0. One služe kako bi se pronašla koordinata traženog emisijskog faktora. Na primjer, u linijama 311-318 se pomoću *for* petlje provjerava nalazi li se trenutna brzina vrtnje između dviju susjednih vrijednosti iz tablice. Kada se pronađu te dvije susjedne vrijednosti, lokalna varijabla `i1` poprima vrijednost koordinate niže vrijednosti brzine vrtnje motora. Ukoliko je tražena brzina

veća od svih očitanih brzina, tj. nalazi se van područja tablice, lokalna varijabla `i1` poprima vrijednost -1. Vrijednost -1 zamišljena je kao kontrolna vrijednost.
U linijama 322-332 traži se na identičan način koordinata srednjeg efektivnog tlaka. Za razliku od brzine vrtnje motora, srednji efektivni tlak može biti i negativan u slučaju kočenja motorom. U tom slučaju, vrijednost `i2` poprima vrijednost 0. Kako pri kočenju motorom, moderni motori prekidaju dovod goriva u cilindar, rad se troši na kompresiju zraka te se ne stvaraju nikakve emisije štetnih tvari.

```
334     if i1==-1 or i2==-1:
335         emission_result [i] = -1
336 
337     elif i1==0 or i2==0:
338         emission_result [i] = 0
339 
340     else:
341         # 1st interpolation, same engine speed
342         ya1 = ef_table[i2][i1]
343         yb1 = ef_table[i2][i1+1]
344 
345         xn1 = ef_table[0][i1]
346         xn2 = ef_table[0][i1+1]
347 
348         #2nd interpolation, same engine speed
349         ya2 = ef_table[i2+1][i1]
350         yb2 = ef_table[i2+1][i1+1]
351 
352         # interpolation for required value
353         xp1 = ef_table[i2][0]
354         xp2 = ef_table[i2+1][0]
355 
356         if ya1 > 0 and yb1 > 0 and ya2 > 0 and yb2 > 0:
357 
358             y1 = ya1+(b-xn1)*(yb1-ya1)/(xn2-xn1)
359 
360             y2 = ya2+(b-xn1)*(yb2-ya2)/(xn2-xn1)
361 
362             y3 = y1+((a-xp1)*(y2-y1)/(xp2-xp1))
363 
364 
365             emission_result [i] = y3 * power [i] / 3600
366 
367         else:
368 
369             emission_result [i] = -1
370 
371     emission_result_cumulative [i] = emission_result_cumulative [i-1] + emission_result [i]
372 
```

Slika 38. Kôd funkcije za izračun potrošnje goriva i proizvedenih emisija (2/2)

Drugi dio funkcije `emission_calculation` računa uz pomoć linearnih interpolacija vrijednost emisijskog faktora za traženi tlak i brzinu vrtnje. Taj algoritam se sastoji od četiri dijela. Prva tri dijela su provjere lokalnih varijabli `i1` i `i2`, tj. koordinata prvih nižih vrijednosti tlaka i brzine vrtnje. Ukoliko bilo koja od njih iznosi -1, odnosno tražena točka se ne može očitati iz tablice, program to zapisuje kao negativan rezultat. Takav rezultat znači da ispitna

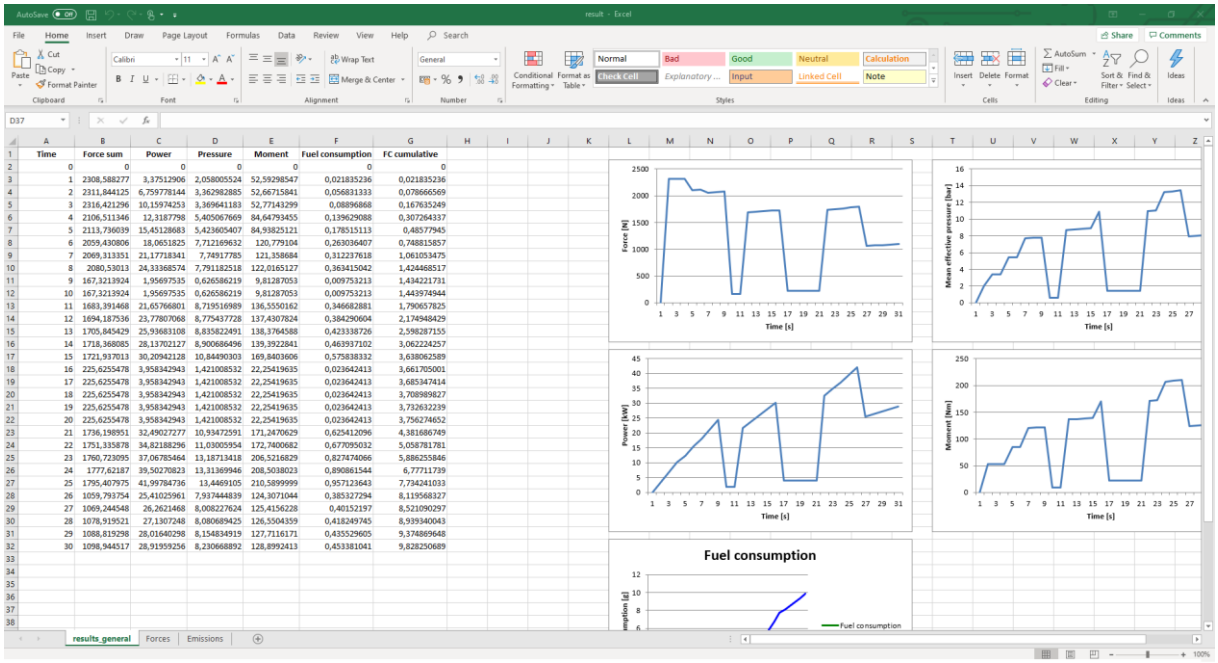
procedura zahtjeva veće performanse vozila. Ukoliko su `i1` ili `i2` jednake 0, odnosno vozilo koči motorom, stvorene emisije su 0. Ukoliko su tražene vrijednosti unutar tablice i lokalne varijable `i1` i `i2` nisu -1 ili 0, traženu vrijednost je potrebno interpolirati.

Za interpolaciju, prvo se provjerava jesu li sve susjedne vrijednosti različite od 0, tj. nalazi li se radna točka ispod krivulje maksimalnog opterećenja motora. Za slučaj da je radna točka iznad krivulje maksimalnog opterećenja motora, program zapisuje rezultat -1 kao kontrolnu vrijednost. U slučaju da se radna točka nalazi ispod krivulje maksimalnog opterećenja, provode se tri interpolacije. Prve dvije interpoliraju emisijski faktor za prvu manju i prvu veću vrijednost brzine vrtnje motora iz tablice, a treća interpolira ta dva rezultata za traženu vrijednost brzine vrtnje. Izračunati konačni rezultat emisijskog faktora se množi sa trenutnom snagom i dijeli sa 3600 kako bi se dobila vrijednost stvorenih emisija u jednoj sekundi.

4.4. Prikaz izračunatih rezultata

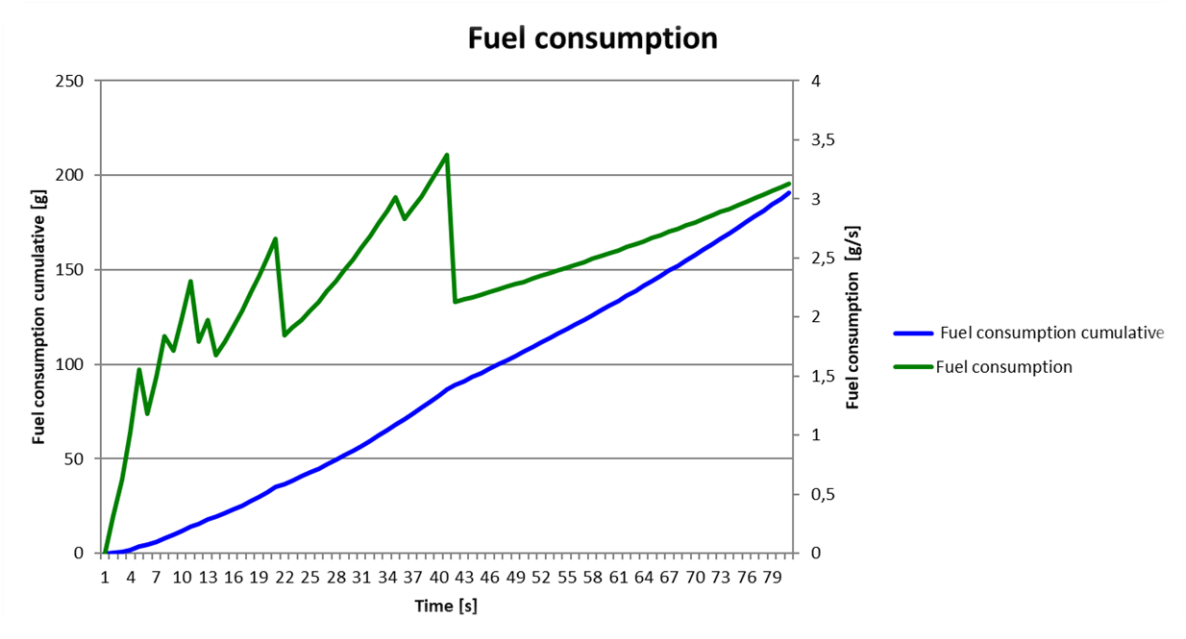
Izračunati rezultati spremaju se u novu datoteku naziva results.xlsx. Datoteka results.xlsx sadrži tri radna lista. Prvi radni list sadrži rezultate s radnim značajkama motora. Na njemu su u stupcima ispisani rezultati za svaki vremenski korak ispitne procedure. Sadrži sedam stupaca, vrijeme [s], ukupna sila otpora [N], snaga motora [kW], srednji efektivni tlak [bar], okretni moment [Nm], trenutna potrošnja goriva [g/s] i kumulativna potrošnja goriva [g]. Trenutna potrošnja goriva pokazuje kolika je potrošnja goriva u gramima po sekundi u svakom vremenskom koraku, dok kumulativna potrošnja goriva pokazuje koliko je grama goriva potrošeno od početka ispitne procedure, odnosno razmatranog zapisa.

