

PyEmission 1.0 User Manual

1. Introduction

PyEmission is a Python library for the estimation of vehicular emissions and fuel consumption. This tool covers a wide range of light-duty motor vehicles including motorcycle, passenger car, passenger truck, and light commercial truck. The tool only takes second-by-second driving cycle and vehicle characteristics data as inputs and generates results of vehicular emissions (CO₂, CO, NO_x, and HC) and fuel consumption. This tool can estimate both tailpipe and corresponding upstream (Well-To-Pump) emissions. The following sections describe all the steps required to use the library from data preparation to output generation.

2. Copy Required Files

Copy the following four files in your current working directory.

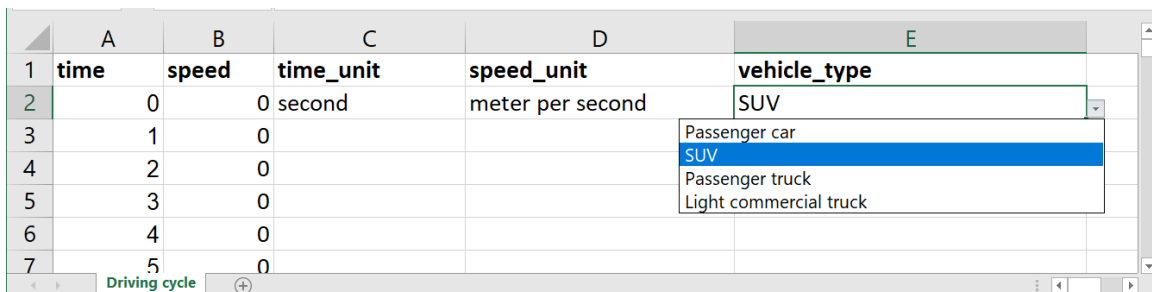
- i. Data.xlsx
- ii. db.pkl
- iii. pyemission.py
- iv. utility_functions.py

3. Data Preparation

It is recommended to use the provided excel file format for data preparation to avoid possible formatting errors. The provided excel file “Data.xlsx” contains a sample driving cycle data with the following four columns (see figure 3.1):

- 1st column: time
- 2nd column: speed of the vehicle
- 3rd column: choose the time unit used in the first column
- 4th column: choose the speed unit used in the second column
- 5th column: choose the vehicle_type from the drop-down menu

When data preparation is complete, close the excel file before running the python program.



	A	B	C	D	E
1	time	speed	time_unit	speed_unit	vehicle_type
2	0	0	second	meter per second	SUV
3	1	0			Passenger car
4	2	0			SUV
5	3	0			Passenger truck
6	4	0			Light commercial truck
7	5	0			

Figure 3.1: Input data file format

4. Import Module

```
from pyemission import GV
```

Here, GV represents Gasoline Vehicle.

5.1. Create a Gasoline Vehicle

```
g = GV()
```

The above code creates a new instance of Gasoline Vehicle, object 'g', using the 'GV' constructor with the following default parameters.

Table 5.1: Default parameters of a Gasoline Vehicle.

Parameters	Default values	Value types	Comment
excel_file_name	"Data.xlsx"	string	Input data Excel file name
sheet_name	"Driving cycle"	string	Name of the 'sheet' inside the Excel file, which contains the driving cycle data
mass	1500	integer or float	Gross mass of the vehicle in Kilograms
frontal_area	2.27	integer or float	Frontal area of the vehicle in Square Meter
mu_rr	0.0127	integer or float	Rolling resistance coefficient between tire and road surface
air_density	1.18	integer or float	Ambient air density in Kilogram per cubic meter
c_d	0.28	integer or float	Aerodynamic drag coefficient
well_to_tank_CO2_emission_factor	16.79	integer or float	Well-to-tank CO2 emission factor for fuel extraction, production, and transportation in Grams per megajoules. Typical value for gasoline is 16.79

Following is an example code where some of the default parameter values have been changed.

```
g = GV(excel_file_name = "My data.xlsx", sheet_name = "Sheet1", mass = 2850,  
       frontal_area = 2.5, mu_rr = 0.015, c_d = 0.3)
```

5.2. GV Class Methods

5.2.1. Create Plots

The following set of codes will create different plots describing the driving cycle characteristics.

```
g.plot_driving_cycle()  
g.plot_tractive_power()  
g.plot_speed_histogram(bins=30)
```

