TECHNOLOGY AND CODE
published: 11 October 2021
doi: 10.3389/fceng.2021.758090

Application of Generalized (Hyper-) Dual Numbers in Equation of State Modeling

Philipp Rehner and Gernot Bauer*

Institute of Thermodynamics and Thermal Process Engineering, University of Stuttgart, Stuttgart, Germany

The calculation of derivatives is ubiquitous in science and engineering. In thermodynamics, in particular, state properties can be expressed as derivatives of thermodynamic potentials. The manual differentiation of complex models can be tedious and error-prone. In this work, we revisit dual and hyper-dual numbers for the calculation of *exact* derivatives and show generalizations to higher order derivatives and derivatives with respect to vector quantities. The methods described in this paper are accompanied by an open source Rust implementation with Python bindings. Applications of the generalized (hyper-) dual numbers are given in the context of equation of state modeling and the calculation of critical points.

Keywords: automatic differentiation, equation of state, thermodynamic models, data structure, numerical methods

OPEN ACCESS

Edited by:
José María Ponce-Ortega,
Michoacana University of San Nicolás de Hidalgo, Mexico

Reviewed by:
Mustafa Salti,
Mersin University, Turkey
Javier Tovar Facio,
National Institute of Ecology and Climate Change, Mexico
Fabricio Nápoles,
Michoacana University of San Nicolás de Hidalgo, Mexico
Luis Fernando Lira-Barragán,
Michoacana University of San Nicolás de Hidalgo, Mexico

****Correspondence:***
Philipp Rehner
rehner@itt.uni-stuttgart.de

Specialty section:
This article was submitted to Computational Methods in Chemical Engineering, a section of the journal Frontiers in Chemical Engineering

Received: *13 August 2021*
Accepted: *20 September 2021*
Published: *11 October 2021*

Citation:
Rehner P and Bauer G (2021) Application of Generalized (Hyper-) Dual Numbers in Equation of State Modeling. Front. Chem. Eng. 3:758090. doi: 10.3389/fceng.2021.758090

1 INTRODUCTION

The calculation of derivatives is required in many branches of physics and engineering. In particular, derivatives are required in solvers for nonlinear equations or optimization problems. Accurate calculations of derivatives are even more important in cases where the derivatives are part of the model itself, rather than just appearing in algorithms that often do not suffer much from appropriate approximations to the Jacobians. The ideal solution would be to not approximate at all and implement all required derivatives of the model by hand. This approach guarantees accurate results and efficiency. However, for complicated models, the probability of errors in the implementation becomes high. The process can be aided by symbolic math toolboxes and automatic code generation. An entirely different approach that requires no additional implementation within the model, is the use of numerical derivatives like forward or central differences, and higher order methods. The simplicity of the approach comes at the price, that the result is inherently an approximation and the achievable precision depends on the chosen step size. For small step sizes, the calculation can become unstable due to machine precision. For large step sizes, the calculation is stable but less accurate. To find the optimal step size for an arbitrary problem can be challenging.

Exact results without differentiation by hand can be obtained using automatic differentiation (Griewank and Walther, 2008). In this approach, the calculations in the model are represented by a calculation tree, that contains all intermediate results and their derivatives. The individual steps in this tree consist of algebraic operations or elementary functions for which the derivatives are known. Depending on the order with which the derivatives are calculated, the methods are referred to as forward or backward accumulation. The method of using (hyper-) dual numbers (Fike and Alonso, 2011) and the generalizations introduced in this work can be understood as an automatic differentiation using operator overloading and forward accumulation. Automatic differentiation with dual numbers is similar to the wider known complex step differentiation (Martins et al., 2003). However, aside from the availability of complex numbers in many programming languages, there is no advantage whatsoever to use complex step differentiation above dual numbers: complex step differentiation can be seen as an improved numerical derivative that alleviates the problems at small step sizes, whereas dual number differentiation always results in exact derivatives within machine precision. Automatic differentiation using (hyper-) dual numbers has been used in various fields of engineering and science such as solving optimal control problems (Agamawi and Rao, 2020) or formulating equations of motion for rigid- and multi-body systems (Cohen and Shoham, 2015; Cohen and Shoham, 2017). Yet, the concept of automatic differentiation with dual numbers is rarely used in chemical engineering and thermodynamics in particular, with the exception of the work of Diewald et al. (2018) and Rehner et al. (2019). With this review of the mathematical background and the openly available implementation in Rust and Python, we aim to change that.

This manuscript is structured as follows: in **section 2** the properties of dual and hyper-dual numbers are briefly revisited and a generalization to higher derivatives and vector valued gradients and Hessians is demonstrated. **Section 3** outlines the implementation in Rust and Python that is available on GitHub. **Section 4** shows applications for generalized (hyper-) dual numbers within thermodynamics and equation of state modeling. This is a particular interesting application for automatic differentiation as state properties can be identified as partial derivatives of thermodynamic potentials.

2 METHODS

This section aims to explain why dual and hyper-dual numbers are useful in the context of automatic differentiation by establishing an isomorphism between the number and derivative calculus. Isomorphism refers to the observation that all derivation rules have a counterpart in the arithmetic of the (hyper-) dual numbers. With this link established, it is shown how it can be reverted to obtain generalized (hyper-) dual numbers for the calculation of higher order and partial derivatives, gradients, Jacobians and Hessians.