Svi ispisani rezultati prikazani su u automatski izrađenim grafovima s njihovim promjenama u vremenu. Izgled radnog lista s radnim značajkama motora prikazan je na slici 39.



Slika 39. Radni list s prikazanim radnim značajkama motora

Na grafu potrošnje goriva prikazane su dvije krivulje, trenutna i kumulativna potrošnja. Primjer takvog grafa prikazan je na slici 40. Radi lakšeg očitavanja, s lijeve strane se nalazi y os kumulativne potrošnje goriva, a s desne y os trenutne potrošnje.



Slika 40. Primjer grafa potrošnje goriva

Drugi radni list sadrži rezultate svih sila otpora. Rezultati su ispisani u pet stupaca, vrijeme, otpor zraka, otpor kotrljanja, otpor ubrzanja i ukupno opterećenje vozila. Svi otpori prikazani su na grafu kao na slici 41.

Slika 41. Primjer grafa sa otporima

Treći radni list sadrži izračunate emisije vozila na zadanoj ispitnoj proceduri. Rezultati su ispisani u 11 stupaca, vrijeme, emisije ugljikovog dioksida, ugljikovog monoksida, ugljikovodika, dušikovih oksida i krutih čestica za trenutno i kumulativno stanje. Za svaku emisiju automatski se generira graf istog tipa kao i graf potrošnje goriva.

4.5. Provjera rada algoritma programa

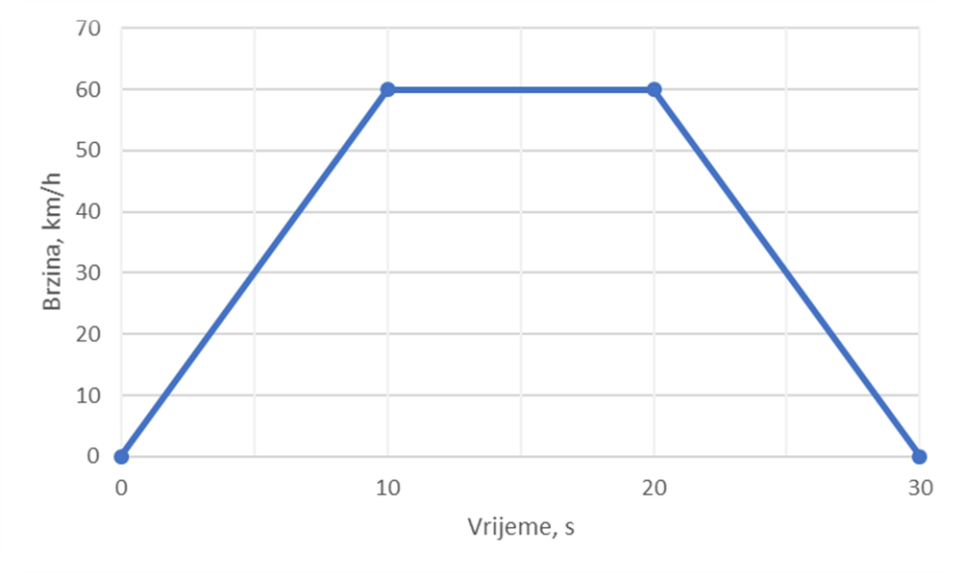
Provjera valjanosti rezultata izračunatih pomoću programa provedena je usporedbom sa ručno izračunatim vrijednostima za svaki korak iste ispitne procedure, a vrijednosti su prikazane u tablicama.

Za ispitnu proceduru izabrano je vozilo Alfa Romeo Mito čiji su ulazni podaci prikazani u tablici 1. Svi rezultati izračunati su istim formulama koje koristi program, a potrebne vrijednosti su očitane iz grafova.

Tablica 1. Podaci o vozilu Alfa Romeo Mito, 2.0 TDI

	Oznaka	Mjerna jedinica	Vrijednost
Ukupna masa vozila	m_v	kg	1300
Čeona površina vozila	A	m^2	1,96
Koeficijent otpora zraka	c_w	-	0,29
Gustoća zraka	ρ_z	kg/m^3	1,204
Ubrzanje gravitacije	g	m/s^2	9,81
Stupanj efikasnosti mjenjača	η_m	-	0,9
Volumen motora	V	m^3	0,001968
Broj taktova	T	-	4
Maksimalna snaga	$P_{e,max}$	kW	103
Dimenzije kotača (pneumatika)	Širina/profil/promjer	mm/-/inch	205/45/17
Dinamički radijus kotača	r_d	m	0,2183713

Izračunate su vrijednosti za tri ispitne procedure. Ispitna procedura A sastavljena je od tri dijela: ubrzavanja iz stanja mirovanja vozila do brzine 60 km/h tokom 10 sekundi, vožnjom na konstantnoj brzini od 60 km/h tokom 10 sekundi i usporavanjem do ponovnog stanja mirovanja tokom 10 sekundi. Na slici 42. grafički je prikazana zadana ispitna procedura.



Slika 42. Ispitna procedura A

Tablica 2. Brzina i ubrzanje za svaki vremenski korak ispitne procedure

Vrijeme, s	Brzina, km/h	Brzina, m/s	Ubrzanje, m/s²
0	0	0,000	0,000
1	6	1,667	1,667
2	12	3,333	1,667
3	18	5,000	1,667
4	24	6,667	1,667
5	30	8,333	1,667
6	36	10,000	1,667
7	42	11,667	1,667
8	48	13,333	1,667
9	54	15,000	1,667
10	60	16,667	1,667
11	60	16,667	0,000
12	60	16,667	0,000
13	60	16,667	0,000
14	60	16,667	0,000
15	60	16,667	0,000
16	60	16,667	0,000
17	60	16,667	0,000
18	60	16,667	0,000
19	60	16,667	0,000
20	60	16,667	0,000
21	54	15,000	-1,667
22	48	13,333	-1,667
23	42	11,667	-1,667
24	36	10,000	-1,667
25	30	8,333	-1,667
26	24	6,667	-1,667
27	18	5,000	-1,667
28	12	3,333	-1,667
29	6	1,667	-1,667
30	0	0,000	-1,667

Ispitna procedura zapisana je u vremenskim koracima od jedne sekunde, te su interpolirane vrijednosti brzine za svaki korak. Iz brzine je prema formuli (5) izračunato ubrzanje vozila.

Tablica 3. Prijenosni omjeri stupnjeva mjenjača i očitani faktori rotirajućih masa

Osovinska redukcija	i_{OR}	2,64829		
Prijenosni omjeri mjenjačke kutije:			$i_{OR}*i_{m,i}$	Očitani k_m
1. stupanj	$i_{m,1}$	3,81	10,089	1,215
2. stupanj	$i_{m,2}$	2,16	5,720	1,09
3. stupanj	$i_{m,3}$	1,48	3,919	1,06
4. stupanj	$i_{m,4}$	1,07	2,833	1,045
5. stupanj	$i_{m,5}$	0,88	2,330	1,04
6. stupanj	$i_{m,6}$	0,74	1,959	1,03

U tablici 3. prikazani su prijenosni omjeri mjenjačke kutije i osovinske redukcije. Uz pomoć njihovog umnoška, očitani su sa slike 25. faktori rotirajućih masa za svaki stupanj prijenosa.

U tablici 4. prikazani su izračunati podaci o brzini vrtnje motora i otporima koji djeluju na vozilo.

