Author response to the review of “MultilayerPy (v1.0): A Python-based framework for building, running and optimising kinetic multi-layer models of aerosols and films”.

We thank both referees for their comments. Changes to the repository have been made and are available via the GitHub page (https://github.com/tintin554/multilayerpy). Upon acceptance of these changes, they will be released as MultilayerPy v1.0.2 on PyPI and the Zenodo archive due to the minor changes in how the models are constructed and some cosmetic changes to the source code, making it more readable and “pythonic”.

Additionally, some tutorial videos are being recorded and will be released in order to guide new users of the software.

Referee 1

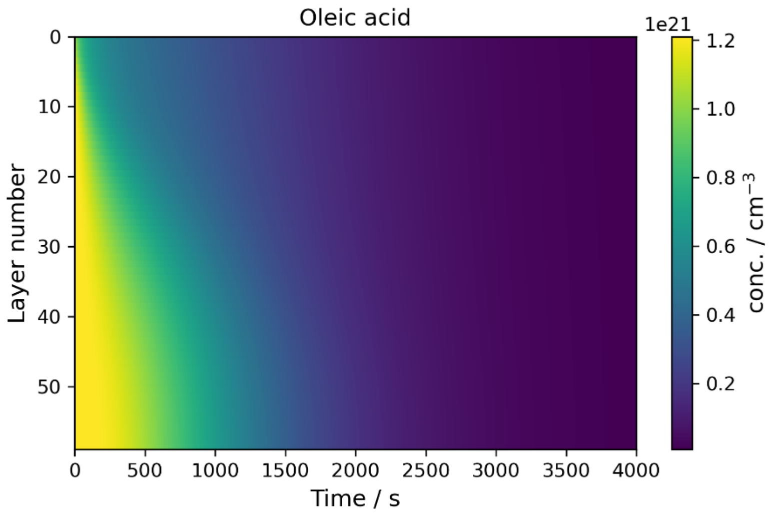
“This work developed an open-source framework MultilayerPy to build and run kinetic multi-layer models. Researchers can utilize this framework to choose a certain reaction scheme, diffusion regime, and model component based on particular aims. It is a reproducible process. Local and global optimization are applied and further tested on the oleic acid-ozone heterogeneous reaction system. The work is well-presented, and the framework makes comparing aerosol models with experiment data easier. I have one major comment and several minor comments.”

We thank the referee for taking the time to review this study and the helpful comments.

Major comments:

(1) Overall, the quantity of case studies is insufficient to assess the framework's potential applications. For example, current case studies are not convincing in testing the multi-layer modeling ability of MultilayerPy. As shown in figure 3, there is no oleic acid concentration gradient between the layers. Could the authors provide more cases to test the ability to represent the layer differences? Also, will it be easy to find other reaction systems to test the differences between KG-SUB and KM-GAP? and how the optimization algorithms will address the differences?

We have now included an extra case study which reproduces our recent work on a semi-solid self-organised (lamellar) form of oleic acid mixed with its sodium salt (sodium oleate). This demonstrates how a concentration gradient between layers can be visualised (see figure below). A new “lamellar phase oleic acid” Jupyter notebook has been created which reproduces this output. We have remained with the oleic acid reaction system due to its popularity as well as the interesting phase behaviour it exhibits. This example also makes use of a custom parameterisation of oligomer diffusivity (resolved in this model), which is also described in the new notebook and addresses point **4** from referee 1.



[Figure 5. The concentration profile of a film of semi-solid (self-organised lamellar) oleic acid during ozonolysis. The film is 0.98 µm thick and 77 ppm ozone was applied to the model.]

We aimed to highlight the differences between KM-SUB and KM-GAP with the comparison presented in Fig. 3. We believe other systems do not offer extra insights into these differences between KM-SUB and KM-GAP – caused by the way KM-SUB and KM-GAP treat model layer thickness (KM-GAP has adaptive layer thicknesses) and the volatilisation of model components. Optimisation algorithms (local and global) will not treat each model differently. However, the optimum model output will be affected by the number of, and the identity of the varying parameters given to these algorithms – as we saw in Fig. 3. Having more data to fit to would constrain these fitting algorithms.

Minor comments:

(2) Line 29--33: Consider including more references about the specific scientific questions KM-SUB and KM-GAP have resolved. That can be the potential application of MultilayerPy.

Additional references have been added.

(3) Line 54: typo ``readable''

Typo corrected.

(4) Line 62: I think the flexibility of adding unique processes to the framework is an important feature. However, the steps to achieve this are not well-documented in the rest of the paper. I recommend adding some descriptions about this at the steps listed in lines 93--100.

This has now been addressed in the response to point **1** where the diffusion gradient and a specialised composition-dependent diffusion regime is employed. We describe how this was done and have included the Jupyter notebook which reproduces the output presented in the updated manuscript.

