

Modelling vehicles' air pollution

An intern's progress, and what's next

Mattéo Boissière

Last updated: August 4, 2023

Special thanks

I would like to thank KTH for providing me with this opportunity - Romain Rumpler and Sacha Baclet especially for their supervision and their guidance, and Pierre Mariotti for the company during lunch breaks and after work hours.

I would also especially like to thank my roommate Paul Borne-Pons, who has done way more data visualization than me, for helping me out of bugs and dead ends that seemed like lost causes and overall being a legend.

Chapter 1

Introduction

One of the first things I noticed in this internship, after years of being used to engineering school and its pretty physics where most things are related to π or e somewhere down the line is that, well, we were now dealing with cars, these complicated manmade objects where who even knows how one particular model functions under the hood - and so, the values you have to deal with look more like this:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	NFR	Sector	Table	Type	Technolog	Fuel	Abatemer	Region	Pollutant	Value	Unit	Cl_lower	Cl_upper	Reference							
2	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gas Oil	NA	Hg	0.053 mg/GJ	0.005	0.53	Pulles et al., (2012)									
3	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	Cd	0.00025 mg/GJ	8.00E-05	0.00075	Nielsen et al., 2012									
4	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	NMVOc	1.6 g/J	0.5	7.6	Nielsen et al., 2010									
5	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	TSP	0.2 g/J	0.05	0.8	BUWAL, 2001									
6	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	PM10	0.2 g/J	0.05	0.8	BUWAL, 2001									
7	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	Benzo(k)f	1.11 µg/J	0.4	3.3	API, 1998									
8	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	Hg	0.1 mg/GJ	0.01	1	Nielsen et al., 2010									
9	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Large stat	Gas Oil	NA	Benzo(k)f	0.0987 mg/GJ	0.0493	0.0987	US EPA (1996), chapter 3.3 ("Less than" value based on method detection limits)									
10	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Large stat	Gas Oil	NA	Indeno[1,	0.187 mg/GJ	0.0937	0.187	US EPA (1996), chapter 3.3 ("Less than" value based on method detection limits)									
11	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Large stat	Gas Oil	NA	Zn	1.81 mg/GJ	0.18	18.1	US EPA (1998), chapter 1.3									
12	1.A.1.a	Public ele	Table_3-2	Tier 2 emi	Stationary	Natural gas	NA	NMVOc	89 g/J	45	135	Nielsen et al., 2010									
13	1.A.1.a	Public ele	Table_3-2	Tier 2 emi	Stationary	Natural gas	NA	PM2.5	2 g/J	1	3	BUWAL, 2001									
14	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Large stat	Gas Oil	NA	NMVOc	37.1 g/J	18.5	55.6	US EPA (1996), chapter 3.3									
15	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Benzo(b)f	37 µg/J	3.7	370	Wenborn et al., 1999									
16	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	HCB	6.7 µg/J	2.2	20.1	Grochowski & Konieczyński, 2008									
17	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Benzo(b)f	29 µg/J	2.9	290	Wenborn et al., 1999									
18	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Gas Turbir	Gaseous Fuels	NA	NOx	153 g/J	92	245	US EPA (2000), chapter 3.1									
19	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Hg	1.4 mg/GJ	0.97	2.25	US EPA (1998), chapter 1.1									
20	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Benzo(a)P	0.736 µg/J	0.245	2.21	US EPA (1998), chapter 1.1									
21	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Cu	9 mg/J	0.23	15.5	Expert judgement derived from Guidebook (2006)									
22	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Zn	90 mg/GJ	0.39	155	Expert judgement derived from Guidebook (2006)									
23	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Hard Coal	NA	Cu	9 mg/GJ	0.23	15.5	Expert judgement derived from Guidebook (2006)									
24	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	CO	13 g/J	0.1	26	European Commission (2006)									
25	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	Cr	9.1 mg/J	6.55	15.3	US EPA (1998), chapter 1.7									
26	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	PCDD/F	10 ng I-TEQ/	5	15	UNEP (2005); Coal fired power boilers									
27	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	Benzo(a)P	1.3 µg/J	0.26	6.5	US EPA (1998), chapter 1.7									
28	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	Indeno[1,	2.1 µg/J	0.4	10.5	US EPA (1998), chapter 1.7									
29	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	Zn	8.8 mg/GJ	0.5	16.8	Expert judgement derived from Guidebook (2006)									
30	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	SOx	1680 g/J	330	5000	See note in Guidebook chapter									
31	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	TSP	10.2 g/J	3.4	30.6	US EPA (1998), chapter 1.7									
32	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	PM10	6.9 g/J	2.3	20.7	US EPA (1998), chapter 1.7									
33	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	As	14.3 mg/GJ	10.3	24.1	US EPA (1998), chapter 1.7									
34	1.A.1.a	Public ele	Table_3-1	Tier 2 emi	Fluid Bed	Brown Coal	NA	PM2.5	2.8 g/J	0.9	8.4	US EPA (1998), chapter 1.7									
35	1.A.1.a	Public ele	Table_3-5	Tier 1 emi	NA	Gas oil	NA	BC	33.5 % of PM2.	28.9	38	Hildemann et al., 1981 & Bond et al., 2006									
36	1.A.1.a	Public ele	Table_3-5	Tier 1 emi	NA	Heavy Fuel Oil	NA	Ni	273 mg/GJ	140	550	US EPA (1998), chapter 1.3									
37	1.A.1.a	Public ele	Table_3-4	Tier 1 emi	NA	Gaseous fuels	NA	PCDD/F	0.5 ng I-TEQ/	0.25	0.75	UNEP (2005); Light fuel oil/natural gas fired powe									

...and so I understood why statistics was a required course in school.

In the report that follows, I will try to detail as best I can the thoughts I've had during this internship - the work I've done, and the work I've haven't done - in the hopes that this can be useful to anyone who'd be interested in picking up the project later on. It is very long, but I promise you, I made it as digestible as I could¹.

We will review the **literature** that was useful for comprehension of the project's theoretical aspects, but also a review of the **software** that helped shape the architecture, and why it is what it is. Along the way, we will see that many great **questions** lay unsolved, and could potentially serve as direction for interesting articles - lacking time, however, I could only lay the groundwork for such topics to be covered!

¹Translation: I put in as many jokes as I could.

Contents

1	Introduction	1
2	Methodology	3
2.1	On the role of ChatGPT/Copilot while learning code	3
2.2	On tutorials	5
2.3	On detailing the setting of our research	5
2.4	On keeping a record of everything	5
3	Literature review, and its implementation in the theoretical model	6
3.1	Current theoretical model	6
3.1.1	Vehicle specific power: definition	6
3.1.2	Vehicle specific power: simplified form	7
3.1.3	This is the part where I go crazy because I trusted the geometrical computations so far and now I don't know anything anymore	8
3.1.4	VSP, but make it discrete	10
3.1.5	Hey let's measure, like, what vehicles emit according to their VSP now maybe .	11
3.2	Literature review, beyond the model I used	12
3.2.1	Diesel v Gasoline	12
3.2.2	Does it always have to be emission factor models?	12
3.2.3	Is VSP always a thing?	12
4	Software review, and its implementation in the project architecture	13
4.1	Current project architecture	13
4.1.1	Using Git for version history	13
4.1.2	Dependencies	14
4.1.3	Necessity of an Internet connection	15
4.1.4	IDE recommendations and workflow	15
4.1.5	Matplotlib, Seaborn, Geoplot: what's up with modules that extend one another?	17
4.2	Software review, beyond the architecture I use	18
4.2.1	The part where I talk about SUMO	18
4.2.2	Data visualization and accessibility	19
4.2.3	Looking into urban geometry	19
5	Results	20
5.1	Program outputs	20
5.2	Some quick possible features for ease of use / software robustness	24
6	Compilation of ideas for future developments	25
6.1	Future directions, topics, questions that papers could answer	25
6.1.1	A tool for carbon footprints: where emission factors shine	25
6.1.2	An economical approach: optimal trajectory for vehicle fleet renovation	26
6.1.3	An individual approach: how could a given vehicle pollute the least they can .	28
7	Conclusion and personal thoughts	29
	Bibliography	30

Chapter 2

Methodology

2.1 On the role of ChatGPT/Copilot while learning code

ChatGPT is a tool I'm sure you know by now, and for good reason: it's very good at **putting into code, an idea you can only express with words**:

The screenshot shows a ChatGPT interface with two code snippets. The first snippet is a Python script for merging two GeoDataFrames based on a 'location' column. It uses the 'merge' function with 'how='left''. The second snippet is an updated version where NaN values in the merged DataFrame are replaced with zeros using the '.fillna(0)' method.

```
python
import geopandas as gpd

# Assuming you have a column named 'location' in both GeoDataFrames that contains the (x,y) coordinates. Also, replace "column_to_update", "column_to_update_accumulated", and "column_to_update_timestep" with the actual column names you want to update.

# Merge the two GeoDataFrames based on the 'location' column
merged_gdf = accumulated_gdf.merge(timestep_gdf, on='location', how='left')

# Update the values in the merged GeoDataFrame
merged_gdf['column_to_update'] = merged_gdf['column_to_update_accumulated']

# Drop unnecessary columns if needed
merged_gdf = merged_gdf.drop(['column_to_update_accumulated', 'column_to_update_timestep'])

# Assign the updated values back to the accumulated_gdf
accumulated_gdf = merged_gdf
```



```
python
import geopandas as gpd

# Assuming you have a column named 'location' in both GeoDataFrames that contains the (x,y) coordinates. Also, replace "column_to_update", "column_to_update_accumulated", and "column_to_update_timestep" with the actual column names you want to update.

# Merge the two GeoDataFrames based on the 'location' column
merged_gdf = accumulated_gdf.merge(timestep_gdf, on='location', how='left')

# Update the values in the merged GeoDataFrame
merged_gdf['column_to_update'] = merged_gdf['column_to_update_accumulated'].fillna(0)

# Drop unnecessary columns if needed
merged_gdf = merged_gdf.drop(['column_to_update_accumulated', 'column_to_update_timestep'])

# Assign the updated values back to the accumulated_gdf
accumulated_gdf = merged_gdf
```

Figure 2.1: ChatGPT can help you find appropriate code...

Figure 2.2: ...and help with debugging if it fails!

ChatGPT is pretty much everywhere in the news, in recent codebases, and it could even be... in this very report???

The screenshot shows a ChatGPT interface with a single code snippet for creating side-by-side images in LaTeX using the 'subfigure' environment.

```
How do I put two images side by side in LaTeX?

To put two images side by side in LaTeX, you can use the `subfigure` environment from the `subcaption` package or the `minipage` environment. Here's how you can achieve it using both methods:
```

Joking aside: I have found ChatGPT to be extremely valuable for **learning new Python modules**: since programming is best learned by trying your hand at it and debugging whatever horrible crap you've made, it has significantly sped up my learning process compared to when I coded without it.

But! There is a procedure to it. I would highly recommend treating ChatGPT kinda like a human you can bother on demand: **wise, but still very much capable of making mistakes**. To avoid falling into the trap of making boilerplate code nobody understands (*and least of all the AI*), I've found it best to stay clear in your demands, focused, and critical of what the algorithm will give you.

ChatGPT personally scratched many itches for me, along the lines of "*I don't know what this silly line does in matplotlib but I've been coding for a while so at this point I'm too afraid to ask*", and that is why I recommend using it.