Tablica 4. Izračunata brzina vrtnje motora i sile otpora vožnje

Vrijeme, s	Brzina vrtnje motora, okr/min	Otpor kotrljanja, N	Faktor kotrljanja, N	Otpor zraka, N	Otpor ubrzanja, N	Ukupni otpor, N
0	800	0,0	0,009	0,0	0,0	0,0
1	1000	116,3	0,009	1,0	2632,5	2749,8
2	1471	117,8	0,009	3,8	2632,5	2754,1
3	2206	119,4	0,009	8,6	2632,5	2760,4
4	2942	120,9	0,009	15,2	2632,5	2768,6
5	2085	122,5	0,010	23,8	2361,7	2507,9
6	2501	124,0	0,010	34,2	2361,7	2519,9
7	2918	125,6	0,010	46,6	2361,7	2533,8
8	2285	127,2	0,010	60,8	2296,7	2484,7
9	2571	128,9	0,010	77,0	2296,7	2502,5
10	2857	130,6	0,010	95,0	2296,7	2522,3
11	2857	130,6	0,010	95,0	0,0	225,6
12	2857	130,6	0,010	95,0	0,0	225,6
13	2857	130,6	0,010	95,0	0,0	225,6
14	2857	130,6	0,010	95,0	0,0	225,6
15	2857	130,6	0,010	95,0	0,0	225,6
16	2857	130,6	0,010	95,0	0,0	225,6
17	2857	130,6	0,010	95,0	0,0	225,6
18	2857	130,6	0,010	95,0	0,0	225,6
19	2857	130,6	0,010	95,0	0,0	225,6
20	2857	130,6	0,010	95,0	0,0	225,6
21	2571	128,9	0,010	77,0	-2296,7	-2090,8
22	2285	127,2	0,010	60,8	-2296,7	-2108,6
23	2000	125,6	0,010	46,6	-2296,7	-2124,5
24	1714	124,0	0,010	34,2	-2296,7	-2138,4
25	1428	122,5	0,010	23,8	-2296,7	-2150,4
26	1143	120,9	0,009	15,2	-2361,7	-2225,5
27	1251	119,4	0,009	8,6	-2361,7	-2233,7
28	1471	117,8	0,009	3,8	-2632,5	-2510,9
29	1000	116,3	0,009	1,0	-2632,5	-2515,2
30	800	0,0	0,009	0,0	0,0	0,0

U tablici 5. prikazane su izračunate radne značajke motora. Negativne vrijednosti u zadnjem dijelu procedure rezultat su negativnih vrijednosti ukupnih otpora, odnosno otpora ubrzanja koji nastaje zbog inercije mase. Pri takvim vrijednostima, tj. pri usporavanju i kočenju vozila, smatra se da ECU motora isključuje ubrizgavanje goriva pa motor ne stvara štetne emisije.

Tablica 5. Izračunate značajke motora

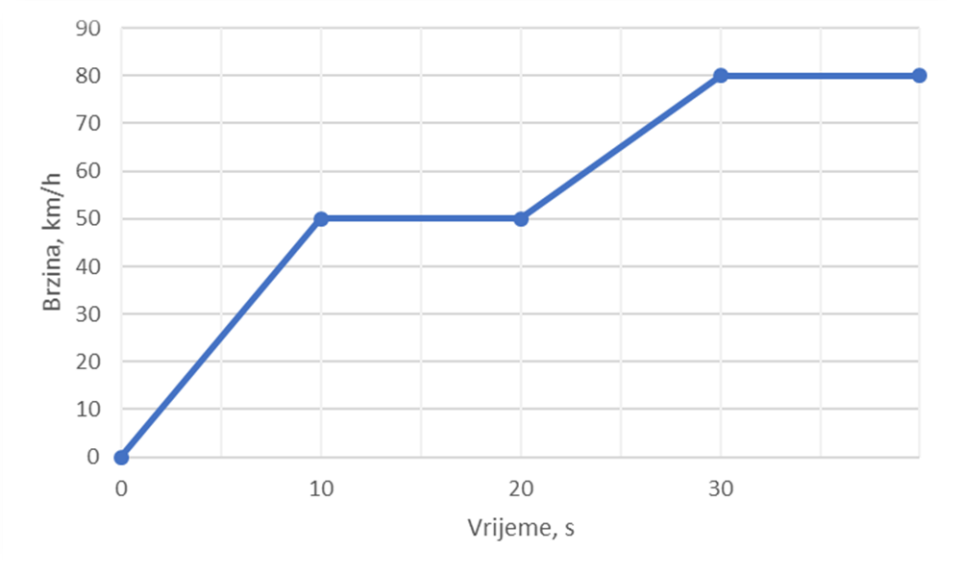
Vrijeme, s	Snaga, kW	Moment, Nm	Tlak, bar
0	0,0	0,0	0,0
1	5,1	66,1	3,1
2	10,2	66,2	4,2
3	15,3	66,4	4,2
4	20,5	66,6	4,3
5	23,2	106,4	6,8
6	28,0	106,9	6,8
7	32,8	107,5	6,9
8	36,8	153,8	9,8
9	41,7	154,9	9,9
10	46,7	156,1	10,0
11	4,2	14,0	0,9
12	4,2	14,0	0,9
13	4,2	14,0	0,9
14	4,2	14,0	0,9
15	4,2	14,0	0,9
16	4,2	14,0	0,9
17	4,2	14,0	0,9
18	4,2	14,0	0,9
19	4,2	14,0	0,9
20	4,2	14,0	0,9
21	-34,8	-129,4	-8,3
22	-31,2	-130,5	-8,3
23	-27,5	-131,5	-8,4
24	-23,8	-132,4	-8,5
25	-19,9	-133,1	-8,5
26	-16,5	-94,4	-8,8
27	-12,4	-94,7	-6,0
28	-9,3	-60,4	-3,9
29	-4,7	-60,5	-2,8
30	0,0	0,0	0,0

U tablici 6. prikazane su vrijednosti brzine vrtnje motora i srednjeg efektivnog tlaka na temelju kojih su se očitale vrijednosti specifične potrošnje na svakom koraku. Te očitane vrijednosti pomnožene su s potrebnom snagom u tom vremenskom koraku i podijeljene s 3600 kako bi se izračunala potrošnja goriva za svaki vremenski korak. Vrijednosti očitane potrošnje goriva i potrošnje izračunate pomoću programa zapisane su u zadnja dva stupca, te su zbrojene u zadnjem redu. U programu su korištene dvije tablice specifične potrošnje s različitim brojem podatka. Prva tablica (tablica g_e, rez1) ima podatke očitane u razmacima od 250 okr/min i 1 bar, a druga (tablica g_e, rez2) u razmacima od 500 okr/min i 2 bar. Također, unošenjem većeg broja radnih točaka u tablicu, odnosno korištenjem tablica „veće rezolucije“ stvara se manja ukupna pogreška.

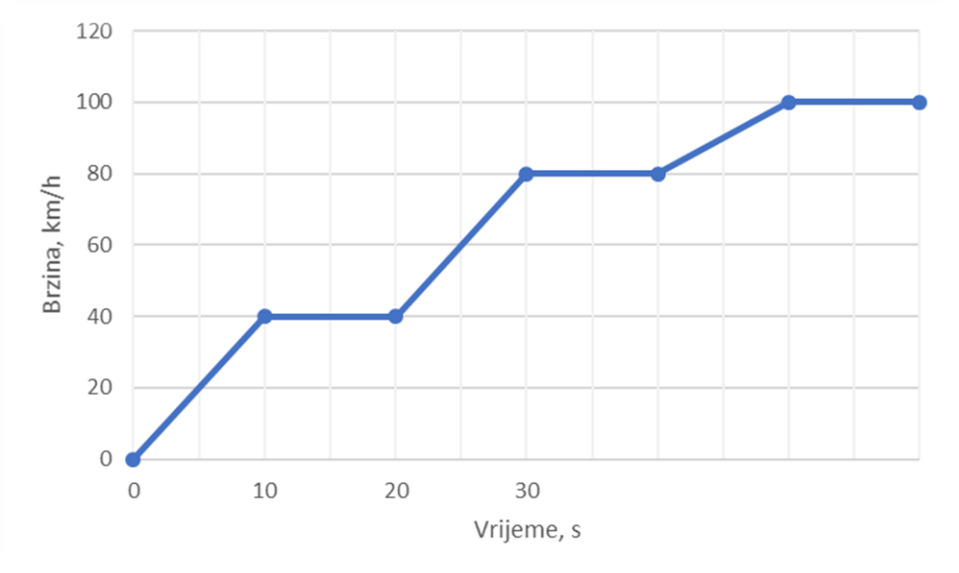
Tablica 6. Usporedba očitanih rezultata i rezultata izračunatih pomoću programa

Brzina vrtnje motora, okr/min	**Tlak, bar**	**Očitana specifična potrošnja, g/kWh**	**Potrošnja, g/s**	**Izračunata potrošnja-p1, g/s**	**Izračunata potrošnja-p2, g/s**
800	0,0	600	0,000	0,000	0,000
1000	3,1	270	0,382	0,377	0,376
1471	4,2	245	0,694	0,670	0,669
2206	4,2	245	1,044	1,025	1,027
2942	4,3	260	1,481	1,474	1,481
2085	6,8	214	1,380	1,398	1,397
2501	6,8	218	1,695	1,692	1,696
2918	6,9	223	2,035	2,035	2,059
2285	9,8	204	2,086	2,081	2,087
2571	9,9	205	2,375	2,386	2,382
2857	10,0	208	2,699	2,740	2,731
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2857	0,9	480	0,557	0,569	0,576
2571	-8,3	0	0,000	0,000	0,000
2285	-8,3	0	0,000	0,000	0,000
2000	-8,4	0	0,000	0,000	0,000
1714	-8,5	0	0,000	0,000	0,000
1428	-8,5	0	0,000	0,000	0,000
1143	-8,8	0	0,000	0,000	0,000
1251	-6,0	0	0,000	0,000	0,000
1471	-3,9	0	0,000	0,000	0,000
1000	-2,8	0	0,000	0,000	0,000
800	0,0	600	0,167	0,167	0,167
		Ukupno:	21,609	21,736	21,834

Isti postupak ponovljen je za druge dvije ispitne procedure. Ispitna procedura B prikazana je na slici 43., a ispitna procedura C na slici 44.