Output

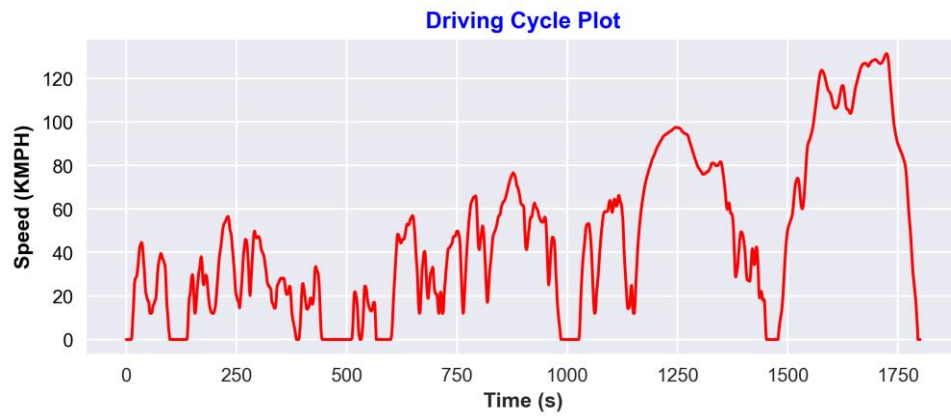


Figure 5.1: Output of `plot_driving_cycle()` method

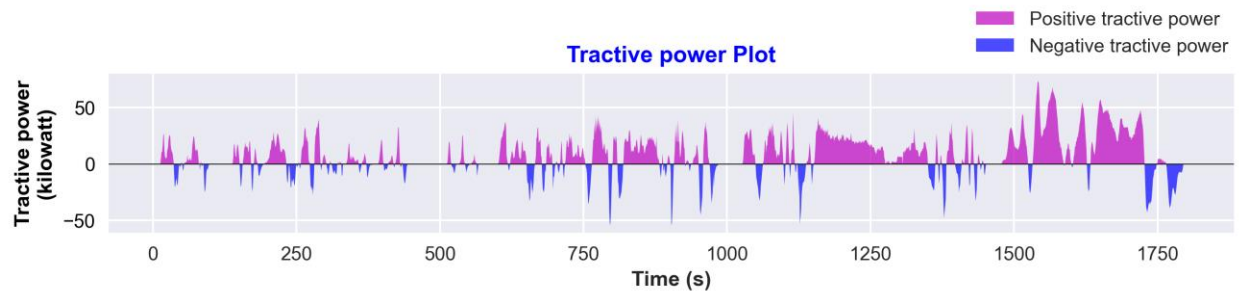


Figure 5.2: Output `plot_tractive_power()` method

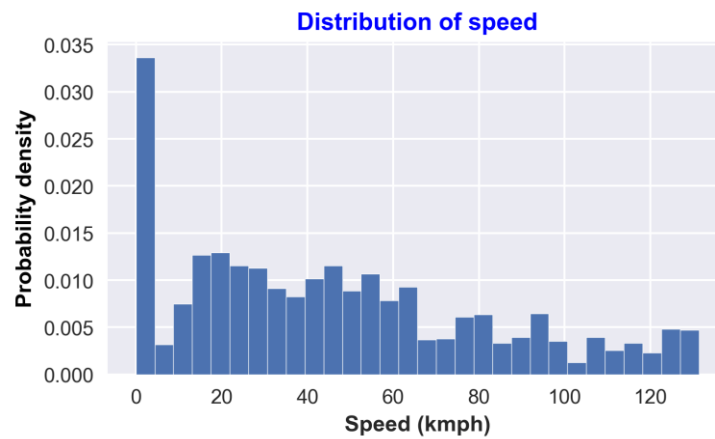


Figure 5.3: Output of `plot_speed_histogram(bins=30)` method

5.2.2. Driving Cycle Statistics

The following methods can be used to obtain the related driving cycle statistics.

Table 5.2: Methods related to driving cycle statistics.

Methods	Description
distance()	Distance of the driving cycle in Kilometer
average_speed()	Average speed in Kilometer per hour
speed_std()	Standard deviation of speed in Kilometer per hour
no_of_stops_per_km()	Number of stops per kilometer
acc_avg()	Average acceleration in meter per square second
dec_avg()	Average deceleration in meter per square second
acc_mode()	Percentage of time on acceleration mode
dec_mode()	Percentage of time on deceleration mode
idling_mode()	Percentage of time on idling mode