< 2 / 2 > BO What are the fig and ax objects in the Python code:
`fig, ax = plt.subplots(figsize=(fsize, fsize))`

In the given Python code, `fig` and `ax` are objects related to creating and managing plots using the matplotlib library.
`fig` is short for "figure" and represents the entire figure or the window in which your plot is displayed. It acts as a container that holds one or more axes (plots). It provides methods and attributes for customizing the overall appearance of the plot, such as the size, title, labels, and more.
`ax` is short for "axes" and represents a specific plot or a set of plots within the figure. It is the area where you can draw your actual data. The `ax` object provides methods and attributes for customizing the individual plot, such as setting the data, adding labels, setting the axis limits, changing the line styles, and more.
The `plt.subplots()` function is used to create a new figure and a set of subplots. It returns a tuple consisting of a figure object (`fig`) and an array of axes objects (`ax`). The `figsize` parameter specifies the size of the figure in inches. The `fsize` variable is likely defined earlier in the code, determining the desired size of the plot.
With the `fig` and `ax` objects assigned, you can then use them to plot and customize your data within the plot. For example, you can use `ax.plot()` to draw lines, `ax.scatter()` to create scatter plots, and various other methods to add labels, titles, legends, and more.

BO What is the particularity of the subplots method? Is it different than the regular plot function? How?

But it is of course not meant to scratch "*do all my work for me please*" itches. And most important of all in this here field of research: GPT answers can absolutely fake data, and should NOT be treated like absolute truth. GPT can be good at clearing the terrain, perhaps, if you are completely lost in a field and would like to know how to get started - but any result that could end up being used somewhere should be double-checked with a reliable source.

And careful not to forget: Google and StackOverflow haven't stopped existing, and in case of doubt, documentation reigns supreme. (*even if sometimes documentation is very poorly written and it's really damn useful to have something that can vulgarize it, I'M LOOKING AT YOU FENICS*)

(Oh, and GitHub also has a tool called Copilot where a GPT is built in your IDE, which means it has access to all of the context in your code and how it interacts! I was not personally able to get it to work, because it checks your location if you want an academic license, and it didn't like that I was in Sweden with the email of a French school. But it is supposed to have a similar effect, if not to say an even more efficient one!)

2.2 On tutorials

Originally, I was inspired by QGIS' very detailed tutorials, and wanted to make a series of Jupyter notebooks that would interactively explain, even to someone with minimal prior knowledge, how everything in the project works. Because in case you haven't figured, on page 5 of this report where I haven't even started to talk about the project itself, *I like explaining things*.

The idea was to take the reader through the reasoning of the modelling, with tests and tries to further an approach from simple to complicated - because if the reasoning is captured and explained, it can be great for learning. For example, an initial idea I've had for a series of tutorials was:

- Only assuming a working Python environment, how do we configure a traffic simulation tool?
- Let's just try a random simulation (and here's how the random works)
- Now, let's try to change the data to be closer to real life.
- What is the emission model?
- What if we are curious about emission models and their differences? (*HBEFA, COPERT, CAL3QHCR...*)
- And what are EURO norms? Can we implement them manually?
- And how does the pollution change when the fleet changes?

And so on. Sadly, it was time-consuming enough to get a simple version of the code to work. Besides, I've come to realize that in the early phases of a project, it's difficult to even know where you're headed, and your direction might drastically change from one day to another. So, making tutorials as a way of advancing when you're not even done, is not as smart as it seems compared to doing it after everything is finished.

2.3 On detailing the setting of our research

I have tried, as much as possible, to keep underlying hypotheses in mind when exploring new concepts, hopefully that will show in later chapters. I have honestly done it perhaps too much, freezing myself from advancing at times, since I'm better at finding shortcomings of an idea than just executing it. But I still encourage the practice - if data is collected for example, I think it's important to know how it was collected exactly. And at least the reasoning should be rigorous!

I will admit that the literature review will be a bit rushed, for the following reason: attempting to do a literature review in the first months left me very confused, and it took me about the whole time of my internship to slowly understand the field and gain just enough perspective to... actually be able to do a halfway decent literature review, but by the time it's too late. Still, I loved this internship very much and I'm very glad I got to learn so much about research! But now I understand what people mean when they say that 5 months is unforgivingly short for a research internship...

2.4 On keeping a record of everything

On citing works with a DOI/ISBN : the citation will link to a complete reference, in the Bibliography section at the end of the document.

On citing works with no DOI/ISBN : the citation will link to a quick reference, at the bottom of the page. In my "Literature gathered" folder, I have included a "By order of presentation" folder, where you will find PDFs of the works cited.

Chapter 3

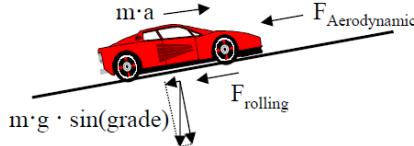
Literature review, and its implementation in the theoretical model

3.1 Current theoretical model

3.1.1 Vehicle specific power: definition

There are a huge amount of variables needed to describe a vehicle's movement and so it is super annoying to model them. But around 1999, MIT Ph.D. student Jose-Luis Jimenez said "*me basically, I'm just built different*"¹ and proposed a new variable: **Vehicle Specific Power**.

Vehicle Specific Power (VSP)



Defined as power over mass, the derivation of VSP is as follows:

$$VSP = \frac{\frac{d}{dt}(E_{kinetic} + E_{potential}) + F_{rolling} \cdot v + F_{aerodynamic} \cdot v}{m} \quad (3.1)$$

with v the speed of the vehicle in m/s , and:

- $E_{kinetic} = \frac{1}{2}m \cdot (1 + \epsilon_i) \cdot v^2$
 m is the **vehicle mass**, but we adjust the mass we're studying by ϵ_i , the **mass factor**. This is because the rotating components (*wheels, gears, shifts...*) have an effect on the vehicle's inertia. The suffix i indicates that ϵ_i depends on whether or not the vehicle is in first, second, third gear...
- $E_{potential} = mgh$ with g the acceleration of gravity, and h the altitude of the vehicle.
- $F_{rolling} = C_R \cdot mg$ with C_R the dimensionless coefficient of rolling resistance.
- $F_{aerodynamic} = \frac{1}{2}\rho_a \frac{C_D \cdot A}{m} (v - v_W)^2$ with ρ_a ambient air density, C_D dimensionless drag coefficient, A frontal area of the vehicle, and v_W the wind speed².

¹Note: he (probably) didn't actually say that

²Jimenez actually writes $v + v_W$ with v_W the headwind into the vehicle, this makes it less clear that v_W is often a negative value, and what we're looking at is really the speed of vehicle relative to the wind. Thus, I allowed myself to change the notation for clarity.

Calculations give the following formula :

$$VSP = v \cdot (a \cdot (1 + \epsilon_i) + g \cdot \sin(\arctan(grade)) + g \cdot C_R) + \frac{1}{2} \rho_a \frac{C_D \cdot A}{m} (v - v_W)^2 \cdot v \quad (3.2)$$

with a the acceleration of the vehicle in m/s^2 , and $grade$ the ratio of vertical rise over horizontal distance. Vehicle Specific Power is in W/kg, but it is commonly expressed in kW/Metric Ton.

One thing that can catch you off guard is that much of Jimenez's thesis, and much of the literature that cites it, write $grade$ as a percentage (%), and that's confusing for two reasons:

- This doesn't make it necessarily crystal clear that $grade$ can be negative, even though it definitely can be. Some literature will make an effort and write "grade ($\pm\%$)", though.
- Well, technically as a ratio of two lengths it could be superior to 1, couldn't it? It's not explained at all, but my guess is that, well, what kind of road has a slope so steep that rise is larger than horizontal distance?? Although physically possible, we just know we'd never run into it. Anyway, we will do some trigonometry later on, and things should make some more sense.

Jimenez showed his work with a presentation¹ during a vehicle emissions workshop (*which is the most digestible for a beginner*) and later during his Ph.D. thesis². The original presentation actually had an additional term for internal friction, but in the thesis Jimenez decided against it, arguing that "*engine friction is generally small compared to the range of values of Specific Power*", and that engine friction "*estimation from roadside measurements is much more difficult and uncertain*".

To Jimenez, the advantages of using VSP as a variable to study vehicle emissions are many:

- The end formula is *roadside-measurable*, which basically means simple to measure directly, whereas previous models needed hard to find data such as engine speed, absolute power, throttle position, etc.
- Citing a previous article that showed emissions are mainly related to engine power and engine speed[2], Jimenez argues that, **although VSP doesn't perfectly capture all effects that might play a role on emissions**, it captures a significant part.

3.1.2 Vehicle specific power: simplified form

This is the part where we want to get a simplified formula for typical vehicles. And this is what I mean when I say we should be careful with hypotheses, it took me several months to realize the VSP formulae that follow aren't a definition but rather a simplification of the definition in specific (yet admittedly common) cases!

This is also where approaches will differ, especially depending on the type of vehicle. Jimenez's choice of "typical values for all parameters" leads to the following formula:

$$VSP = v \cdot (1.1 \cdot a + 9.81 \cdot grade + 0.132) + 3.02 \cdot 10^4 \cdot (v + v_W)^2 \cdot v \quad (3.3)$$

A slight variant formula has been adopted into the methodology of the US Environment Protection Agency as of 2002 and later cited in many works as USEPA 2002³ or Frey et al. 2002⁴. Before we dive into that, Jimenez writes the following, concerning his simplifications:

- "*Note that these expressions are based on average values of the rolling resistance coefficient, the aerodynamic drag term coefficient ($C_D \cdot A/m$), and the value of air density at 20°C (68°F). Better estimates of the actual values of these parameters should be used whenever possible.*"

¹J. L. Jiménez, VSP origin presentation, 1999

²J. L. Jiménez, VSP detailed thesis, 1999 - definition of VSP page 54

³US EPA, Onboard Emissions Analysis, 2002

⁴According to Coelho et al. 2009[4], Frey et al. 2002 is Publication EPA420-R-02-027 which I have left in the folder, but also in that paper they cite Frey et al. 2002 and I can't really find Frey et al. 2002? I don't know what to believe anymore Mom please pick me up I'm scared

- "The ideal gas law ($Pv/T = P/(\rho T) = \text{constant}$) can be used to correct to other temperature and pressure conditions. The formula is $\rho = \rho_0 * (P/P_0) * (T_0/T) = 1.207 * (P/101.33) * (293.16/T)$, with ρ in kg/m^3 , P in kiloPascals (kPa), and T in Kelvin (K). This correction may be important, e.g. if the measurements are performed at -10°C (-14°F) and 1 atmosphere the air density will be 10% higher than at 20°C ."
- Rigorously $\cos(\arctan(\text{grade}))$ should be used instead of grade, but the error of this approximation is small (less than 1% relative error for grades below 11%).

