Check for updates

NIST Internal Report
NIST IR 8549

temo: teqp-based model optimization

Ian Bell

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8549

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Internal Report
NIST IR 8549

temo: teqp-based model optimization

Ian Bell
Applied Chemicals and Materials Division
Material Measurement Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8549

December 2024

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

Certain equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

NIST Technical Series Policies
Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on 2024-11-26

How to cite this NIST Technical Series Publication:
Bell I (2024) temo: teqp-based model optimization. (National Institute of Standards and Technology, Gaithersburg, MD), NIST IR 8549. https://doi.org/10.6028/NIST.IR.8549

Author ORCID iDs
Ian Bell: 0000-0003-1091-9080

Contact Information
ian.bell@nist.gov

NIST IR 8549
December2024

Abstract

The `temo` library is a set of building blocks that facilitate the automation of fitting models to experimental data for thermodynamic properties. The library is written in Python to enable broad re-use and is based upon the suite of thermodynamic equations of state and modeling tools available in the `teqp` library. The flexibility of the provided methods allow the user to extend the capabilities of `temo` for their own particular problems.

Keywords

Optimization; model fitting; equation of state.

Table of Contents

List of Figures

1. Introduction

In the engineering thermodynamics community, the most accurate thermodynamic mixture models are obtained from the multi-fluid mixture approach[1–7]. In this approach, equations of state (EOS) for the pure fluids are combined in a corresponding states approach with a mixing rule which contains four adjustable parameters for each binary pair of indices ij in the mixture: $\beta_{T,ij}$, $\gamma_{T,ij}$, $\beta_{v,ij}$, $\gamma_{v,ij}$, with T for the temperature reducing function and v for the volume reducing function. Fitting these parameters can yield, for simple enough mixtures, a quite reasonable representation of mixture behavior for binary pairs. The process of fitting four adjustable parameters can be carried out without specialized code, as existing software implementations (e.g., CoolProp [8]) allow the interaction parameters to be modified in code.

The refrigeration industry has requirements for highly accurate thermodynamic models, and the fluids are in general not especially difficult to measure with high accuracy (the temperature and pressure ranges match those of metrology-grade instruments (e.g., Refs. [9–11]). These high-accuracy measurements require a more advanced mixture model approach including a departure function, which is an empirical contribution to the mixture model. The mathematical complexity of the departure function means that one needs rather more specialized optimization approaches.

The availability of highly-accurate experimental measurements motivates the development of a tool that can make the process of fitting mixture models more straightforward. The target userbase is researchers with moderate expertise in Python, but limited expertise in fitting thermodynamic mixture models. There are many such researchers who utilize less accurate models because easy-to-use tools for fitting these models do not exist.

2. Implementation and architecture

The core code of `temo` is written in Python, enabling broad re-use due to the flexibility with which the code can be modified by the user for their own purpose. `temo` has a tight connection with `teqp`[12], which is written in C++ for speed. Otherwise, `temo` leverages the scientific python stack (SciPy, NumPy, pandas) for data loading and analysis. In that way the user can integrate the code of `temo` with their own code.

The library `temo` is a set of functional building blocks that can be combined together to carry out a process that normally requires significant expertise. The goal is to provide a few bare bones examples demonstrating the core capabilities of the library, and the user is then expected to copy the provided examples and modify them to suit their own problems.

The process follows roughly the familiar extract, transform, load (ETL) paradigm. Several data files containing experimental data in the CSV (comma-separated values) format (though the delimiter need not be a comma to assist with internationalization) are loaded, and where necessary, routines are provided for adding guess values for iterative calcula-

