manuscript No.
(will be inserted by the editor)

Detailed exhaust modeling using Cantera reactor networks.

Anthony S. Walker

Received: date / Accepted: date

Abstract As society continues to relay on the burning of fuel to extract energy, it is important to understand the impacts our actions have on our health and the environment. `PyPlume` is a software developed for the purpose of modeling exhausts to help understand these impacts. The software focuses on building complicated reactor networks to more practically simulate the exhaust-atmosphere interaction and understand the composition and effects of exhaust products on the environment. The software is the result. A tool that can be used for further study based on a researcher's needs.

1 Introduction & Motivation

As society continues to relay on the burning of matter to extract energy, it is important to understand the impact that our actions have on our health and the environment. Is all the fuel we burn detrimental to our health in the long term? To what degree can we safely continue our way of life? Are we destroying other forms of life on the planet as well with our behavior? The list of important questions goes on and answering them is no minor feat.

Chemistry plays an important role in many aspects of today's society such as transportation, healthcare, energy production, and environmental safety and protection. In regards to transportation, the role only gets more important as Airbus predicts that air traffic will double in the next 15 years[2]. Meanwhile, estimates place aircraft contrails as the most substantial factor in radiative forcing [3,10] as cited in [4]. The idea that pollutants affect the environment is not novel; there has been a noticeable impact on the climate from aerosols for quite some time [8]. Environmental impacts aside, studies suggest negative impacts on respiratory and cardiovascular health as a result of aerosol particles [1,9].

This aforementioned work is only a small fraction of studies on this subject matter but these questions remain important as our reliance increases. Answering these questions more accurately requires scientific study to develop and model the lifespan and behavior of products from burning matter. Small scale phenomena can be analyzed experimentally to determine constituents and potential impacts. However, how do you experimentally model wildfire that burns hundreds of thousands of acres with an immense variety of different

A. Walker
School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA
E-mail: walkanth@oregonstate.edu

fuels both living and dead? How do you model the burning of a jet fuel which has thousands of different chemical species to track? How do you follow the products from both scenarios throughout the atmosphere across the world? The answer would seem to be computational modeling, but it has it's limitations as well. First and foremost, modeling such complicated phenomena requires a lot of computing resources. The models can be simplified to use less resources but at the cost of accuracy [5] as cited in [7].

Limitations aside, improvements in the ability to accurately model reacting flows are necessary to continue improving our understanding of impacts on the environment and our health. This is the core idea of `PyPlume`. It is a modeling software with the goal of being an extensible tool for more in depth research studies on the evolution of chemical species from burning various fuels [11].

2 Methods

`PyPlume` was built with Cantera[6] at it's core. The goal was to create a versatile `Python` tool that can generate complicated exhaust models with a reactor network approach. While complicated models are currently constrained by expense, I am working towards implemented methods to drastically accelerate the solution process and reduce this expense. This package was created with the intention of using these methods on a particular application such as a more rigorous study of aircraft exhaust. `PyPlume` focuses on the exhaust portion of the system but also includes a reservoir-combustor network to generate the input conditions and, an atmospheric reservoir for mass entrainment into the exhaust network. The general idea is shown in Figure 1.

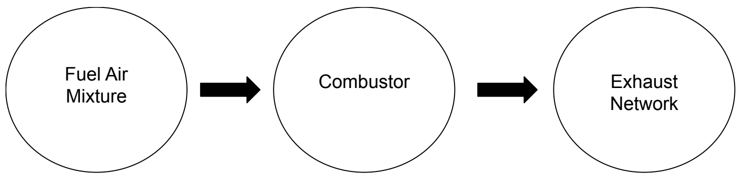


Fig. 1: General concept for modeling exhaust networks in pyplume.

So, how do we produce a more in-depth study with a set of well-stirred ideal gas reactors? The main idea that I had was the ability to relax the well mixed nature of well-stirred reactors. This can be achieved with multiple connected reactors, entrainment, and mass conservation. `PyPlume` uses an adjacency matrix approach to creating and connecting the exhaust reactor network which makes this possible, e.g., mass entrainment into an outer reactor will provide a different concentration than inner reactors. The second idea was to use larger mechanisms that include more of elementary reactions. This is also possible with the package because the mechanism can be independently specified for every reactor if desired.

