PROPTI – A Generalised Inverse Modelling Framework

To cite this article: Lukas Arnold et al 2018 J. Phys.: Conf. Ser. 1107 032016

View the article online for updates and enhancements.

You may also like

- Transition temperature and the equation of state from lattice QCD: Wuppertal-Budapest results
  Szabolcs Borsanyi, Gergely Endrodi, Zoltan Fodor et al.

- Application of rule-based data mining techniques to real time ATLAS Grid job monitoring data
  R Ahrens, T Harenberg, S Kalinin et al.

- Energy-Variation Analysis and Orbit-Complexity Quantification
  Fotios Kasolis and Markus Clemens
PROPTI – A Generalised Inverse Modelling Framework

Lukas ARNOLD, Tristan HEHNEN, Patrick LAUER, Corinna TRETIN, and Ashish VINAYAK

1 Civil Safety Research, Forschungszentrum Jülich, Germany
l.arnold@fz-juelich.de
2 Civil Safety Research, Forschungszentrum Jülich, Germany
tristan.hehnen@cern.ch
3 Department of Fire and Explosion Safety, University of Wuppertal, Germany
lauer@uni-wuppertal.de
4 Department of Fire and Explosion Safety, University of Wuppertal, Germany
trettin@uni-wuppertal.de
5 Civil Safety Research, Forschungszentrum Jülich, Germany
a.vinayak@fz-juelich.de

ABSTRACT
Simulation of pyrolysis involves the knowledge of reaction kinetics and thermo-physical parameters, which are in general not directly measurable. Inverse modelling provides means to deduce the needed parameters from experimental data. This complex process involves the coupling of a simulation model with an optimisation method as well as the handling of a large amount of data. All of these processes are prone to errors and therefore a unified and automated implementation is beneficial for the whole community. In this contribution, a software to carry out this process is proposed. PROPTI is an open source tool written in Python, that is meant to provide a framework for inverse modelling of parameters in computer simulation, with emphasis on pyrolysis modelling in fire simulation. Its generalised formulation allows the usage of any simulation model in combination with various experimental data. The underlying optimisation library allows the utilisation of HPC systems. After presenting the concept of this framework, two examples are shown to illustrate the process and demonstrate the capabilities.

KEYWORDS: pyrolysis, FDS, inverse modelling, optimisation, HPC, framework
INTRODUCTION

Pyrolysis describes a thermo-chemical decomposition of solids releasing gaseous species, some of which are combustible. Thus, modelling pyrolysis is essential to simulate burning of solid materials. Often the heat release rate of a fire scenario is used to represent a design fire. This implicitly prescribes the pyrolysis and therefore the fire development. In this approach, environmental conditions, like ventilation and geometry, have no self consistent impact on the fire. To overcome these restrictions, the pyrolysis reaction rate can be computed, based on the material properties, temperature and environmental conditions. One possible estimation of the reaction rate is formulated following the Arrhenius equation, with its material specific set of reaction kinetic parameters.

To compute pyrolysis the material temperature has to be determined. This implies that the reaction kinetic parameters are not sufficient to fully describe the degradation behaviour. Thermo-physical parameters, like density or heat capacity, are necessary, as well as other parameters, like layer thickness. Real materials are complex mixtures of different chemical components. This makes it very difficult to design experiments to directly gain the needed parameters. Therefore, a possible approach to obtain these parameters is an inverse modelling process (IMP). [5]

As a general concept, given experimental data and a model, the target of the IMP is to identify the set of model parameters, which best represent the experimental observation. To achieve this objective, an IMP builds upon an optimisation algorithm. A simulation of the current parameter set is carried out, then the simulation results are compared to the target data. Based on this result the optimisation algorithm proposes a new set of parameters for the next iteration. This loop is repeated until convergence criteria are satisfied. This approach is shown in figure 1.

Fig. 1. Inverse modelling process.

A challenge of this approach is to couple the simulation model to an optimisation algorithm, and to process the resulting data. Matala investigated the usage of different optimisation strategies [8], Wang used a genetic algorithm in combination with FDS to investigate road tunnel fires [12], Bal and Rein investigated effects of inverse pyrolysis modelling on the estimated parameters[2], and many more. One of the most notable contributions is Gpyro by Lautenberger [7], which provides a highly detailed (3D) pyrolysis model and boundary conditions for FDS. It also provides means for material property estimation with a collection of optimisation algorithms.