3.1.3 This is the part where I go crazy because I trusted the geometrical computations so far and now I don't know anything anymore

It is interesting to see how the EPA have adapted his formula. For light-duty vehicles (*which is a fancy way of saying basically cars*), here's what the EPA writes :

$$VSP_{\text{light-duty}} = v \cdot (1.1 \cdot a + 9.81 \cdot \arctan(\sin(\text{grade})) + 0.132) + 0.000302 \cdot v^3 \quad (3.4)$$

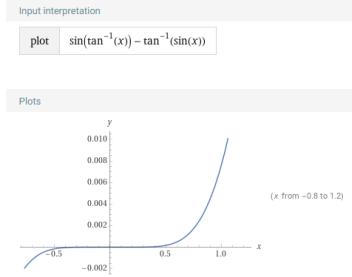
And here is the formula that many papers use, one of them being Fernandes 2017[7], where they cite the 2002 EPA report as source:

$$VSP_{\text{light-duty}} = v \cdot (1.1 \cdot a + 9.81 \cdot \sin(\arctan(\text{grade})) + 0.132) + 0.000302 \cdot v^3 \quad (3.5)$$

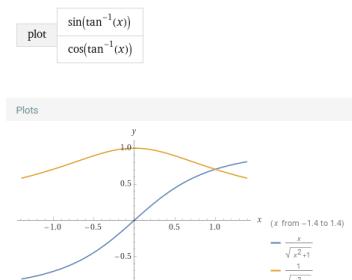
For buses, the mass factor and resistance coefficients chance, and the EPA gives the following formula, again in USEPA 2002:

$$VSP_{\text{bus}} = v \cdot (a + 9.81 \cdot \sin(\text{grade}) + 0.0094) + 0.42 \cdot v^3 \quad (3.6)$$

Now, we can immediately see that from here on out, the speed of the headwind that goes into the vehicle is neglected, which is understandable enough. But another observation we could make is that they write $\arctan(\sin(\text{grade}))$ in the EPA report, but it is never explained and never reused in later papers. Granted, \sin \arctan and \arctan \sin are similar functions, but they are not the same, so it seems to me like a mistake...



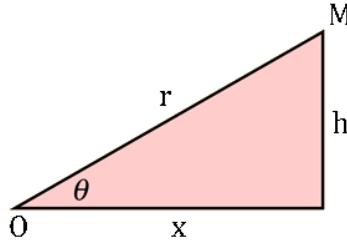
And even weirder that, although not a huge deal since Jimenez considers it to be negligible, either I'm bad at reading Jimenez's footnotes (*which is very much possible*) or everyone seems to have confused $\sin \arctan$ for $\cos \arctan$, which are NOT the same function. So what's up? Did Jimenez / the PA publish an erratum later on or has just everyone ever been quoting papers wrong this whole time?? I hope not.



I will attempt to do the calculations myself. As a reminder, we're supposed to get :

$$VSP_{\text{potential part}} = \frac{\frac{d}{dt} E_{\text{potential}}}{m} = g \frac{dh}{dt} = v \cdot g \cdot \sin(\arctan(\text{grade}))$$

where h is altitude from sea level. Suppose in the following diagram that, at instant t , the car is at $M(t)$, and so r , h and x are also functions of time. $\theta(t)$ can also be, too, since the slope can change and not be an exact triangle. (*Wait, if a road is curved, is it even valid to attempt a proof with this kind of "infinitesimal triangles" kind of logic? I honestly am not sure.*)



First off, it's common for slopes to be given in degrees, so let's just say we have ϕ the angle in degrees, then let's remember that the formula would be: $\theta = \frac{\pi}{180} \cdot \phi$.

(This paragraph exists because I'm not quite sure if Kaveh's CSVs are in degrees or radians. Not that it really matters because we've put everything at 0, but uh, just in case. The code could use some double-checking.)

Now then. It seems to me that, if grade is defined as the ratio of vertical rise over horizontal distance, then it's just $\frac{h}{x}$, which some of you may recognize as $\tan(\theta)$. So, okay, $\theta = \arctan(\text{grade})$. Then, potential energy would be the $mgh = mgr \cdot \sin(\theta)$.

Okay, I think I see what's going on.

- If we assume θ NOT to vary with time, then $g \frac{dh}{dt} = g \cdot \frac{dr}{dt} \cdot \sin(\theta) = v \cdot g \cdot \sin(\arctan(\text{grade}))$ (hurray!)
- If we assume $\theta(t)$ to vary with time (which it absolutely can!), then I feel like we have:

$$\begin{aligned} VSP_{\text{potential}} &= g \frac{dh}{dt} = g \left(\frac{dr}{dt} \sin(\theta) + r \cdot \frac{d(\sin(\theta))}{dt} \right) \\ &= v \cdot g \cdot \sin(\theta) + g \cdot r \cdot \frac{d\theta}{dt} \cdot \cos(\theta) \\ &= v \cdot g \cdot \sin(\arctan(\text{grade})) + g \cdot r \cdot \frac{d \arctan(\text{grade})}{dt} \cos(\arctan(\text{grade})) \\ VSP_{\text{potential}} &= VSP_{\text{potential, static } \theta} + g \cdot r(t) \cdot \frac{1}{1 + \text{grade}^2(t)} \cdot \frac{d\text{grade}}{dt}(t) \cdot \cos(\arctan(\text{grade}(t))) \end{aligned} \tag{3.7}$$

I... don't think I've ever seen this formula everywhere? And maybe this can be simplified even further but I'm not too sure how? I get that this can be negligible, if we assume flat surfaces everytime, then yes, everything is equal to zero, and it's whatever. And if we assume constant grade, then yes, the first formula is correct, and it's whatever. But I don't see this precision in papers, I see $9.81 \cdot \arctan(\sin(\text{grade}))$ confidently, and I feel like the discrepancies between papers here and there and Jimenez's hasty footnotes are kind of what I've just done, but more confusedly. Which is also equal to zero if we assume flat surfaces everytime anyway. But that's not what they say. So what's the big deal.

Anyway... I still have a lot of things I want to write so let's move on. So, we're at page 10 and all we have so far is a VSP. What do we do with it now ?

3.1.4 VSP, but make it discrete

By all logical means, if pollution is dependant on VSP (*and most likely other random phenomena but unless you want to die of exhaustion let's listen to Jimenez and assume we can neglect those*), then surely, for each type of vehicle there is, a certain VSP will give a certain amount of pollution.



Figure 3.1: Clio 2 Campus Bye Bye version 1.2 16V de 75 ch finition *Authentique*

And so we can figure out, say, the function :

$$f_{\text{Clio 2 Campus Bye Bye}} : \begin{cases} \mathbb{R} \rightarrow \mathbb{R} \\ VSP \mapsto \text{emitted pollutant of choice} \end{cases}$$

Great, so who wants to take an infinity of measures to account for every possible value of VSP? Yeah, me neither.

So, the EPA decided to opt for a binning approach, dividing good old continuous VSP into 14 discrete "VSP modes" - and again many later papers have followed suit. If your inner statistician is wondering how exactly they have determined these 14 modes to be the most relevant, there is more than enough to read in the famed EPA420-R-02-027 report. But the modes are as follows:

VSP Mode	Definition
1	$VSP < -2$
2	$-2 \leq VSP < 0$
3	$0 \leq VSP < 1$
4	$1 \leq VSP < 4$
5	$4 \leq VSP < 7$
6	$7 \leq VSP < 10$
7	$10 \leq VSP < 13$
8	$13 \leq VSP < 16$
9	$16 \leq VSP < 19$
10	$19 \leq VSP < 23$
11	$23 \leq VSP < 28$
12	$28 \leq VSP < 33$
13	$33 \leq VSP < 39$
14	$39 \leq VSP$

Now, once we have estimated the VSP mode of a given vehicle at a given time, how do we get an idea of how much it's polluting?

3.1.5 Hey let's measure, like, what vehicles emit according to their VSP now maybe

We're getting there! So, the year is 2004, and the US EPA releases MOVES⁵. And it's pretty great! As we've been leading up to so far, the idea is that, now that we've put driving patterns in different modes, **we can test many vehicles to operate in that pattern and obtain the average pollution rate in g/s for a given pollutant (CO , NO_x , HC)**. A more detailed explanation can be found in a 2010 overview of the model⁶.

This is an example of something called an **emission factor model**, very abundant due to its simplicity: now that we choose to consider that the pollution of a given vehicle in a given mode will follow a certain fixed rate (*in grams per second*), all we have to do is add those factors at each timestep of a simulation to obtain the pattern of pollution.⁰

This figure is from a 2006 paper by Coelho et al.[3], it is actually based on table A-4 of EPA's aforementioned 2002 report, Publication EPA420-R-02-027.

Table 2
Mean Values for NO_x , HC , CO_2 and CO emissions (g/s) for VSP modes for vehicles with odometer reading < 50 000 miles and engine displacement < 3.5 L

Definition	Pollutant			
	NO_x	HC	CO_2	CO
$VSP^a < -2$	0.0009	0.0004	1.6711	0.0078
$-2 \leq VSP < 0$	0.0006	0.0003	1.4580	0.0039
$0 \leq VSP < 1$	0.0003	0.0004	1.1354	0.0033
$1 \leq VSP < 4$	0.0012	0.0004	2.2333	0.0083
$4 \leq VSP < 7$	0.0017	0.0005	2.9199	0.0110
$7 \leq VSP < 10$	0.0024	0.0007	3.5253	0.0170
$10 \leq VSP < 13$	0.0031	0.0008	4.1075	0.0200
$13 \leq VSP < 16$	0.0042	0.0010	4.6350	0.0292
$16 \leq VSP < 19$	0.0051	0.0011	5.1607	0.0355
$19 \leq VSP < 23$	0.0059	0.0014	5.6325	0.0551
$23 \leq VSP < 28$	0.0076	0.0021	6.5348	0.1138
$28 \leq VSP < 33$	0.0121	0.0034	7.5852	0.2076
$33 \leq VSP < 39$	0.0155	0.0049	9.0242	0.4418
$39 \leq VSP$	0.0179	0.0109	10.0884	0.8823

Source: North Carolina State University, 2002.

^a As computed by Eq. (5).

And this is what my code has been based on! Recall that all that is needed to calculate VSP (*and thus VSP mode*) is speed, acceleration, and slope, all of which I have! That is, **assuming dimensions of a standard light-duty vehicle** (*which... it took me several months to realize I was actually reviewing mostly US literature instead of European literature. And I would wager that personal vehicles are larger in the US, actually... Maybe, in order to be thorough, we could recalculate usual resistance coefficients with a Swedish fleet.*

⁵Publication EPA420-P-04-019, 2004. Actually not really a research paper, but the draft(!) of a user guide for a software implementation of the MOVES model. Didn't stop it from being cited plenty!

⁶Koupal et al., MOVES overview, 2010

⁰and if it was in grams per unit of distance, then we'd have to multiply by distance... This is the methodology most carbon footprint calculation follows actually! But we'll come back to that.

3.2 Literature review, beyond the model I used

COPERT, MOVES, VERSIT+, PHEM... If you search blindly, you will only find too many vehicle emission models! For a brief explanation of how each model **works**, the JRC 2013 report[6] is the best place to start.

Now, for brief explanations of how each model **performs**, (*sometimes with a critical view!*), that's what the PDFs in the "Literature gathered" folder are for! I find **López-Martínez 2017**[8] and **Albuquerque et al. 2020**[1] to be particularly good starts to get a wide overview of tools already developed: their advantages, their shortcomings, etc, up to 2020 at least.

3.2.1 Diesel v Gasoline

Coelho et al. actually have a 2009 paper[4] about how MOVES 2004 mostly covers gasoline vehicles, and so they wonder just how much the VSP method developed for "LDGVs" (*Light-Duty Gasoline Vehicles*) is transferable to "LDDVs" (*Light-Duty Diesel Vehicles*).

Table 4
VSP normalized average emission rates for CO₂, CO, NO_x, and HC by VSP mode between EURO III diesel Skoda Octavia TDI 1.9L (D) and EURO IV gasoline VW Polo 1.4 L (G).

