

FeO_s - An Open-Source Framework for Equations of State and Classical Density Functional Theory

Philipp Rehner,^{†,¶} Gernot Bauer,^{‡,¶} and Joachim Gross^{*,‡}

[†]*Energy and Process Systems Engineering, Department of Mechanical and Process Engineering, ETH Zurich, Tannenstrasse 3, 8092 Zurich, Switzerland*

[‡]*Institute of Thermodynamics and Thermal Process Engineering, University of Stuttgart, Pfaffenwaldring 9, 70569 Stuttgart, Germany*

[¶]*These authors contributed equally to this work.*

E-mail: gross@itt.uni-stuttgart.de

Abstract

In this work, we present an open-source software package, referred to as FeO_s – *Framework for Equations of State and Classical Density Functional Theory*. FeO_s is a collection of interfaces and data types that can be used (1) to implement thermodynamic equations of state and Helmholtz energy functionals for classical density functional theory, and (2) to compute thermodynamic properties of pure substances and mixtures, phase equilibria, and interfacial properties such as surface tensions and adsorption isotherms. The framework is written in the Rust programming language with a complete Python interface and is designed with a focus on usability and extensibility. It is openly available on GitHub (<https://github.com/feos-org/feos>). Equations of state can be implemented in Rust, yielding performant code, or as a Python `class`, which is useful for prototyping and with less emphasis on execution speed. In both cases, the user has to implement a single function: the Helmholtz energy. FeO_s then uses generalized (hyper-) dual numbers to evaluate the Helmholtz energy as well as

the required exact partial (higher-order) derivatives. Using this type of automatic differentiation delivers performance without the need for implementing any analytical derivatives. The performance is further enhanced by a caching mechanism that avoids duplicate model evaluations.

Together with the core interfaces and functionalities for equations of state and classical density functional theory, we provide implementations for multiple models such as the PC-SAFT equation of state (with homo- and heterosegmented group contribution methods) and Helmholtz energy functionals (including segment-based functionals). To showcase a selection of FeO_s' features, an example study of the adsorption of biogas in porous media using the PC-SAFT functional is provided.

Introduction

Thermodynamic equations of state (EoS) are fundamental tools in thermal and chemical engineering¹⁻³ They enable calculating properties of multi-component systems as a function of experimentally accessible quantities such as temperature, pressure, and composition. Equations of state differ in the breadth of their applicability and complexity, and, depending on the field of application, there are requirements on functionality, robustness, precision, and computational speed. A milestone in the description of fluid phases was provided by van der Waals' equation of state,⁴ which is based on a (coarse) molecular model and first led to description of vapor/liquid coexistence. Prominent modifications of the model by van der Waals that preserve the cubic volume-dependence are: Redlich-Kwong,⁵ Soave-Redlich-Kwong,⁶ and Peng-Robinson.⁷ Due to their fast evaluation times, cubic EoS are still widely used in technical applications. However, they lack a physical basis when describing non-spherical, polar, or hydrogen-bonding substances and mixtures. Modern EoS are developed based on a molecular model, i.e., a description of (pair-wise) intra- and intermolecular interactions. Whereas theories for fluids with simple spherical intermolecular interactions were developed in the 1970s,⁸ the field made a leap with a development of M.S. Wertheim, who derived a

description for highly directional interactions.^{9–12} Wertheim’s theory led to the statistical associating fluid theory (SAFT),¹³ which regards molecules as chains of spherical segments (thus accounting for the non-spherical shape of real molecules due to covalent bonds) and allows for short-ranged attractive (hydrogen) bonds. The success of SAFT led to the development of a plethora of derived models, of which the most used ones are PC-SAFT,^{14–17} SAFT-VR-Mie¹⁸ and soft-SAFT.^{19,20} A combination of a cubic equation of state with the association contribution from TPT1 was published as cubic + association (CPA).^{21–23} Finally, TPT1 and SAFT can resolve individual segments on a molecule, leading to the development of heterosegmented EoS like SAFT- γ -Mie²⁴ and gc-PC-SAFT.²⁵

Cubic and SAFT-type EoS aim to describe fluids based on a few either macroscopic (cubic) or microscopic (SAFT) properties. The small number of parameters ensures a robust extrapolation to state points for which no experimental data is available and the molecular model underlying these EoS ensure a meaningful description of mixtures with few binary interaction parameters or even predictions of mixture properties. However, experimental data is abundant for some fluids, and reference equations of state that use a large number of adjustable parameters to represent the experimental data of those substances have been developed. Reference EoS were published for pure components like water,²⁶ CO₂,²⁷ and nitrogen,²⁸ but also for mixtures like natural gas or related systems.²⁹

While EoS are used to compute properties of homogeneous fluid phases and to model phase equilibria and phase stability, they cannot model properties of microscopically inhomogeneous systems like fluids at interfaces or in porous media. Being able to model these phenomena is essential for dynamic processes such as droplet coalescence or formation of micelles or engineering applications such as adsorption. Classical density functional theory³⁰ (DFT) is a framework that extends fluid theories (i.e. equations of state based on a molecular model) to inhomogeneous systems. In DFT, the system is described by the grand potential, which can be expressed as a functional of the (partial) density *profiles*. Although DFT is a formalism that has been used and studied in research for decades, it is not a commonly

used method in industry as of today, although it can be used to predict properties that are difficult or expensive to measure, such as surface tensions of mixtures,³¹ contact angles³² and adsorption isotherms.^{33,34} In addition, it can provide insights into phenomena that are experimentally difficult to assess, such as the accumulation of light-boiling or amphiphilic molecules at interfaces.³⁵

Within the last years, multiple open-source packages in the field of thermodynamics and equations of state were published, each with its own focus. *CoolProp*³⁶ and *thermo*³⁷ provide databases, correlations for properties such as vapor pressures and activity coefficients, and equations of state. *Thermopack*^{38,39} (written in Fortran) focuses on equations of state with an emphasis on robust algorithms for phase equilibria and critical points, including multi-phase flashes to be used within computational fluid dynamics simulations. *phasepy*^{40,41} provides Python implementations of equations of state and algorithms for phase equilibria, including square gradient theory which can be used to describe density profiles across vapor liquid interfaces. *teqp*⁴² (written in C++) and *Clapeyron.jl*⁴³ (written in Julia) – similar to FeO_s – both utilize automatic differentiation, which circumvents the need to implement analytic derivatives of the Helmholtz energy. All of these projects have a common feature: they provide an interface to a dynamic, high-level programming language. *Clapeyron.jl* is implemented in the Julia programming language, while the others are either fully implemented in Python (*phasepy*, *thermo*) or provide Python bindings (*CoolProp*, *Thermopack*, *teqp*). Clearly, for a modern toolkit, this is a necessity as it enables access to a broader range of users and allows using tools such as Jupyter notebooks that make research more transparent and reproducible.

A recent survey of the ‘Working Party of Thermodynamics and Transport Properties of the European Federation of Chemical Engineering’ summarizes the most important gaps and concerns raised by industry in the field of applied thermodynamics and outlines specific requirements for model frameworks.⁴⁴ In a subsequent publication, the authors outline the “main directions that [they] believe the applied thermodynamics community should adopt

in the coming decade”:³

- There is a need for methods to assess the properties of fluids under confinement, at interfaces, and in the presence of external fields.
- Users must be able to parameterize the model and assess its uncertainties and range of applicability.
- In the same vein, access to these models and parameters has to be transparent - ideally in a standardized form and including the data used for the parameterization.
- Finally, there is increasing demand for ongoing education, training, and collaboration.

FeO_s is well suited to address these important topics. The implementation of DFT in FeO_s is designed with respect to the modeling of industrially relevant problems. It provides ad hoc functionalities to describe confined media and interfaces or to specify external potentials. Furthermore, FeO_s provides utilities to optimize model parameters within a couple of lines of code. All of this is possible from within Jupyter notebooks without sacrificing performance. With the simple installation and availability on all major platforms, results can easily be shared and reproduced.

