# Hafnium Labs

## Predicting Chemistry

September 2022

# Agenda

Intro to Hafnium Labs and Q-props

Examples:

- High-fidelity dynamic simulation of $CO_2$ + impurities
- Reactive electrolyte systems with amines + $CO_2/H_2S$
- Beyond state-of-art thermodynamics – polar PC-SAFT for Benzene-Cyclohexane

# What we do

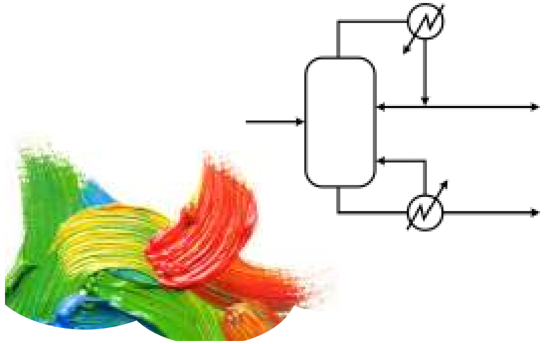
Hafnium Labs solves one of the hardest challenges in chemical R&D:
Obtaining ***reliable*** physical property data ***fast***

Our **Q-props** software sets a gold standard for property modeling to help digitalize R&D and enable:

**BETTER AND SAFER DESIGN**

**FASTER DEVELOPMENT**

**BROADER EXPLORATION**



# Our mission

*Establish a gold standard for obtaining physical properties of molecules and mixtures*

# Background and approach

Founded in 2016

- Industry need: **Digitalization of chemistry** requires reliable physical properties – often as critical input to modeling tools
    - **Customer industries**: Energy, chemicals, consumer goods, pharma, mining, and engineering
- Our approach: First tool to take a **universal and continuously improving approach**, providing a one-stop solution

Working on a universal solution, we put **more resource into physical properties** than any individual projects (or most companies) can justify, with >€3M already invested in R&D

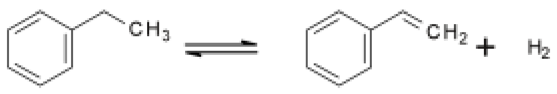
We work closely with customers to define good proof-of-concept projects, after which broader deployment can be planned

# Reliable digital designs require reliable physical properties

## Example: Influence of physical properties on process simulation results

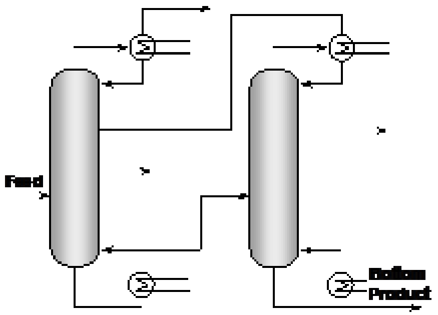
### A simple problem?

Styrene is separated from ethylbenzene by distillation:



*Ethylbenzene*
*$T_{boil}$ = 136°C*

*Styrene*
*$T_{boil}$ = 145°C*



### 3 different simulators give vastly different separations

Bottom styrene concentration:

- Simulator 1: **90%**
- Simulator 2: **81%**
- Simulator 3: **71%**

Same mathematical models but <u>each simulator uses different physical property data</u>

**→ Wrong physical properties can ruin a digital design**

### The problems go way beyond simple examples

Little/no data for green chemicals

Adding new compounds and data is time-consuming and error-prone

Lack reliable uncertainties to rationalize design factors

Solids, electrolytes and reactions are often neglected – but cause costly failures, e.g. corrosion

*Example from R. Dohrn, O. Pfohl, Fluid Phase Equilibria,* ***2002****, 194-197, 15-29*

# Q-props: One-stop solution for all physical property needs

| | | **Product** | **Service** |
|---|---|---|---|
| **Predict** | • Predict pure and pseudo-pure compound properties | ✓ | ✓ |
| **Screen** | • Identify compounds or mixtures with specific set of properties | ✓ | ✓ |
| **Property Package** | • Get reliable physical property package for any system<br>• Integrate in process simulator | ✓ | |
| **Model** | • Test, tune, and validate thermodynamic models<br>• Check reliability of all streams of a flowsheet | ✓ | ✓ |
| **UnitOps** | • Combine any offering with unit operation modeling<br>• Analyze effect of uncertainties and integrate in process simulator | ✓ | |