VSP mode	Pollutant							
	CO ₂ (g/s)		CO (mg/s)		NO _x (mg/s)		HC (mg/s)	
	G	D	G	D	G	D	G	D
1	0.63	0.21	0.50	0.03	0.23	1.29	0.03	0.14
2	1.05	0.61	0.27	0.07	0.68	2.62	0.03	0.11
3	1.02	0.73	0.15	0.14	0.60	3.38	0.03	0.11
4	2.07	1.50	0.58	0.25	1.75	6.05	0.07	0.17
5	2.79	2.34	1.14	0.29	2.52	9.36	0.10	0.20
6	3.47	3.29	1.76	0.69	3.34	12.53	0.15	0.23
7	4.31	4.20	4.05	0.58	4.04	15.48	0.22	0.24
8	5.19	4.94	6.13	0.64	2.63	17.82	0.35	0.23
9	5.81	5.57	9.06	0.61	3.51	21.32	0.42	0.24
10	6.43	6.26	18.97	1.01	2.89	32.53	0.52	0.28
11	7.37	7.40	44.98	1.15	1.27	55.75	0.62	0.37

(For some reason I can't seem to understand, the VSP mode stops at 11 instead of 14?)

The standard distribution of gasoline vs diesel in vehicle fleets justifies that in our current model, we can prioritize modelling gasoline, and then maybe later we'll look into diesel.

3.2.2 Does it always have to be emission factor models?

Not at all! They're handy for several reasons, but definitely not an end-all be-all model depending on the situation we're emodelling. A 2013 paper⁷ by Matjaz Knez makes a pretty good inventory of what types of modeling there are, and writes the following:

"Emission factor models function with a simple calculation method and do not require large amounts of input data. The estimation of the emissions is expressed by the use of an emissions factor related to one type of vehicle and a specific driving mode (i.e. urban, rural or motorway). Emission factors are derived from the mean values of repeated measurements over a particular driving cycle and are usually expressed in mass of pollutant per unit distance. Emission factor models are commonly used in the development of national and regional emission inventories. This approach is not accurate on microscale, regarding the emission factors are based on average driving characteristics."

3.2.3 Is VSP always a thing?

VSP is mainly the US EPA's thing, really, and was central to the development of their MOVES model. But there are many models that don't necessarily use VSP!

And this is where I'm proudest of my code, and the section that follows is all about that. The gist of it is, **procedures and objects are kept separate and cleanly organized, then called by a central script**. This means that, if you want to keep the heart of the code, but change the model used to study emissions, it should only take modifying of a few lines here and there!

⁷M. Knez, Vehicular emission models, 2013

Chapter 4

Software review, and its implementation in the project architecture

4.1 Current project architecture

4.1.1 Using Git for version history

Branches

Git is often said to be a powerful tool for working on code collaboratively: developers can work at the same time on separate features by creating different branches and merging them. It's weird to get the hang of if you're not used to it, but it can prove damn useful. But not if you're coding by yourself, right? Surely if you're coding by yourself it's just as fine to edit everything on one single file, maybe create some duplicate versions if you're unsure about yourself, but no need to go through this git commit / git branch nonsense, right? **WRONG**.

I did manual versioning when I started, and it quickly proved to be annoying. All it takes is to start working on a feature from a stable version, and then later have an idea for another feature that works fine and makes your job easier, but then feature n°1 is ruining everything and needs debugging, so you get back to a stable version for testing, but the stable version is also missing feature n°2 that works just fine, to make you go "*uuurgghhh okay I wish I could work on these things separately actually*". Which is exactly what branches are for! **In a good workflow, main is always a stable version, and messy experimentation (new features, debugging...) goes on in separate branches.**

Commits

Good Git etiquette requires that, everytime you've made a minor adjustment to your code and you'd like to save your changes, you write the following lines¹:

```
1 git add .\src\ 2
2 git commit -m "Descriptive message of what you just did"
3 git pull
4 git push 3
```

Doing this allows for one additional godsend methodology: **as long as you've committed your changes, it's fine, you can always go back**. Let's say for example you've messed everything up tremendously and need to go back quite a few commits. This is where the command `git reflog` comes in. Typing it in the terminal will give you the following output:

¹If you're very shy with command line inputs, GitHub Desktop is a great tool!

²Or any other file/directory you want to commit. Remember that tab autocompletion is a thing!

³Assuming you've configured local and remote repositories correctly, which is its own bag of worms but still worth it in the long run.

```

1 $ git reflog
2 4423af3 (HEAD -> main) HEAD@{0}: commit: Some commit message
3 a8e25c9 HEAD@{1}: commit (merge): Merge branch 'development'
4 edca205 HEAD@{2}: commit: [THE COMMIT THAT RUINED EVERYTHING]
5 f34e91a HEAD@{3}: commit: Implement new feature XYZ
6 1f23684 HEAD@{4}: commit: Fix bug ABC
7 98e9034 HEAD@{5}: commit: Refactor code
8 c548ef2 HEAD@{6}: commit: Initial commit

```

See how those commits have hash codes ? That list is from most recent to least recent, so we'd like to go back to commit `f34e91a` on line 5, right? All we need to do now is :

```

1 $ git checkout -b newbranchname f34e91a

```

This will create a new branch called `newbranchname` that will have the code in its exact state at time of commit `f34e91a`. And then you can merge it back to main, add whatever features you want more delicately, and it's all good!

4.1.2 Dependencies

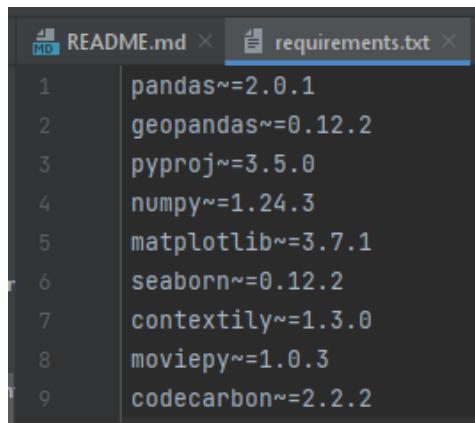
Many Python modules are needed to make the project work, and the complete list can be found in `requirements.txt`. As is written in the README, the command:

```

1 pip install - requirements.txt

```

should install everything that is needed. Let's go through them quickly¹:



```

pandas~=2.0.1
geopandas~=0.12.2
pyproj~=3.5.0
numpy~=1.24.3
matplotlib~=3.7.1
seaborn~=0.12.2
contextily~=1.3.0
moviepy~=1.0.3
codecarbon~=2.2.2

```

- **Pandas** is a well-known and powerful module for handling complex data sheets/CSVs within Python.
- **GeoPandas** builds on top of Pandas for handling specifically geographical data.
- **Pyproj** is a cartographic projections library, which has been used for a CRS reprojection operation within the code.
- **Numpy** is a well-known and powerful module for handling complex numerical operations within Python.
- **Matplotlib** is a well-known and powerful module whose "pyplot" library is extremely useful for plotting results within Python.
- **Seaborn** is a data visualization module that builds on top of matplotlib and is especially made to be suited for operating with Pandas dataframes.
- **Contextily** is a tile map library, which has been used for placing a portion of OpenStreetMaps in the background of a geographical Matplotlib plot.
- **CodeCarbon** is a Python module that computes carbon emissions due to the power given to the computer to run the program.

Figure 4.1: Modules and their versions, at time of writing. **Python 3.11** was used during internship.

CodeCarbon is mostly a small, experimental feature that I found fun to learn about. It can be enabled or disabled at will in the program. When enabled, it will print additional logs to the console and produce an Excel in the output folder with emissions in kgCO2eq.

CO2 emissions due to computation are mostly an issue in 1) **countries with a carbonated electricity mix**, which isn't the case here, and 2) **programs that require heavy computing**, such as PDE solving and AI training, which isn't the case here either. But it's still interesting!

¹I'm starting to think I don't actually know the meaning of that word

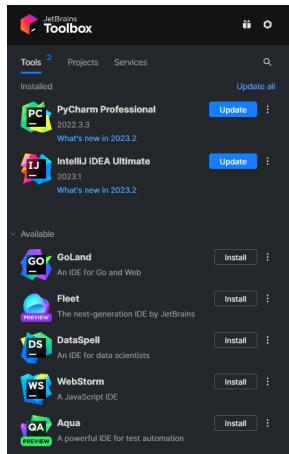
4.1.3 Necessity of an Internet connection

Once you have cloned the Git, most of the work is done locally, except for two actions that require an internet connection:

- Retrieving a background map for display (*called "basemap"*) from OpenStreetMaps
- If CodeCarbon is enabled, a location check to adjust the electricity mix to that of the current country.

Perhaps some work could be done to make this work offline, but that's not a priority.

4.1.4 IDE recommendations and workflow

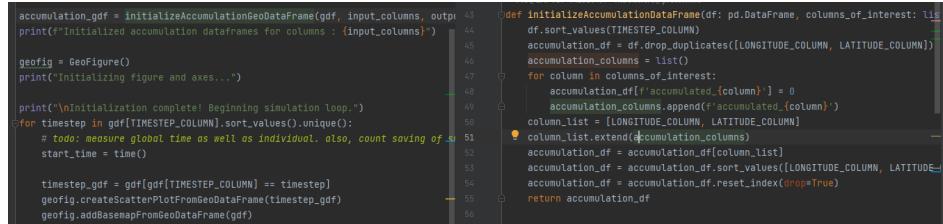


An IDE (Integrated Development Environment) is a software with features to help develop code more efficiently. It's much, much better than just using IDLE!

Visual Studio Code is a solid all-around choice, but the **JetBrains Toolbox** has great specialized tools for a specific language (*PyCharm for Python, IntelliJ for Java, etc etc...*).

There are free versions, but it is possible to get an **academic license** for the paying "professional" versions with extra features: I used my @student-cs.fr email address to confirm that I was a student for example, and it should be accessible to members of KTH as well.

Here's some among many examples of workflow can be improved with an IDE (*following examples are from PyCharm, but probably exist elsewhere.*):



```

accumulation_gdf = initializeAccumulationGeoDataFrame(gdf, input_columns, output_columns)
print("Initialized accumulation dataframes for columns : ", input_columns)

geofig = Geofigure()
print("Initializing figure and axes...")

print("\nInitialization complete! Beginning simulation loop...")
for timestep in gdf[TIMESTEP_COLUMN].sort_values().unique():
    # todo: measure global time as well as individual. also, count saving of data
    start_time = time()

    timestep_gdf = gdf[gdf[TIMESTEP_COLUMN] == timestep]
    geofig.createScatterPlotFromGeoDataFrame(timestep_gdf)
    geofig.addBaseMapFromGeoDataFrame(gdf)

```

```

def initializeAccumulationDataFrame(df: pd.DataFrame, columns_of_interest: list):
    df.sort_values(TIMESTEP_COLUMN)
    accumulation_df = df.drop_duplicates([LONGITUDE_COLUMN, LATITUDE_COLUMN])
    accumulation_columns = list()
    for column in columns_of_interest:
        accumulation_df[f'accumulated_{column}'] = 0
        accumulation_columns.append(f'accumulated_{column}')
    column_list = [LONGITUDE_COLUMN, LATITUDE_COLUMN]
    column_list.extend(accumulation_columns)
    accumulation_df = accumulation_df[column_list]
    accumulation_df = accumulation_df.sort_values([LONGITUDE_COLUMN, LATITUDE_COLUMN])
    accumulation_df = accumulation_df.reset_index(drop=True)
    return accumulation_df

```

Figure 4.2: Doing **ctrl+click** on a method being called will jump to where it is defined