FeO_s is developed with two use-cases in mind. First, it provides implementations for equations of state and Helmholtz energy functionals for DFT together with algorithms for critical point and phase equilibrium calculations as well as solvers for density profiles in multiple dimensions and coordinate systems. It can therefore be used as a toolkit to compute thermodynamic properties of homogeneous and inhomogeneous systems. And second, it provides interfaces and data types that can be used to extend the code, e.g. to implement new models and algorithms.

Equations of state in FeO_s are implemented in terms of additive Helmholtz energy contributions. To circumvent the need for implementing analytical (partial) derivatives of the Helmholtz energy contributions, which is a tedious, time consuming, and error-prone task,

FeO_s utilizes generalized (hyper-) dual numbers for computing *exact* derivatives. Generalized (hyper-) dual numbers are used to determine exact partial derivatives of the Helmholtz energy without programming any partial derivative. For detailed information about the implementation and how generalized (hyper-) dual numbers can be utilized in the context of thermodynamic models, such as equations of state, the interested reader is referred to an earlier work of our group.⁴⁵

FeO_s is purely written in the Rust programming language. Rust is a strongly typed, compiled language, including features for functional programming, that produces machine code comparable in performance to C++ and Fortran while offering additional safety features and an expressive type system. Furthermore, it comes with its own build system, package- and dependency manager, unit- and integration tests, as well as documentation builder and benchmarking system. Using the *PyO3*⁴⁶ library, FeO_s is compiled into a feature-complete Python extension module including type-checked interfaces (which are NumPy compatible) and proper error handling without any actual code written in Python. The Helmholtz energy function that has to be implemented for each model is generic over any generalized (hyper-) dual number. This means, depending on the derivative that is needed, the Helmholtz energy model is evaluated with the suitable generalized (hyper-) dual number data type. For example, evaluating the Helmholtz energy without partial derivatives just requires real-valued input variables, while the first partial derivative requires dual numbers (one real and one non-real value) as input. The Rust compiler supports *monomorphization*, which in our context generates, without further requirements from the software-developer, various Helmholtz energy functions for the different (hyper-) dual numbers at compile time. This leads to very efficient code because there is no penalty (e.g. virtual table look-ups) at run-time. Users who implement their own model can write code without expert-knowledge of (hyper-) dual numbers as if regular floating point numbers were used. We provide arithmetic operations (using *operator overloading*) and mathematical functions (such as trigonometric and Bessel functions) for generalized (hyper-) dual numbers.

The paper is structured as follows. First, we present the goals underlying the development of FeO_s and how these goals are reflected in the concepts of the code. Then we present the formal structure of the code. Subsequently, the general usage of the code is explained. We then list the implemented models and some of FeO_s ' unique features. Finally we present an example system that highlights the package's capabilities and close with a conclusion.

Goals of FeO_s

FeO_s is developed with two major goals in mind: usability and extensibility. The package can be used as a toolkit (using models and functionalities already implemented) or as a platform for new algorithms and models (using low-level data types and interfaces).

Usability demands that FeO_s is very easy to install and to use. We achieve this by providing a Python interface that mirrors all features of the underlying Rust library. The library is compiled into a Python wheel for every major platform and uploaded to the Python Package Index (PyPI) and can therefore be installed without having to compile the library, simply by typing `pip install feos` in your terminal. A Python interface allows using Jupyter notebooks, which are excellent for exploratory coding, supported by detailed documentation for objects and functions directly accessible from within notebooks. Besides learning and teaching, notebooks are suited for publishing research studies in a transparent and reproducible manner. Furthermore FeO_s offers low-barrier access to DFT functionalities like adsorption isotherms. Solving DFT problems requires a substantial initial programming effort, which currently limits the use of DFT approaches to few expert-groups. With FeO_s we aim to bring DFT methods to a broader and diverse community of researchers.

Extensibility means that the code is structured so that new algorithms and models can be implemented as easily as possible. We achieve this by formulating an EoS as a sum of Helmholtz energy contributions where the thermodynamic conditions (vector of moles or number of molecules \mathbf{N} , volume V , and temperature T) for which the Helmholtz energy

is evaluated are passed as generalized (hyper-) dual numbers. The `FeOs` framework picks the appropriate type of dual number depending on the required derivative to calculate a given property. Then, from a single call to the user-supplied Helmholtz energy function, the derivatives are obtained without requiring manually written analytic derivatives. The implementation of generalized (hyper-) dual numbers in Rust was previously done by the authors in a separate package `num-dual`.

For developing a new equation of state model or for a swift assessment of a model, where performance is not of major concern, the Helmholtz energy can alternatively be implemented in Python. The user provides a Python `class` that contains a method in which the Helmholtz energy is evaluated. With operator overloading and NumPy compatibility, the method can be written just like a standard Python method. Internally, the same generalized (hyper-) dual numbers are used to evaluate the Python code. Changes can be made to the EoS implementation in Python without requiring a recompilation of the `FeOs` library. This advantage in development speed comes at the cost of needing run-time polymorphism to evaluate operations for arbitrary types in Python as opposed to the compile-time polymorphism in Rust which is part of the reason for its high computational efficiency.

Code Structure

`FeOs` consists of three separate Rust libraries: `feos-core`, `feos-dft` and `feos`. The `feos-core` library defines data types and traits for equations of state, data types that store thermodynamic conditions (objects called `State`), and algorithms for, e.g., volume (or density) iteration, phase equilibrium and stability, and critical point calculations. In Rust, *traits* define abstract shared behavior similar to interfaces in other languages. The `feos-dft` library defines data types and traits for Helmholtz energy functionals and other data types and algorithms used in the context of DFT, e.g., density profiles, external potentials, adsorption isotherms, and convolution methods. The `feos-dft` library extends `feos-core` in the sense

that it reuses data types and algorithms where possible. In particular, Helmholtz energy functionals can be used to calculate bulk properties just like equations of state. Finally, `feos` contains the implemented models and higher-level utilities, such as tools for parameter optimization, and builds the Python package. With the separation into distinct libraries, `feos-core` can be compiled independently from the other two parts and can be used in standalone implementations of equation of state models.

The individual models, the `feos-dft` library, and utilities such as parameter optimization, are gated behind Rust *features*, i.e., conditional compilation directives. A feature can optionally be omitted during compilation which makes developing new algorithms and models easier and faster. For example, when implementing a new EoS, one can initially deactivate all features (decreases compile time and amount of code one has to consider) and incrementally add features and implement and test the needed interfaces.

A similar approach is used for different model capabilities. Whereas every EoS model requires a function for the Helmholtz energy, implementing, e.g., correlation functions for entropy scaling is optional. If the entropy scaling trait is not implemented, the code is compiled without the functionalities that depend on entropy scaling. Therefore, users can directly see which features are available from a given model, and developers can incrementally implement and test new code.

General Usage

Calculating a property in FeO_s is a three-step process that is visualized in fig. 1:

First, for a given mixture or pure substance, model parameters have to be read and used to instantiate the equation of state object. These model parameters do not appear in any mandatory interface in FeO_s – developers can freely decide how to handle parameters specific to their model. However, FeO_s provides methods to read parameters from JSON files and previously published parameters are available from the FeO_s GitHub repository in the