2.1 Dual Numbers

Similar to complex numbers, dual numbers are formed by adjoining a new element ε with the property $\varepsilon^2 = 0$. A dual number x can thus be written in the form

$$x = x_0 + x_1\varepsilon, \qquad x_0, x_1 \in \mathbb{R} \tag{1}$$

The product of x with a dual number $y = y_0 + y_1\varepsilon$ is

$$xy = x_0 y_0 + \left(x_0 y_1 + x_1 y_0\right)\varepsilon. \tag{2}$$

For the application in thermodynamic modeling and other parts of physics, the most interesting property, however, is the *exact* Taylor expansion

$$f(x) = f(x_0) + f'(x_0)x_1\varepsilon. \tag{3}$$

All higher order terms cancel since they contain the factor $\varepsilon^2 = 0$. Therefore, the derivative of a (real) function $f(x)$ can be calculated exactly by evaluating the function using dual numbers and taking the ε-component f_1 of the result, as

$$f(x_0 + \varepsilon) = \underbrace{f(x_0)}_{f_0} + \underbrace{f'(x_0)}_{f_1}\varepsilon. \tag{4}$$

As opposed to numerical differentiation methods (forward, backward, or central differences), calculating the derivative with dual numbers does not introduce an error by truncating the Taylor expansion (the truncation is exact for dual numbers). Therefore the necessity to find an optimal step size is eliminated and we can simply choose $x_1 = 1$.

The multiplication, **Eq. 2**, of two dual numbers follows the same rules as the product rule in calculus and the Taylor expansion, **Eq. 3**, is analogous to the chain rule. With the chain rule, the product rule and the elementwise addition, the derivatives of arbitrarily complex functions are known. Therefore, dual numbers form an isomorphism to the first derivative, and the derivatives of any model, that is written using analytical functions can be calculated exactly using dual numbers. As an example, the division can be written as a product of the numerator with the divisor raised to the power of minus one:

$$\frac{x}{y} = xy^{-1} = (x_0 + x_1\varepsilon)\left(y_0^{-1} - y_0^{-2}y_1\varepsilon\right) = \frac{x_0}{y_0} + \frac{x_1 y_0 - x_0 y_1}{y_0^2}\varepsilon \tag{5}$$

The result of this composition is the quotient rule in calculus. Additional functions like exponentials

$$e^x = e^{x_0} + x_1 e^{x_0}\varepsilon \tag{6}$$

or logarithms

$$\ln x = \ln x_0 + \frac{x_1}{x_0}\varepsilon \tag{7}$$

follow from **Eq. 3**.

2.2 Hyper-Dual Numbers

As an extension to dual numbers, hyper-dual numbers were introduced by Fike and Alonso (2011). Hyper-dual numbers contain two extra dimensions ε_1 and ε_2 with the property $\varepsilon_1^2 = \varepsilon_2^2 = (\varepsilon_1\varepsilon_2)^2 = 0$. Thus a hyper dual number x can be written as

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + x_{12}\varepsilon_1\varepsilon_2, \qquad x_0, x_1, x_2, x_{12} \in \mathbb{R} \tag{8}$$

For a function $f(x, y)$, the exact Taylor expansion using hyper dual numbers is

$$\begin{aligned} f(x_0 + \varepsilon_1, y_0 + \varepsilon_2) = f(x_0, y_0) + f_x(x_0, y_0)\varepsilon_1 + f_y(x_0, y_0)\varepsilon_2 \\ + f_{xy}(x_0, y_0)\varepsilon_1\varepsilon_2. \end{aligned} \tag{9}$$

Therefore, hyper dual numbers can be used to calculate exact second partial derivatives $f_{xy}(x, y)$ by setting $x_1 = y_2 = 1$ and all other derivative parts to zero. The first partial derivatives $f_x(x, y)$ and $f_y(x, y)$ are always calculated simultaneously. The product of two hyper dual numbers x and y

$$\begin{aligned} xy = x_0 y_0 + \left(x_0 y_1 + x_1 y_0\right)\varepsilon_1 + \left(x_0 y_2 + x_2 y_0\right)\varepsilon_2 \\ + \left(x_0 y_{12} + x_1 y_2 + x_2 y_1 + x_{12} y_0\right)\varepsilon_1\varepsilon_2 \end{aligned} \tag{10}$$

again corresponds to the product rule for partial derivatives and the chain rule can be written as

$$f(x) = f(x_0) + f'(x_0)x_1\varepsilon_1 + f'(x_0)x_2\varepsilon_2 + \left(f''(x_0)x_1x_2 + f'(x_0)x_{12}\right)\varepsilon_1\varepsilon_2. \quad (11)$$

To calculate derivatives using dual or hyper-dual numbers, **Eqs. 3**, **11** are used to implement the required elementary functions (exponentials, powers, trigonometric functions, etc.) and **Eqs. 2**, **10** are used to overload the multiplication operator. Then, by the virtue of **Eqs. 4**, **9**, the derivatives of arbitrary functions are available by setting the first derivative part(s) of the input variable(s) to 1.