These ideas led to the development of three models within `PyPlume`: `simple`, `linear`, and `grid`. The simple model was used primarily in developing and testing and is shown in Figure 2.

The `grid` and `linear` models were developed as different takes on achieving a less well mixed system but they have yet to be formally tested and verified to this end. These models aside, `PyPlume` can be used

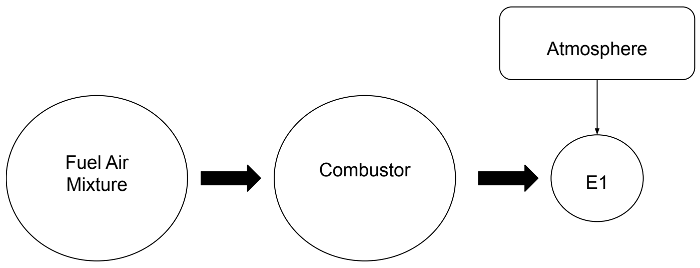


Fig. 2: Simple model for testing and developing.

to develop other models by providing an adjacency matrix to the software. I suspect the ability to generate complex reactor networks with matrices will be the most useful for my application. For example, to produce the `linear`, and `grid` model shown in Figures 3 and 4, the adjacency matrices would respectively be

$$\begin{bmatrix} 0&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&1&1&0\\ 0&0&0&0&0&0&1&1&1&1&0\\ 0&0&0&0&0&0&1&1&1&1&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&0&1&1&0&0&1&0 \end{bmatrix} \quad and \quad \begin{bmatrix} 0&1&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0\\ 1&1&0&1&1&0&1&1&0&1&0 \end{bmatrix}.$$

These adjacency matrices could be further used to specify reverse flow into reactors, i.e., moving upstream. While it has not been tested in this scenario, it is possible. These connections are `MassFlowController` objects from `Cantera` and continuity is maintained in a simple manner, i.e., mass flow out of a reactor is divided by the number of sinks.

$$\dot{m}_i = \frac{m_i}{n_{s,i}\tau(t)}, \tag{1}$$

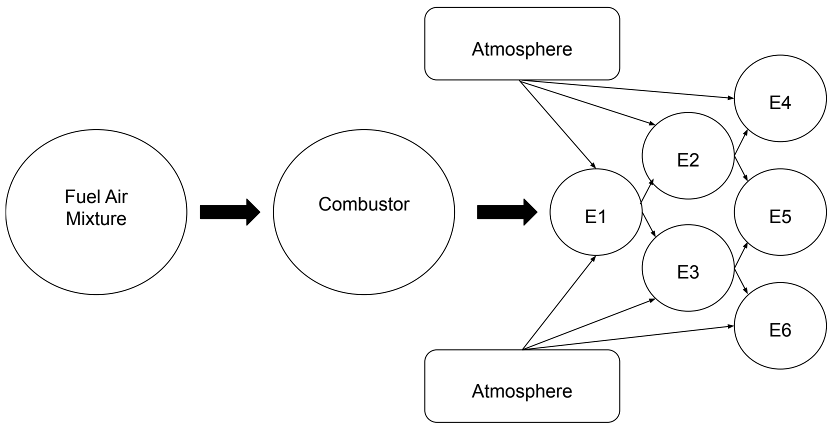


Fig. 3: Linear model for developing a less well mixed system.

where $\dot{m}_i$ is the mass flow from a specific reactor, m_i is mass of that reactor, $n_{s,i}$ is the number of sinks for that reactor, and $\tau(t)$ is the residence time as a function of clock time. $\tau(t)$ is specified, m_i is also determine from an initial specified value but evolves throughout the simulation, and $n_{s,i}$ is the sum of the i^{th} row of the adjacency matrix. The details of this can be found in the `model.py` file of the package.