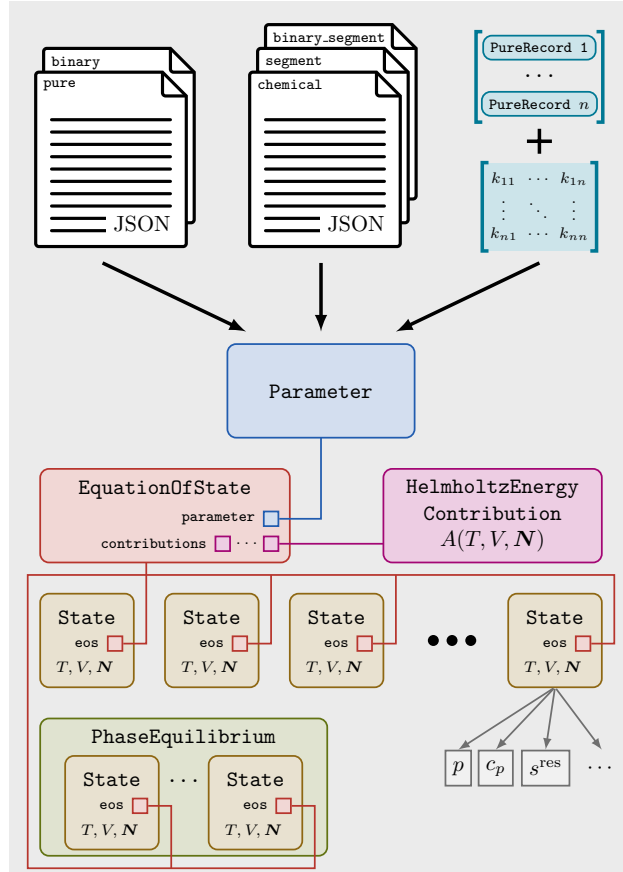


Figure 1: Relationship between different objects in FeO_s. A line with a square represents a stored reference (pointer). For example, each state object stores a pointer to a single equation of state while an equation of state stores one or more pointers to Helmholtz energy contributions.

form of JSON files. Instead of reading from files, parameter objects can also be instantiated by directly passing a list of pure component parameters and optionally a matrix of binary interaction parameters. If a group contribution method is implemented for the model, the parameters can be generated by reading chemical information like group counts and the corresponding segment parameters. Depending on the parameters passed, the EoS can create one or multiple `HelmholtzEnergyContribution` objects in which the actual calculation of the Helmholtz energy takes place. Explicitly keeping track of individual contributions is helpful for more complex models like PC-SAFT as they can be turned on or off depending on the considered molecular interactions.

Once an equation of state is instantiated (for a single substance or mixture), the second step is specifying the thermodynamic conditions. This can be done by providing a combination of state variables such as temperature, pressure, density, composition, enthalpy, and entropy. Other options are critical conditions or phase equilibria. All of the above methods use the EoS object as input and return `State` objects. A `State` internally stores the natural variables of the Helmholtz energy and a reference to the EoS object. A single instantiated EoS object can be reused in any number of states. For phase equilibria calculations like flash, bubble point or dew point calculations, the resulting object is a `PhaseEquilibrium` that contains a `State` object for every phase in the system.

The last step is calculating the property of interest by invoking a method on the `State` object. For example, to calculate the pressure, one would invoke the `pressure` method of the previously generated `State`. At this point, internally and invisible to the user, FeO_s transforms the data types of the stored state variables to the correct dual numbers depending on which partial derivatives are needed for the property. Listing 1 shows how the steps from parameters to properties are performed in practice using the Python interface.

Splitting the property calculation into the definition of thermodynamic conditions and the actual property calculation means that the iterative calculation of densities or temperatures only must be done once for calculating possibly multiple properties. Additionally, already

Listing 1: The three steps needed to compute a property in FeO_s .

```
1  # Step 1: read parameters from json file...
2  parameters = PcSaftParameters.from_json(
3      ['methane'],
4      'parameters/gross2001.json'
5  )
6  # ...and build equation of state
7  eos = EquationOfState.pcsaft(parameters)
8
9  # Step 2: define thermodynamic conditions
10 state = State(eos, temperature=300*KELVIN, pressure=1*BAR)
11
12 # Step 3: calculate property
13 chemical_potential = state.chemical_potential()
```

evaluated partial derivatives can be cached in the **State** object. This allows a user to write, e.g., a process model in a very convenient way without worrying about possibly expensive function calls because – once a **State** object is created – FeO_s automatically reuses already computed derivatives of the Helmholtz energy.

The three steps outlined above are similar for DFT. For bulk properties, Helmholtz energy functionals can simply be treated as equation of state objects. Internally, this is accomplished by evaluating the functional for a zero-dimensional density profile; for the user, the interfaces remain exactly the same. For inhomogeneous systems, the second step (i.e., the state creation) is split into an initialization step and the solution of the equilibrium condition. The initialization step depends on the system under study (vapor-liquid interfaces, fluids adsorbed in nanopores, etc.). For the solution, different solvers are implemented that can be adjusted to increase either the performance or the stability as needed. From the converged density profile, interfacial properties like surface tensions, adsorption, or solvation free energies can be determined.

For frequently occurring tasks, such as phase diagrams or adsorption isotherms, FeO_s

provides utility functions with a convenient interface. Internally, these utilities perform the same steps (repeatedly defining new conditions and calculating the property of interest), but they can profit from optimizations like using results from previous iterations as initial conditions to increase the stability and the performance of the calculations.

Provided Models and Features

As of this writing, FeO_s supports a simple implementation of the Peng-Robinson EoS, which is intended primarily as a pedagogic example, an implementation of the PC-SAFT equation of state^{14–17} and respective Helmholtz energy functionals,^{31,35,47} the uv-theory,⁴⁸ the perturbed truncated and shifted (PeTS) Lennard-Jones equation of state⁴⁹ and respective Helmholtz energy functionals, and the SAFT-VRQ Mie equation of state^{50,51} including Helmholtz energy functionals.⁵² The implementation of PC-SAFT includes associating¹⁵ and polar contributions (referred to as PCP-SAFT: dipole-dipole,¹⁷ quadrupole-quadrupole¹⁶ and dipole-quadrupole⁵³). In addition, both the homosegmented and heterosegmented group contribution approaches for PC-SAFT²⁵ are available.

We provide all commonly used static properties that can be calculated from partial derivatives of the Helmholtz energy in extensive, mole- and mass-specific forms (where appropriate). Partial derivatives of the Helmholtz energy, pressure, and (logarithmic) fugacity with respect to density (volume), temperature, and amount of substance for each component are also provided. For PC-SAFT, model equations for transport coefficients from entropy scaling are implemented. As of this writing, these can be used to calculate the viscosity (using substance-specific or group contribution parameters) of pure substances and mixtures,^{54,55} as well as thermal conductivities⁵⁶ and self-diffusion coefficients⁵⁷ of pure substances. Parameter files, including published correlation parameters, are provided on GitHub in the FeO_s repository.

DFT algorithms are generically implemented for 1D, 2D, and 3D and Cartesian, cylin-

drical, and spherical coordinate systems. We provide utilities for 1D interfaces (e.g., initial density profiles, surface tension diagrams for 1D planar interfaces) and 1D and 3D pores and adsorption calculations. External potentials can either be chosen from our implementations (hard wall, Lennard-Jones 9-3, Steele, free energy averaged, atomic positions plus force field) or provided as `numpy.ndarray` from within Python.

Most interfaces (both in Rust and Python) use dimensioned quantities via the `quantity` Rust library. This has multiple advantages. First, interfaces can check units and provide meaningful error messages. For example, we can report that molar densities must be used instead of erroneously provided mass densities. Second, we can provide a single interface for functions where, e.g., pressure or temperature are possible inputs and decide what algorithm to use based on the input’s unit. And third, the code is easier to read since in- and output units are unambiguous, and it is simple to perform unit conversions.

Example showcase: biogas adsorption

In this section, we showcase some of `FeOs`’ features using biogas and the PC-SAFT Helmholtz energy model. Biogas is modeled as a binary mixture of methane and carbon dioxide (CO_2). In a first step, we assess the accuracy with which the PC-SAFT equation of state correlates the binary phase behavior of the methane/ CO_2 mixture. Pure component parameters are read from JSON files. Parameter-sets from publications are available from the `FeOs` GitHub repository and they are listed in table 1.

Table 1: PC-SAFT pure component parameters for methane and CO_2 .

component	m	σ	ε/k_B	Q	reference
methane	1	3.7039	150.03	-	¹⁴
CO_2	1.6298	3.0867	163.34	3.9546	¹⁶

Here, the parameters for the non-polar methane and the quadrupolar CO_2 were published in different publications and are therefore read from separate input files.