2.3 Generalizations

A detailed mathematical investigation of the properties of dual and hyper dual numbers is important in a general context. However, from a science and engineering point of view, the isomorphism between the number and properties of derivatives is all that matters. Dual numbers are usually presented as numbers with certain properties. Then, by identifying the isomorphism to the chain and product rules, the exact calculation of derivatives follows as an application. There is nothing stopping us from reversing the process: start with a certain application, like the calculation of third or higher order derivatives and generate a number that is isomorphic to the chain and product rules. In other words, we think about the number as data structures, that contain a function value and its corresponding derivatives simultaneously.

2.3.1 Higher Order Ordinary Derivatives

For ordinary derivatives of an arbitrary order all derivatives up to the highest order are required for intermediate steps. Therefore, a generalized dual number of order K can be written as

$$x = x_0 + \sum_{i=1}^{K} x_i \nu_i, \qquad \forall i:\ \ x_i \in \mathbb{R} \quad (12)$$

where the additional dimensions are referred to with the letter ν as opposed to ε to indicate, that they represent additional derivative orders instead of additional variables. The multiplication of two generalized dual numbers x and y can be written as

$$xy = x_0y_0 + \sum_{i=1}^{K}\sum_{j=0}^{i}\binom{i}{j}x_j y_{i-j}\nu_i \quad (13)$$

The general expression of the chain rule requires the application of Faà di Bruno's formula

$$f(x) = f(x_0) + \sum_{i=1}^{K}\sum \frac{i!}{k_1!\cdot\cdots\cdot k_i!} f^{(k_1+\cdots+k_i)}(x_0)\prod_{m=1}^{i}\left(\frac{x_m}{m!}\right)^{k_m}\nu_i \quad (14)$$

where the inner sum sums over all tuples of non-negative integers $(k_1, \ldots, k_i)$ with $\sum_{m=1}^{i} m k_m = i$. The general implementation is cumbersome and presumably slow. Therefore it is appropriate to implement these generalized dual numbers for a fixed value of K. For $K = 3$, the product rule

$$xy = x_0y_0 + (x_0y_1 + x_1y_0)\nu_1 + (x_0y_2 + 2x_1y_1 + x_2y_0)\nu_2 + (x_0y_3 + 3x_1y_2 + 3x_2y_1 + x_3y_0)\nu_3 \quad (15)$$

and the chain rule

$$f(x) = f(x_0) + f'(x_0)x_1\nu_1 + \left(f''(x_0)x_1^2 + f'(x_0)x_2\right)\nu_2 + \left(f'''(x_0)x_1^3 + 3f''(x_0)x_1x_2 + f'(x_0)x_3\right)\nu_3 \quad (16)$$

simplify accordingly. The i-th derivative ($i \leq K$) of a real function $f(x)$ can be identified as the ν_i component of f evaluated using generalized dual numbers.

$$f(x_0 + \nu_1) = f(x_0) + \sum_{i=1}^{K} f^{(i)}(x_0)\nu_i \quad (17)$$

2.3.2 Gradients and Jacobians

With marginal changes to the formulation, a vector dual number can be defined that is able to calculate the gradient of a function with respect to an arbitrary amount of variables. To do so, the scalar dimension ε is replaced with a vector $\boldsymbol{\varepsilon} = \begin{pmatrix}\varepsilon_1 & \ldots & \varepsilon_N\end{pmatrix}^\intercal$ with N the number of variables. The vector dual number x can then be written as

$$x = x_0 + \boldsymbol{x}_1^\intercal\boldsymbol{\varepsilon}, \qquad x_0 \in \mathbb{R},\ \boldsymbol{x}_1 \in \mathbb{R}^N. \quad (18)$$

The product rule

$$xy = x_0y_0 + (x_0\boldsymbol{y}_1 + y_0\boldsymbol{x}_1)^\intercal\boldsymbol{\varepsilon} \quad (19)$$

and the chain rule

$$f(x) = f(x_0) + f'(x_0)\boldsymbol{x}_1^\intercal\boldsymbol{\varepsilon} \quad (20)$$

are modified accordingly. The most relevant use case is that for a function $f(\boldsymbol{x})$ of N variables $\boldsymbol{x} = \begin{pmatrix}x_1 & \ldots & x_N\end{pmatrix}^\intercal$, $\boldsymbol{\varepsilon}$ can be chosen as an array of N elements. Then the gradient ∇f can be determined from the dual part of $f(\boldsymbol{x}_0 + \boldsymbol{\varepsilon})$:

$$f(\boldsymbol{x}_0 + \boldsymbol{\varepsilon}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^\intercal\boldsymbol{\varepsilon} \quad (21)$$

For array valued functions F, this generalizes to the Jacobian J, as

$$F(\boldsymbol{x}_0 + \boldsymbol{\varepsilon}) = F(\boldsymbol{x}_0) + J(\boldsymbol{x}_0)\boldsymbol{\varepsilon} \quad (22)$$