# Q-props integrates with process simulation tools end-to-end

0. Import

1. Get all data in one place

2. Create project DBs of best possible property data *with* uncertainties

3. Identify best model and understand its uncertainties

4. Export or improve

5. Run simulations

**Import from process simulation:**

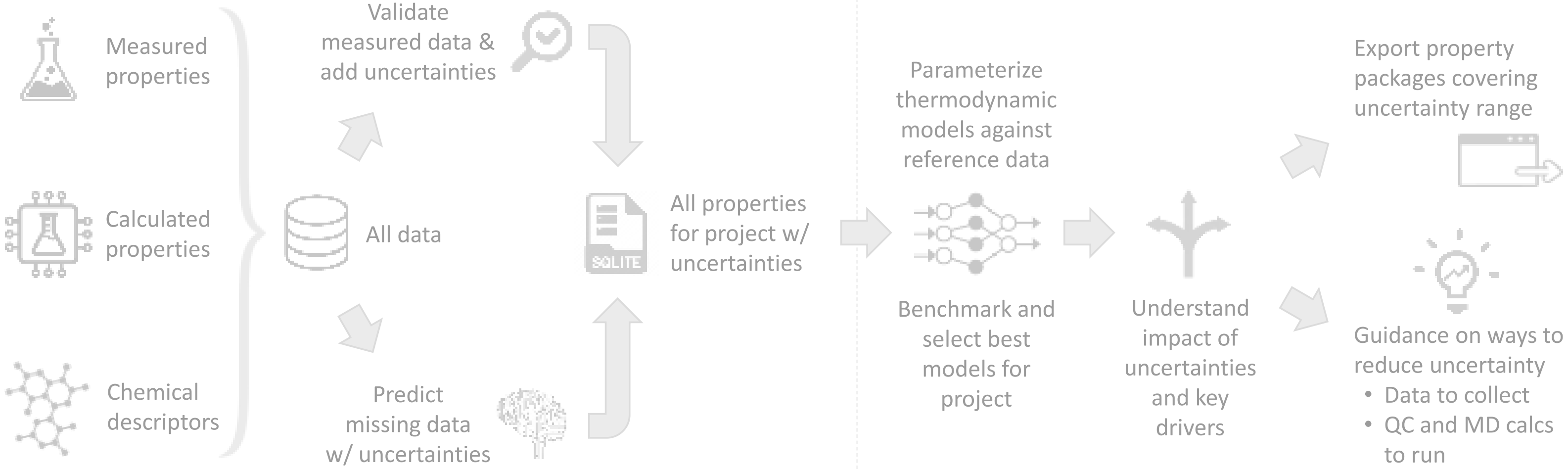
1. Compounds

2. Conditions

3. Simulator physical property predictions (for comparison inside Q-props)

**Input loaded into Q-props**

→ Q-props runs steps 1-4 automatically



Measured properties

Calculated properties

Chemical descriptors

Validate measured data & add uncertainties

All data

Predict missing data w/ uncertainties

All properties for project w/ uncertainties

Parameterize thermodynamic models against reference data

Benchmark and select best models for project

Understand impact of uncertainties and key drivers

Export property packages covering uncertainty range

Guidance on ways to reduce uncertainty
- Data to collect
- QC and MD calcs to run