Slika 43. Ispitna procedura B



Slika 44. Ispitna procedura C

Rezultati svih ispitnih procedura prikazani su u tablici 7.

Prva ispitna procedura sastoji se od 30 očitanih radnih točaka, odnosno 20 radnih točaka u kojima je vrijednost potrošnje različita od 0. Druga ispitna procedura sastoji se od 40 radnih točaka, a treća od 60.

Tablica 7. Usporedba ukupne potrošnje goriva tijekom provjere rada programa uporabom 3 različite ispitne procedure

		Ukupna potrošnja dobivena ručnim izračunom	Vrijednosti ukupne potrošnje izračunate uz pomoć programa	
			Tablica g_e, rez1	Tablica g_e, rez2
Ispitna procedura A	Ukupno:	21,608	21,735	21,834
Ispitna procedura B	Ukupno:	40,906	41,026	41,709
Ispitna procedura C	Ukupno:	69,1628	69,486	70,917

Root-mean-square deviation (RMSD) je metoda koja služi za izračun razlika između izmjerenih vrijednosti i vrijednosti dobivenih određenim modelom. Vrijednosti izračunate tom metodom pokazuju pogreške metode. RMSD metoda služi kao mjera točnosti za usporedbu pogrešaka predviđanja različitih modela za određeni skup podataka. RMSD vrijednost je uvijek pozitivna. Vrijednost 0 ukazivala bi na savršeno poklapanje podataka. Dakle, niže RMSD vrijednosti označavaju veću točnost modela kojim se predviđa neka pojava.

U ovom slučaju, RMSD metodom izračunate su razlike rezultata simulacije od rezultata očitanih vrijednosti. RMSD vrijednost izračunata je pomoću formule:

$$RMSD = \sqrt{\frac{\sum_i^N (x_{1,t} - x_{2,t})^2}{N}} \qquad (17)$$

gdje su:

$x_{1,t}$ - vrijednosti ručno izračunate potrošnje goriva,

$x_{2,t}$ - vrijednosti izračunate potrošnje goriva pomoću programa,

i – vremenski korak,

N – ukupan broj vremenskih koraka.

Izračunate devijacije prikazane su u tablici 8.

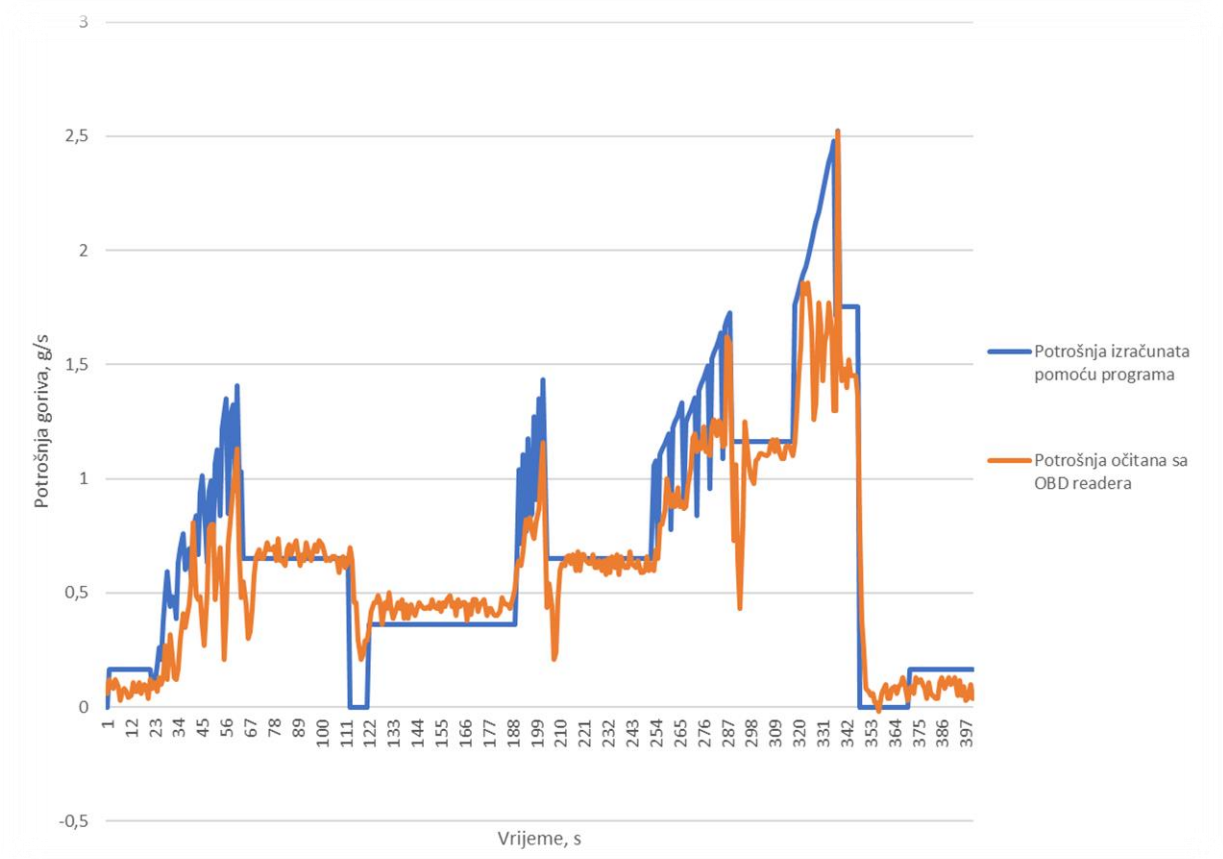
Tablica 8. Devijacija ručno izračunatih rezultata i rezultata dobivenih pomoću simulacije korištenjem tablica različite rezolucije

	Tablica g_e, rez1	Tablica g_e, rez2
Ispitna procedura A	0,0124	0,0147
Ispitna procedura B	0,0318	0,0417
Ispitna procedura C	0,0229	0,0598

Iz prikazanih rezultata ukupne potrošnje i izračunatih devijacija RMSD metodom može se vidjeti da su razlike konačnih rezultata manje ukoliko su tablice specifične potrošnje i emisijskih faktora veće rezolucije, tj. imaju više očitanih radnih točaka.

4.6. Usporedba rezultata s izmjerenim podacima

Kako bi se pokazala smislenost rada programa, uspoređeni su izmjereni podaci sa vozila i rezultati izračunati simulacijom. Podaci sa vozila očitani su pomoću tzv. *OBD readera* (engl. *On-bord diagnostics*) za vrijeme trajanja ispitne procedure. Ista ispitna procedura zadana je i programu, te su uneseni podaci za isto vozilo. Mapa specifične potrošnje nije poznata zbog čega se rezultati ne podudaraju. Unatoč tome, podaci se mogu usporediti na nekoliko načina. Na slici 45. prikazan je graf s uspoređenim potrošnjama goriva.



Slika 45. Usporedba potrošnje izračunate pomoću programa i očitane sa OBD readera

Iz grafa se može vidjeti da se promjene rasta, pada i konstantne potrošnje podudaraju u vremenu. Također, rezultati mjerenja i simulacije istog su reda veličine. Iz toga se može zaključiti da program izračunava smislene otpore vožnje i nastale emisije i potrošnju goriva.

Treba naglasiti da točnost rezultata značajno ovisi o točnosti korištenih mapa (dijagrama) specifične efektivne potrošnje goriva i emisijskih faktora.

4.7. Upute za program

Razvojno okruženje

Program je napisan u programskom jeziku Python (verzija Python 3.7.6.). Za editiranje je korišten program Spyder (verzija 4.0.1.) iz programskog paketa Anaconda. Programski paket moguće je preuzeti sa službene stranice [28]. U njemu su sadržane sve korištene biblioteke programskih rutina, a njihova dokumentacija može se vidjeti na njihovim službenim stranicama [25], [26], [27].