2.3.3 Vectorized Second Partial Derivatives and Hessians

Vector hyper dual numbers can also be defined and allow the calculation of second partial derivatives with respect to an arbitrary number of variables. Both $\boldsymbol{\varepsilon}_1$ and $\boldsymbol{\varepsilon}_2$ are vectors and thus the vector hyper dual number x in general can be written as

$$x = x_0 + \boldsymbol{x}_1^\intercal\boldsymbol{\varepsilon}_1 + \boldsymbol{x}_2^\intercal\boldsymbol{\varepsilon}_2 + \boldsymbol{\varepsilon}_1^\intercal\boldsymbol{x}_{12}\boldsymbol{\varepsilon}_2, \quad x_0 \in \mathbb{R},\ \boldsymbol{x}_1 \in \mathbb{R}^M,\ \boldsymbol{x}_2 \in \mathbb{R}^N,\ \boldsymbol{x}_{12} \in \mathbb{R}^{M\times N} \quad (23)$$

with the product rule

$$xy = x_0y_0 + (x_0\boldsymbol{y}_1 + y_0\boldsymbol{x}_1)^\intercal\boldsymbol{\varepsilon}_1 + (x_0\boldsymbol{y}_2 + y_0\boldsymbol{x}_2)^\intercal\boldsymbol{\varepsilon}_2 + \boldsymbol{\varepsilon}_1^\intercal(x_0\boldsymbol{y}_{12} + \boldsymbol{x}_1\boldsymbol{y}_2^\intercal + \boldsymbol{y}_1\boldsymbol{x}_2^\intercal + y_0\boldsymbol{x}_{12})\boldsymbol{\varepsilon}_2, \quad (24)$$

and the chain rule

$$f(x) = f(x_0) + f'(x_0)\boldsymbol{x}_1^\intercal\boldsymbol{\varepsilon}_1 + f'(x_0)\boldsymbol{x}_2^\intercal\boldsymbol{\varepsilon}_2 + \boldsymbol{\varepsilon}_1^\intercal(f''(x_0)\boldsymbol{x}_1\boldsymbol{x}_2^\intercal + f'(x_0)\boldsymbol{x}_{12})\boldsymbol{\varepsilon}_2 \quad (25)$$

To save computation time for the case of simple Hessians, only one of the first derivatives needs to be accounted for as they are identical. The vector hyper dual number can then be written as

$$x = x_0 + \boldsymbol{x}_1^\intercal\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^\intercal\boldsymbol{x}_{12}\boldsymbol{\varepsilon}, \qquad x_0 \in \mathbb{R},\ \boldsymbol{x}_1 \in \mathbb{R}^N,\ \boldsymbol{x}_{12} \in \mathbb{R}^{N\times N} \quad (26)$$

instead. The Hessian H (and the gradient that comes for free) of a scalar function $f(\boldsymbol{x})$ can then be calculated from

$$f(\boldsymbol{x}_0 + \boldsymbol{\varepsilon}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^\intercal\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^\intercal H(\boldsymbol{x}_0)\boldsymbol{\varepsilon}. \quad (27)$$

2.3.4 Higher Order Partial Derivatives

Higher order derivatives can be added by implementing new dual numbers with additional fields for the derivatives and extended sets of product and chain rules. However, by allowing the elements of any generalized dual number to be a generalized dual number themselves, higher order derivatives can be obtained without additional implementation work (Szirmay-Kalos, 2021). This becomes apparent, when a hyper dual number x is written as

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + x_{12}\varepsilon_1\varepsilon_2 = (x_0 + x_1\varepsilon_1) + (x_2 + x_{12}\varepsilon_1)\varepsilon_2 \quad (28)$$

showing that it can simply be represented as a dual number for which the coefficients are themselves dual numbers. By avoiding redundancies in the calculation, a dedicated implementation for hyper dual numbers can be expected to be faster than the recursive version. However, the recursive implementation offers great flexibility when it comes to higher order derivatives for which a dedicated implementation becomes tedious. An example for which the usage of recursive dual numbers is useful is given in **section 4.2**.

3 IMPLEMENTATION

Our implementation is done in the Rust programming language, an open source language that was released in its first stable version in 2015 and has been increasingly adapted and used since then. It is an attractive choice for a numerical library because it is a strongly typed, compiled language that generates efficient machine code while offering high-level language constructs (e.g. pattern matching), elegant error handling, allows for generic programming and has automatic code creation features (macros). A very valuable feature of the Rust compiler is the so-called borrow checker that prevents operations that would lead to undefined behavior (null or dangling pointers) or data races at compile time.

Besides the compiler, Rust ships with *Cargo*, a package manager that resolves dependencies of external packages, compiles the code and its dependencies (by calling the compiler, *rustc*), runs tests and benchmarks and builds the documentation. The bindings to Python are written in pure Rust using *PyO3*, a Rust package that allows to build native Python extension modules.

Dual numbers as presented in this work are implemented in Rust as structured types (*struct*s, similar to classes in other languages). Different from other object oriented languages, Rust provides inheritance and polymorphism in form of *trait*s, i.e. a collection of methods that extend the data type's functionality. Overloading of operators is realized by implementing the respective traits e.g. the *Add* trait for addition, *Sub* for subtraction, and so on.

Dual numbers are generic over their inner data types and the number of variables. The generic data type allows to specify the precision, but also to define recursive dual numbers. The number of variables is specified using Rusts *min_const_generics* feature. Therefore, the size of the arrays is known at compile time and scalar dual numbers can be obtained as a special case of the generic vector dual numbers without any overhead. Our library defines a trait, *DualNum*, which is implemented for all dual numbers as well as floating point numbers (*f64* and *f32*), and contains a number of useful traits for numerical operations (binary arithmetic operations, trigonometric functions, exponential and logarithm as well as spherical Bessel functions, comparison operators and so on). This trait can be used to write generic functions whose arguments and results can be any dual number types, and—since the compiler generates a version for each dual number type at compile time (a concept called monomorphization)—no checks at runtime are necessary. Since the *DualNum* trait is build upon the more generic *Num* trait (which is part of the Rust *num* package), dual numbers work seamlessly with other Rust packages that offer functions or structures parameterized over *Num*. For example, it is possible to write a generic function that uses the fast Fourier transform (using the *RustFFT* package) for which (partial) derivatives can be computed by simply calling the function with a dual number as argument instead of a floating point number.