***'Infinite' database of reliable reference properties***

*(pure compound and binary/ternary mixture data)*

***Best property model for mixture and impact of uncertainties***

Q-props

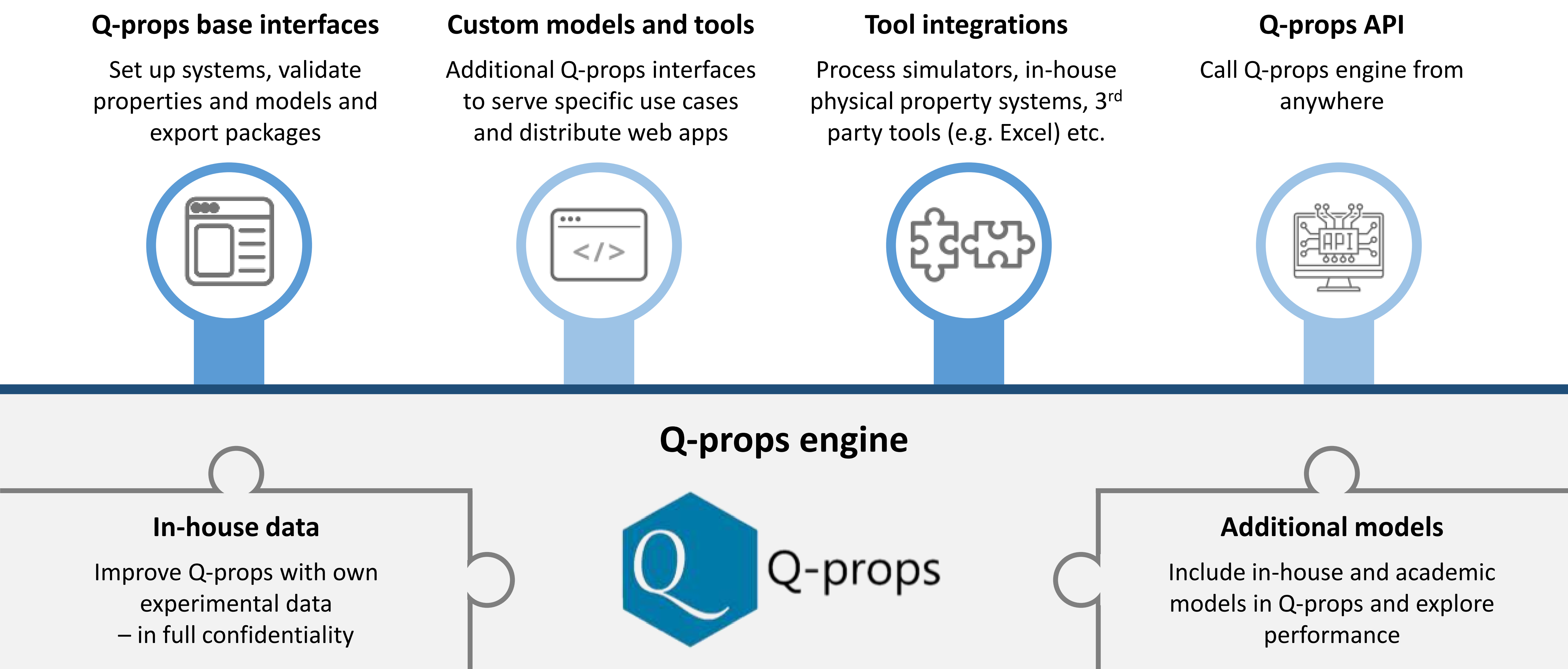
**Run scenarios**

Exported packages can be loaded into process simulator to assess impact of physical property uncertainties on process simulation results

**Replace property engine**

Packages from Q-props can be used for single streams or entire flowsheet

# Q-props is built for extensions and integrations



### Q-props base interfaces

Set up systems, validate properties and models and export packages

### Custom models and tools

Additional Q-props interfaces to serve specific use cases and distribute web apps

### Tool integrations

Process simulators, in-house physical property systems, 3rd party tools (e.g. Excel) etc.