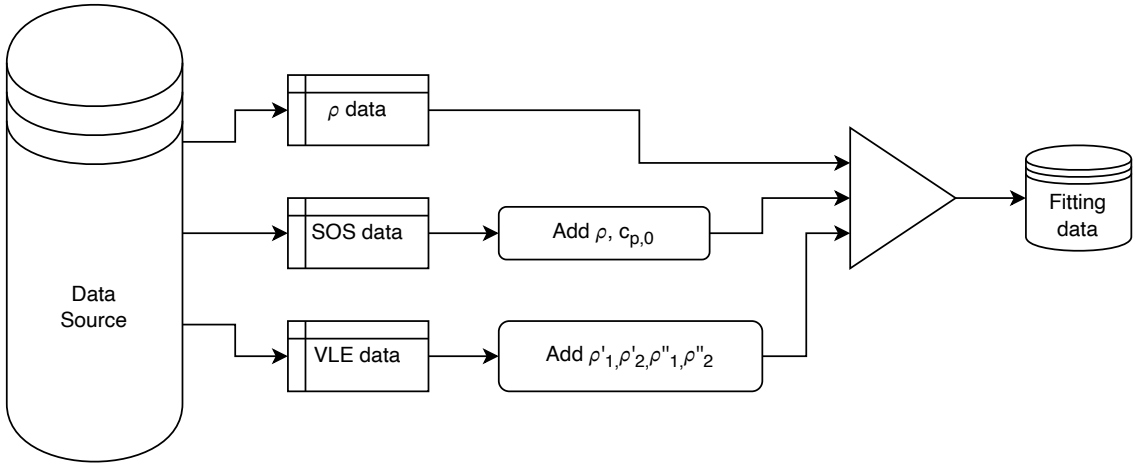

Fig. 1. Schematic of the data loading pipeline for a fitting problem in which density, speed of sound (SOS), and vapor-liquid-equilibrium (VLE) data are available.

tions as needed. Once these data files are loaded, they are passed into the optimization class. The optimization class is provided as a stub class that is meant to be extended by the user. The cost function can be added to and modified by the user.

Routines are also provided for post-processing the results and automatically generating the necessary publication-ready plots. In this way the entire pipeline from experimental data to publication-ready models, validation data, and supporting information for the manuscript can be collected in one location. The provided code will expedite the process of model fitting and make the process more reproducible.

The user starts by writing a script in Python that loads the experimental data with loading functions from the `temo.fit.data_loaders` module. These functions load different kinds of experimental data

- speed of sound: `load_SOS`
- density data: `load_PVT`
- vapor-liquid-equilibrium data: `load_VLE`
- critical point data: `load_crit`
- cross-second virial coefficient data: `load_B12`

A generic schematic of the process is shown in Fig. 1 for a mixture for which density, speed of sound, and vapor-liquid-equilibria data are available. An example of this case are the refrigerant models developed in the last few years [6, 7].

Not all kinds of data may be available for a given model, but in previous works [6, 7] it was found that fitting PVT, speed of sound, and VLE data resulted in a very reliable model because these experimental data were found to be self-consistent. Furthermore, these data were characterized by very small expanded uncertainties (below 0.1% combined expanded uncertainties for speed of sound and density in general and below 1% for vapor pressure).

The next core functionality provided by `temo` is the generation of mutant models with the help of `teqp` in the `temo.fit.mutant_factories` module. Models in `teqp` can be specified in the form of JSON-based (JSON: javascript object notation) data structures. The mutant factory functions map from an input vector of variables being fit to the JSON data structure needed by `teqp`. An instantiated model is returned from the selected factory function. An example mutant function for an exponential term could read

The optimizer function (selected by the user) minimizes the cost function to yield the "best" model. There is no unambiguous definition of the "best" model because this is a multi-objective optimization process; fine-tuning of the cost function may be required. For a given vector of independent variables, a model is generated by the mutant generation functions, and the cost function is evaluated. After many calls to the cost function, the global minimum of the cost function is hopefully obtained. However, most of the optimization approaches that are well-suited to this type of problem are neither deterministic nor guaranteed to converge to the global minimum of the cost function. Nevertheless, the differential evolution algorithm[13] has been used with good results for fitting accurate departure functions and interaction parameters.