Listing 2: Reading PC-SAFT parameters from multiple different json files.

```
1 parameters = PcSaftParameters.from_multiple_json([
2     (['methane'], 'parameters/pcsaft/gross2001.json'),
3     (['carbon dioxide'], 'parameters/pcsaft/gross2005_fit.json')
4 ])
5 eos = EquationOfState.pcsaft(parameters)
6 functional = HelmholtzEnergyFunctional.pcsaft(parameters)
```

The parameters can be used to initialize the model, which can be either an equation of state (line 5 of listing 2) or a Helmholtz energy functional (line 6 of listing 2). Helmholtz energy functionals can be directly used to calculate bulk properties and phase equilibria, just like an equation of state object. In addition to that they provide functions for the calculation of properties in inhomogeneous systems using classical DFT.

Because it is a reoccurring task, the calculation of a binary phase diagram can be done in a single line with `FeOs` which is shown in listing 3. The utility is provided by the `PhaseDiagram` class that is imported from the `feos.eos` module.

Listing 3: Phase diagrams for binary mixtures either at constant temperature or pressure.

```
1 # vapor-liquid equilibria at constant temperature
2 vle_t = PhaseDiagram.binary_vle(eos, 230*KELVIN)
3 # alternatively, vapor-liquid equilibria at constant pressure
4 vle_p = PhaseDiagram.binary_vle(eos, 20*BAR)
5 # plot results using matplotlib
6 plt.plot(vle_t.liquid.molefracs[:,0], vle_t.liquid.pressure/BAR)
```

The second argument to `PhaseDiagram.binary_vle` in listing 3 can be a temperature (line 2) or a pressure (line 4). The framework infers the type of diagram that is supposed to be generated based on the unit of the argument. The properties of the individual vapor and liquid states are available directly from the `PhaseDiagram` object and are also stored as dimensioned quantities.

FeO_s provides a framework for optimizing parameters of equations of state, in particular by calculating target functions for different properties. The target (or cost) functions can be provided to any optimization framework, e.g. `scipy.optimize`, to perform a parameter optimization. In this example, we optimize a binary interaction parameter k_{ij} to experimental data of vapor-liquid equilibria of methane/CO₂ at two different temperatures. The optimization results are shown in fig. 2. While the phase diagram as predicted by PC-SAFT solely based on k_{ij} , i.e. on Berthelot-Lorentz combining rules (dashed lines) predicts the experimental data⁵⁸ qualitatively well, the agreement can be improved by using the optimized value of the binary interaction parameter $k_{ij} = -0.0192$.

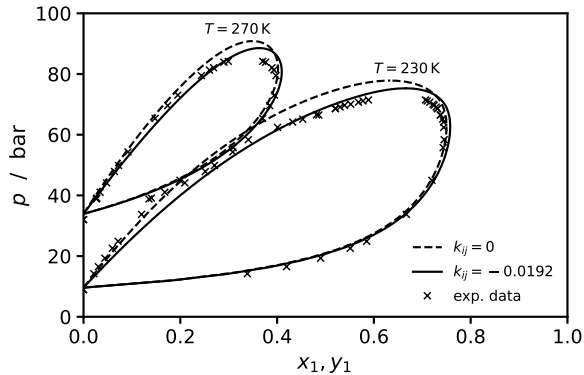


Figure 2: Vapor-liquid equilibria of the binary mixture methane/CO₂ at $T = 230$ K and $T = 270$ K. Comparison between experimental data,⁵⁸ PC-SAFT predictions ($k_{ij} = 0$), and PC-SAFT fits ($k_{ij} = -0.0192$).

Next we use the equation of state to calculate dynamic properties via entropy scaling. For PC-SAFT, FeO_s provides the correlation functions of Lötgering-Lin et al.^{54,55} for viscosities and those of Hopp et al. for thermal conductivity⁵⁶ and self diffusion.⁵⁷ For given PC-SAFT parameters, correlation parameters of pure substances can be adjusted very quickly. The viscosity of a mixture can then be predicted using simple combining rules for the correlation parameters.⁵⁵ For the methane/CO₂ mixture this is shown in fig. 3 where the correlation parameters for the pure substances were adjusted to the same experimental data that was used in the work of Lötgering-Lin et al.⁵⁵ and the mixture predictions are compared to experimental data.^{59,60} Listing 4 shows example code for the calculation of the mixture

Listing 4: Dynamic properties such as the viscosity can be calculated from a thermodynamic state just like static properties.

```

1 saft_results = []
2 for t, p, x in experimental_data:
3     # thermodynamic conditions read from experimental data
4     state = State(
5         eos,
6         temperature=t,
7         pressure=p,
8         molefracs=x
9     )
10    # store viscosity from entropy scaling
11    saft_results.append(state.viscosity())

```

viscosity.

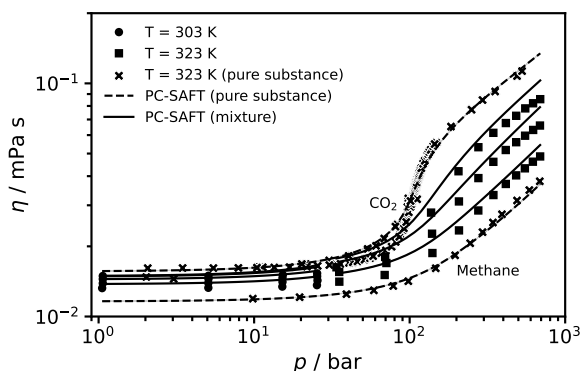


Figure 3: Viscosity of the binary mixture methane/CO₂ for different compositions at two isotherms. Lines represent calculations from PC-SAFT via entropy scaling, symbols represent experimental data. Parameters for the viscosity of pure substances (dashed lines) were adjusted to experiments. Mixture viscosities are predictions.

In listing 5 we illustrate how to predict the surface tension of the considered mixture using the PC-SAFT DFT model for the range of compositions covered in the phase diagram (fig. 2). To calculate the surface tension of a planar interface, the system is set up in 1D Cartesian coordinates. Here we discretize the interfacial region using 1024 grid points and a width of $300 \text{ \AA} = 30 \text{ nm}$. The resulting surface tensions σ for the two temperatures $T = 230 \text{ K}$

Listing 5: A diagram for surface tensions can be directly generated from a list of phase equilibria.

```

1  # surface tension for pre-calculated VLE's
2  sft = SurfaceTensionDiagram(
3      vle.states,          # phase equilibria
4      n_grid=1024,        # number of grid points
5      l_grid=300*ANGSTROM # width
6  )
7  # plot surface tension using matplotlib
8  plt.plot(
9      sft.liquid.molefracs[:,0],
10     sft.surface_tension/(MILLI*NEWTON/METER)
11 )
12 # plot density profile
13 plt.plot(
14     sft[0].profiles[12].z/(NANO*METER),
15     (sft[0].profiles[12].density/(KILO*MOL/METER**3)).T
16 )

```

and $T = 270$ K are shown in fig. 4. As an example, the calculated density profiles for the marked data point (defined through index 12 in listing 5) is shown in the inset. The density profiles show how the light-boiling methane accumulates in the interface.

The optimized PC-SAFT binary interaction parameter is then used to assess the adsorption properties of the biogas. The reference is an experimental study by Ottiger et al.⁶¹ of methane/ CO_2 adsorption on dry coal. For the DFT model we assume that the porous medium consists of spherical pores with an effective diameter. Because the pore structure of the coal is less homogeneous than ordered porous media like zeolithes or metal organic frameworks, we also treat the effective pore volume and the effective specific surface area as independently variable adjustable parameters. The interactions between the fluid and the solid are described with a Steele potential⁶² adapted to the spherical geometry⁶³ using Lorenz-Berthelot combining rules. The solid-solid energy and size parameters ε_{ss} and σ_{ss}

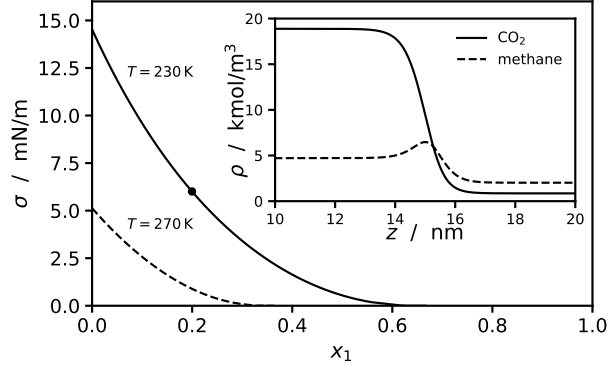


Figure 4: Surface tension of the system methane/ CO_2 at $T = 230$ K and $T = 270$ K as predicted by the PC-SAFT Helmholtz energy functional. The inset shows the density profiles of one selected interface on the 230 K isotherm.

are adjusted together with the pore size, the pore volume and the specific surface area to the pure component isotherms for methane and CO_2 . The optimized parameters are then used to predict the adsorption isotherms for binary mixtures with varying feed concentrations.