Data types with generic parameters cannot be directly exposed to Python, since these parameters (and consequently the size of the data type) have to be known at compile time. Therefore, we offer several specific parameter combinations as Python classes, e.g. scalar dual and hyper dual numbers with 64 bit floating point numbers as inner data types. The Python data types work seamlessly with most of *numpy*'s math functions so that existing code rarely has to be modified. The source code for the Rust package including the Python bindings is available on GitHub under the name *num-dual*. The code is also published on the Rust package repository crates. io and the python packages for Windows, Linux and macOS are available from the Python Package Index (PyPI).

4 APPLICATIONS

The usage of the generalized (hyper-) dual numbers in the last section is by no means restricted to a specific field in science or engineering. However, in this work we want to focus on the application in equation of state modeling. Besides the documentation of the provided data types and functions, the github repository contains a comprehensive example which illustrates the methods and algorithms discussed below in form of a Jupyter notebook using the Peng-Robinson equation of state.

4.1 Calculation of Thermodynamic Properties

The calculation of state properties in thermodynamics is a particularly interesting application of generalized (hyper-) dual numbers because properties can be written as partial derivatives of a thermodynamic potential. In modern equations of state, this thermodynamic potential is usually the Helmholtz energy A with its characteristic variables temperature T, volume V, and number of particles of each species $\boldsymbol{n}$. In this and the subsequent sections bold symbols indicate arrays over all components. With first order partial derivatives, the entropy S, the pressure p, and the chemical potentials $\boldsymbol{\mu}$ can be obtained as

$$S = -\left(\frac{\partial A}{\partial T}\right)_{V,\boldsymbol{n}} \quad p = -\left(\frac{\partial A}{\partial V}\right)_{T,\boldsymbol{n}} \quad \boldsymbol{\mu} = \left(\frac{\partial A}{\partial \boldsymbol{n}}\right)_{T,V} \tag{29}$$

Using dual numbers, the scalar properties can be calculated as

$$A(T+\varepsilon, V, \boldsymbol{n}) = A - S\varepsilon \text{ and } A(T, V+\varepsilon, \boldsymbol{n}) = A - p\varepsilon \tag{30}$$

With vector dual numbers, the chemical potential can be calculated in a single function evaluation, as

$$A(T, V, \boldsymbol{n}+\boldsymbol{\varepsilon}) = A + \boldsymbol{\mu}^\intercal \boldsymbol{\varepsilon}. \tag{31}$$

With scalar dual numbers, the individual components have to be evaluated individually, as

$$A(T, V, n_i+\varepsilon, n_{j\neq i}) = A + \mu_i \varepsilon \tag{32}$$

For second partial derivatives hyper dual numbers are required. The derivatives with respect to scalar variables are obtained from

$$A(T+\varepsilon_1, V+\varepsilon_2, \boldsymbol{n}) = A - S\varepsilon_1 - p\varepsilon_2 - \left(\frac{\partial p}{\partial T}\right)_{V,\boldsymbol{n}} \varepsilon_1\varepsilon_2 \tag{33}$$

which demonstrates how the first partial derivatives for the variables associated with ε_1 and ε_2 are intrinsically calculated together with the second partial derivatives. This characteristic can be used to avoid redundancies in the calculation for specific property evaluations where both first and second order partial derivatives are required or more generally by caching all results of the property evaluation for a given (T, V, $\boldsymbol{n}$) state. Second partial derivatives with respect to the same variable can also be calculated using hyper dual numbers.

$$A(T+\varepsilon_1+\varepsilon_2, V, \boldsymbol{n}) = A - S\varepsilon_1 - S\varepsilon_2 - \left(\frac{\partial S}{\partial T}\right)_{V,\boldsymbol{n}} \varepsilon_1\varepsilon_2 \tag{34}$$

$$A(T, V+\varepsilon_1+\varepsilon_2, \boldsymbol{n}) = A - p\varepsilon_1 - p\varepsilon_2 - \left(\frac{\partial p}{\partial V}\right)_{T,\boldsymbol{n}} \varepsilon_1\varepsilon_2. \tag{35}$$

However, without any loss of information, the performance can be slightly improved by switching to second order dual numbers.

$$A(T+\nu_1, V, \boldsymbol{n}) = A - S\nu_1 - \left(\frac{\partial S}{\partial T}\right)_{V,\boldsymbol{n}} \nu_2 \tag{36}$$

$$A(T, V+\nu_1, \boldsymbol{n}) = A - p\nu_1 - \left(\frac{\partial p}{\partial V}\right)_{T,\boldsymbol{n}} \nu_2 \tag{37}$$

by avoiding the duplicate calculation of the same first derivative.