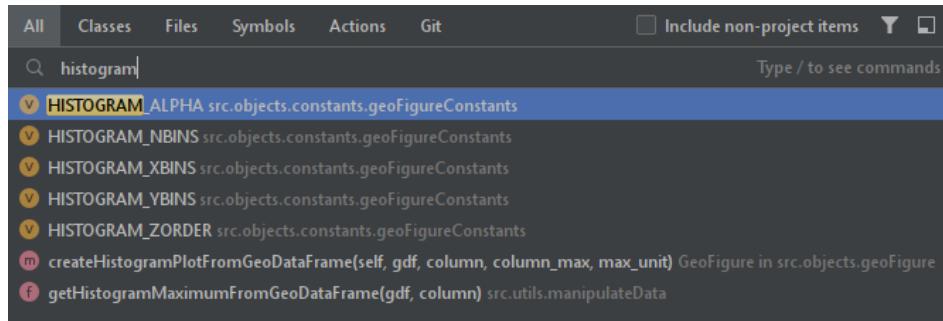


Figure 4.3: A search function that will search through the entire complex project architecture via **Navigate -> Search everywhere**

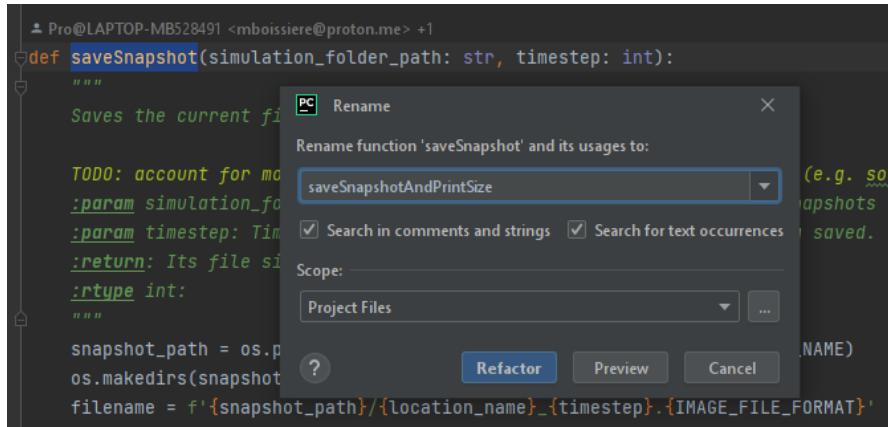


Figure 4.4: Rename an object or method via right-click \rightarrow Refactor \rightarrow Rename... will automatically rename all other usages in the project

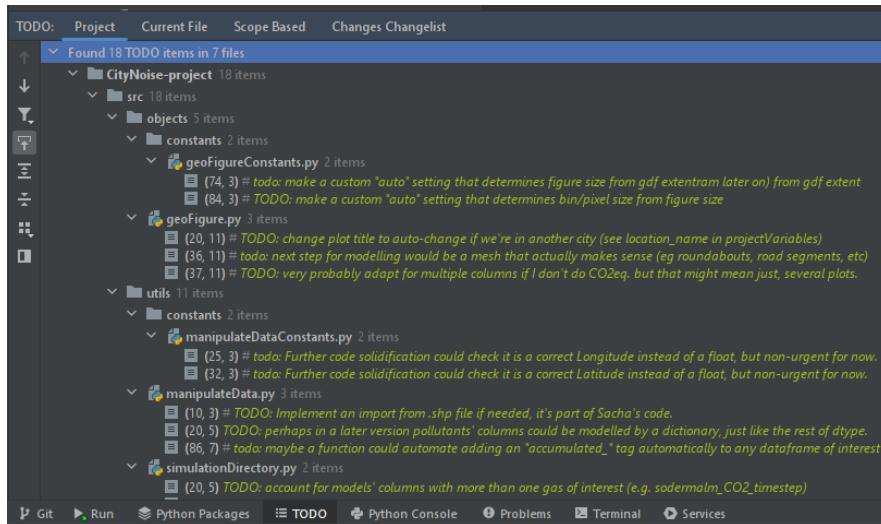


Figure 4.5: Everytime you write "TODO" in a comment, it will be stored in a TODO tab, so it's easy to see where there can be quick fixes or features to be implemented

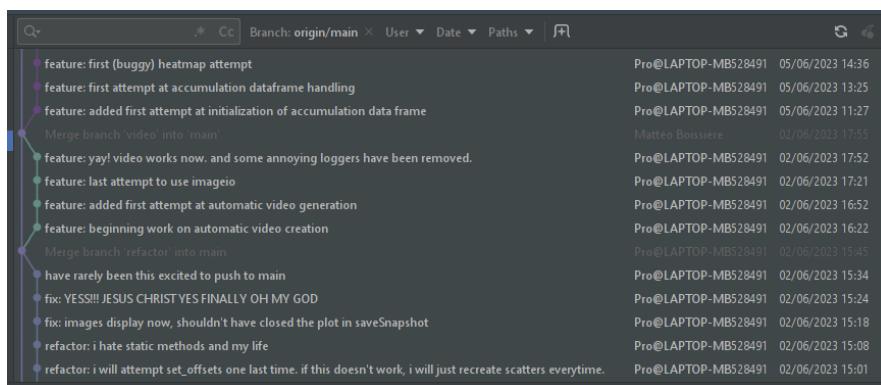


Figure 4.6: A visualization of the Git log via Git \rightarrow Show Git Log (there are also buttons for push/pull/commit functionalities if you're still shy with the command line!)

And many more I don't have the time to name if I want to move on.

4.1.5 Matplotlib, Seaborn, Geoplot: what's up with modules that extend one another?

So, when people create Python modules, they often build on top of existing modules. After all, technically, **this very project is sort of a Python module that was built on top of matplotlib, seaborn, etc!**⁴

Usually, **the more a module is specialized** (*building functions from previous modules...*), the easier it is to do things if you so happen to work in the field it covers. But! You also **lose some control**, working with auxilliary functions that reduce the number of variables for you and hide away the complicated stuff that goes on. So what does this all mean in practice? Let's look at an example.

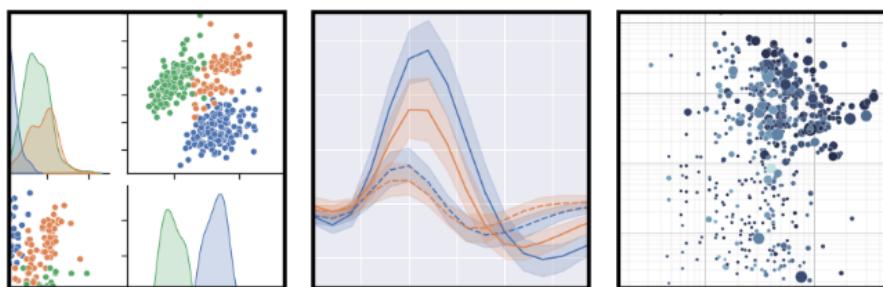
geoplot: geospatial data visualization



`geoplot` is a high-level Python geospatial plotting library. It's an extension to `cartopy` and `matplotlib` which makes mapping easy: like `seaborn` for geospatial. It comes with the following features:

Geoplot is a tool extending matplotlib, specifically made for plotting geographical data. Sounds perfect for our internship right? Not exactly. Although it has many interesting uses, and makes it for example a breeze to plot two types of graph on top of one another, **it does not implement 2D histograms**, which are (*at least currently*) central to the project.⁵ This is why we will prefer to use numpy in combination with matplotlib, as well as pandas in combination with seaborn.

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Geopandas also has its own plotting tools, which sometimes work super well, and sometimes break down confusingly. So now you know why! And if you're ever stuck: matplotlib is the bottom of the barrel and super annoying to use, but it's also the safest choice!

⁴Sort of, because it would actually need a tiny bit of extra refactoring for it to be entirely true, there would have to be `__init__.py` files here and there, but that's beside the point.

⁵It is also based on shapely which is kinda broken right now and so to get it to work you actually have to write "`pip install geopandas; pip install geoplot; pip uninstall shapely; pip install shapely --no-binary shapely`", with a ! at the beginning of each line if you're writing it at the top of a Jupyter notebook. Anyway.

4.2 Software review, beyond the architecture I use

4.2.1 The part where I talk about SUMO

So, SUMO is pretty dang great. And it was not at all the subject of my internship - my job was to analyse the CSVs produced by SUMO. But SUMO does a lot of cool things and so I still feel somewhat obligated to talk about them.

SUMO User Documentation

"Simulation of Urban MObility" (SUMO) is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks. It allows for intermodal simulation including pedestrians and comes with a large set of tools for scenario creation. It is mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center. SUMO is licensed under the EPL 2.0. The source code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the EPL 2.0 are satisfied: GPL2 or later."

Please use the mailing lists for sharing your results or for asking questions. Answers to some common questions may also be found in the FAQ.

If you use SUMO, please tell us about your Publications.

Note

When citing SUMO in general please use our current reference publication: "Microscopic Traffic Simulation using SUMO" ©, Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücke, Johannes Rummel, Peter Wagner, and Eva-Maria Wießner, IEEE Intelligent Transportation Systems Conference (ITSC), 2018.

What if I don't want to work with SUMO?

Well, your loss, because as we'll see it is an extremely developed tool with many features. But I agree its source code, and formats of file prints, aren't the most beginner-friendly. If you want to work without spending time learning about SUMO, someone did make a quick generator for random traffic : <https://github.com/ethanpng2021/creatorbox>

Installation and virtual environments

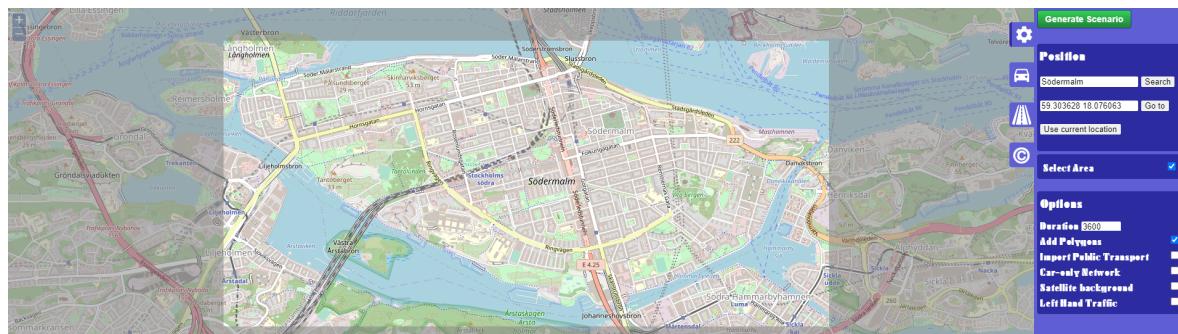
Sometimes computer people will tell you to install things in virtual environments, and you will say *"that doesn't even make sense, computer man, how can an environment even be virtual, like whatever I will just click on the default installation button"*. This was the moment where I understood what computer people were on about.

Basically, creating a virtual environment for installation allows you to completely control where your work folder is, and to install dependencies locally, without them interfering elsewhere. It's an entirely localized Python setup, which is useful when:

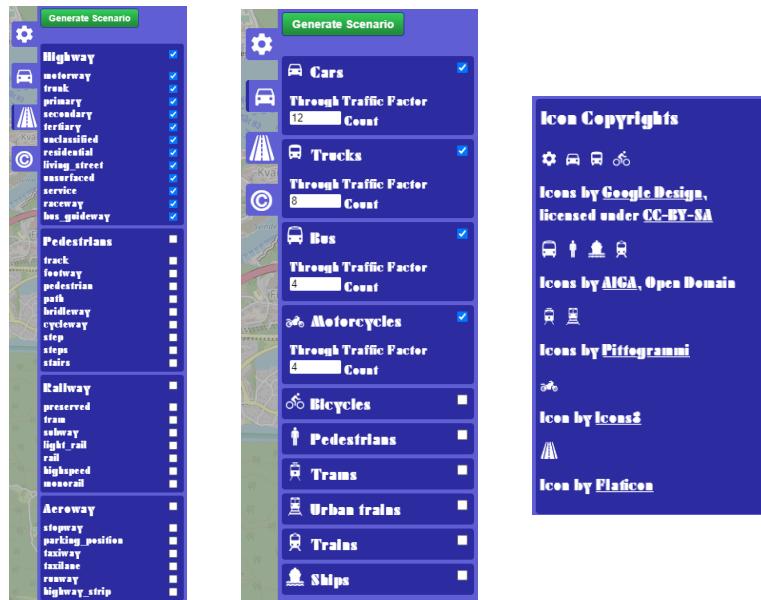
- The directory of installation by default is called `C:\Pro\Program Files` and you can't get SUMO to work because one line of code in SUMO reads the path and bugs out because of the SPACE in "Program Files".
- The directory of installation by default is weird permissions-wise and so you get "permission denied" errors even though it is your own computer.