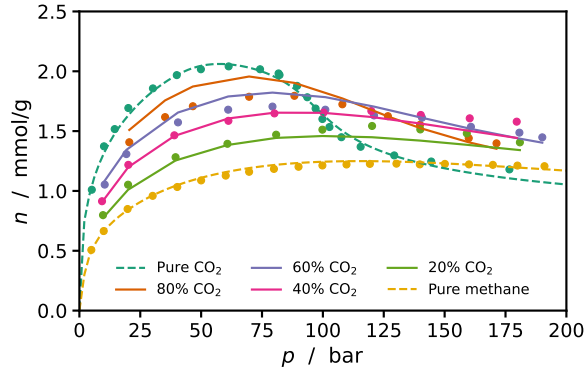


Figure 5: Adsorption isotherms for methane/ CO_2 at $T = 318.15$ K on a dry coal at different feed compositions. The solid parameters are fitted to the experimental data⁶¹ of the two pure component isotherms (dashed lines). From that the adsorption behavior of the mixture is predicted.

The results of the study are shown in fig. 5. The DFT model based on the PC-SAFT Helmholtz energy functional can describe the adsorption behavior of the two pure components well up to high pressures. For the mixtures, in the experimental setup by Ottiger et al.⁶¹ the CO_2 concentration of the bulk system is lower compared to the feed due to the

Listing 6: Calculation of an adsorption isotherm for a one-dimensional spherical pore modelled by a Steele potential.

```

1  # define external potential
2  potential = ExternalPotential.Steele(sigma_ss, epsilon_k_ss, rho_s)
3  # define pore - here 1D spherical with Steele potential
4  pore = Pore1D(
5      geometry=Geometry.Spherical,
6      pore_size,
7      potential,
8      n_grid
9  )
10 # define thermodynamic conditions
11 pressures = SIArray1.linspace(0.01*BAR, 200*BAR, 100)
12 temperature = 318.15*KELVIN
13 # specify solver: 10 Picard steps followed by Newton solver
14 solver = (DFTSolver()
15     .picard_iteration(max_iter=10, damping_coefficient=0.1)
16     .newton()
17 )
18 # calculate adsorption isotherm
19 isotherm = Adsorption1D.adsorption_isotherm(
20     functional, temperature, pressures, pore, solver
21 )

```

adsorption in the coal. Therefore, only pressure points with given experimental data (including the composition of the bulk phase) are included in the isotherm to ensure an accurate comparison.

Listing 6 exemplifies the calculation of a single adsorption isotherm. First, the external potential is defined (line 2). The potential, together with the geometry (the coordinate system), the size of the pore, and the number of grid points, defines the pore structure. For a given range of pressures and a temperature, the adsorption isotherm can readily be calculated (line 19-21). The solver can be customized and can be provided as optional argument. Here we use 10 steps of a Picard iteration followed by a Newton scheme.

Performance and benchmarks

In this section we present some performance characteristics and benchmarks of FeO_s. This is not an in-depth analysis – rather it is intended to give an impression of how long frequently used routines and tasks run. The presented timings are given for the binary system of methane/CO₂ (including quadrupolar contributions to the Helmholtz energy) and were obtained on a desktop machine running Linux with an Intel[®] Core[™] i7-7700K CPU (4.2 GHz). The code was compiled with full optimization (the build manager’s `release` flag) including link-time optimization (`lto = true`) which is the configuration we use to build the published Python wheels. A single CPU was utilized.

Table 2: Execution time t of several tasks for the binary system methane/CO₂ (unless stated otherwise). Timings were obtained from within Jupyter-notebooks using the IPython *timeit* command.

EoS (methane/CO ₂ , $T = 230$ K)		t
critical point		0.76 ms
T, p -flash	$p = 40$ bar, $x_1^{\text{feed}} = 0.5$	1.13 ms
bubble point	$x_1^{\text{liq}} = 0.5$	2.01 ms
phase diagram	50 compositions	49.70 ms
DFT (1024 grid points)		t
surface tension	$T = 230$ K, $p = 40$ bar	0.09 s
surface tension diagram	50 compositions	14.20 s
pore	$p = 190$ bar, $x_1 = 0.6$	0.13 s
adsorption isotherms (fig. 5)	241 calculations (200 pure, 41 mixture)	18.70 s
Parameter adjustment		t
k_{ij}	input: 156 vapor liquid equilibria total VLE calculations: 800	0.79 s
entropy scaling (CO ₂ , 4 parameters)	153 data points	0.72 s
pore parameters (5 parameters)	41 pressures: 21 for methane, 20 for CO ₂ 1024 grid points	73.00 s

Table 2 lists execution times for multiple frequently used tasks from within a Jupyter-notebook. For these timings we are not optimizing any input parameters with regards to execution speed. Almost all algorithms that we provide in FeO_s allow for modifications

of initial values, convergence tolerances, as well as discretisation and solver parameters (in the case of DFT) to fine-tune the behaviour of the algorithm for the system of interest. Modifying these parameters can drastically change execution speed and stability. In FeO_s , the default parameters can be obtained from the documentation and are generally chosen with respect to stability over speed.

For typical tasks of an EoS, we report timings for a critical point calculation, a temperature-pressure flash (for given composition), a bubble-point calculation (with equimolar feed) and the sequential calculation of VLE's for 50 compositions (phase diagram, cf. fig. 2). All tasks are performed at $T = 230\text{ K}$ for the methane/ CO_2 mixture including the optimized binary interaction parameter k_{ij} . While it is possible to produce a phase diagram by manually calculating multiple VLE's, the `PhaseDiagram` feature makes use of previously calculated conditions and thus is usually faster.

For DFT, we present the calculation of the surface tension for a given VLE, a surface tension diagram across 50 compositions (cf. fig. 4), the solution of the density profile for a given temperature, pressure and bulk composition in a one-dimensional spherical pore as well as the adsorption isotherms shown in fig. 5. All DFT calculations are performed in a one-dimensional coordinate system (either Cartesian or spherical) with 1024 grid points. To solve the density profiles, we use a Picard iteration (10 steps) with a damping constant of 0.1 followed by a Newton method.^{64,65} The convergence tolerance is 10^{-11} . Similar to the phase diagram, the algorithms for surface tension diagrams and adsorption isotherms can utilize density profiles from prior results or neighboring thermodynamic conditions making them considerably more effective than naive sequential calculations.

A common task in engineering is the adjustment of model parameters. For EoS, typical parameter regression tasks are pure-substance parameters, binary interaction parameters, and parameters of entropy scaling correlation functions. FeO_s provides utilities for all three tasks in the form of so-called `DataSet` and `Estimator` objects. A `DataSet` takes experimental data (thermodynamic conditions as well as a target property) as input and provides a

`DataSet.predict` method, that returns the prediction of the property for given a thermodynamic model. For example `DataSet.vapor_pressure(psat, t)` can be used to construct a data set for experimental vapor pressures (`psat`) at given temperatures (`t`). An `Estimator` object can contain multiple `DataSet` objects and – for a given EoS – return the predictions or relative differences with respect to the stored experimental data. Using the `Estimator` object, a cost function can be written in two or three lines of Python code.