Priprema ulazne datoteke

Ulazne podatke za simulaciju potrebno je unijeti u datoteku *input.xlsx*. Ime datoteke za ulazne podatke ne smije se mijenjati zbog toga što program datoteku traži upravo prema unaprijed definiranom imenu. Na prvom radnom listu imena *Vehicle info* potrebno je unijeti podatke o vozilu. Ćelije za unos podataka nalaze se pored njihovih naziva i mjernih jedinica u kojima je potrebno unijeti podatak, te su označene žutom bojom. Za dimenzije pneumatika potrebno je unijeti podatke kao što je pokazano na primjeru na radnom listu. Prijenosni omjeri pojedinih stupnjeva mjenjača unose se za svaki stupanj, neovisno koliko mjenjač ima stupnjeva prijenosa. Ćelije za unos podataka *Upshift engine speed* i *Downshift engine speed* namijenjena su za podatke brzine vrtnje motora pri kojima se definira način prebacivanja u viši, odnosno niži stupanj prijenosa. Podaci bi trebali biti unutar graničnih vrijednosti za normalan rad motora. Npr. ako je unesena vrijednost za *Upshift engine speed* 3000 okr/min, program će izračunati za svaki vremenski korak u kojem se prijeđe brzina vrtnje motora od 3000 okr/min kao da se u tom trenutku prebacuje u viši stupanj i izračunati će novu brzinu ovisnu o višem stupnju prijenosa.

Radni list *Test procedure* namijenjen je za unos tražene ispitne procedure. On se sastoji od dva stupca, vremena i brzine zadane procedure. Nije potrebno unositi podatke za svaku sekundu ispitne procedure. Program provjerava unesene podatke i nakon toga dijeli ispitnu proceduru na konstantne vremenske korake od jedne sekunde, odnosno radi s frekvencijom 1 Hz. Program vrijednosti brzine za svaki korak izračunava linearnom interpolacijom dvije najbliže unesene točke ispitne procedure.

Tablice specifične potrošnje goriva i emisijskih faktora se unose na ostale radne listove. Za izračun svake emisije potrebno je unijeti tablicu s emisijskim faktorima. Poželjno je da tablica bude očitana što preciznije, da su razmaci između radnih točaka što manji. Ukoliko emisiju neke tvari nije potrebno izračunati ili za nju nije poznat dijagram emisijskih faktora, u tablicu

je moguće unijeti samo jednu radnu točku s iznosima 0 za brzinu vrtnje, srednji efektivni tlak i emisijski faktor ili ostaviti već postojeću generičku tablicu.

Pokretanje programa

Program se može pokrenuti sa bilo kojeg mjesta, neovisno gdje je spremljen. Nužno je da se nalazi u istoj mapi sa datotekom s ulaznim podacima *input.xlsx*. Ukoliko se nalaze u istoj mapi i datoteka s ulaznim podacima je spremljena pod tim imenom program se može pokrenuti.

Pokretanje programa može se izvršiti na dva načina. Prvi je putem bilo kojeg Python interpretera u čijoj su verziji instalirane navedene biblioteke koje se koriste u ovom programu. Potrebno je otvoriti programski kôd python.py u željenom interpreteru i samo pokrenuti izvršavanje kôda. Ukoliko se datoteka s ulaznim podacima ne nalazi u istoj mapi ili neka od biblioteka nije instalirana, interpreter će dati upozorenje i programski kôd se neće izvršiti.

Drugi način pokretanja programa je pomoću aplikacije *program.exe*. Datoteku s ulaznim podacima je potrebno staviti u istu mapu s navedenim imenom i pokrenuti aplikaciju. Ukoliko datoteka s ulaznim podacima nije spremljena pod tim imenom ili se ne nalazi u istoj mapi, programski kôd se neće izvršiti.

U oba načina potrebno je pravilno unijeti sve potrebne ulazne podatke. Ukoliko su svi podaci pravilno uneseni, nakon izvršavanja programskog kôda, generirati će se datoteka *results.xlsx*. Svakim narednim pokretanjem programa, stara datoteka s rezultatima će se prebrisati, pa je potrebno preimenovati datoteku ukoliko se rezultati žele sačuvati.

5. Zaključak

U sklopu ovog rada prikazan je problem emisija štetnih tvari iz vozila s motorima s unutarnjim izgaranjem, razlozi i metode njihovih mjerenja i problemi dugotrajnosti i skupoće. Kao jedno od rješenja problema mjerenja emisija vozila, napravljen je jednostavan alat za određivanje potrošnje goriva i emisija motora za zadani radni ciklus, te je prikazan i opisan način njegovog rada. Funkcionalnost rada programa provjerena je provjerom algoritma programa po kojemu radi i usporedbom sa rezultatima mjerenja. Algoritam programa provjeren je usporedbom analitičkog izračuna sila otpora vožnje i značajki rada motora. Nakon izračuna potrebnih elemenata, očitavanjem podataka iz grafova i njihovim izračunom za svaki vremenski korak zadane ispitne procedure izračunata je potrošnja goriva. Zatim su uspoređeni rezultati analitičkog izračuna i vrijednosti izračunate uz pomoć programa. Nastale razlike u rezultatima mogu se pripisati ograničenim brojem radnih točaka za radno područje motora u tablici specifične potrošnje, gdje su odstupanja manja u simulaciji u kojoj je korištena tablica sa većim brojem radnih točaka. Smislenost rezultata programskog alata provjerena je i usporedbom rezultata simulacije i rezultata mjerenja emisija na određenoj ispitnoj proceduri. Zbog manjka identične mape specifične potrošnje testiranog vozila korištena je generička verzija, zbog čega rezultati nisu identični. Iz rezultata se moglo vidjeti da se periodi povećanja, smanjenja i konstantne potrošnje goriva poklapaju, te su istog reda veličine.

Cilj je bio prikupiti i druge mjerene podatke s radnih strojeva i vozila, ali zbog specifičnih uvjeta izrade diplomskog rada u pandemijskim uvjetima, neke od planiranih stavki nisu mogle biti ostvarene.

S druge strane, kako je pokazano da programski algoritam ispravno funkcionira, otvara se mogućnost za nastavak rada na ovom programu i daljnje unaprjeđenje kako programa tako i postupaka evaluacije potrošnje goriva i emisija pojedinog vozila ili radnog stroja pokretanog motorom s unutarnjim izgaranjem.

Kao moguća poboljšanja programa, moguće bi bilo dodati dodatne režime rada motora, kao npr. pokretanje hladnog motora i rad motora u ovisnosti o okolišnoj temperaturi, te dodati potprograme nekih sustava kao što je start-stop sustav. Daljnja poboljšanja su moguća i kod korištenju samog programa i datoteka ulaznih podataka i rezultata. Radi bržeg i lakšeg upravljanja programom, potrebno bi bilo dodati manualni izbor datoteke s ulaznim podacima koja neće trebati unaprijed određeno ime. Kod generiranja datoteke rezultata, potrebno je modificirati program da se stare datoteke rezultata ne prepisuju novom, već generiraju pod drugim imenom.

Literatura

[1] Springer, George: Engine Emissions, Springer US, 1973.

[2] Bari, Saiful: Diesel engine - Combustion, Emissions and Condition Monitoring, IntechOpen, 2013.

[3] Izvješće Europskog parlamenta o emisijama CO_2, https://www.europarl.europa.eu/news/hr/headlines/society/20190313STO31218/emisije-co2-u-prometu-eu-a-cinjenice-i-brojke, pristupljeno: 27.06.2020.

[4] Izvješća Europske agencije za okoliš, https://www.eea.europa.eu/themes/transport, pristupljeno: 30.06.2020.

[5] Izvješća Europske agencije za okoliš, https://www.eea.europa.eu/hr/themes/prijevoz/intro, pristupljeno:30.06.2020.

[6] Raspodjela energije vozila, https://www.fueleconomy.gov/feg/atv.shtml, pristupljeno: 28.06.2020.

[7] Uredba (EZ) br 433/2009, https://eur-lex.europa.eu/legal-content/HR/TXT/HTML/?uri=CELEX:32009R0443&from=HR, pristupljeno: 05.07.2020.

[8] Uredba (EZ) br 510/2011, https://eur-lex.europa.eu/legal-content/HR/TXT/HTML/?uri=CELEX:32011R0510&from=HR, pristupljeno: 05.07.2020.

[9] Direktiva 70/220/EEC, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:31970L0220&from=EN, pristupljeno 05.07.2020.

[10] Podaci o WLTP ciklusu, https://www.wltpfacts.eu/, pristupljeno: 25.06.2020.

[11] Delphi, podaci o emisijama vozila, https://www.delphi.com/newsroom/press-release/delphi-technologies-releases-updated-worldwide-emissions-standards-guide, pristupljeno: 29.06.2020.

[12] Europski emisijski standardi, https://dieselnet.com/standards/eu/ld.php, pristupljeno: 25.05.2020.

[13] JRC, CO2MPAS, službena stranica, https://co2mpas.io/, pristupljeno: 01.07.2020.