The OSM Web Wizard

SUMO's OSM (*OpenStreetMaps*) Web Wizard (*accessed by launching WebWizard.py*) makes it super user-friendly to obtain a random generation of traffic data over a given location (*any given location!*), that can then be viewed in the SUMO app.



It also allows for simulation of cars, trucks, buses and motorcycles. Which is good, because as we've seen earlier, we can adapt VSP formulas and the like to model these different types of vehicles.



SUMO already does a lot of calculations actually

SUMO developers actually 'discuss environmental issues in their documentation! And SUMO not only simulates, but also directly calculates emissions of CO₂, CO, HC, NO_x, and even PM_x⁶! Here is the documentation explaining how it works:

- <https://sumo.dlr.de/docs/Simulation/Output/EmissionOutput.html>
- <https://sumo.dlr.de/docs/Models/Emissions.html>

Now if we've been paying attention, the first question we should be asking is "*so what emissions model do they use?*" - and the answer is a mix of PHEM and HBEFA. Some explanatory literature can be found here.

4.2.2 Data visualization and accessibility

Late in my internship, I have found an excellent site **that could help make the visualization of our data accessible to everyone**. It allows you to upload an image (*a screenshot from the simulation, for example*) and see what it looks like for different types of colorblind people!

So, it can always be nice to check if the colormap you picked is actually all that inclusive. The site is : <https://www.color-blindness.com/coblis-color-blindness-simulator/>

4.2.3 Looking into urban geometry

The SIRANE model[11][10] is interesting, it focuses heavily on how fluids behave around a given city architecture, (fluid mechanics, reflection on surface buildings and such) to show how pollutants are concentrated in urban areas. Our program does 2D histograms (*i.e. a simple grid mesh*), perhaps we could try and make a mesh that looks at roads, intersections, perhaps using .shp files and QGIS software⁷? There is one thing I haven't implemented at all from Sacha's script, and that is import of .shp files. I have pretty much only worked on import from CSVs.

⁶Does this render my entire work kinda useless? The answer is shush I don't want to think about it.

⁷I'm saying this as if it's easy. It's very much not. Still, there are some methods in GeoPandas (such as "SpatialIndex.nearest") that help find the closest road segment (or point) for each vehicle position.

Chapter 5

Results

5.1 Program outputs

Here are some screenshots of the end states for both the 5min and the 2h simulation of Södermalm.

For the 5 min simulation, a complete output folder can be found in the rest of the files I attach to this report.

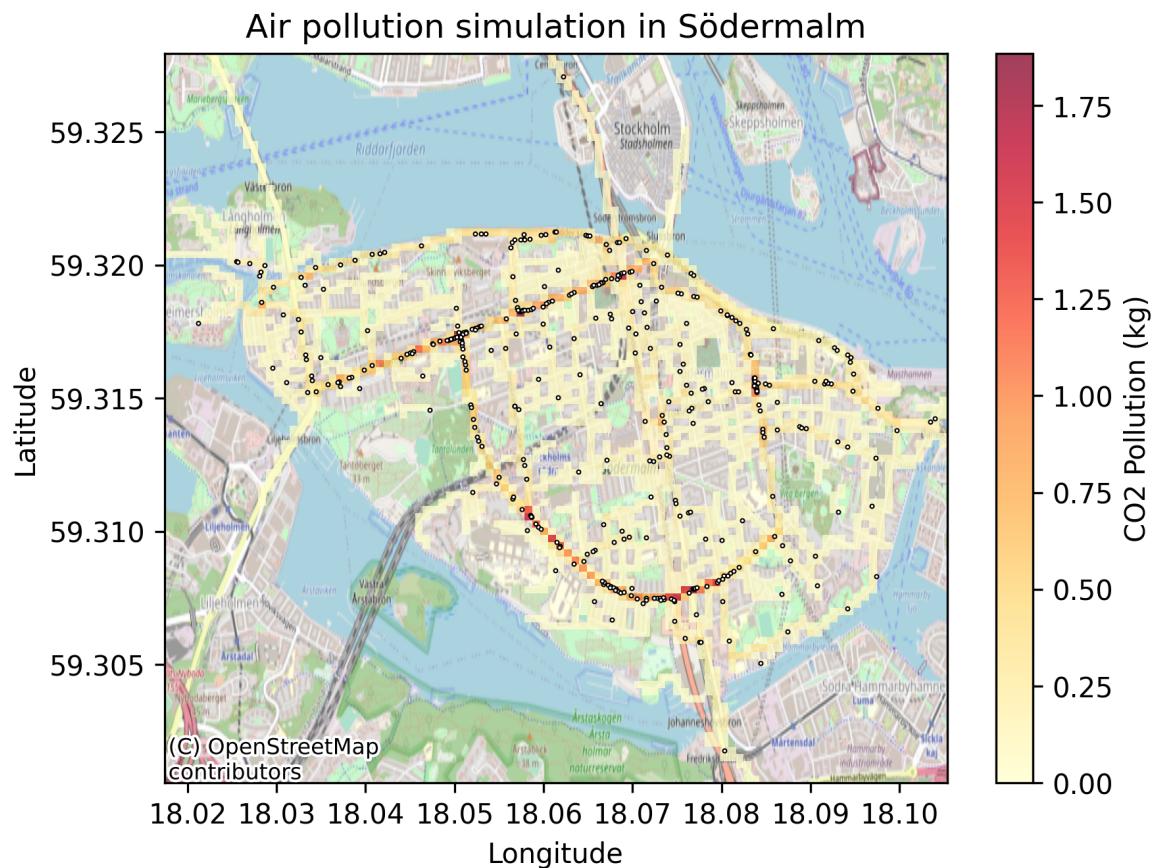


Figure 5.1: End state of the 5 min simulation on Södermalm

For the 2h simulation, I went up to approximately 1h30 of simulation time, but received a memory error after some time. I had plenty of storage space, so it wasn't about the size of the screenshots (*even though they are very heavy*).

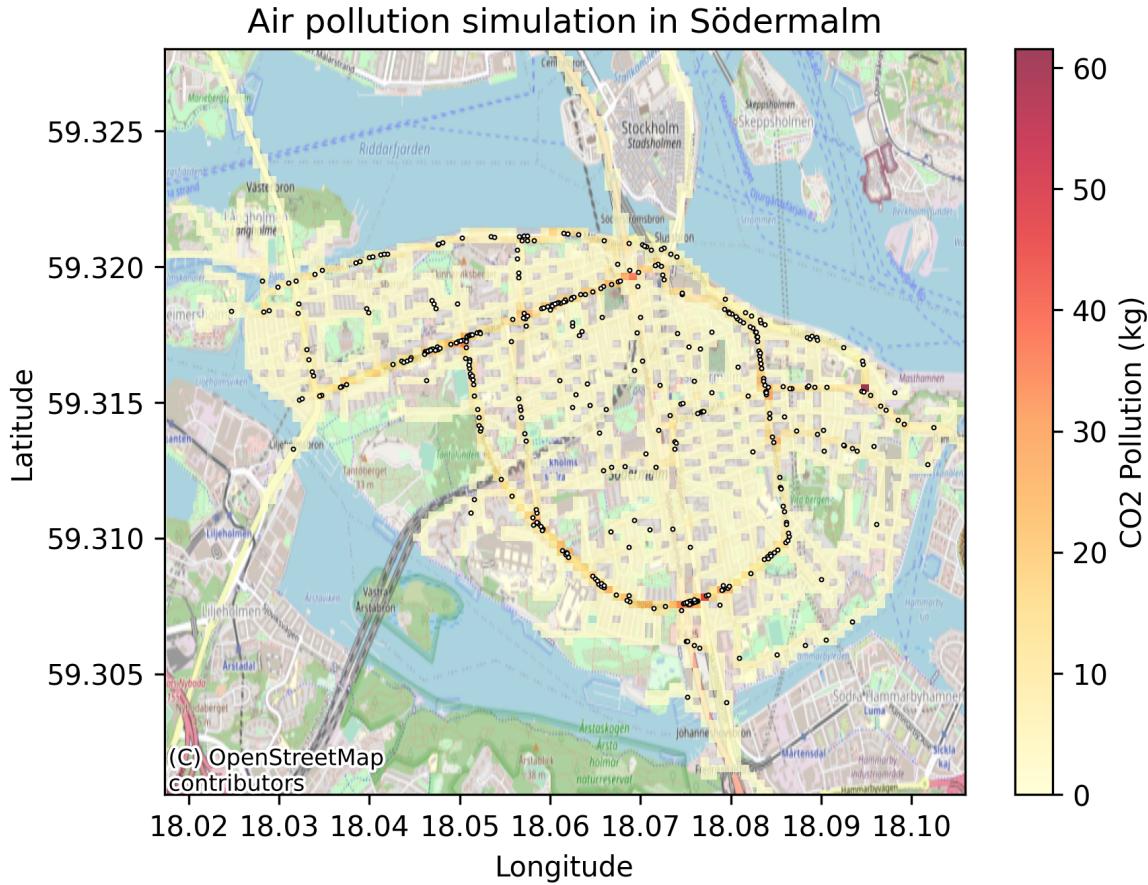


Figure 5.2: End state of the 2h simulation on Södermalm (*actually stuck at 1h35min51s*)

The error log is as follows:

```

1  == Snapshot 20748 ==
2  Generation time: 2.46 seconds
3  File size: 1.74 MB
4
5  == Snapshot 20749 ==
6  Generation time: 2.40 seconds
7  File size: 1.74 MB
8
9  == Snapshot 20750 ==
10 Generation time: 2.46 seconds
11 File size: 1.74 MB
12 Traceback (most recent call last):
13   File "C:\Users\Pro\Documents\GitHub\CityNoise-project\src\main.py", line 66,
14     in <module> accumulation_gdf =
15 addAccumulationDataFromGeoDataFrame(accumulation_gdf, timestep_gdf, input_columns)
16
17   File "C:\Users\Pro\Documents\GitHub\CityNoise-project\src\utils
18 \manipulateData.py",
19   line 119, in addAccumulationDataFromGeoDataFrame
20     merged_gdf = accumulation_gdf.merge(timestep_gdf, on='geometry', how='left')
21
22   File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages

```



```
23     \geopandas\geodataframe.py", line 1470, in merge
24         result = DataFrame.merge(self, *args, **kwargs)
25             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
26 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
27 \pandas\core\frame.py", line 9848, in merge
28     return merge(
29         ^^^^
30 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
31 \pandas\core\reshape\merge.py", line 158, in merge
32     return op.get_result(copy=copy)
33         ^^^^^^^^^^^^^^^^^^^^^^
34 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
35 \pandas\core\reshape\merge.py", line 807, in get_result
36     result = self._reindex_and_concat(
37         ^^^^^^^^^^^^^^^^^^
38 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
39 \pandas\core\reshape\merge.py", line 798, in _reindex_and_concat
40     result = concat([left, right], axis=1, copy=copy)
41         ^^^^^^^^^^^^^^^^^^
42 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
43 \pandas\core\reshape\concat.py", line 385, in concat
44     return op.get_result()
45         ^^^^^^^^^^
46 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
47 \pandas\core\reshape\concat.py", line 623, in get_result
48     return cons(new_data).__finalize__(self, method="concat")
49         ^^^^^^
50 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
51 \geopandas\geodataframe.py", line 32, in __geodataframe_constructor_with_fallback
52     df = GeoDataFrame(*args, **kwargs)
53         ^^^^^^^^^^^^^^
54 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
55 \geopandas\geodataframe.py", line 173, in __init__
56     self["geometry"] = _ensure_geometry(self["geometry"].values, crs)
57         ^^^^^^
58 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
59 \geopandas\geodataframe.py", line 1443, in __setitem__
60     super().__setitem__(key, value)
61 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
62 \pandas\core\frame.py", line 3960, in __setitem__
63     self._set_item(key, value)
64 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
65 \pandas\core\frame.py", line 4153, in _set_item
66     value = self._sanitize_column(value)
67         ^^^^^^
68 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
69 \pandas\core\frame.py", line 4881, in _sanitize_column
70     return sanitize_array(value, self.index, copy=True, allow_2d=True)
71         ^^^^^^
72 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
73 \pandas\core\construction.py", line 551, in sanitize_array
74     subarr = data.copy()
75         ^^^^
76 File "C:\Users\Pro\AppData\Local\Programs\Python\Python311\Lib\site-packages
77 \geopandas\array.py", line 963, in copy
78     return GeometryArray(self.data.copy(), crs=self._crs)
79         ^^^^^^
80 numpy.core._exceptions._ArrayMemoryError:
81 Unable to allocate 7.58 MiB for an array with shape (993826,) and data type object
82
83 Process finished with exit code 1
```

It is likely that our code isn't memory efficient - contrary to C++ for example, Python doesn't require you to strictly control your memory allocation. Perhaps we operate on large Pandas DataFrames, make copies at each timestep, and though we think that we're overwriting old memory, we might actually not be.

```

print("\nInitialization complete! Beginning simulation loop.")
for timestep in gdf[TIMESTEP_COLUMN].sort_values().unique():
    # todo: measure global time as well as individual. also, count saving of snapshot
    start_time = time()

    timestep_gdf = gdf[gdf[TIMESTEP_COLUMN] == timestep]
    geofig.createScatterPlotFromGeoDataFrame(timestep_gdf)
    geofig.addBasemapFromGeoDataFrame(gdf)
    geofig.adjustAxesFromGeoDataFrame(gdf)
    accumulation_gdf = addAccumulationDataFromGeoDataFrame(accumulation_gdf, timestep_gdf, input_columns)
    geofig.createHistogramPlotFromGeoDataFrame(accumulation_gdf, 'accumulated_CO2', converted_max, unit)
    end_time = time()

    elapsed_time = end_time - start_time
    print(f"\n- Snapshot {timestep} =-")
    print(f"Generation time: {elapsed_time:.2f} seconds")

    # todo: refactor savesnapshot honestly its weird for it to include file size
    file_size_bytes = saveSnapshot(simulation_folder_path, timestep)
    file_size, file_size_unit = convertFileSize(file_size_bytes)
    print(f"File size: {file_size:.2f} {file_size_unit}")

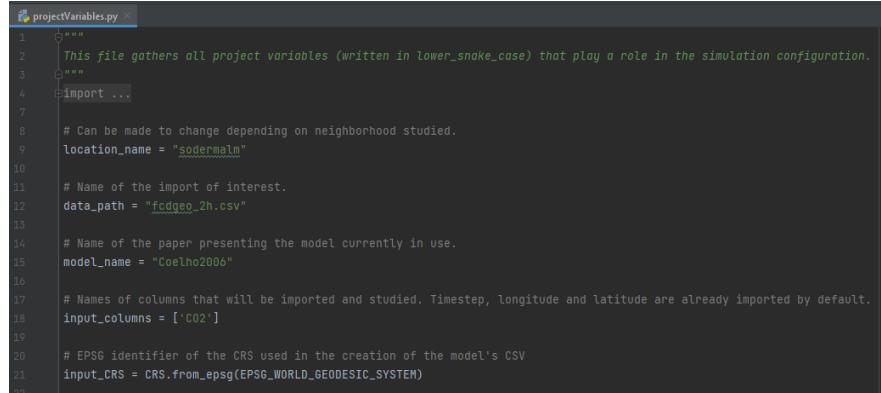
print(f"\n{plot_type} snapshots saved.")

```

Figure 5.3: Excerpt of the code within the loop over timesteps

If data size ends up becoming a huge problem, I've heard that **PySpark**, a framework for distributed computing, is adapted for pandas and used by people who work in big data.

It should be noted that I've only run the code with CO₂ so far, but it should be possible to also run the program with HC, NOx, and CO.



A screenshot of a code editor showing a Python file named `projectVariables.py`. The code defines several variables:

```

projectVariables.py
1 """
2     This file gathers all project variables (written in lower_snake_case) that play a role in the simulation configuration.
3 """
4 import ...
5
6 # Can be made to change depending on neighborhood studied.
7 location_name = "södermalm"
8
9 # Name of the import of interest.
10 data_path = "fcddgo_2h.csv"
11
12 # Name of the paper presenting the model currently in use.
13 model_name = "Coelho2006"
14
15 # Names of columns that will be imported and studied. Timestep, longitude and latitude are already imported by default.
16 input_columns = ['CO2']
17
18 # EPSG identifier of the CRS used in the creation of the model's CSV
19 input_CRS = CRS.from_epsg(EPSG_WORLD_GEODESIC_SYSTEM)
20
21

```

Figure 5.4: Excerpt of `projectVariables.py`, which can be used to setup important variables

`projectVariables.py` gathers all the important variables, however so far implementation isn't complete enough for setup to be automatic.

For example, `model_name` is just the name for a file path: if one wants to try a different model, one needs to put in the `models` directory an actual CSV representative of the model in the way that `Coelho2006.csv` is. Same with `location_name`: the string is for changing the name in screenshots and the like, but if one actually wants to study a region that isn't Södermalm, we'd need a corresponding CSV of traffic simulation in the `data` directory.

5.2 Some quick possible features for ease of use / software robustness

- A simple "total emissions over entire region" could be displayed in the figure on top of the spatial distribution, so for example if the colorbar shows a maximum of 3kg of CO₂ we don't think "oh, well that's pretty low actually" (*when there could be like 50 pixels each having 3kg of CO₂ actually*)
- Pollution stays on the plot forever, when some models at least implement some kind of dissipation¹. It could be as simple as, when we add `timestep_gdf` to `accumulation_gdf`, keep it in memory to subtract it back a given number of timesteps later. It could be more complex and based on actual chemistry. But so far there's nothing.
- The dynamic simulation over all timesteps is very pretty to watch, but sometimes we just want to look at the end state real quick! And not save gigabytes of screenshots on our computer! We should be able to select a "mode" between end-state and dynamic, so that we can tell the program to just give us the screenshot of the end state if that's all we want. Plus, a function already returns the end state, so there wouldn't be much work (`getEndValuesFromGeoDataFrame` in `manipulateData.py`, originally made to get the end maximum so that the colorbar is fixed)
- Being able to start and end the simulation at timesteps of choice could also be helpful. For example, it could help temporarily circumvent the fact that I ran out of memory space to compute 2h of simulation in one go (*I could then compute one hour, and then the next*)
- Another possibly useful mode switch that should be enabled through project variables: **pretty mode and fast mode** (which would affect the quality of the images: basemap zoom from contextily, figure DPI in `pyplot.savefig...`
- There are no actual tests in the program! I mean, I did test it manually by running it and saying "hm, looks good". But in projects that are bound to be scaled, it is good software engineering practice to add a folder dedicated to automatic tests! A quick introduction to different types of tests: <https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing>
- There is already a good amount of logging via console prints, and archiving in the output folder. **But there are specialized logging modules that could help be even more efficient for potential bug detection.**

¹ And again, this all depends on the angle we have: do we care about local effects of pollution, or about a carbon footprint output? That affects whether or not we care about dissipation! See next chapter.

Chapter 6

Compilation of ideas for future developments

6.1 Future directions, topics, questions that papers could answer

So that's all well and nice, but what we have now is the skeleton for an idea of a tool that could potentially be useful for something. But papers tend to answer a very specific question, or test a very specific hypothesis, and that is something we need to identify before getting lost in all the possibilities. Where's the direction? What are we even researching here? What is love? Baby don't hurt me? No more?

Although I haven't had the time to move in any of these directions myself, my review the field and past knowledge have allowed me to at least think about them. **So, I present to you some potential ideas that papers could follow, and some of the first impressions and thoughts I have about them.**

6.1.1 A tool for carbon footprints: where emission factors shine

So, one other place where emission factors are plentiful is **carbon footprint calculations**. And for good reason: the idea is to calculate your impact due to many different things (*food, transport, purchases, etc*) who emit CO₂ directly or indirectly in very different ways.

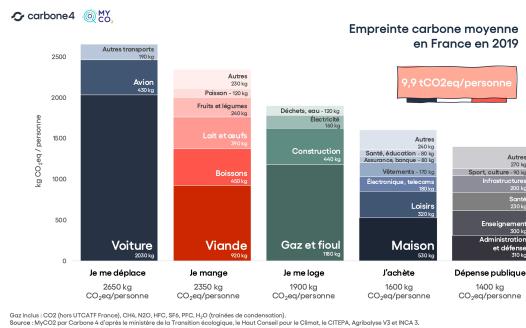


Figure 6.1: Pardon my French (Source : Carbone 4)

So, watering it all down to a simple emission factor makes it much easier to sum all those contributions together, and to simplify some variations due to complicated phenomena because *Actually Calculating Rigorously Everything would be an absolute nightmare.*

This means that many many factors are all averages, gathered over all types of vehicles (*gasoline, diesel, electric...*) and all types of usages (*short, medium, long distance travel...*). So, if you want to calculate the emissions of your 2002 Subaru Outback going from Kiruna to Abisko or whatever, it won't be all that precise.

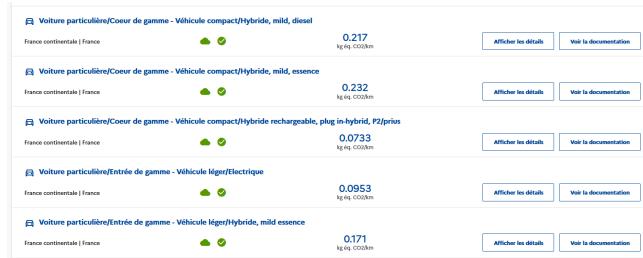


Figure 6.2: Pardon my French again (*Source : ADEME*)

So, why not work on a tool that would easily allow users to say "okay, I want to go from here to here with this vehicle specifically, what did I just emit?", with, granted some uncertainty still because there are a gazillion different ways one can model vehicle emissions that are differently suited depending on the situation etc etc, but probably way less uncertainty than the current usual method for carbon footprint calculations?