The modeling is the primary function of `PyPlume` but it also has other functionalities. It includes methods for plotting data and constituents from the reactor network in various combinations based on user input. It

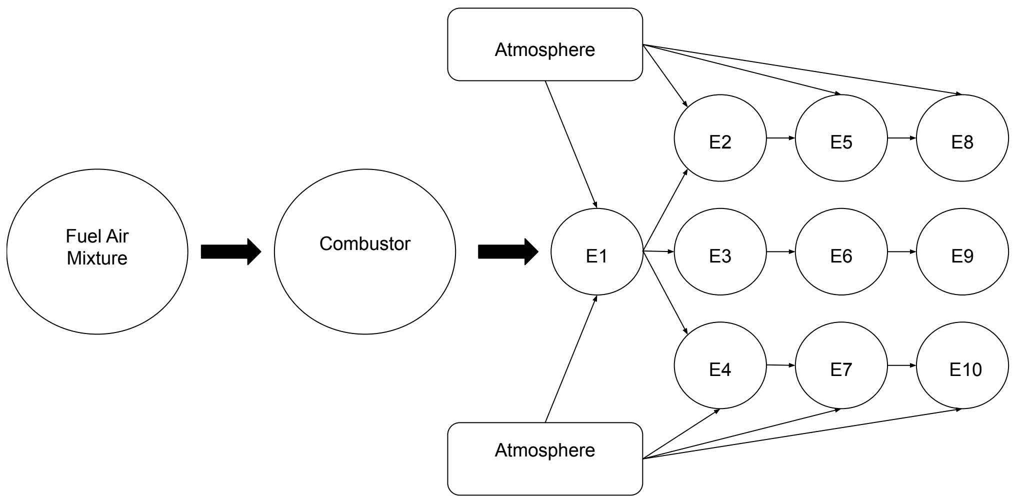


Fig. 4: Grid model for developing a less well mixed system.

includes methods for testing the currently implemented code, and it also has a command line interface for all of the currently implemented modules. The methods used in the modules are not necessarily critical to this section but more added functionalities—so I omitted discussion of them here.

3 Results

The major result of this project is `PyPlume` itself. During development, it was not applied to a specific application but built for future needs of my project. So this section will focus on it's use as it is the primary result. The source code for `PyPlume` is available on Github[1]. It is also available on PyPi[2] and Anaconda[3].

I believe the easiest way to use `PyPlume` is through `conda` via
`conda install -c anthony-walker -c cantera -c conda-forge pyplume`. `PyPlume` has five submodules, four of which are fully implemented for the current version:

1. `pyplume.mech`
2. `pyplume.model`
3. `pyplume.figures`
4. `pyplume.output`
5. `pyplume.statistics` - (not fully implemented)

All of these submodules have both a command line and a script interface.

3.1 Command line

In this section, I will walk through use on the command line. The `mech` package offers capabilities to manage your mechanism files as well as use mechanisms already provided by the package. `pyplume.mech -h` will show all of the available options to the user—Figure 6. Use of this package however is completely optional; Cantera provides a built in set of mechanism files which can be used with the `model` interface.

[1] https://github.com/SoftwareDevEngResearch/pyplume
[2] https://pypi.org/project/pyplume/
[3] https://anaconda.org/anthony-walker/pyplume

```
pyplume.mech -a test.cti
pyplume.model simple -tf 1 --steady --verbose --mech test.cti
pyplume.figures simple.hdf5 -d -w -p 'CO2' 'mass'
```

Fig. 5: Usage of CLI interface on an example

```
usage: pyplume.mech [-h] [-r] [-l] [-a ADD] [-d DELETE] [-t]
This is the commandline interface for managing mechanism files of PyPlume.
optional arguments:
  -h, --help            show this help message and exit
  -r, --restore         set this flag to restore mechanism files.
  -l, --list            set this flag to list mechanism files.
  -a ADD, --add ADD     this can be used to add a mechanism file to the codes
                        internal data.
  -d DELETE, --delete DELETE
                        this can be used to delete a mechanism file to the
                        codes internal data.
  -t, --test            set this flag to run test functions.
```

Fig. 6: Commandline interface help for `pyplume.mech`

The `model` interface—shown in Figure 7—can be used with `pyplume.model -h`. It has one positional argument which requires a specified model (`simple`, `grid`, `linear`). Otherwise, it gives the capabilities to run a simulation as desired based on certain flags.