Example

```
print(f'Distance traveled           : {g.distance()} km')
print(f'Average speed              : {g.average_speed()} KMPH')
print(f'Standard deviation of speed : {g.speed_std()} KMPH')
print(f'Number of stops per kilometer : {g.no_of_stops_per_km()}')
print(f'Average acceleration         : {g.acc_avg()} m/s2')
print(f'Average deceleration         : {g.dec_avg()} m/s2')
print(f'Percentage of time on acceleration mode : {g.acc_mode()}%')
print(f'Percentage of time on deceleration mode : {g.dec_mode()}%')
print(f'Percentage of time on idling mode      : {g.idling_mode()}%')
```

Output

```
Distance traveled           : 23.194 km
Average speed              : 46.361 KMPH
Standard deviation of speed : 36.108 KMPH
Number of stops per kilometer : 0.345
Average acceleration         : 0.388 m/s2
Average deceleration         : -0.446 m/s2
Percentage of time on acceleration mode : 44.753%
Percentage of time on deceleration mode : 38.923%
Percentage of time on idling mode      : 13.27%
```

5.2.3. Fuel Consumption and Emission

The following methods can be used to get the amount of fuel burnt by the engine and the corresponding emissions.

Table 5.3: Methods related to fuel consumption and CO₂ emission.

Methods	Description
<code>pump_to_wheel_CO2()</code>	Tailpipe or pump to wheel CO2 emission in grams
<code>pump_to_wheel_CO()</code>	Tailpipe or pump to wheel CO emission in grams
<code>pump_to_wheel_NOx()</code>	Tailpipe or pump to wheel NOx emission in grams
<code>pump_to_wheel_HC()</code>	Tailpipe or pump to wheel HC emission in grams
<code>pump_to_wheel_CO2_per_km()</code>	Tailpipe or pump to wheel CO2 emission per kilometer (grams/km)
<code>pump_to_wheel_CO_per_km()</code>	Tailpipe or pump to wheel CO emission per kilometer (grams/km)
<code>pump_to_wheel_NOx_per_km()</code>	Tailpipe or pump to wheel NOx emission per kilometer (grams/km)
<code>pump_to_wheel_HC_per_km()</code>	Tailpipe or pump to wheel HC emission per kilometer (grams/km)
<code>well_to_pump_CO2()</code>	Well to Pump CO2 emission due to fuel extraction, production, and transportation in grams
<code>well_to_wheel_CO2()</code>	Well to Wheel CO2 emission in grams
<code>well_to_wheel_CO2_per_km()</code>	Well to Wheel CO2 emission per kilometer (grams/km)
<code>fuel_burnt()</code>	Amount of fuel burnt in milliliters
<code>mpg()</code>	Miles per Gallon (MPG)

Example

```
print(f'Tailpipe or Pump to Wheel CO2 emission : {g.pump_to_wheel_CO2()} grams')
print(f'Tailpipe or Pump to Wheel CO emission : {g.pump_to_wheel_CO()} grams')
print(f'Tailpipe or Pump to Wheel NOx emission : {g.pump_to_wheel_NOx()} grams')
print(f'Tailpipe or Pump to Wheel HC emission : {g.pump_to_wheel_HC()} grams')
print(f'Tailpipe CO2 emission per kilometer : {g.well_to_wheel_CO2_per_km()} grams/km')
print(f'Tailpipe CO emission per kilometer : {g.well_to_wheel_CO_per_km()} grams/km')
print(f'Tailpipe NOx emission per kilometer : {g.well_to_wheel_NOx_per_km()} grams/km')
print(f'Tailpipe HC emission per kilometer : {g.well_to_wheel_HC_per_km()} grams/km')
print(f'Well to Pump CO2 emission : {g.well_to_pump_CO2()} grams')
print(f'Well to Wheel CO2 emission : {g.well_to_wheel_CO2()} grams')
print(f'Well to Wheel CO2 emission per kilometer: {g.well_to_wheel_CO2_per_km()} grams/km')
print(f'Fuel burnt : {g.fuel_burnt()} ml')
print(f'Miles per Gallon (MPG) : {g.mpg()}')
```

Output

```
Tailpipe CO2 emission : 5958.536 grams
Tailpipe CO emission : 16.287 grams
Tailpipe NOx emission : 0.697 grams
```

Tailpipe HC emission	: 0.157 grams
Tailpipe CO2 emission per km	: 256.904 grams/km
Tailpipe CO emission per km	: 0.702 grams/km
Tailpipe NOx emission per km	: 0.03 grams/km
Tailpipe HC emission per km	: 0.007 grams/km
Well to Pump CO2 emission	: 1444.847 grams
Well to Wheel CO2 emission	: 7403.383 grams
Well to Wheel CO2 emission per km	: 319.199 grams/km
Fuel burnt	: 2516.199 ml
Miles per Gallon (MPG)	: 21.681