### Q-props API

Call Q-props engine from anywhere

## Q-props engine

### In-house data

Improve Q-props with own experimental data
– in full confidentiality

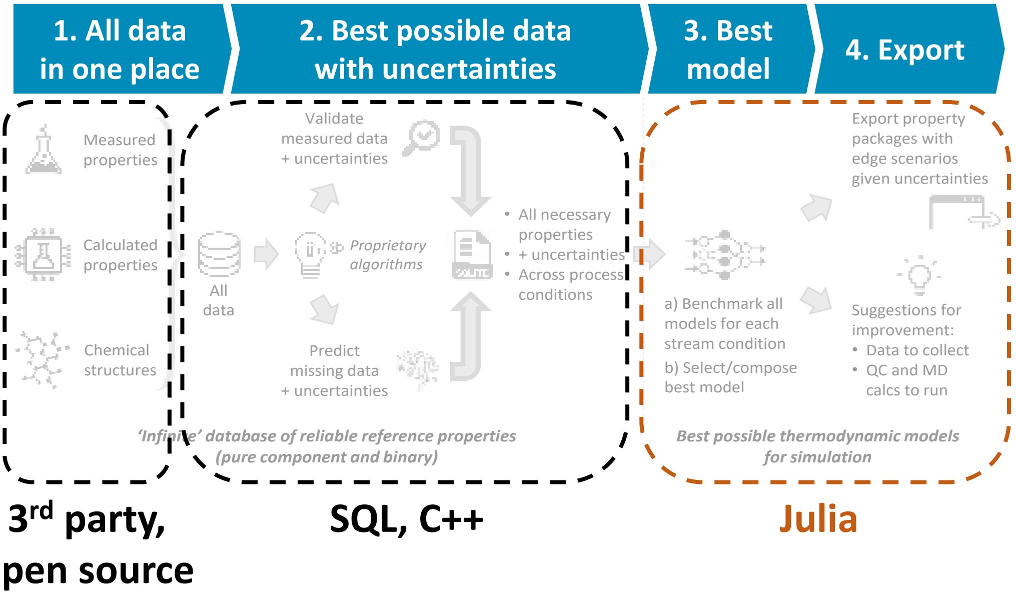
### Additional models

Include in-house and academic models in Q-props and explore performance

# Embedding Q-props through CAPE-OPEN

# We have built core Q-props components using the Julia programming language

## Julia is core to Q-props modeling engine



1. All data in one place

2. Best possible data with uncertainties

3. Best model

4. Export

Measured properties

Calculated properties

Chemical structures

All data

Validate measured data + uncertainties

*Proprietary algorithms*

Predict missing data + uncertainties

- All necessary properties
- + uncertainties
- Across process conditions

*'Infinite' database of reliable reference properties (pure component and binary)*

a) Benchmark all models for each stream condition

b) Select/compose best model

Export property packages with edge scenarios given uncertainties

Suggestions for improvement:

- Data to collect
- QC and MD calcs to run

*Best possible thermodynamic models for simulation*

**3rd party, open source**

**SQL, C++**

**Julia**

**Other languages**

- C#/.NET Core for ETL workflows
- Python for automation and Jupyter notebooks
- HTML/Javascript for frontends (Jupyter)

## Julia goodies for developers

**Julia can be used for both prototypes and production**
→ Viable alternative to FORTRAN and C++ but also Python and MATLAB

**Built-in package-manager for dependency management in large projects**

- Eases maintenance and extensibility
- Internal package repository

**Julia is stable (currently at version 1.8)**

- More than 8000 packages available on Github

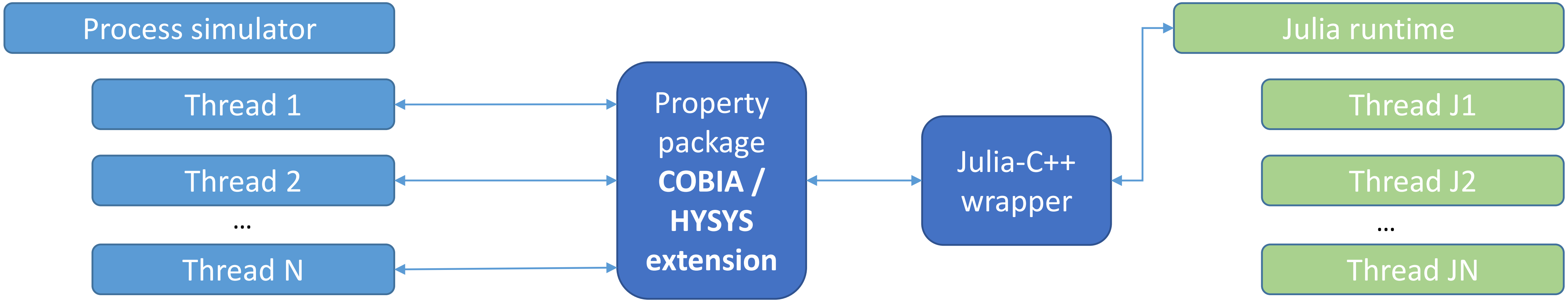
**Nice features for thermodynamics modeling**

- Support for unicode names enables standard symbols in applied thermodynamics, e.g. β, γ, ω, σ, ϕ, Γ
- Excellent support for unit of measurement, automatic/algorithmic differentiation etc.

# Integrating Julia with CAPE-OPEN or Native HYSYS Thermo

We used COBIA v/1.0 to implement unit operation and property package wrappers (v 1.1/1.2)

- COBIA acts as a wrapper for COM (COMBIA).
- Julia C API **must** be called from a single thread
    - C++ acts as wrapper for synchronizing calls to Julia via std::future
    - Julia 1.9 will allow full multi-threading support



Things to watch out for:

- Julia uses UTF-8 but CAPE-OPEN uses wide strings (16-bit)
- Use e.g. std::wstring_convert<std::codecvt_utf8_utf16<char16_t>>

# Examples: Simulation of pure fluids w/ impurities

# New processes are pushing existing tools to the limit: Accurate properties for Brayton cycle w/ supercritical CO2

### Many potential applications

- Concentrating solar power (CSP)
- Waste heat recovery
- Geothermal
- CO2 Sequestration
- …

### Brayton cycle w/ $sCO_2$

- Single-phase fluid (> 31 $^{o}$C)
- Temperatures up to 1000 $^{o}$C and pressures up to 35MPa
- Advanced configurations may lead to smaller turbomachinery than steam (up to 20x smaller)