The partial derivative of the chemical potential with respect to mole numbers can be computed efficiently using vector hyper dual numbers in the Hessian form, **Eq. 26**,

$$A(T, V, \boldsymbol{n}+\boldsymbol{\varepsilon}) = A + \boldsymbol{\mu}^\intercal\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^\intercal\left(\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{n}}\right)_{T,V} \boldsymbol{\varepsilon} \tag{38}$$

or in the general form, **Eq. 23**.

$$A(T+\varepsilon_1, V, \boldsymbol{n}+\boldsymbol{\varepsilon}_2) = A - S\varepsilon_1 + \boldsymbol{\mu}^\intercal\boldsymbol{\varepsilon}_2 + \varepsilon_1\left(\frac{\partial \boldsymbol{\mu}}{\partial T}\right)_{V,\boldsymbol{n}} \boldsymbol{\varepsilon}_2 \tag{39}$$

$$A(T, V+\varepsilon_1, \boldsymbol{n}+\boldsymbol{\varepsilon}_2) = A - p\varepsilon_1 + \boldsymbol{\mu}^\intercal\boldsymbol{\varepsilon}_2 + \varepsilon_1\left(\frac{\partial \boldsymbol{\mu}}{\partial V}\right)_{T,\boldsymbol{n}} \boldsymbol{\varepsilon}_2 \tag{40}$$

where $M = 1$ in **Eq. 23** and thus ε_1 simplifies to a scalar.

4.2 Calculation of Critical Points

A commonly used algorithm for the calculation of critical points for a system with given mole fractions $\mathbf{z}$ was proposed by Heidemann and Khalil (1980) and refined by Michelsen and Mollerup (2004). The matrix $\boldsymbol{M}$ is defined as

$$M_{ij} = \sqrt{z_i z_j}\left(\frac{\partial^2 \beta A}{\partial n_i \partial n_j}\right)_{T,V} \tag{41}$$

with z_i the mole fraction of component i, $\beta = \frac{1}{k_\mathrm{B}T}$ the inverse temperature, and k_B the Boltzmann constant. Further, the variable s is introduced, that acts on the mole numbers $\mathbf{n}$ via

$$n_i = z_i + s u_i \sqrt{z_i} \tag{42}$$

with $\mathbf{u}$ the eigenvector corresponding to the smallest eigenvalue λ_1 of $\mathbf{M}$. Then, the two criticality conditions can be written as

$$\left.\frac{\partial^2 \beta A}{\partial s^2}\right|_{s=0} = \sum_i \sum_j u_i u_j M_{ij} = \lambda_1 = 0 \tag{43}$$

and

$$\left.\frac{\partial^3 \beta A}{\partial s^3}\right|_{s=0} = 0. \tag{44}$$

The second criticality condition is usually evaluated using a numerical derivation. This is particularly undesirable in this context, because the numerical error does appear in the residual function. Therefore it influences not only the convergence but the result itself.

The method can be enhanced using dual numbers. First the matrix $\boldsymbol{M}$ is evaluated using scalar or vector hyper-dual numbers as shown in **section 4.1**. The calculation of the smallest eigenvalue λ_1 and the corresponding eigenvector $\mathbf{u}$ is unaffected by the presence of dual numbers. The second criticality condition is calculated with one single evaluation of the Helmholtz energy using the third order dual numbers introduced in **section 2.3.1**. By setting

$$n_i = z_i + u_i\sqrt{z_i}v_1, \tag{45}$$

the v_3 component of the Helmholtz energy A (T, V, $\mathbf{n}$) is the second criticality condition.

The critical temperature and either density or volume are iterated using a Newton scheme. The Jacobian can be approximated numerically, however, if recursive dual numbers and the calculation of eigenvalues and eigenvectors for dual numbers are available, it can be evaluated exactly. The temperature and volume are set to

$$T^{\mathrm{dual}} = T + \begin{pmatrix}1 & 0\end{pmatrix}\boldsymbol{\varepsilon},\ V^{\mathrm{dual}} = V + \begin{pmatrix}0 & 1\end{pmatrix}\boldsymbol{\varepsilon}. \tag{46}$$

The entries of $\mathbf{M}$ (themselves dual numbers) are calculated by setting

$$n_k^{\mathrm{hd-dual}} = z_k + \delta_{ik}\varepsilon_1 + \delta_{jk}\varepsilon_2 \tag{47}$$

with the Kronecker delta δ_{ij} and evaluating the Helmholtz energy

$$M_{ij}^{\mathrm{dual}} = \sqrt{z_i z_j}A\left(T^{\mathrm{dual}}, V^{\mathrm{dual}}, \mathbf{n}^{\mathrm{hd-dual}}\right)_{12} \tag{48}$$

The eigenvalue $\lambda_1^{\mathrm{dual}}$ and the corresponding eigenvector $\mathbf{u}^{\mathrm{dual}}$ are calculated as shown in **Supplementary Material**. Finally, the second criticality condition is determined by setting

$$n_i^{\mathrm{d3-dual}} = z_i + u_i^{\mathrm{dual}}\sqrt{z_i}v_1 \tag{49}$$

and evaluating $A(T^{\mathrm{dual}}, V^{\mathrm{dual}}, \mathbf{n}^{\mathrm{d3-dual}})_3$. The Jacobian is now available from the gradient parts of the two criticality conditions. Here it was assumed that operators are available for all combinations of dual and recursive hyper-dual numbers. In practice this poses difficulties with respect to the implementations and is therefore likely not the case. Then, T^{dual} and V^{dual} can simply be lifted to the higher order dual number by setting all derivative parts to zero.