I like this approach, however I will admit it has two flaws:

- literature on pollutant emissions cares a lot about CO₂, but also about HC, CO, and NOx. It would be of course possible, but also a bit of a shame, to reduce these to a CO₂ equivalent, because these three pollutants have local effects that are different than CO₂ but also annoying (*I don't know enough chemistry to know what those effects are exactly, but it came up frequently in literature*).
- this could mean simulating long itineraries, which is something that SUMO kinda struggles with. I recall trying out a simulation over all of Stockholm (*instead of only Södermalm*), and it kinda broke down at some point, there were traffic jams that would take entire streets... In fact, even Södermalm-sized areas are tough to deal with, I've only found Gamla Stan-sized areas to be somewhat stable.

Just an additional resource before I go: I've cited ADEME before, but the EEA (*European Environmental Agency*) also have their own emissions factor database, which I then assume is at the European level (*whereas I think ADEME is French, but I haven't double-checked*).

6.1.2 An economical approach: optimal trajectory for vehicle fleet renovation

Questions are already more interesting if we wish to have direct action or to influence policy, because we're hitting broader, more systemic questions. Say we want to complete our net zero transition by 2050 (*like, for real, and not relying on carbon offsets whose legitimacy has been disproved¹ many times*). This could be studied through an economic lens (*what incentives exactly could work and change behavior on a large scale, how much would cost a renovation or recycling of the fleet*), while keeping a very strong grasp on technical aspects (*knowledge of the current fleet and how it could pollute on average, and that's where the simulation comes in*).

Perhaps some cars will be better scrapped because they're so old and polluting and have done their time, or perhaps some cars are at the beginning of their life cycle and will be better renovated. Perhaps some car owners actually use their cars very little and could be convinced to favor a rent economy and public transport for their travels, or perhaps the area where they live makes it so that they really really need their car and we'd better incentivize them to carpool or renovate and such.

This one is partly inspired by work I did in a previous internship at the French Ministry of Ecological Transition, where the goal was to use computational optimization (*CPLEX/Gurobi simplex solver, with constraints expressed with the Pyomo module*) to minimize a cost function with technical constraints, to do some prospective modelling that aims at a carbon-neutral 2050 while trying to minimize actual monetary cost.

¹<https://www.theguardian.com/environment/2023/jan/18/revealed-forest-carbon-offsets-biggest-provider-worthless-verra-aoe>

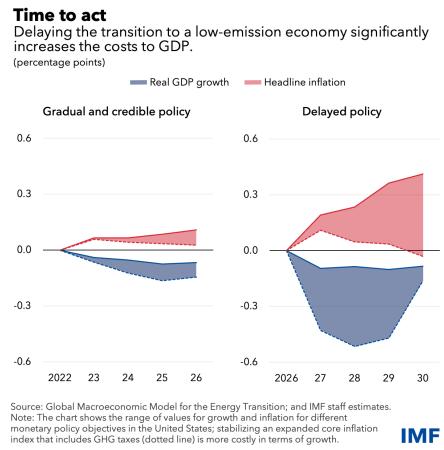


Figure 6.3: We know that we'd better invest money in the fight against climate change - but how exactly?

It was all about trajectory: for example in France we know that we need to renovate our buildings, whose energy efficiency is represented via grades G (*poorest isolation*) to A (*brand-new building*). **We know that we need to renovate them, but what's the best strategy?** Do we try to renovate D-buildings until they become C-buildings? Or B-buildings? Do we demolish F-buildings? What's the plan here?

We could ask ourselves these questions, but with the transport sector in Stockholm (*EURO norms, electrification, modal report...*). For the uninitiated, a solid introduction to what EURO norms² are: <https://www.rac.co.uk/drive/advice/emissions/euro-emissions-standards/>. On top of that, a table of emission factors by EURO norm can be found here³: https://en.wikipedia.org/wiki/European_emission_standards#Emission_standards_for_passenger_cars.

What would pollution look like with different distributions of EURO IV/V/VI in the vehicle fleet⁴? Can we incentivize people to go electric, perhaps showing benefits of relying on gas anymore? Would the loss of pollution be worth the cost of production of the car? Norway has a lot of electric cars, how are charging stations spread throughout the countries and cities? Are people happy about it?

One opinion I will add: one area in Stockholm will give vehicle owner fines if their EURO emission standard is too low (links here to explanation and geometry). Perhaps this can help fund part of the transition, and it is a noble goal, but it should not be considered the only possible source of income. My first argument for that, is that consumers being punished without being helped will only feel attacked by the system and view ecology as an annoyance, which is the opposite of what we want.

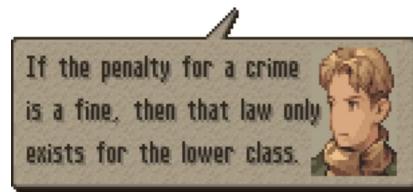


Figure 6.4: And for my second argument I will quote a relic of philosophy from the year 1997: Final Fantasy Tactics for the PlayStation 1.

²Although it'd be scientifically interesting to simulate traffic with varying distributions of EURO norms, it is sadly hard to know if such simulations would reflect the truth, since manufacturers have been cheating EURO norm tests for a while now: https://en.wikipedia.org/wiki/European_emission_standards#%22Cycle_beating%22_controversy

³Featured here are passenger cars, but also motorcycles, trucks and buses...

⁴If we want to test a certain geographical area, one paper has a fleet composition for France[9]! There is also a website that seems like it could give Swedish fleet, but uh. I don't speak Swedish.

Anyway, these are all very ambitious questions, I will admit! All of which I'm woefully unprepared to tackle head-on! So far in this paper we haven't talked at all about the economy or logistics of electrification, so it would be a bold and completely new direction to take. But it could be super interesting! Because of this, in the `resources` folder of the GitHub repository, I have left reports from The Shift Project, which are extremely interesting (*but long, and, I'm so sorry about this, French*) papers addressed to complete beginners about how we could go about this.

6.1.3 An individual approach: how could a given vehicle pollute the least they can

As we know, "every tenth of a degree matters" when it comes to global warming. With noise pollution already being a harm for biodiversity, air pollution is also a matter of public health. So, we have chosen to look into air pollution models, to gain a better of understanding of vehicles' behavior through simulations.

The angle is the following: a significant research is done on many ways to reduce emissions (*public transit infrastructure and modal report, etc*) but there are still some instances where people will need cars. We could look at an individual's perspective and see what they can do to pollute the least possible.

For example, it's been shown that with regular traffic (*which can be coordinated with "green waves"*, see *De Coensel 2012 [5]*), vehicles don't have to accelerate/decelerate all the time and this helps reduce pollution. What if we imagined some kind of Citymapper-like routing app, that would have access to a realistic traffic simulation based on current traffic with HERE's API (https://developer.here.com/documentation/traffic/dev_guide/topics/what-is.html), and from there on gives you the least polluting itinerary to your destination. Would it be the same as the one with least traffic, or would it be slightly different? Perhaps driving near a park would help early capture of pollutants? What if the function we're seeking to minimize is multi-varied, and so we could see the influence of one factor over another?

- Advantages: this allows some scientifically interesting questions! It kind of gets at the part of your brain that wants to optimize things. And later, we could see how that pollution loss could be aggregated if we assume everyone to have this "heroic" behavior.
- Disadvantages: this is far from my favorite, because all of this could be super mega negligible in the long run, and feed into individual consumer guilt-tripping, when many researchers and activists agree that this is NOT the way to go. A dose of pragmatism is needed in this issue - for example, it is useless to be a superhero when it comes to the way you use your car, if you then board a plane to go somewhere that is accessible by train. Even considering a more local scale (*i.e. neighborhoods and local chemical interactions of pollutants themselves*) where this carbon footprint argument doesn't fly (*pun intended*), I would rather consider structural, the-best-car-is-no-car approaches. And then, well if you need to use the car one time because you're moving or going to a very isolated suburb, even the most fervent activists (*I happen to know*) will tell you there's no shame in that, and there's no use stressing over teensy emissions.

Chapter 7

Conclusion and personal thoughts

I don't have any personal thoughts. My head is empty.

Thank you for reading! I am a bit disappointed in myself in that it took me a long time to understand things, and I sometimes feel like I could've gone farther than I did. Still, I'm proud of having written this report! I feel like I've written all of my thoughts down, and so I will have transmitted what little knowledge I've accumulated the best I can!

I sincerely hope this report will be useful! And even though I won't be working here anymore, I'd love to have feedback on it, or updates/answers to whatever silly questions I've had¹.

¹Though criticism can always be good for whatever later papers I could always write in my life and I welcome it, I'm of course not entitled to your time anymore! Like, the lab and the job is of course the focus etc etc. But also just to say that I don't not care completely. I don't know if that's clear. I think I love footnotes a little too much.

Bibliography

- [1] Francisco D.B. Albuquerque, Munjed A. Maraqa, Rezaul Chowdhury, Timur Mauga, and Mohammed Alzard. Greenhouse gas emissions associated with road transport projects: current status, benchmarking, and assessment tools. *Transportation Research Procedia*, 48:2018–2030, 2020.
- [2] Matthew Barth, Theodore Younglove, Tom Wenzel, George Scora, Feng An, Marc Ross, and Joseph Norbeck. Analysis of Modal Emissions From Diverse In-Use Vehicle Fleet. *Transportation Research Record: Journal of the Transportation Research Board*, 1587(1):73–84, January 1997.
- [3] Margarida C. Coelho, Tiago L. Farias, and Nagui M. Rousphail. Effect of roundabout operations on pollutant emissions. *Transportation Research Part D: Transport and Environment*, 11(5):333–343, September 2006.
- [4] Margarida C. Coelho, H. Christopher Frey, Nagui M. Rousphail, Haibo Zhai, and Luc Pelkmans. Assessing methods for comparing emissions from gasoline and diesel light-duty vehicles based on microscale measurements. *Transportation Research Part D: Transport and Environment*, 14(2):91–99, March 2009.
- [5] B. De Coensel, A. Can, B. Degraeuwe, I. De Vlieger, and D. Botteldooren. Effects of traffic signal coordination on noise and air pollutant emissions. *Environmental Modelling & Software*, 35:74–83, July 2012.
- [6] European Commission. Joint Research Centre. Institute for Energy and Transport. *Overview of emission and traffic models and evaluation of vehicle simulation tools*. Publications Office, LU, 2013.
- [7] P. Fernandes, M.C. Coelho, and N.M. Rousphail. Assessing the impact of closely-spaced intersections on traffic operations and pollutant emissions on a corridor level. *Transportation Research Part D: Transport and Environment*, 54:304–320, July 2017.
- [8] José M. López-Martínez, Felipe Jiménez, F. Javier Páez-Ayuso, M. Nuria Flores-Holgado, Angélica N. Arenas, Blanca Arenas-Ramirez, and Francisco Aparicio-Izquierdo. Modelling the fuel consumption and pollutant emissions of the urban bus fleet of the city of Madrid. *Transportation Research Part D: Transport and Environment*, 52:112–127, May 2017.
- [9] Simon Martinet, Yao Liu, Liliane Jean-Soro, Mathieu Goriaux, and Michel André. In-situ estimation of non-regulated pollutant emission factors in an urban area of Nantes, France, with fleet composition characterization. *Transportation Research Part D: Transport and Environment*, 76:193–210, November 2019.
- [10] L. Soulhac, P. Salizzoni, P. Mejean, D. Didier, and I. Rios. The model SIRANE for atmospheric urban pollutant dispersion; PART II, validation of the model on a real case study. *Atmospheric Environment*, 49:320–337, March 2012.
- [11] Lionel Soulhac, Pietro Salizzoni, F.-X. Cierco, and Richard Perkins. The model SIRANE for atmospheric urban pollutant dispersion; part I, presentation of the model. *Atmospheric Environment*, 45(39):7379–7395, December 2011.