[14] Kontekst nastanka programa CO2MPAS, https://e3p.jrc.ec.europa.eu/articles/co2mpas, pristupljeno: 02.07.2020.

[15] Prezentacija programa CO2MPAS, https://ec.europa.eu/jrc/sites/jrcsh/files/co2mpas-introduction_20oct-morning_2.pdf, pristupljeno 02.07.2020.

[16] Princip rada programa CO2MPAS, https://www.tuvsud.com/en/e-ssentials-newsletter/automotive-essentials/e-ssentials-3-2017/co2mpas---authorized-manipulation, pristupljeno: 02.07.2020.

[17] Makridis M., Mattas K., Mogno C., Ciuffo B., Fontaras G.,The impact of automation and connectivity on traffic flow and CO2 emissions, A detailed microsimulation study, Atmospheric Environment, Volume 226, 2020.

[18] Ciuffo, B. & Fontaras, Georgios. Models and scientific tools for regulatory purposes: The case of CO 2 emissions from light duty vehicles in Europe. Energy Policy Volume 109., 2017.

[19] Opći podaci o programu VECTO, http://inconvenienttruck.eu/testingtruckco2emissions/, pristupljeno 26.06.2020.

[20] Opći podaci o programu VECTO, https://www.continental-tires.com/transport/fleetsolutions/co2-regulations-vecto/infographic, pristupljeno 26.06.2020.

[21] Europska komisija, opći podaci o programu VECTO, https://ec.europa.eu/clima/policies/transport/vehicles/vecto_en, pristupljeno: 26.06.2020.

[22] Izvješće i upute za program VECTO, https://ec.europa.eu/clima/sites/clima/files/transport/vehicles/docs/sr7_lot4_final_report_en.pdf, pristupljeno 26.06.2020.

[23] Mahalec, Ivan, Lulić, Zoran, Kozarac, Darko: Motori s unutarnjim izgaranjem, FSB Zagreb, 2012.

[24] Naunheimer, Harald, Bertsche, Bernd, Ryborz, Joachim, Novak, Wolfgang, Fietkau, Peter: Automotive transmissions: Fundamentals, selection, design and application, 2011.

[25] Dokumentacija biblioteke NumPy, https://numpy.org/, pristupljeno 25.06.2020.

[26] Dokumentacija biblioteke Xlrd, https://xlrd.readthedocs.io/, pristupljeno 25.06.2020.

[27] Dokumentacija biblioteke XlsxWriter, https://xlsxwriter.readthedocs.io/, pristupljeno 25.06.2020.

[28] Programski paket Anaconda, https://www.anaconda.com/products/individual, pristupljeno 25.06.2020.

PRILOZI

I. Kôd programa

Prilog I.