In contrast to the above, the presented framework in this contribution is not aimed to develop a new model for heat transfer or pyrolysis itself. It strives to provide a generic method interface of how to perform inverse modelling as an open source software application. Therefore, users need to spend less time to set up a framework on their own, but can focus on the actual task at hand. Thus, it is highly beneficial for the fire safety community.

A flexible and open source IMP framework called PROPTI [1] is proposed. It focuses on handling the experimental data sets and the communication between simulation software and optimisation algorithms. Different packages of optimisation algorithms can be linked to PROPTI, which allows to chose algorithms in a simple fashion. Up to now, SPOTPY [4] has been linked, but connections to other packages are easily possible. Virtually any simulation software can be used, since PROPTI handles the coupling in a general way. However, due to the authors current research activities, the focus is on Fire Dynamics Simulator (FDS) [10]. An appropriate documentation, examples and tutorials on the usage of PROPTI are provided.
FRramework
The PROPTI framework is written in Python 3 and consists of a set of object oriented modules and convenience scripts, which ease typical application scenarios. It depends on common Python modules, as well as the optimisation package SPOTPY [4]. The framework is written in a very general way, with no emphasis on any special simulation model or experimental setup. However, there exist formal requirements for the definition of the model’s input parameters and the formatting of simulation and experimental data. The optimisation process is able to consider multiple experiments and has no restriction on which parameters shall be optimised. For demonstration purposes, the modelling of thermogravimetric analysis (TGA) experiments with FDS is shown in the following section. Before the IMP is started, a sensitivity analysis can be conducted to quantify the individual impact of each parameter and eventually reduce the number of parameters to be optimised. Optimisation methods can easily be swapped and arbitrary algorithm packages can be linked to PROPTI.

To process experimental data sets and results of the simulation model, they need to be stored as simple text files. This requirement is directly satisfied by FDS if output devices are used, otherwise an additional post-processing has to be triggered. The data sets do not have to be defined at the exactly same evaluation points, as PROPTI will use interpolation methods for data comparison.

Data and Model Definition
As the initial step, a configuration file must be created, that utilises Python syntax. It contains details about where to find the experimental data, which optimisation method to be applied and which parameters are to be examined. Information on the simulation model execution, including the path to the executable, the execution mode (serial or parallel) as well as input file templates must be provided. The experimental data is a set of data files that cover multiple experiments or repetitions. Thus, their relationship to the corresponding simulation setups, i.e. input file templates, has to be defined as well.

Execution of the Optimisation
Once the configuration file is set up, the optimisation procedure can be started. The main phases, which are consequently repeated, are basically:

1. PROPTI calls the optimiser to define a set of current parameter values to be evaluated.
2. PROPTI sets up working directories for each simulation run.
3. PROPTI uses the provided information to replace the place holders in the template file, or multiple files if needed, with the current parameter values.
4. PROPTI executes the models.
5. PROPTI extracts the simulation data and passes it to the optimiser.
6. The optimiser compares the experimental and simulation data and generates a fitness value.
7. The optimiser decides if further evaluations are needed, based on the fitness value.

A further advantage of PROPTI is an additional set of tools, that provides means for automated pre- and post-processing within the IMP. Due to the usage of placeholders and Python syntax, complex logic dependencies between parameters can be represented. Since the optimisation process can be computationally demanding, it benefits from high performance computing (HPC) systems and therefore PROPTI is designed with HPC application in mind. Depending on the algorithm used, PROPTI is able to simultaneously execute simulations with these parameter sets. This concept allows to use shared (via sub-processes), as well as distributed (via MPI) memory systems and is already implemented in PROPTI. It has been successfully used on CERN HPC-clusters and the JURECA supercomputer [6].