The final interface available to the command line is `pyplume.figures -h`—shown in Figure 8. This interface allows the plotting of data produced by the `pyplume.model` interface.

These three interfaces can be combined to produce data and figures as desired. This example places `test.cti` in the mechanism files (assuming that it exists in the directory). It then integrates to the final time of 1, advances to steady state, and plots the resulting data.

The resulting data was omitted for brevity purposes but the plots generated are shown in Figure 9.

3.2 **Script**

These same results can be produced in a python script like Figure 10. This however is only a very limited case used for illustration purposes. The code has much more capability if configured appropriately which I believe to be a relatively simple process in comparison to other codes. This may be biased though because I wrote the code.

```
usage: pyplume.model [-h] [-ss] [-t0 [T0]] [-tf [TF]] [-m [MECH]] [-dt [DT]]
                     [-t] [-v]
                     [{simple,grid,linear}]

This is the commandline interface for running an exhaust network.

positional arguments:
  {simple,grid,linear}  This is a required arguement that specifies the model
                        which will be used. Currently implemented choices are
                        simple, grid, and linear.


optional arguments:
  -h, --help            show this help message and exit
  -ss, --steady         set this flag run to steady state after integration
  -t0 [T0]              Initial integration time
  -tf [TF]              Final integration time
  -m [MECH], --mech [MECH]
                        mechanism file
  -dt [DT]              Integration time interval
  -t, --test            set this flag to run test functions.
  -v, --verbose         set this flag to run print statements during the
                        process.
```

Fig. 7: Command line interface help for `pyplume.model`

4 Conclusions & Future Work

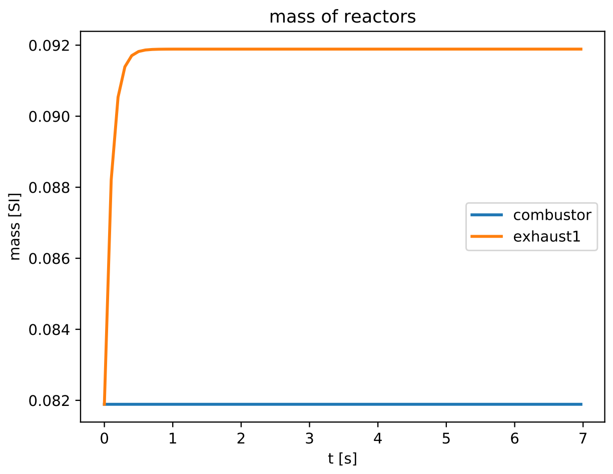
In this paper, I have presented the basic functionality of `PyPlume` but it can be used for much more complicated scenarios. It consists of four currently implemented modules for modeling and graphing the results of exhaust reactor networks. These modules are `mech`, `model`, `figures`, `output`. With these modules and proper configuration of initial conditions, I should be able to run fairly complicated reactor network simulations with relative ease.

In the future, I hope to complete the package's fifth module which was intended to be statistical analysis of the results. I also hope to complete more extensive testing suites for the other modules and set up some kind of continuous integration. As it stands, the only real theory behind the code is that of Cantera's[6]; I hope to include relevant exhaust modeling research in the form of class methods.

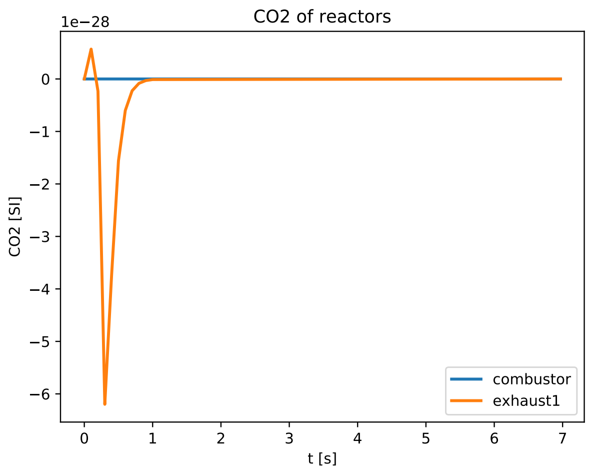
```
usage: pyplume.figures [-h] [-t] [-v] [-w] [-d] [-p PROPERTY [PROPERTY ...]]
                       [-r REACTORS [REACTORS ...]]
                       [filename]
This is the commandline interface for plotting results from the exhaust
network.
positional arguments:
  filename              filename used for plotting.


optional arguments:
  -h, --help            show this help message and exit
  -t, --test            set this flag to run test functions.
  -v, --verbose         set this flag to run corresponding print statements
                        through code.
  -w, --write           write plots to file as they are generated.
  -d, --display         set this flag to display plots on screen.
  -p PROPERTY [PROPERTY ...], --property PROPERTY [PROPERTY ...]
                        Plot a specific property
  -r REACTORS [REACTORS ...], --reactors REACTORS [REACTORS ...]
                        Specify reactors integer indices to plot property for
                        a subset of reactors
```