Kôd programa

```
# Software name:
# Simple tool for determining fuel consumption and engine emissions for a given duty cycle
#
#
# ime programskog alata:
# Jednostavan alat za određivanje potrošnje goriva i emisija motora za zadani radni ciklus
#
#
# author: Boris Babić
#




import numpy as np
import xlrd
import xlsxwriter
from sys import exit



book_input = xlrd.open_workbook('input.xlsx')

sheet_vt = book_input.sheet_by_name('Test procedure')

sheet_vehicle_info = book_input.sheet_by_name('Vehicle info')

sheet_fuel_consumption = book_input.sheet_by_name('fuel_consumption')
sheet_ef_co2 = book_input.sheet_by_name('ef_CO2')
sheet_ef_co = book_input.sheet_by_name('ef_CO')
sheet_ef_hc = book_input.sheet_by_name('ef_HC')
```

```
sheet_ef_nox = book_input.sheet_by_name('ef_NOx')
sheet_ef_pm = book_input.sheet_by_name('ef_PM')


# loading vehicle data

# general data
vehicle_mass = sheet_vehicle_info.cell_value (4, 3)
vehicle_surface = sheet_vehicle_info.cell_value (5, 3)
air_resistance_coefficient = sheet_vehicle_info.cell_value (6, 3)
air_density = sheet_vehicle_info.cell_value (7, 3)
g = sheet_vehicle_info.cell_value (8, 3)
transmission_ef_coef = sheet_vehicle_info.cell_value (9, 3)
engine_volume = sheet_vehicle_info.cell_value (10, 3)
stroke = sheet_vehicle_info.cell_value (11, 3)
power_max = sheet_vehicle_info.cell_value (12, 3)

# tyre dimensions
tyre_HR_1 = sheet_vehicle_info.cell_value (14, 1) / 1000
tyre_HR_2 = sheet_vehicle_info.cell_value (14, 2) / 1000
tyre_R = sheet_vehicle_info.cell_value (14, 3) * 0.0254 / 2
tyre_HR = tyre_HR_1 * tyre_HR_2

tyre_r_dyn = 0.97 * (tyre_R + tyre_HR)


# changing gears data
engine_speed_upshift = sheet_vehicle_info.cell_value (18, 2)
engine_speed_downshift = sheet_vehicle_info.cell_value (19, 2)
engine_idle_speed = sheet_vehicle_info.cell_value (21, 2)



# axle reduction
```

```
i_axle = sheet_vehicle_info.cell_value (4, 7)


# loading gearbox ratios

gear_ratio = np.array (sheet_vehicle_info.cell_value (6, 7))
n=7
m=0
while m == 0:
    x = sheet_vehicle_info.cell_value (n, 7)
    if x !="":
        gear_ratio = np.append(gear_ratio, x)
        n += 1
    else:
        m=1



# loading test procedure (time and velocity)

time = np.zeros(shape=(sheet_vt.nrows-1), dtype=int)
velocity = np.zeros(shape=(sheet_vt.nrows-1), dtype=float)

for i in range(sheet_vt.nrows-1):
    time [i]=sheet_vt.cell_value (i+1,0)
    velocity [i]=sheet_vt.cell_value (i+1,1)


for i in range (1, time.size):
    if time[i-1] > time[i]:
        workbook_results = xlsxwriter.Workbook ("results.xlsx")
        sheet_warning = workbook_results.add_worksheet("warning")
        cell_format_red = workbook_results.add_format({'bold': True, 'font_color': 'red'})
```

```
        sheet_warning.write(1,1, "Warning: time in test procedure is not valid. Please check.", cell_format_red)
        workbook_results.close()
        exit()



# adding time and velocity values for 1 Hz intervals

for i in range (time[-1]):
  if i !=time [i] and i!=0:
      time = np.insert(time, i, i)

      v_interpolation = velocity[i-1]+((time[i]-time[i-1])*(velocity[i]-velocity[i-1])/(time[i+1]-time[i-1]))
      velocity = np.insert(velocity, i, v_interpolation)

velocity = velocity / 3.6




#   acceleration calculation

acceleration = np.zeros(shape=(time.size), dtype=float)

for i in range (1, time.size):
    acc = (velocity [i] - velocity [i-1])
    acceleration [i]= acc
```

```
# engine speed calculation

engine_speed = np.zeros(shape=(time.size), dtype=float)
gear = np.zeros(shape=(time.size), dtype=int)


j=0

for i in range (time.size):
    eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)

    if velocity [i] == 0:
        eng_spd = engine_idle_speed
        engine_speed [i] = eng_spd
    else:
        if eng_spd > engine_speed_upshift and acceleration[i]>0:
            if j == gear_ratio.size-1:
                engine_speed [i] = eng_spd
            else:
                j +=1
                eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)
                engine_speed [i] = eng_spd
        elif eng_spd < engine_speed_downshift and acceleration[i]<0:
            if j == 0:
                eng_spd = engine_speed_downshift
                engine_speed [i] = eng_spd
            else:
                j -= 1
                eng_spd = velocity [i] * 30 * gear_ratio [j] * i_axle / (np.pi * tyre_r_dyn)
                engine_speed [i] = eng_spd
        elif eng_spd<engine_idle_speed:
            engine_speed [i]=engine_speed_downshift
        else:
```

```
            engine_speed [i] = eng_spd

    gear [i] = j

#
# resistance calculation
#

# rolling resistance calculation

force_rolling = np.zeros(shape=(time.size), dtype=float)
rolling_resistance_coefficient = np.zeros(shape=(time.size), dtype=float)

for i in range (1, time.size):
    v = velocity [i] * 3.6
    rolling_resistance_coefficient [i] = 0.0090 + 0.0020*(v/100) + 0.0003*((v/100)**4)

    if velocity[i]==0:
        force_rolling [i] = 0
    else:
        force_rolling [i] = vehicle_mass * rolling_resistance_coefficient [i] * g

# air resistance calculation

force_airdrag = np.zeros(shape=(time.size), dtype=float)

for i in range (time.size):
    force_airdrag [i] = 0.5 * (velocity[i])**2 * vehicle_surface * air_resistance_coefficient *
air_density



# acceleration resistance calculation

force_acc = np.zeros(shape=(time.size), dtype=float)
```

```
km_factor = np.zeros(shape=(gear_ratio.size), dtype=float)

for i in range (km_factor.size):
   red = i_axle * gear_ratio [i]

   km_factor [i] = 0.0013*red**2 + 0.0068*red +1.013

for i in range (time.size):
   if velocity [i]==0:
      force_acc[i]=0
   else:
      force_acc [i] = vehicle_mass * acceleration [i] * km_factor [gear[i]]



# forces sum

force_all = force_airdrag + force_rolling + force_acc




# required power calculation

power = np.zeros(shape=(time.size), dtype=float)

limit_power = False

for i in range (time.size):
   power [i] = force_all [i] * velocity [i] / (1000 * transmission_ef_coef)

   if power [i] > power_max:
      limit_power = True
```

```
# mean effective pressure calculation

pressure_ef = np.zeros(shape=(time.size), dtype=float)

for i in range (time.size):
   pressure_ef [i] = power [i] / (engine_volume * (engine_speed [i] / (30 * stroke)) * 100)




# moment calculation

moment = np.zeros(shape=(time.size), dtype=float)


for i in range (time.size):
   moment [i] = (force_all [i] * tyre_r_dyn) / (gear_ratio[gear[i]] * i_axle * transmission_ef_coef)


#
#
#
# function for loading table of emission factors

def table_loading (table_name, sheet_name):
   for c in range(sheet_name.ncols-1):
      for r in range(sheet_name.nrows-1):
         table_name [r][c] = sheet_name.cell_value(r+1, c+1)


table_fuel_consumption = np.zeros(shape=(sheet_fuel_consumption.nrows-1, sheet_fuel_consumption.ncols-1))
```

```
table_loading (table_fuel_consumption, sheet_fuel_consumption)

table_ef_co2 = np.zeros(shape=(sheet_ef_co2.nrows-1, sheet_ef_co2.ncols-1))

table_loading (table_ef_co2 , sheet_ef_co2 )

table_ef_co = np.zeros(shape=(sheet_ef_co.nrows-1, sheet_ef_co.ncols-1))

table_loading (table_ef_co , sheet_ef_co )

table_ef_hc = np.zeros(shape=(sheet_ef_hc.nrows-1, sheet_ef_hc.ncols-1))

table_loading (table_ef_hc, sheet_ef_hc)

table_ef_nox = np.zeros(shape=(sheet_ef_nox.nrows-1, sheet_ef_nox.ncols-1))

table_loading (table_ef_nox, sheet_ef_nox)

table_ef_pm = np.zeros(shape=(sheet_ef_pm.nrows-1, sheet_ef_pm.ncols-1))

table_loading (table_ef_pm, sheet_ef_pm)

#

#
# calculation of emission or fuel consumption in 1 Hz intervals
```

```
def emission_calculation (ef_table, sheet_name, emission_result, emission_result_cumulative):

    for i in range (1, time.size):
        a= pressure_ef [i]
        b= engine_speed [i]

        i1=0
        i2=0

        # finding values of engine speed for interpolation

        for ia in range (1, sheet_name.ncols-2):
            x1=ef_table[0][ia]
            x2=ef_table[0][ia+1]
            if b >= x1 and b < x2:
                i1=ia

            elif b >= ef_table[0][sheet_name.ncols-2]:
                i1=-1

        # finding values of mean effective pressure for interpolation

        for ib in range (1, sheet_name.nrows-2):
            x1=ef_table[ib][0]
            x2=ef_table[ib+1][0]
            if a >= x1 and a < x2:
                i2=ib

            elif a < 0:
                i2=0

            elif a >= ef_table[sheet_name.nrows-2][0]:
```

```
            i2=-1

    if i1==-1 or i2==-1:
        emission_result [i] = -1

    elif i1==0 or i2==0:
        emission_result [i] = 0

    else:
        # 1st interpolation, same pressure
        ya1 = ef_table[i2][i1]
        yb1 = ef_table[i2][i1+1]

        xn1 = ef_table[0][i1]
        xn2 = ef_table[0][i1+1]

        #2nd interpolation, same pressure
        ya2 = ef_table[i2+1][i1]
        yb2 = ef_table[i2+1][i1+1]

        # interpolation for required value
        xp1 = ef_table[i2][0]
        xp2 = ef_table[i2+1][0]

        if ya1 > 0 and yb1 > 0 and ya2 > 0 and yb2 > 0:

            y1 = ya1+(b-xn1)*(yb1-ya1)/(xn2-xn1)

            y2 = ya2+(b-xn1)*(yb2-ya2)/(xn2-xn1)

            y3 = y1+((a-xp1)*(y2-y1)/(xp2-xp1))

            if power[i]==0:
                emission_result [i] = y3 / 3600
```

```
            else:
                emission_result [i] = y3 * power [i]

        else:

            emission_result [i] = -1

    emission_result [i] = emission_result [i] / 3600
    emission_result_cumulative [i] = emission_result_cumulative [i-1] + emission_result [i]

# calculation of fuel consumption
fuel_consumption = np.zeros(shape=(time.size), dtype=float)
fuel_consumption_cumulative = np.zeros(shape=(time.size), dtype=float)
emission_calculation (table_fuel_consumption, sheet_fuel_consumption, fuel_consumption,
fuel_consumption_cumulative)


emission_co2 = np.zeros(shape=(time.size), dtype=float)
emission_co2_cumulative = np.zeros(shape=(time.size), dtype=float)

emission_calculation (table_ef_co2, sheet_ef_co2, emission_co2, emission_co2_cumulative)


emission_co = np.zeros(shape=(time.size), dtype=float)
emission_co_cumulative = np.zeros(shape=(time.size), dtype=float)

emission_calculation (table_ef_co, sheet_ef_co, emission_co, emission_co_cumulative)