To give an idea of the performance of a parameter optimization, we adjust the binary interaction parameter k_{ij} and the four parameters of the viscosity correlation function for CO_2 . For the optimization of k_{ij} a total of 156 experimental vapor liquid equilibrium data points is used. As solver we utilize the `least_squares` function of the `scipy.optimize` module with default settings, i.e. a Trust Region Reflective algorithm, 2-point Jacobian calculation, and all termination tolerances set to 10^{-8} . For the initial value of $k_{ij} = 0.0$ and the distance of the experimental datum to the predicted VLE as residuum, the solver converges within 4 iterations with a total of 8 evaluations of the cost function – a total of 800 vapor liquid equilibrium calculations – in less than a second. The same solver (and settings) is used to adjust the four viscosity correlation parameters of pure CO_2 .

Finally, we conduct an optimization of five parameters characterizing the solid fluid interactions in a porous system. The five parameters are adjusted to pure-component adsorption isotherms of methane and CO_2 . All DFT calculations use the same number of grid points (1024) and solver configurations (for the least squares solver and the DFT solver) as noted above. The least squares solver converges within 13 iterations and a total of 86 cost function evaluations. In every function evaluation, two adsorption isotherms with 21 pressures for methane and 20 pressures for CO_2 are calculated. Hence, 3526 equilibrium density profiles are computed within a little more than a minute.

A major contribution to the performance of FeO_s is the fast and exact (up to machine precision) calculation of partial derivatives using generalized (hyper-) dual numbers. In table 3 we show how different generalized (hyper-) dual numbers change the execution time of

the Helmholtz energy function which is at the heart of all property evaluations. We investigate the derivatives that are cached within a `State` object, i.e. the first partial derivative utilizing a dual number, second partial derivatives utilizing a hyper-dual number, and third derivatives utilizing a dual number with three non-real parts (dual3).

The computation times shown in table 3 demonstrate how automatic differentiation is faster than numerical differentiation using forward or central differences, which need multiple f64 evaluations of the Helmholtz energy. Also considering the added accuracy of dual numbers, numerical differentiation can be wholeheartedly disregarded as a serious alternative to (efficient) automatic differentiation. Compared to manual implementations of partial derivatives, generalized (hyper-) dual numbers provide an enormous reduction of the programming and testing effort. Finally, the resulting code is easily maintainable and readable, which puts the method ahead of the last alternative, i.e., partial derivatives generated by external code generation frameworks.

Table 3: Slowdown $\delta_D = t_D/t_{f64}$ of execution time of the Helmholtz energy function when evaluated with different dual numbers versus 64 bit floating-point numbers. Timings were obtained in Rust for the system methane/CO₂ at $T = 230$ K, $p = 40$ bar, $x_1 = 0.15$. The *result* column lists return values from a single call to the Helmholtz energy function.

data type	result	t	δ_D
f64	A	0.883 μ s	1.00
dual	$A, \left(\frac{\partial A}{\partial V}\right)_{\mathbf{N},T}$	1.064 μ s	1.20
dual2	$A, \left(\frac{\partial A}{\partial V}\right)_{\mathbf{N},T}, \left(\frac{\partial^2 A}{\partial V^2}\right)_{\mathbf{N},T}$	1.344 μ s	1.52
hyperdual	$A, \left(\frac{\partial A}{\partial V}\right)_{\mathbf{N},T}, \left(\frac{\partial A}{\partial T}\right)_{\mathbf{N},V}, \left(\frac{\partial^2 A}{\partial V \partial T}\right)_{\mathbf{N}}$	1.423 μ s	1.61
dual3	$A, \left(\frac{\partial A}{\partial V}\right)_{\mathbf{N},T}, \left(\frac{\partial^2 A}{\partial V^2}\right)_{\mathbf{N},T}, \left(\frac{\partial^3 A}{\partial V^3}\right)_{\mathbf{N},T}$	1.528 μ s	1.73

The computation speed also depends on the chosen model. To assess the influence of the different Helmholtz energy contributions of the PC-SAFT equation of state, table 4 shows the computation times for the evaluation of the first derivative of the Helmholtz energy for 16 binary mixtures. The ratio of computation time of each system compared to the most

simple mixture (hexane/heptane) is given in parentheses. The binary mixtures consist of molecules with different polarities and hence different Helmholtz energy contributions. For all molecules, the Helmholtz energy contains the hard-sphere, chain and dispersion contributions. The molecules can then be grouped in four categories: non-polar molecules (hexane and heptane; no additional contribution), dipolar molecules (acetone, dimethyl ether: additional dipole-dipole contribution), quadrupolar molecules (carbon dioxide, acetylene; additional quadrupole-quadrupole contribution), associating molecules (ethanol, 1-propanol; additional association contribution).

Table 4: Computation times for the evaluation of the first derivative of the residual PC-SAFT Helmholtz energy for 16 mixtures with different polarities. The values in parentheses denote the slowdown of the evaluation compared to the fastest, non-polar system of hexane and heptane.

	heptane	dimethyl ether	acetylene	1-propanol
hexane	0.726 μ s (1.00)	1.120 μ s (1.54)	1.125 μ s (1.55)	0.982 μ s (1.35)
acetone	1.138 μ s (1.57)	1.319 μ s (1.82)	2.011 μ s (2.77)	1.389 μ s (1.91)
carbon dioxide	1.130 μ s (1.56)	2.093 μ s (1.87)	1.372 μ s (1.89)	1.475 μ s (2.03)
ethanol	0.996 μ s (1.37)	1.527 μ s (2.10)	1.605 μ s (2.21)	5.055 μ s (6.97)

The results demonstrate how adding contributions to the model increases computation times. Due to efficient implementations of multi-sums, mixtures with a single polar component can be computed faster than those with two dipolar or two quadrupolar components. Mixtures with dipolar and quadrupolar components have particularly high computation times because not only have the dipole and quadrupole contribution be accounted for but also an extra contribution for dipole-quadrupole interactions introduced by Vrabec and Gross.⁵³ For single associating components analytic expressions for the fraction of non-bonded sites are implemented. Therefore, the increase in computation times is rather mild for systems with a single associating component. If two or more associating components are present in the system, the fraction of non-bonded sites is calculated iteratively using the algorithm by Michelsen,⁶⁶ thus leading to significantly higher computation times.

Conclusion

FeO_s is as a thermodynamics toolkit providing models and algorithms for equations of state and classical density functional theory. It simple to set up and provides a level of usability and a set of features that is unique in the thermodynamics community and thus it can be used to research and implement new equations of state and Helmholtz energy functionals, to build process models, or simply as a tool to compute thermodynamic properties. It is fully usable from Python and with its expressive interface and easy setup it is well suited for teaching.

Classical density functional theory, as a framework for predicting inhomogeneous systems (e.g. interfacial tensions or adsorption properties), requires a high initial programming effort, which currently limits the widespread use of DFT approaches. With FeO_s we aim to facilitate the use of DFT methods in research and engineering. FeO_s has implementations to solve 1-, 2-, and 3-dimensional DFT problems in Cartesian, cylindrical or spherical coordinate systems.

FeO_s is not only a toolkit; it is structured so that adding new models and algorithms is as convenient as possible. We hope that this open-source contribution lowers the barrier to use DFT in engineering applications and provides the tools necessary to allow for faster adoption of newly developed models into industry.

A recent survey of the ‘Working Party of Thermodynamics and Transport Properties of the European Federation of Chemical Engineering’³ identifies essential gaps and concerns for the field of applied thermodynamics: (1) Methods to assess the properties of fluids under confinement, at interfaces, and in the presence of external fields. (2) The ability to parameterize models and assess their uncertainties and range of applicability. (3) Transparent access to these models and parameters - ideally in a standardized form and including the data used for the parameterization. (4) Demand for ongoing education, training, and collaboration. This work shows that FeO_s addresses these four topics.

Acknowledgement

This work was funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy EXC 2075 – 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). PR acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 497566159.

Supporting Information Available

Jupyter notebooks of the example system used in the text that can be used to generate all plots. An additional notebook exemplifies how PC-SAFT parameters and correlation parameters for the viscosity (via entropy-scaling) can be adjusted to experimental data. We note that these notebooks are also part of examples hosted on GitHub. We update examples on GitHub when changes to interfaces are made so that notebooks hosted on GitHub should be preferred over this SI.