We have also updated point 5 of the list as follows. Changes in **bold**:

[Model simulator object creation to run and save model outputs. **There is potential to add custom parameterisation (see *Case study 3*).**]

(5) Figure 3: Would it be better to combine (b) and (c) together? It will make the differences more evident.

We have now combined (b) and (c) together.

(6) Line 252: Besides the codes, it would be easy for readers to follow if the steps to create the cases are briefly documented.

The Jupyter notebooks reproducing these case studies document the steps. We have increased the detail of the description of each step in the case study building process. Both the Jupyter notebooks and the corresponding .html files in the docs folder have been updated.

The main text has been updated to clarify that the notebooks describe each step of the model building process.

(7) Line 280: How did you get the optimized value of $\alpha_{\rm s,0, ozone}$? Did you use the same model data used for the MCMC sampling procedure? I am thinking of training and validating data differences in Machine learning.

Yes, we used the same data to initially optimise the model and then sample the parameter space. The optimised value was obtained from the differential evolution algorithm. This was then used as a starting point for the MCMC algorithm to walk around the parameter space so that we could quantify parameter uncertainty.

The idea of splitting the experimental data into training and validation sets is interesting. This would require many datapoints to justify using this method. The model-experiment systems presented here have too few datapoints to attempt this. However, models could be fitted to experiments with high time resolution kinetic data, where there are many datapoints to split into training and validation data.

Referee 2

This manuscript develops an open-source package which can build kinetic multi-layer models. These models are useful in studying bulk-surface interactions for particles and the interplay between chemistry and thermodynamics. The ozonolysis of oleic acid system, which has been studied with both experimental work and kinetic modeling, is used as the most extensive example. Optimization of model parameters with experimental data is also explored via MCMC sampling techniques. Overall, this manuscript is easy to follow and this code base is going to be an extremely valuable tool for the community, though I believe a few additions to the notebooks code be helpful for those less familiar with code development (which is where this manuscript has the potential to have the biggest impact).

We thank the referee for taking the time to review this manuscript and the helpful suggestions.

Major comments

This manuscript intends to reduce the workload for those attempting to use kinetics modeling, but there are parts of the notebooks which I believe to be a bit difficult for novice users (regardless of their knowledge of the Python language). I spent quite a bit of time playing with the documents, and first would like to thank the authors, as I believe this tool is going to be quite useful for many in the field moving forward. I have a few major notes to make about the code base:

(1) What about the case where a user doesn't want to consider a reaction? A good example for using this model could be in the case of water condensation, which is explored in the KM-GAP case studies (3.1). It seems that all chemical components need to be defined via the reaction scheme, but in the case of water diffusing through an organic aerosol, perhaps you are not interested in any further chemistry. One could just create a set of reactions with rate constants all equal to 0 so the chemistry doesn't occur, but this doesn't seem like a 'best practice' way to code this up. Perhaps I missed how to do this more elegantly, though, in which case it would be good to highlight this in the code or manuscript.

We thank the referee for this helpful comment. We have now edited the code so that the user is able not to supply reactants or products to the reaction scheme object, telling MultilayerPy that there are no reactions occurring – the reaction scheme creation is still necessary so that the user must be explicit in not wanting reactions to occur. It is also possible for the user not to include a specific component in the reaction scheme so that it only diffuses between layers and potentially evaporates from the surface (depending on if it is set as volatile). This allows the user to include reactive and non-reactive species in the model.

An additional line in the MultilayerPy crash course notebook has been added which describes how one would do this. An extra sentence in point 2 of the package description in the main text has also been added to highlight that reactions and composition-dependent diffusion are optional.

(2) In general, the units are difficult to follow. While Table A1 provides the unit description for a KM-SUB application, Table S2 provides unit values. There is a challenge in the code in identifying what the true input values are. Take parameter `delta_1`: it is reported as 0.8 nm but you have to input the unit as cm. If you take both tables together then you can decipher what's happening, but I think it might be useful just to comment off the units after the parameters in the dictionary. It was just difficult for me to change parameters on a whim to representative values.

We have now changed the units in table S2 to be consistent with the main text. We have also commented the units in the tutorial notebooks for clarity.

(3) Consider a ‘docs’ sect would be useful beyond the provided notebooks to understanding this code base.

There is a docs folder in the source code repository but we understand that it is not written in prose as it is automatically generated from the source code. As mentioned at the beginning of this document, a series of videos are being recorded which will guide the user in installing and running MultilayerPy, covering the key features of the package and how it is arranged. As feedback from the community arrives, an “FAQ” document will be updated and will address common issues. This document has been started in the repository linked at the top of this document.

We have now added a section on basic installation to the manuscript. There is also more markdown text in the crash course notebook.

Minor comments

(4) Line 200: You comment that MCMC cannot be used to compare two different models. What exactly do you imply by this? MCMC in the context of this paper is used to sample the posterior distribution and could for different reaction systems extract the values of, for instance, accommodation coefficients. These coefficients are themselves comparable to the extent that the model represents the physical system. So, in that sense you can compare ‘between models’. Essentially, I am just wondering what you originally intended to say.