The optimization script provided with `temo` is configured to do multiple repeats of the optimization to increase the probability of finding the true minimum of the cost function, and to allow for fitting multiple models at the same time. The individual optimization problems are embarassingly parallel, so the `multiprocessing` package in python is used to parallelize the optimization runs. Each run result is stored in a `.tar.xz` file with a unique name to avoid file-writing collision. Furthermore, the use of a compressed `.tar.xz` format allows for metadata to be collected at a quite granular level during the optimization for analysis in post-processing. The metadata associated with the "best" model doesn't change very often and compression keeps the archive file from growing unmanageably by eliminating the files with non-changing contents. Otherwise, the uncompressed archive is hundreds of MB of largely identical files; the compressed results are less than a MB, which can itself be archived without incurring large storage requirements. The output archive includes the script used for the fitting as well as all input files to enable reproducibility of the entire fitting process.

2.1. Worked Example

The `fit_models.py` script provided with the code shows a complete example of doing the entire fitting process. It is the complete workflow that was used in the previous publi-

cations, with some additional refinement for user-friendliness. The parallelism is enabled in the script, and at completion, a uniquely-named archive in the `.tar.xz` file format is created, ready for analysis and publication.

3. Binary Mixture VLE

The development of mixture models for multicomponent mixtures is based up on the development of models for binary mixtures; the multicomponent model is based upon the combination of the binary models. Thus some helper functions were developed for working with binary mixtures, in particular to use the isochoric tracing formalism to fit one or more mixing parameters for binary mixtures based on vapor-liquid equilibrium data. Two helper classes were developed:

3.1. BinaryVLEIsothermFitter

This class wraps methods required for doing fitting of models for experimental data along an isotherm. There are convenience methods for construction of models based on model parameters as well as a cost function that can be passed to the user's optimization function of choice. This allows for straightforward fitting of one or more interaction parameters

3.2. BinaryVLEIsothermInterpolator

This class takes the trace from `BinaryVLEIsothermFitter` and constructs a number of interpolation objects from the `scipy.interpolate.interp1d` module. These interpolation objects allow for easy mapping between the arclength tracing parameter which is monotonic by construction and other variables like pressure and temperature to facilitate inter-interpolation between the variables.

4. Reuse potential

The software library `temo` is the first full-featured public library for fitting highly accurate thermodynamic models. The goal is to have a well-written and well-engineered software tool that becomes the *de facto* approach for fitting the models described in this work. There are already several research groups that have adopted `temo`, and a number of models are actively being fit with `temo`.

Although the focus in `temo` has been to develop mixture models based upon the multi-fluid model approach, there is no fundamental limitation that prevents its use for fitting other kinds of models. Indeed the previous versions of the codebase that would become formalized as `temo` have already been used to fit interaction parameters for cubic equations of state and the PC-SAFT equation of state. Adding a different EOS to `temo` only requires to write a new mutant factory for the kind of model, all other calculations are mostly model-agnostic.

4.1. Support

For support, users are requested to open a topic for discussion at https://github.com/usnistgov/temo/discussions. Then, if a bug is identified, the discussion topic can be converted to an issue, and the bug and resolution can be linked. In some cases it is not possible to communicate via github, and the primary author can be emailed for information.

5. Software Engineering

5.1. Location

Archive

Name: data.nist.gov
Persistent identifier: https://doi.org/10.18434/mds2-3632
License: Public domain for NIST
Publisher: Ian Bell
Version published: 0.1.0

Code repository

Name: github
Persistent identifier: https://github.com/usnistgov/temo
License: Public domain for NIST

5.2. Quality control and documentation

The underlying library `teqp` is covered by unit testing on all major operating systems. Documentation of the routines in `temo` are provided in sphinx-generated documentation. This documentation is auto-posted on readthedocs, and re-built at each commit to the github repository. The documentation explains how to use each element of the code, working up to a complete example of fitting data with `temo`. The documentation is generated from jupyter notebooks that can be downloaded and re-run by an interested user. This is the same approach used in `teqp`.

5.3. Operating system

windows, mac, linux

5.4. Programming language

Python 3.9+ (but 3.12+ is recommended)

5.5. Dependencies

If `teqp` needs to be re-compiled (not common, likely if a newer version of python is used for which binary wheels are not presently available), a compiler is needed to compile `teqp`, otherwise all packages are available in binary form.