References

- (1) Kontogeorgis, G. M.; Folas, G. K. *Thermodynamic models for industrial applications: from classical and advanced mixing rules to association theories*; Wiley: Chichester, 2009.
- (2) Bortz, M.; Asprion, N. *Simulation and Optimization in Process Engineering: The Benefit of Mathematical Methods in Applications of the Chemical Industry*; Elsevier: Amsterdam, 2022.
- (3) de Hemptinne, J.-C.; Kontogeorgis, G. M.; Dohrn, R.; Economou, I. G.; ten Kate, A.; Kuitunen, S.; Fele Žilnik, L.; De Angelis, M. G.; Vesovic, V. A View on the Future

- of Applied Thermodynamics. *Industrial & Engineering Chemistry Research* **2022**, *61*, 14664–14680.
- (4) Van der Waals, J. D. Over de Continuïteit van den Gas-en Vloeistofoestand. Ph.D. thesis, University of Leiden, 1873.
 - (5) Redlich, Otto.; Kwong, J. N. S. On the Thermodynamics of Solutions. V. An Equation of State. Fugacities of Gaseous Solutions. *Chem. Rev.* **1949**, *44*, 233–244.
 - (6) Soave, G. Equilibrium constants from a modified Redlich-Kwong equation of state. *Chem. Eng. Sci.* **1972**, *27*, 1197–1203.
 - (7) Peng, D.-Y.; Robinson, D. B. A New Two-Constant Equation of State. *Ind. Eng. Chem. Fundam.* **1976**, *15*, 59–64.
 - (8) Hansen, J.-P., McDonald, I. R., Eds. *Theory of Simple Liquids (Fourth Edition)*, fourth edition ed.; Academic Press: Oxford, 2013.
 - (9) Wertheim, M. S. Fluids with highly directional attractive forces. I. Statistical thermodynamics. *J. Stat. Phys.* **1984**, *35*, 19–34.
 - (10) Wertheim, M. S. Fluids with highly directional attractive forces. II. Thermodynamic perturbation theory and integral equations. *J. Stat. Phys.* **1984**, *35*, 35–47.
 - (11) Wertheim, M. S. Fluids with highly directional attractive forces. III. Multiple attraction sites. *J. Stat. Phys.* **1986**, *42*, 459–476.
 - (12) Wertheim, M. S. Fluids with highly directional attractive forces. IV. Equilibrium polymerization. *J. Stat. Phys.* **1986**, *42*, 477–492.
 - (13) Chapman, W. G.; Gubbins, K. E.; Jackson, G.; Radosz, M. New reference equation of state for associating liquids. *Ind. Eng. Chem. Res.* **1990**, *29*, 1709–1721.

- (14) Gross, J.; Sadowski, G. Perturbed-Chain SAFT: An Equation of State Based on a Perturbation Theory for Chain Molecules. *Ind. Eng. Chem. Res.* **2001**, *40*, 1244–1260.
- (15) Gross, J.; Sadowski, G. Application of the Perturbed-Chain SAFT Equation of State to Associating Systems. *Ind. Eng. Chem. Res.* **2002**, *41*, 5510–5515.
- (16) Gross, J. An equation-of-state contribution for polar components: Quadrupolar molecules. *AIChE J.* **2005**, *51*, 2556–2568.
- (17) Gross, J.; Vrabec, J. An equation-of-state contribution for polar components: Dipolar molecules. *AIChE J.* **2006**, *52*, 1194–1204.
- (18) Lafitte, T.; Apostolakou, A.; Avendaño, C.; Galindo, A.; Adjiman, C. S.; Müller, E. A.; Jackson, G. Accurate statistical associating fluid theory for chain molecules formed from Mie segments. *J. Chem. Phys.* **2013**, *139*, 154504.
- (19) Blas, F. J.; Vega, L. F. Thermodynamic behaviour of homonuclear and heteronuclear Lennard-Jones chains with association sites from simulation and theory. *Mol. Phys.* **1997**, *92*, 135–150.
- (20) Llorell, F.; Pàmies, J. C.; Vega, L. F. Thermodynamic properties of Lennard-Jones chain molecules: Renormalization-group corrections to a modified statistical associating fluid theory. *J. Chem. Phys.* **2004**, *121*, 10715–10724.
- (21) Kontogeorgis, G. M.; Voutsas, E. C.; Yakoumis, I. V.; Tassios, D. P. An Equation of State for Associating Fluids. *Ind. Eng. Chem. Res.* **1996**, *35*, 4310–4318.
- (22) Kontogeorgis, G. M.; Michelsen, M. L.; Folas, G. K.; Derawi, S.; von Solms, N.; Stenby, E. H. Ten Years with the CPA (Cubic-Plus-Association) Equation of State. Part 1. Pure Compounds and Self-Associating Systems. *Ind. Eng. Chem. Res.* **2006**, *45*, 4855–4868.

- (23) Kontogeorgis, G. M.; Michelsen, M. L.; Folas, G. K.; Derawi, S.; von Solms, N.; Stenby, E. H. Ten Years with the CPA (Cubic-Plus-Association) Equation of State. Part 2. Cross-Associating and Multicomponent Systems. *Ind. Eng. Chem. Res.* **2006**, *45*, 4869–4878.
- (24) Papaioannou, V.; Lafitte, T.; Avendaño, C.; Adjiman, C. S.; Jackson, G.; Müller, E. A.; Galindo, A. Group contribution methodology based on the statistical associating fluid theory for heteronuclear molecules formed from Mie segments. *J. Chem. Phys.* **2014**, *140*, 054107.
- (25) Sauer, E.; Stavrou, M.; Gross, J. Comparison between a Homo- and a Heterosegmented Group Contribution Approach Based on the Perturbed-Chain Polar Statistical Associating Fluid Theory Equation of State. *Ind. Eng. Chem. Res.* **2014**, *53*, 14854–14864.
- (26) Wagner, W.; Pruß, A. The IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use. *J. Phys. Chem. Ref. Data* **2002**, *31*, 387–535.
- (27) Span, R.; Wagner, W. A New Equation of State for Carbon Dioxide Covering the Fluid Region from the Triple-Point Temperature to 1100 K at Pressures up to 800 MPa. *J. Phys. Chem. Ref. Data* **1996**, *25*, 1509–1596.
- (28) Span, R.; Lemmon, E. W.; Jacobsen, R. T.; Wagner, W.; Yokozeki, A. A Reference Equation of State for the Thermodynamic Properties of Nitrogen for Temperatures from 63.151 to 1000 K and Pressures to 2200 MPa. *J. Phys. Chem. Ref. Data* **2000**, *29*, 1361–1433.
- (29) Kunz, O.; Wagner, W. The GERG-2008 Wide-Range Equation of State for Natural Gases and Other Mixtures: An Expansion of GERG-2004. *J. Chem. Eng. Data* **2012**, *57*, 3032–3091.

- (30) Evans, R. The nature of the liquid-vapour interface and other topics in the statistical mechanics of non-uniform, classical fluids. *Adv. Phys.* **1979**, *28*, 143–200.
- (31) Sauer, E.; Gross, J. Classical Density Functional Theory for Liquid–Fluid Interfaces and Confined Systems: A Functional for the Perturbed-Chain Polar Statistical Associating Fluid Theory Equation of State. *Ind. Eng. Chem. Res.* **2017**, *56*, 4119–4135.
- (32) Sauer, E.; Terzis, A.; Theiss, M.; Weigand, B.; Gross, J. Prediction of Contact Angles and Density Profiles of Sessile Droplets Using Classical Density Functional Theory Based on the PCP-SAFT Equation of State. *Langmuir* **2018**, *34*, 12519–12531.
- (33) Sauer, E.; Gross, J. Prediction of Adsorption Isotherms and Selectivities: Comparison between Classical Density Functional Theory Based on the Perturbed-Chain Statistical Associating Fluid Theory Equation of State and Ideal Adsorbed Solution Theory. *Langmuir* **2019**, *35*, 11690–11701.
- (34) Kessler, C.; Eller, J.; Gross, J.; Hansen, N. Adsorption of light gases in covalent organic frameworks: comparison of classical density functional theory and grand canonical Monte Carlo simulations. *Microporous Mesoporous Mater.* **2021**, *324*, 111263.
- (35) Rehner, P.; Bursik, B.; Gross, J. Surfactant Modeling Using Classical Density Functional Theory and a Group Contribution PC-SAFT Approach. *Ind. Eng. Chem. Res.* **2021**, *60*, 7111–7123.
- (36) Bell, I. H.; Wronski, J.; Quoilin, S.; Lemort, V. Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp. *Ind. Eng. Chem. Res.* **2014**, *53*, 2498–2508.
- (37) Caleb Bell and Contributors, Thermo: Chemical properties component of Chemical Engineering Design Library (ChEDL). <https://github.com/CalebBell/thermo>, visited Jan 2023.