MCMC does sample from the posterior distribution, however this distribution cannot be compared to that of another model applied to the same data (e.g. a model with more components or an extra process). This is because MCMC only draws samples *proportional* to the posterior as it ignores the Bayesian evidence (*Z* in eq. 2). It is true that one could compare the parameter distribution from fits with two different model fits.

Our point is that posterior distributions from two different models sampled with the same data are not comparable due to the lack of *Z*, which would scale model-data fits to make their posterior distributions comparable. There exist algorithms such as nested sampling (Skilling, 2004), which aim to calculate this *Z* value in order to compare two different models with the same experimental data – calculating a so-called “Bayes factor”. This is the subject of ongoing work for this kind of modelling, though these algorithms are computationally very expensive when applied to these models – which is the major barrier.

We have clarified this paragraph. Changes in **bold**:

[“…Because of this, MCMC cannot be used to **select the best of** two different models **applied to the same data** as $p(D|\theta)$ would not be normalised to the same scale. **Though it is possible to compare parameter estimations from different models**.”]

(5) Further, I have a question about the Emcee package: are the final extracted parameters based on the lowest value of the cost function in the markov chain? Or is a separate optimizer run on that cost function fit to optimize the parameters? While MCMC itself tends toward the best fit parameters, it does not create the best fit parameters.

The final parameters from MCMC sampling are the mean of those parameter distributions. If the user wants to find the minimum of a cost function, global optimisation is required as MCMC is not an optimiser. In this paper we started the MCMC procedure from the global minimum, as recommended by Hogg and Foreman-Mackey (2018).

One can chose either to report the optimised value from global optimisation (i.e. the parameters with the lowest cost function) or the mean of the varied parameter distributions from the MCMC sampling procedure. These values (global minimum and MCMC mean) are available in the output .csv file that can be created after the model fitting and sampling process. MCMC is not an optimisation algorithm but is a very useful tool for estimating parameter uncertainty in a statistically robust way (rather than running a global optimisation algorithm many times, which would take much longer).

(6) Line 262: The authors state that differences in the model performance result from a lack of time-resolved particle size information after 17 min. While this point is subtle, the differences after 17 min are because of the physical differences in the model parameters, which in part result from the optimization on only 17 min of data.

We thank the referee for this comment. We have clarified our point to say that the decrease in particle size due to nonanal evaporation increases the concentration of oleic acid in the particle, leading to the model decay profile (Fig. 3(b)) and concentration profile (Fig. 3(d)). Changes in **bold**:

[“Time-resolved particle size information is not known for this experiment. **This will affect the number concentration of oleic acid in the particle in the KM-GAP model, which considers nonanal evaporation and resulting particle size change,** accounting for the significant differences seen between the KM-SUB and KM-GAP outputs at times greater than the last experimental datapoint (~17 min).”]

(7) Line 301: You reference the strong correlation observed in Fig S1. I am wondering if in the 2d histogram the burn in period is removed –it appears you can see the walk as evidenced in the extremities on both the 1 and 2D histograms. It is not untypical in many applications to remove the MCMC burn-in for visualizing the posterior. With more restricted bounds it may be easier to view the correlation.

This plot does include the burn-in period. Looking at the output, the walkers seem to occasionally wander to other regions of high probability, which likely also involve correlation between these two parameters.

It is also not unusual to observe a “banana” shaped 2-D distribution in such corner plots, especially if the distribution is multi-modal – which is probably the case here (Hogg and Foreman-Mackey, 2018).

(8) Table S2: check the labeling of [3]nonanal and [4]products and the consistency in the subscripts of the variables in the table. I think they may be switched.

We thank the referee for pointing out this mistake and have updated the table and scheme accordingly.

(9) In the simulate code (`simulate.py`), if you turn the save figs on, the figures save on a 0-based indexing rather than a 1-based indexing. The problem here is that for someone not privy to coding, this may be confusing in interpreting which reaction component is which (because the components are listed into the model with a 1-based indexing).

We thank the referee for pointing this out and have changed the code so that plots are saved with a 1-based indexing.

Editorial

Line 11: insert ‘the’ before ‘particle and film level’

Inserted.

Line 33: replace ‘were’ with ‘are’

Replaced.

Line 306: Should be ‘encourage’ (drop the s).

The s is dropped.

Author corrections

In applying our response to the referees, we noticed that the layer numbering in the heatmap plots in Fig. 3 was inversed. This has now been corrected.

References

D. Hogg and D. Foreman-Mackey, *Data Analysis Recipes: Using Markov Chain Monte Carlo*, *Astrophys. J. Suppl. Ser.*, 236, 2018, doi: 10.3847/1538-4365/aab76e.

J. Skilling, *Nested Sampling*, *AIP Conf. Proc.*, 735, 2004, doi: 10.1063/1.1835238.