Fig. 8: Command line interface help for `pyplume.figures`



(a) Plot of mass produced by the `PyPlume`.



(b) Plot of CO_2 produced by the `PyPlume`.

Fig. 9: Plots generated by `pyplume.figures`.

```
import pyplume

pyplume.mech.mechFileAdd('test.cti')

pm = pyplume.model.PlumeModel.gridModel(mechs=['test.cti','air.cti','test.cti'])

pm.buildNetwork()

[pm(t) for t in np.arange(0.1,1.1,0.1)]

pm.steadyState()

fgk = pyplume.figures.figureGenerationKit("./simple.hdf5")

fgk.plotProperty(['CO2','mass'],save=True)
```

Fig. 10: Script interface to match the commandline example.

References

1. Health Effects of Organic Aerosols. 2014. URL: http://www.narsto.org/event.src?ID, doi:10.1080/08958370701866008.
2. Global Market Forecast Airbus. Global Networks Global Citizens, 2019-2038, 2019.
3. Ulrike Burkhardt and Bernd Kärcher. Global radiative forcing from contrail cirrus. *nature.com*, 2011. URL: www.nature.com/natureclimatechange, doi:10.1038/NCLIMATE1068.
4. Fabio Caiazzo, Akshat Agarwal, Raymond L. Speth, and Steven R.H. Barrett. Impact of biofuels on contrail warming. *Environmental Research Letters*, 12(11), 11 2017. doi:10.1088/1748-9326/aa893b.
5. Kenneth E Christian, William H Brune, Jingqiu Mao, and Xinrong Ren. Global sensitivity analysis of GEOS-Chem modeled ozone and hydrogen oxides during the INTEX campaigns. *Atmos. Chem. Phys*, 18:2443–2460, 2018. doi:10.5194/acp-18-2443-2018.
6. David G Goodwin, Raymond L Speth, Harry K Moffat, and Bryan W Weber. Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes. \url{https://www.cantera.org}, 2018. doi:10.5281/zenodo.1174508.
7. Makoto M. Kelp, Christopher W. Tessum, and Julian D. Marshall. Orders-of-magnitude speedup in atmospheric chemistry modeling through neural network-based emulation. 8 2018. URL: http://arxiv.org/abs/1808.03874.
8. J. Murray Mitchell. The Effect of Atmospheric Aerosols on Climate with Special Reference to Temperature near the Earth's Surface. *Journal of Applied Meteorology*, 10(4):703–714, 8 1971. doi:10.1175/1520-0450(1971)010<0703:teoaao>2.0.co;2.
9. Ulrich Pöschl. Atmospheric aerosols: Composition, transformation, climate and health effects, 11 2005. doi:10.1002/anie.200501122.
10. Thomas F. Stocker, Dahe Qin, Gian Kasper Plattner, Melinda M.B. Tignor, Simon K. Allen, Judith Boschung, Alexander Nauels, Yu Xia, Vincent Bex, and Pauline M. Midgley. *Climate change 2013 the physical science basis: Working Group I contribution to the fifth assessment report of the intergovernmental panel on climate change*, volume 9781107057999. Cambridge University Press, 1 2013. doi:10.1017/CBO9781107415324.
11. Anthony Walker. pyplume. 6 2020. URL: https://doi.org/10.5281/zenodo.3891934#.XuPwuNU5_pY.mendeley, doi:10.5281/ZENODO.3891934.