Analysis
After the model evaluations are finished, the optimiser provides the parameter set, which best fits the experimental data sets. General means of analysing the simulation results are provided by PROPTI. It is possible to create plots, that show the development of each parameter during the IMP. Furthermore, an overview of the fitness value development is provided. FDS input files with the best parameter set can be generated for each simulation setup.
EXAMPLES

The examples demonstrate general capabilities of PROPTI on deliberately simple setups. Mock-up material parameters have been generated, utilising a simulation of a thermogravimetric analysis (TGA) experiment from the FDS examples (tga_analysis.fds). This fabricated experiment is used to verify the capability of the IMP to reproduce known values.

The selected material consists of three components, which together define a material sample of cellulose. The first component is water, that evaporates around 100 °C, and for simplicity is neglected here. The two remaining components are each described by the same material parameters, like density or heat of combustion, and differ only in reaction kinetics. Here, the FDS parameters REFERENCE_TEMPERATURE and REFERENCE_RATE have been chosen, as described in the FDS user guide. Obviously, these parameters are easily estimated by fitting a triangle to each peak, as described in literature already [9]. As stated above, these examples are chosen for the sake of demonstrating the procedure.

The first example demonstrates that the combination of PROPTI and the SCE-UA algorithm is able to converge to a known solution. The second example demonstrates how multiple experiments, and simulations thereof, can be used to gain a more robust parameter set. Here, the material is subjected to three different heat fluxes, by using three heating rates in both the experiment and the simulation: 5 K/min, 10 K/min and 15 K/min (HEATING_RATE). More sophisticated setups, which include a wider variety of experiments or more repetitions, are possible as well and are part of authors current research.

As an optimisation algorithm the SPOTPY implementation of the Shuffled Complex Evolutionary Algorithm (SCE-UA) [3] was chosen. Four complexes were used, based on the four optimisation parameters and the maximum number of repetitions, i.e. the generated parameter sets, was set to 600. Note, in the second example three different simulation setups are involved, therefore up to 600 times three model executions are needed.

Example 1

After 600 repetitions, which corresponds to 16 generations, the best parameter set is close to the original parameter values. A comparison between the parameters themselves is provided in table 1. Figure 2a provides a comparison between the mock-up experimental data and the model’s response using the best parameter set. Figure 2b shows the development of the fitness value during the run of example 1. The fitness value is defined as a negative value and the optimiser tries to maximise it. Each generation is represented as a boxplot. Thus, it is easier to judge the development of the fitness value, specifically for generations that each contain hundreds of simulations.

For assessing the parameter sensitivity, a Fourier Amplitude Sensitivity Test (FAST) [11] is conducted. The output is shown in table 2 with $s_1$, the individual influence of a single parameter, and $s_t$, the sensitivity of the model to a single parameter dependent on all other parameters.

As a visualisation of the sensitivity assessment, the development of the parameters with the lowest and the highest values are shown. The parameter that got the lowest significance value during the FAST assessment is the reference rate of component 2. Its development over the IMP is shown in figure 3a. The highest significance value was received by the reference temperature of component 1, shown in figure 3b. These figures represent the expected development, based on the sensitivity analysis.

Table 1. Comparison of known parameters from the prescribed FDS example and the SCE-UA results.

| Known parameter                  | Example 1 | Example 2 |
|----------------------------------|-----------|-----------|
| Reference temperature component 1| 315.0     | 315.006   | 315.008   |
| Reference rate component 1       | 0.0056    | 0.00559347| 0.00559759|
| Reference temperature component 2| 430.0     | 429.992   | 430.007   |
| Reference rate component 2       | 0.0075    | 0.00750615| 0.00749932|
(a) Comparison between the mock-up experimental (original FDS) data and the simulation response of the best parameter set.

(b) Development of the fitness value during the inverse modelling process, where generations are represented as boxplots.

Fig. 2. Results of the first example.

Table 2. Sensitivity of each parameter on the simulation response, computed with FAST.

| Parameter                          | $s_1$    | $s_2$    |
|-----------------------------------|----------|----------|
| Reference temperature component 1 | 0.352860 | 0.579744 |
| Reference rate component 1        | 0.319699 | 0.402163 |
| Reference temperature component 2 | 0.028856 | 0.140844 |
| Reference rate component 2        | 0.030123 | 0.068708 |

(a) Development of the reference rate of component 2 during the inverse modelling process.

(b) Development of the reference temperature of component 1 during the inverse modelling process.

Fig. 3. Parameter development in the first example.