4.3 Cross Association

In equations of state that are based on thermodynamic perturbation theory by Wertheim (1984a), Wertheim (1984b), like the statistical associating fluid theory (SAFT) family (Chapman et al., 1989; Gross and Sadowski, 2001; Lafitte et al., 2013) or the cubic plus association (CPA) equation of state (Kontogeorgis et al., 2006), association contributions are used to model highly directional short range interactions like hydrogen bonds. The Helmholtz energy contribution for the cross association is given by

$$\beta A^{\mathrm{assoc}} = \sum_i n_i \sum_{A_i} N_{A_i}\left(\ln\chi^{A_i} - \frac{\chi^{A_i}}{2} + \frac{1}{2}\right) \tag{50}$$

with N_{A_i} the number of association sites of kind A on molecule i and χ^{A_i} the corresponding fraction of non-bonded sites. What makes this contribution relevant in the context of generalized (hyper-) dual numbers is, that the fraction of non-bonded sites χ^{A_i} is determined by the implicit set of equations

$$\chi^{A_i} = \left(1 + \sum_j \rho_j \sum_{B_j} N_{B_j}\chi^{B_j}\Delta^{A_iB_j}\right)^{-1} \tag{51}$$

The exact expression for the association strength $\Delta^{A_iB_j}$ differs between different equations of state, but the equations shown here are valid for all variants. To be able to automatically determine partial derivatives of the Helmholtz energy, the partial derivatives of the monomer fractions χ^{A_i} are also required. With an appropriate iterative method, these derivatives can be obtained with hardly any additional implementations using generalized (hyper-) dual numbers.

The state of the art iteration method for the monomer fraction was once again presented by Michelsen (2006). Applied to the notation used above, the problem is reformulated as a minimization of the property

$$\begin{aligned} Q = &\sum_i \rho_i \sum_{A_i} N_{A_i}\left(\ln\chi^{A_i} - \chi^{A_i} + 1\right) \\ &- \frac{1}{2}\sum_i \sum_j \rho_i\rho_j \sum_{A_i}\sum_{B_j} N_{A_i}N_{B_j}\chi^{A_i}\chi^{B_j}\Delta^{A_iB_j} \end{aligned} \tag{52}$$

with respect to the variables χ^{A_i}. The minimum of Q is obtained when the gradients

$$g_{A_i} = \frac{\partial Q}{\partial\chi^{A_i}} = \rho_i N_{A_i}\left(\frac{1}{\chi^{A_i}} - 1 - \sum_j \rho_j \sum_{B_j} N_{B_j}\chi^{B_j}\Delta^{A_iB_j}\right) \tag{53}$$

vanish. The stationary points of **Eq. 53** coincide with the solutions of **Eq. 51**. The solution is found by using the Newton iteration scheme

$$\boldsymbol{\chi}^{(k+1)} = \boldsymbol{\chi}^{(k)} + \Delta\boldsymbol{\chi}, \qquad \hat{\boldsymbol{H}}\Delta\boldsymbol{\chi} + \boldsymbol{g} = 0 \tag{54}$$

with the modified Hessian $\hat{\boldsymbol{H}}$ with entries

$$\begin{aligned} \hat{H}_{A_iB_j} = &-\frac{\rho_i N_{A_i}\delta_{A_iB_j}}{\chi^{A_i}}\left(1 + \sum_k \rho_k \sum_{C_k} N_{C_k}\chi^{C_k}\Delta^{A_iC_k}\right) \\ &- \rho_i\rho_j N_{A_i}N_{B_j}\Delta^{A_iB_j} \end{aligned} \tag{55}$$

The modification of the Hessian ensures that it is negative definite. As shown in **Supplementary Material**, when **Eq. 54** is iterated from some initial value using generalized (hyper-) dual numbers, the derivatives are calculated automatically. To speed up the computation, it is advisable to first solve the nonlinear equation for the real part. Then, as the equations for the derivatives are linear, the resulting χ^{A_i} can be lifted to the relevant dual number and as many steps of **Eq. 54** need to be applied as there are derivatives, i.e. one for dual numbers, two for hyper-dual numbers and three for third order dual numbers. With the derivatives of the monomer fractions in place, the Helmholtz energy contribution and its derivatives are available from **Eq. 50**.

5 DISCUSSION

Generalized (hyper-) dual numbers enable the calculation of exact derivatives of scalar and vector valued functions which is especially valuable when derivatives are part of the (physical) model. Derivatives no longer need to be implemented or approximated which leads to less error prone and faster development. Depending on the model, an implementation using dual numbers can be more costly to evaluate compared to hand-written derivatives. However, in the context of thermodynamic equations of state shown in this work, using the right type of dual number in combination with caching prior results, this disadvantage can be compensated.

Thermodynamic equations of state greatly benefit from generalized (hyper-) dual numbers because thermodynamic properties are computed from (partial) derivatives of a single function, the thermodynamic potential. We showed above that generalized (hyper-) dual numbers can be used to efficiently compute these properties even for complex models that contain implicit expressions like the association contribution in SAFT. In some cases, like the calculation of critical points, the algorithms themselves can be simplified using generalized (hyper-) dual numbers. Because only a single model equation has to be implemented (the Helmholtz energy), separation of model agnostic algorithms, like phase equilibria and stability analysis, and the model equation is simple and leads to more maintainable and extensible code. We believe that from a research point of view this enables faster and easier development of new models and algorithms, and consequently, it will enable faster adaption and transfer to industry.