- (38) Hammer, M.; Aasen, A.; Wilhelmsen, Ø. Thermopack. <https://github.com/thermotools/thermopack>, visited Jan 2023.
- (39) Wilhelmsen, Ø.; Aasen, A.; Skaugen, G.; Aursand, P.; Austegard, A.; Aursand, E.; Gjennestad, M. Aa.; Lund, H.; Linga, G.; Hammer, M. Thermodynamic Modeling with Equations of State: Present Challenges with Established Methods. *Ind. Eng. Chem. Res.* **2017**, *56*, 3503–3515.
- (40) Chaparro, G.; Mejía, A. Phasepy: A Python based framework for fluid phase equilibria and interfacial properties computation. *J. Comput. Chem.* **2020**, *41*, 2504–2526.
- (41) Mejía, A.; Müller, E. A.; Chaparro Maldonado, G. SGTPy: A Python Code for Calculating the Interfacial Properties of Fluids Based on the Square Gradient Theory Using the SAFT-VR Mie Equation of State. *J. Chem. Inf. Model.* **2021**, *61*, 1244–1250.
- (42) Bell, I. H.; Deiters, U. K.; Leal, A. M. M. Implementing an Equation of State without Derivatives: teqp. *Ind. Eng. Chem. Res.* **2022**, *61*, 6010–6027.
- (43) Walker, P. J.; Yew, H.-W.; Riedemann, A. Clapeyron.jl: An Extensible, Open-Source Fluid Thermodynamics Toolkit. *Ind. Eng. Chem. Res.* **2022**, *61*, 7130–7153.
- (44) Kontogeorgis, G. M.; Dohrn, R.; Economou, I. G.; de Hemptinne, J.-C.; ten Kate, A.; Kuitunen, S.; Mooijer, M.; Žilnik, L. F.; Vesovic, V. Industrial Requirements for Thermodynamic and Transport Properties: 2020. *Industrial & Engineering Chemistry Research* **2021**, *60*, 4987–5013, PMID: 33840887.
- (45) Rehner, P.; Bauer, G. Application of Generalized (Hyper-) Dual Numbers in Equation of State Modeling. *Front. Chem. Eng.* **2021**, *3*.
- (46) PyO3 Project and Contributors, PyO3. <https://github.com/PyO3/pyo3>, visited Jan 2023.

- (47) Mairhofer, J.; Xiao, B.; Gross, J. A classical density functional theory for vapor-liquid interfaces consistent with the heterosegmented group-contribution perturbed-chain polar statistical associating fluid theory. *Fluid Phase Equilib.* **2018**, *472*, 117–127.
- (48) van Westen, T.; Gross, J. Accurate thermodynamics of simple fluids and chain fluids based on first-order perturbation theory and second virial coefficients: uv-theory. *J. Chem. Phys.* **2021**, *155*, 244501.
- (49) Heier, M.; Stephan, S.; Liu, J.; Chapman, W. G.; Hasse, H.; Langenbach, K. Equation of state for the Lennard-Jones truncated and shifted fluid with a cut-off radius of 2.5σ based on perturbation theory and its applications to interfacial thermodynamics. *Mol. Phys.* **2018**, *116*, 2083–2094.
- (50) Aasen, A.; Hammer, M.; Ervik, Å.; Müller, E. A.; Wilhelmsen, Ø. Equation of State and Force Fields for Feynman–Hibbs-corrected Mie Fluids. I. Application to Pure Helium, Neon, Hydrogen, and Deuterium. *Journal of Chemical Physics* **2019**, *151*, 064508.
- (51) Aasen, A.; Hammer, M.; Müller, E. A.; Wilhelmsen, Ø. Equation of State and Force Fields for Feynman–Hibbs-corrected Mie Fluids. II. Application to Mixtures of Helium, Neon, Hydrogen, and Deuterium. *Journal of Chemical Physics* **2020**, *152*, 074507.
- (52) Hammer, M.; Bauer, G.; Stierle, R.; Gross, J.; Wilhelmsen, Ø. Classical density functional theory for interfacial properties of hydrogen, helium, deuterium, neon and their mixtures. *Manuscript submitted for publication*.
- (53) Vrabec, J.; Gross, J. Vapor-Liquid Equilibria Simulation and an Equation of State Contribution for Dipole-Quadrupole Interactions. *J. Phys. Chem. B* **2008**, *112*, 51–60.
- (54) Lötgering-Lin, O.; Gross, J. Group Contribution Method for Viscosities Based on Entropy Scaling Using the Perturbed-Chain Polar Statistical Associating Fluid Theory. *Ind. Eng. Chem. Res.* **2015**, *54*, 7942–7952.

- (55) Lötgering-Lin, O.; Fischer, M.; Hopp, M.; Gross, J. Pure Substance and Mixture Viscosities Based on Entropy Scaling and an Analytic Equation of State. *Ind. Eng. Chem. Res.* **2018**, *57*, 4095–4114.
- (56) Hopp, M.; Mele, J.; Hellmann, R.; Gross, J. Thermal Conductivity via Entropy Scaling: An Approach That Captures the Effect of Intramolecular Degrees of Freedom. *Ind. Eng. Chem. Res.* **2019**, *58*, 18432–18438.
- (57) Hopp, M.; Mele, J.; Gross, J. Self-Diffusion Coefficients from Entropy Scaling Using the PCP-SAFT Equation of State. *Ind. Eng. Chem. Res.* **2018**, *57*, 12942–12950.
- (58) Webster, L. A.; Kidnay, A. J. Vapor-Liquid Equilibria for the Methane-Propane-Carbon Dioxide Systems at 230 K and 270 K. *J. Chem. Eng. Data* **2001**, *46*, 759–764.
- (59) Kestin, J.; Yata, J. Viscosity and Diffusion Coefficient of Six Binary Mixtures. *The Journal of Chemical Physics* **1968**, *49*, 4780–4791.
- (60) Kestin, J.; Ro, S. T. The Viscosity of Nine Binary and Two Ternary Mixtures of Gases at Low Density. *Berichte der Bunsengesellschaft für physikalische Chemie* **1974**, *78*, 20–24.
- (61) Ottiger, S.; Pini, R.; Storti, G.; Mazzotti, M. Competitive adsorption equilibria of CO₂ and CH₄ on a dry coal. *Adsorption* **2008**, *14*, 539–556.
- (62) Steele, W. A. The physical interaction of gases with crystalline solids: I. Gas-solid energies and properties of isolated adsorbed atoms. *Surf. Sci.* **1973**, *36*, 317–352.
- (63) Siderius, D. W.; Gelb, L. D. Extension of the Steele 10-4-3 potential for adsorption calculations in cylindrical, spherical, and other pore geometries. *J. Chem. Phys.* **2011**, *135*, 084703.
- (64) Mairhofer, J.; Gross, J. Numerical aspects of classical density functional theory for one-dimensional vapor-liquid interfaces. *Fluid Phase Equilib.* **2017**, *444*, 1–12.

- (65) Sears, M. P.; Frink, L. J. D. A new efficient method for density functional theory calculations of inhomogeneous fluids. *J. Comput. Phys.* **2003**, *190*, 184–200.
- (66) Michelsen, M. L. Robust and Efficient Solution Procedures for Association Models. *Ind. Eng. Chem. Res.* **2006**, *45*, 8449–8453.