Example 2

The previous example is extended to three different simulation setups for different heating rates. Thus, each parameter set needs to be evaluated under three different conditions and all results are compared to their respective (mock-up) experimental data. In this case, the same parameter set is written into three FDS input files, which represent the different experimental conditions. The simulation setups enable the user to specify the different relations between individual experimental and simulation data series. As a result of the IMP, an optimal parameter set is generated to recreate the material behaviour well over all experimental conditions. The simulation results of the best parameter set are shown in figure 4a, which match the target mass loss rates reasonably well. An overview of the development of the fitness values for example 2 is provided in the boxplots in figure 4b. The fitness value converges again.
SUMMARY & OUTLOOK

A framework to estimate parameters for pyrolysis modelling, utilising an inverse modelling process is proposed in this work. It provides a guided starting point for scientific and engineering projects to model fire propagation. Its generality allows to use arbitrary simulation software as well as optimisation algorithms. Furthermore, its standardisation supports the knowledge transfer among its users. The basic capabilities and features were demonstrated and verified, based on simple TGA data sets. The authors welcome anyone to make use of this framework and contact the authors in case of questions. As this is an ongoing project, the authors encourage contributions to this project.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the computing time granted (project jjsc27) by the JARA-HPC Vergabegremium and VSR commission on the supercomputer JURECA [6] at Forschungszentrum Jülich.

REFERENCES

[1] L. Arnold, T. Hehnen, P. Lauer, C. Trettin, and A. Vinayak. “FireDynamics/propti,” March. Zenodo. Mar. 2018. doi:10.5281/zenodo.1188756.
[2] N. Bal, and G. Rein. “On the Effect of Inverse Modelling and Compensation Effects in Computational Pyrolysis for Fire Scenarios.” Fire Safety Journal 72: 68–76. 2015. doi:10.1016/j.firesaf.2015.02.012.
[3] Q. Duan, V. K. Gupta, and S. Sorooshian. “Shuffled Complex Evolution Approach for Effective and Efficient Global Minimization.” Journal of Optimization Theory and Applications 76 (3). Springer: 501–521. 1993.
[4] T. Houska, P. Kraft, A. Chamorro-Chavez, and L. Breuer. “SPOTting Model Parameters Using a Ready-Made Python Package.” PloS One 10 (12). Public Library of Science: e0145180. 2015.
[5] M. M. Khan, A. Towarson, and M. Chaos. “Combustion Characteristics of Materials and Generation of Fire Products.” In SFPE Handbook of Fire Protection Engineering, edited by M. J. Hurley et al., 1143–1232. Springer New York, New York, NY. 2016. doi:10.1007/978-1-4939-2565-0_36.
[6] D. Krause, and P. Thörnig. “JURECA: General-Purpose Supercomputer at Jülich Supercomputing Centre.” Journal of Large-Scale Research Facilities JLSRF 2: 62. 2016.
[7] C. Lautenberger. “Gpyro3D: A Three Dimensional Generalized Pyrolysis Model.” INTERNATIONAL ASSOCIATION FOR FIRE SAFETY SCIENCE. 2014.
[8] A. Matala, C. Lautenberger, and S. Hostikka. “Generalized Direct Method for Pyrolysis Kinetic Parameter Estimation and Comparison to Existing Methods.” Journal of Fire Sciences 30 (4). Sage Publications Sage UK: London, England: 339–356. 2012.
[9] K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk, and K. J. Overholt. Fire Dynamics Simulator User’s Guide. Apr. 2018.
[10] K. McGrattan, R. McDermott, W. Melli, G. Forney, J. Floyd, S. Hostikka, and A. Matala. “Modeling the Burning of Complicated Objects Using Lagrangian Particles.” In Conference Proceedings of the Twelfth International Interflame Conference, 743–753. 2010.
[11] A. Saltelli, S. Tarantola, and K.-S. Chan. “A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output.” Technometrics 41 (1). Taylor & Francis Group: 39–56. 1999.
[12] X. Wang. “Fire Dynamics Simulator (FDS) Pyrolysis Model Analysis of Heavy Goods Vehicle Fires in Road Tunnels.” Phdthesis, University of Canterbury, Private Bag 4800, Christchurch, New Zealand. 2017.