DATA AVAILABILITY STATEMENT

The source code developed for this study can be found on GitHub (https://github.com/itt-ustutt/num-dual) and crates.io (https://crates.io/crates/num-dual). Compiled Python packages for Windows, Linux and macOS are available from PyPI (https://pypi.org/project/num_dual).

AUTHOR CONTRIBUTIONS

PR and GB have contributed equally to design and implementation of the num-dual Rust and Python libraries as well as the scientific research and the documentation process.

ACKNOWLEDGMENTS

We thank Joachim Gross for his continued support and the opportunity to pursue this project, and Rolf Stierle and Elmar Sauer for their valuable input on the manuscript.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fceng.2021.758090/full#supplementary-material

REFERENCES

Agamawi, Y. M., and Rao, A. V. (2020). Cgpops. *ACM Trans. Math. Softw.* 46, 1–38. doi:10.1145/3390463

Bartholomew-Biggs, M. C. (1998). Using Forward Accumulation for Automatic Differentiation of Implicitly-Defined Functions. *Comput. Optimization Appl.* 9, 65–84. doi:10.1023/A:1018382103801

Chapman, W. G., Gubbins, K. E., Jackson, G., and Radosz, M. (1989). Saft: Equation-Of-State Solution Model for Associating Fluids. *Fluid Phase Equilibria* 52, 31–38. doi:10.1016/0378-3812(89)80308-5

Cohen, A., and Shoham, M. (2015). Application of Hyper-Dual Numbers to Multibody Kinematics. *J. Mech. Robotics* 8, 011015. doi:10.1115/1.4030588

Cohen, A., and Shoham, M. (2017). Application of Hyper-Dual Numbers to Rigid Bodies Equations of Motion. *Mechanism Machine Theor.* 111, 76–84. doi:10.1016/j.mechmachtheory.2017.01.013

Diewald, F., Heier, M., Horsch, M., Kuhn, C., Langenbach, K., Hasse, H., et al. (2018). Three-dimensional Phase Field Modeling of Inhomogeneous Gas-Liquid Systems Using the Pets Equation of State. *J. Chem. Phys.* 149, 064701. doi:10.1063/1.5035495

Fike, J., and Alonso, J. (2011). "The Development of Hyper-Dual Numbers for Exact Second-Derivative Calculations," in 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Orlando, Fl, USA, January 7, 2011. (Reston, VA: AIAA), 886. doi:10.2514/6.2011-886

Griewank, A., and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Chennai, TN: SIAM.

Gross, J., and Sadowski, G. (2001). Perturbed-chain Saft: An Equation of State Based on a Perturbation Theory for Chain Molecules. *Ind. Eng. Chem. Res.* 40, 1244–1260. doi:10.1021/ie0003887

Heidemann, R. A., and Khalil, A. M. (1980). The Calculation of Critical Points. *Aiche J.* 26, 769–779. doi:10.1002/aic.690260510

Kontogeorgis, G. M., Michelsen, M. L., Folas, G. K., Derawi, S., von Solms, N., and Stenby, E. H. (2006). Ten Years with the Cpa (Cubic-plus-association) Equation of State. Part 1. Pure Compounds and Self-Associating Systems. *Ind. Eng. Chem. Res.* 45, 4855–4868. doi:10.1021/ie051305v

Lafitte, T., Apostolakou, A., Avendaño, C., Galindo, A., Adjiman, C. S., Müller, E. A., et al. (2013). Accurate Statistical Associating Fluid Theory for Chain Molecules Formed from Mie Segments. *J. Chem. Phys.* 139, 154504. doi:10.1063/1.4819786

Martins, J. R. R. A., Sturdza, P., and Alonso, J. J. (2003). The Complex-step Derivative Approximation. *ACM Trans. Math. Softw.* 29, 245–262. doi:10.1145/838250.838251

Michelsen, M. L. (2006). Robust and Efficient Solution Procedures for Association Models. *Ind. Eng. Chem. Res.* 45, 8449–8453. doi:10.1021/ie060029x

Michelsen, M., and Mollerup, J. (2004). *Thermodynamic Modelling: Fundamentals and Computational Aspects*. Holte, Denmark: Tie-Line Publications.

Rehner, P., Aasen, A., and Wilhelmsen, Ø. (2019). Tolman Lengths and Rigidity Constants from Free-Energy Functionals-General Expressions and Comparison of Theories. *J. Chem. Phys.* 151, 244710. doi:10.1063/1.5135288

Szirmay-Kalos, L. (2021). Higher Order Automatic Differentiation with Dual Numbers. *Period. Polytech. Elec. Eng. Comp. Sci.* 65, 1–10. doi:10.3311/PPee.16341

Wertheim, M. S. (1984b). Fluids with Highly Directional Attractive Forces. Ii. Thermodynamic Perturbation Theory and Integral Equations. *J. Stat. Phys.* 35, 35–47. doi:10.1007/BF01017363

Wertheim, M. S. (1984a). Fluids with Highly Directional Attractive Forces. I. Statistical Thermodynamics. *J. Stat. Phys.* 35, 19–34. doi:10.1007/BF01017362

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.