emission_hc = np.zeros(shape=(time.size), dtype=float)
emission_hc_cumulative = np.zeros(shape=(time.size), dtype=float)

emission_calculation (table_ef_hc, sheet_ef_hc, emission_hc, emission_hc_cumulative)
```

```
emission_nox = np.zeros(shape=(time.size), dtype=float)
emission_nox_cumulative = np.zeros(shape=(time.size), dtype=float)

emission_calculation (table_ef_nox, sheet_ef_nox, emission_nox, emission_nox_cumulative)



emission_pm = np.zeros(shape=(time.size), dtype=float)
emission_pm_cumulative = np.zeros(shape=(time.size), dtype=float)

emission_calculation (table_ef_pm, sheet_ef_pm, emission_pm, emission_pm_cumulative)



#
#
#
# writing results in xlsx file
#
#

# make Excel file
workbook_results = xlsxwriter.Workbook ("results.xlsx")

#
#
# sheet with general results
#
#
# add sheet to result Excel file
sheet_results_general = workbook_results.add_worksheet("General results")



cell_format_bold = workbook_results.add_format({'bold': True, 'align': 'center'})
```

```
cell_format_red = workbook_results.add_format({'bold': True, 'font_color': 'red'})

sheet_results_general.set_column(0, 4, 12)
sheet_results_general.set_column(5, 6, 16)

sheet_results_general.write(0,0, "Time", cell_format_bold)
sheet_results_general.write(0,1, "Force sum", cell_format_bold)
sheet_results_general.write(0,2, "Power", cell_format_bold)
sheet_results_general.write(0,3, "Pressure", cell_format_bold)
sheet_results_general.write(0,4, "Moment", cell_format_bold)
sheet_results_general.write(0,5, "Fuel consumption", cell_format_bold)
sheet_results_general.write(0,6, "FC cumulative", cell_format_bold)



sheet_results_general.write_column(1,0, time)
sheet_results_general.write_column(1,1, force_all)
sheet_results_general.write_column(1,2, power)
sheet_results_general.write_column(1,3, pressure_ef)
sheet_results_general.write_column(1,4, moment)
sheet_results_general.write_column(1,5, fuel_consumption)
sheet_results_general.write_column(1,6, fuel_consumption_cumulative)

# if max power reached

if limit_power == True:
   sheet_results_general.write(1,8, "Warning: max power reached", cell_format_red)

# chart - force
chart_1 = workbook_results.add_chart({'type': 'line'})
chart_1.add_series({'values':  ['General results', 1, 1, time.size, 1]})
chart_1.set_x_axis({'name': 'Time [s]'})
chart_1.set_y_axis({'name': 'Force [N]'})
chart_1.set_legend({'none': True})
sheet_results_general.insert_chart ('L2', chart_1)
```

```
# chart - power
chart_2 = workbook_results.add_chart({'type': 'line'})
chart_2.add_series({'values':  ['General results', 1, 2, time.size, 2]})
chart_2.set_x_axis({'name': 'Time [s]'})
chart_2.set_y_axis({'name': 'Power [kW]'})
chart_2.set_legend({'none': True})
sheet_results_general.insert_chart ('L17', chart_2)

# chart - pressure
chart_3 = workbook_results.add_chart({'type': 'line'})
chart_3.add_series({'values':  ['General results', 1, 3, time.size, 3]})
chart_3.set_x_axis({'name': 'Time [s]'})
chart_3.set_y_axis({'name': 'Mean effective pressure [bar]'})
chart_3.set_legend({'none': True})
sheet_results_general.insert_chart ('T2', chart_3)

#chart - moment
chart_4 = workbook_results.add_chart({'type': 'line'})
chart_4.add_series({'values':  ['General results', 1, 4, time.size, 4]})
chart_4.set_x_axis({'name': 'Time [s]'})
chart_4.set_y_axis({'name': 'Moment [Nm]'})
chart_4.set_legend({'none': True})
sheet_results_general.insert_chart ('T17', chart_4)

# chart fuel consumption
chart_fc = workbook_results.add_chart({'type': 'line'})
chart_fc.set_title({'name': 'Fuel consumption'})

chart_fc.add_series({
   'values': ['General results', 1, 5, time.size, 5],
   'name': 'Fuel consumption',
   'line':   {'color': 'green'},
   'y2_axis': 1,
```

```
})
chart_fc.add_series({
   'values': ['General results', 1, 6, time.size, 6],
   'name': ' Fuel consumption cumulative',
   'line':   {'color': 'blue'},
})

chart_fc.set_x_axis({'name': 'Time [s]'})
chart_fc.set_y_axis({'name': 'Fuel consumption cumulative [g]'})
chart_fc.set_y2_axis({'name': 'Fuel consumption  [g/s]'})
sheet_results_general.insert_chart ('L32', chart_fc)



# sheet with force results
#
#
sheet_results_force = workbook_results.add_worksheet("Forces")

sheet_results_force.set_column(0, 10)
sheet_results_force.set_column(1, 4, 20)

sheet_results_force.write(0,0, "Time", cell_format_bold)
sheet_results_force.write(0,1, "Air resistance", cell_format_bold)
sheet_results_force.write(0,2, "Rolling resistance", cell_format_bold)
sheet_results_force.write(0,3, "Acceleration resistance", cell_format_bold)
sheet_results_force.write(0,4, "Force sum", cell_format_bold)



sheet_results_force.write_column(1,0, time)
sheet_results_force.write_column(1,1, force_airdrag)
sheet_results_force.write_column(1,2, force_rolling)
sheet_results_force.write_column(1,3, force_acc)
sheet_results_force.write_column(1,4, force_all)
```

```
chart_forces = workbook_results.add_chart({'type': 'line'})

chart_forces.set_title({'name': 'Forces'})

chart_forces.add_series({
   'values': ['Forces', 1, 1, time.size, 1],
   'name': 'Air resistance',
   'line':   {'color': 'orange'},
})
chart_forces.add_series({
   'values': ['Forces', 1, 2, time.size, 2],
   'name': 'Rolling resistance',
   'line':   {'color': 'blue'},
})
chart_forces.add_series({
   'values': ['Forces', 1, 3, time.size, 3],
   'name': 'Acceleration resistance',
   'line':   {'color': 'green'},
})
chart_forces.add_series({
   'values': ['Forces', 1, 4, time.size, 4],
   'name': 'Force sum',
   'line':   {'color': 'red'},
})

chart_forces.set_x_axis({'name': 'Time [s]'})
chart_forces.set_y_axis({'name': 'Force [N]'})
sheet_results_force.insert_chart ('G2', chart_forces,
                                  {'x_scale': 2, 'y_scale': 2})


# sheet with emission results
```

```
#
#
sheet_results_emission = workbook_results.add_worksheet("Emissions")

#
#
# CO2
#
sheet_results_force.set_column(0, 10)
sheet_results_emission.set_column(1, 10, 15)

sheet_results_emission.write(0,0, "Time", cell_format_bold)
sheet_results_emission.write(0,1, "CO2", cell_format_bold)
sheet_results_emission.write(0,2, "CO2_cumulative", cell_format_bold)



sheet_results_emission.write_column(1,0, time)
sheet_results_emission.write_column(1,1, emission_co2)
sheet_results_emission.write_column(1,2, emission_co2_cumulative)

# chart co2
chart_co2 = workbook_results.add_chart({'type': 'line'})
chart_co2.set_title({'name': 'CO2'})

chart_co2.add_series({
   'values': ['Emissions', 1, 1, time.size, 1],
   'name': 'CO2',
   'line':   {'color': 'green'},
   'y2_axis': 1,
})
chart_co2.add_series({
   'values': ['Emissions', 1, 2, time.size, 2],
   'name': ' CO2_cumulative',
   'line':   {'color': 'blue'},
```

```
})

chart_co2.set_x_axis({'name': 'Time [s]'})
chart_co2.set_y_axis({'name': 'CO2 cumulative [g]'})
chart_co2.set_y2_axis({'name': 'CO2 [g/s]'})
sheet_results_emission.insert_chart ('M2', chart_co2)




#
#
# CO
#
sheet_results_emission.write(0,3, "CO", cell_format_bold)
sheet_results_emission.write(0,4, "CO_cumulative", cell_format_bold)



sheet_results_emission.write_column(1,3, emission_co)
sheet_results_emission.write_column(1,4, emission_co_cumulative)

# chart CO
chart_co = workbook_results.add_chart({'type': 'line'})
chart_co.set_title({'name': 'CO'})

chart_co.add_series({
   'values': ['Emissions', 1, 3, time.size, 3],
   'name': 'CO',
   'line':   {'color': 'green'},
   'y2_axis': 1,
})
chart_co.add_series({
   'values': ['Emissions', 1, 4, time.size, 4],
   'name': ' CO_cumulative',
```

```
    'line':   {'color': 'blue'},
})

chart_co.set_x_axis({'name': 'Time [s]'})
chart_co.set_y_axis({'name': 'CO cumulative [g]'})
chart_co.set_y2_axis({'name': 'CO [g/s]'})
sheet_results_emission.insert_chart ('M17', chart_co)



#
#
# HC
#
sheet_results_emission.write(0,5, "HC", cell_format_bold)
sheet_results_emission.write(0,6, "HC_cumulative", cell_format_bold)



sheet_results_emission.write_column(1,5, emission_hc)
sheet_results_emission.write_column(1,6, emission_hc_cumulative)

# chart HC
chart_hc = workbook_results.add_chart({'type': 'line'})
chart_hc.set_title({'name': 'HC'})

chart_hc.add_series({
    'values': ['Emissions', 1, 5, time.size, 5],
    'name': 'HC',
    'line':   {'color': 'green'},
    'y2_axis': 1,
})
chart_hc.add_series({
    'values': ['Emissions', 1, 6, time.size, 6],
    'name': ' HC_cumulative',
    'line':   {'color': 'blue'},
```

```
})

chart_hc.set_x_axis({'name': 'Time [s]'})
chart_hc.set_y_axis({'name': 'HC cumulative [g]'})
chart_hc.set_y2_axis({'name': 'HC [g/s]'})
sheet_results_emission.insert_chart ('M32', chart_hc)

#
#
# NOx
#
sheet_results_emission.write(0,7, "NOx", cell_format_bold)
sheet_results_emission.write(0,8, "NOx_cumulative", cell_format_bold)



sheet_results_emission.write_column(1,7, emission_nox)
sheet_results_emission.write_column(1,8, emission_nox_cumulative)

# chart NOx
chart_nox = workbook_results.add_chart({'type': 'line'})
chart_nox.set_title({'name': 'NOx'})

chart_nox.add_series({
   'values': ['Emissions', 1, 7, time.size, 7],
   'name': 'NOx',
   'line':   {'color': 'green'},
   'y2_axis': 1,
})
chart_nox.add_series({
   'values': ['Emissions', 1, 8, time.size, 8],
   'name': ' NOx_cumulative',
   'line':   {'color': 'blue'},
})
```

```
chart_nox.set_x_axis({'name': 'Time [s]'})
chart_nox.set_y_axis({'name': 'NOx cumulative [g]'})
chart_nox.set_y2_axis({'name': 'NOx [g/s]'})
sheet_results_emission.insert_chart ('U2', chart_nox)

#
#
# PM
#
sheet_results_emission.write(0,9, "PM", cell_format_bold)
sheet_results_emission.write(0,10, "PM_cumulative", cell_format_bold)



sheet_results_emission.write_column(1,9, emission_pm)
sheet_results_emission.write_column(1,10, emission_pm_cumulative)

# chart PM
chart_pm = workbook_results.add_chart({'type': 'line'})
chart_pm.set_title({'name': 'Particulate Matter'})

chart_pm.add_series({
   'values': ['Emissions', 1, 9, time.size, 9],
   'name': 'PM',
   'line':   {'color': 'green'},
   'y2_axis': 1,
})
chart_pm.add_series({
   'values': ['Emissions', 1, 10, time.size, 10],
   'name': ' PM_cumulative',
   'line':   {'color': 'blue'},
})

chart_pm.set_x_axis({'name': 'Time [s]'})
chart_pm.set_y_axis({'name': 'PM cumulative [g]'})
```

```
chart_pm.set_y2_axis({'name': 'PM [g/s]'})
sheet_results_emission.insert_chart ('U17', chart_pm)

workbook_results.close()
```