Here is the `environment.yml` environment specification for conda:

```
name: temo
dependencies:
- python=3.12
- matplotlib
- scipy
- pandas
- pip
- pip:
  - teqp
  - CoolProp
  - "git+https://github.com/usnistgov/temo.git"
```

Acknowledgments

This material is based upon work supported by the US Army Corps of Engineers and the Department of Defense Strategic Environmental Restoration Development Program under contract No. WP23-3793: Holistic Optimization of Air-Conditioning Systems for Military Use with Low-GWP Refrigerants.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Army Corps of Engineers or the Department of Defense Strategic Environmental Restoration Development Program.

Partial funding for this work was provided by NIST. Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

References

[1] Kunz O, Klimeck R, Wagner W, Jaeschke M (2007) *The GERG-2004 Wide-Range Equation of State for Natural Gases and Other Mixtures* (VDI Verlag GmbH).

[2] Kunz O, Wagner W (2012) The GERG-2008 Wide-Range Equation of State for Natural Gases and Other Mixtures: An Expansion of GERG-2004. *J Chem Eng Data* 57:3032–3091. https://doi.org/10.1021/je300655b

[3] Gernert J, Span R (2016) EOS-CG: A Helmholtz energy mixture model for humid gases and CCS mixtures. *J Chem Thermodyn* 93:274–293. https://doi.org/10.1016/j.jct.2015.05.015

[4] Neumann T, Herrig S, Bell IH, Beckmüller R, Lemmon EW, Thol M, Span R (2023) EOS-CG-2021: A Mixture Model for the Calculation of Thermodynamic Properties of CCS Mixtures. *Int J Thermophys* 44(12). https://doi.org/10.1007/s10765-023-03263-6

[5] Lemmon EW, Jacobsen RT (2004) Equations of State for Mixtures of R-32, R-125, R-134a, R-143a, and R-152a. *J Phys Chem Ref Data* 33(2):593–620. https://doi.org/10.1063/1.1649997

[6] Bell IH (2022) Mixture Models for Refrigerants R-1234yf/134a, R-1234yf/1234ze(E), and R-134a/1234ze(E) and Interim Models for R-125/1234yf, R-1234ze(E)/227ea, and R-1234yf/152a. *J Phys Chem Ref Data* 51(1):013103. https://doi.org/10.1063/5.0086060

[7] Bell IH (2023) Mixture Model for Refrigerant Pairs R-32/1234yf, R-32/1234ze(E), R-1234ze(E)/227ea, R-1234yf/152a, and R-125/1234yf. *J Phys Chem Ref Data* 52(1):013101. https://doi.org/10.1063/5.0135368

[8] Bell IH, Wronski J, Quoilin S, Lemort V (2014) Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp. *Ind Eng Chem Res* 53(6):2498–2508. https://doi.org/10.1021/ie4033999

[9] McLinden MO, Perkins RA (2023) A Dual-Path Pulse-Echo Instrument for Liquid-Phase Speed of Sound and Measurements on p-Xylene and Four Halogenated-Olefin Refrigerants [R1234yf, R1234ze(E), R1233zd(E), and R1336mzz(Z)]. *Ind Eng Chem Res* https://doi.org/10.1021/acs.iecr.3c01720

[10] Rowane AJ, Perkins RA (2022) Speed of Sound Measurements of Binary Mixtures of 1,1,1,2-Tetrafluoroethane, 2,3,3,3-Tetrafluoropropene, and 1,3,3,3-Tetrafluoropropene Refrigerants. *Int J Thermophys* 43(4). https://doi.org/10.1007/s10765-021-02966-y

[11] Rowane AJ, Rasmussen EG, McLinden MO (2022) Liquid-Phase Speed of Sound and Vapor-Phase Density of Difluoromethane. *J Chem Eng Data* 67(10):3022–3032. https://doi.org/10.1021/acs.jced.2c00441

[12] Bell IH, Deiters UK, Leal AMM (2022) Implementing an Equation of State Without Derivatives: teqp. *Ind Eng Chem Res* 61(17):6010–6027. https://doi.org/10.1021/acs.iecr.2c00237

[13] Storn R, Price K (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J Global Opt* 11(4):341–359. https://doi.org/10.1023/A:1008202821328