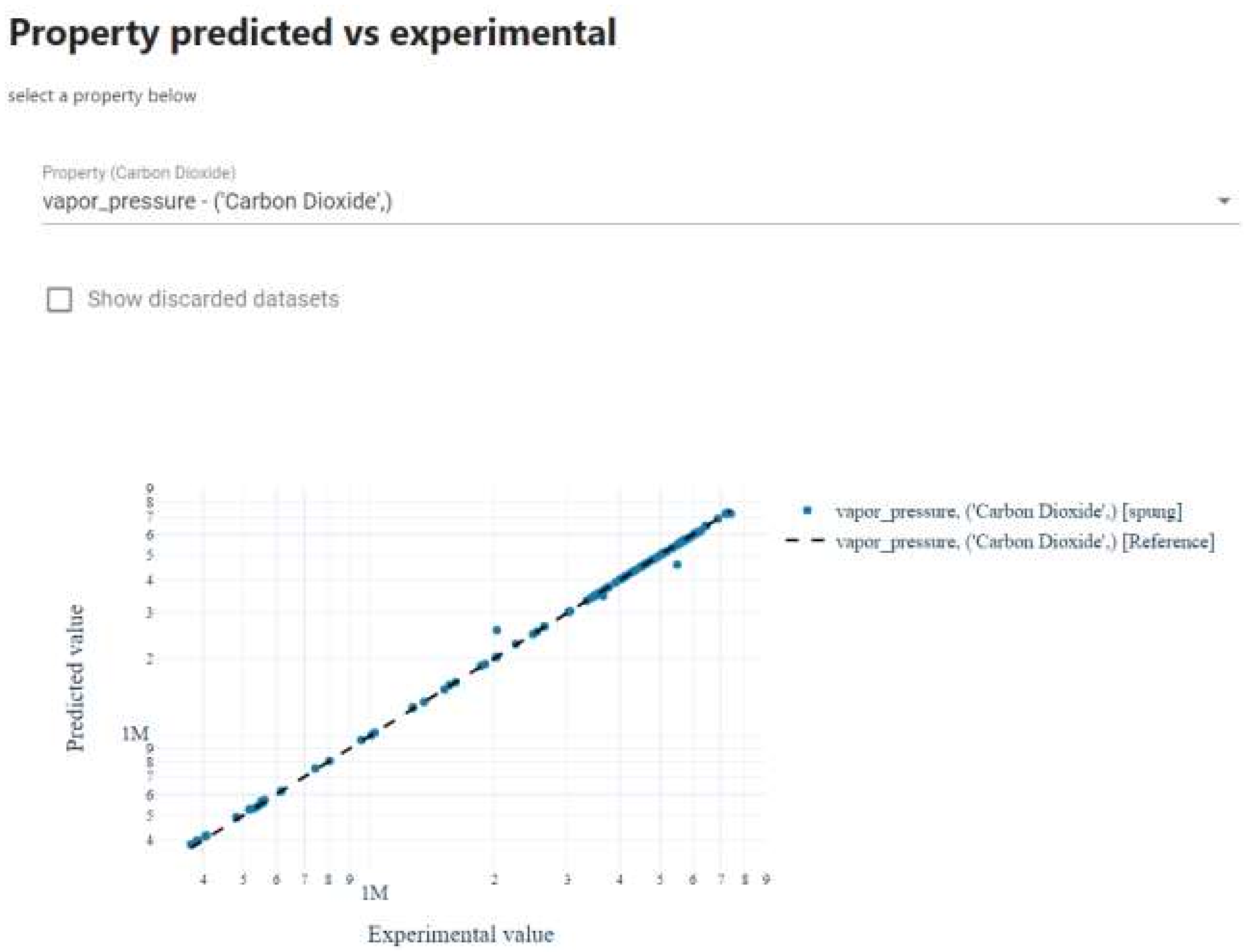
### Challenges existing tools

- Cubic EoS or MBWR cannot consistently represent properties over such wide temperature/pressure ranges
- Span-Wagner EoS (REFPROP) is suitable, but about an order of magnitude slower than cubic EoS
- Instabilities in flash are observed with existing commercial tools leading sometimes to slowdown

### A Q-props Model was set up

- Validation of Q-props for properties of pure $CO_2$ against experimental data
- Extension to mixtures through SPUNG principle
- (Demo)

# Validation examples for CO2



## Property predicted vs experimental

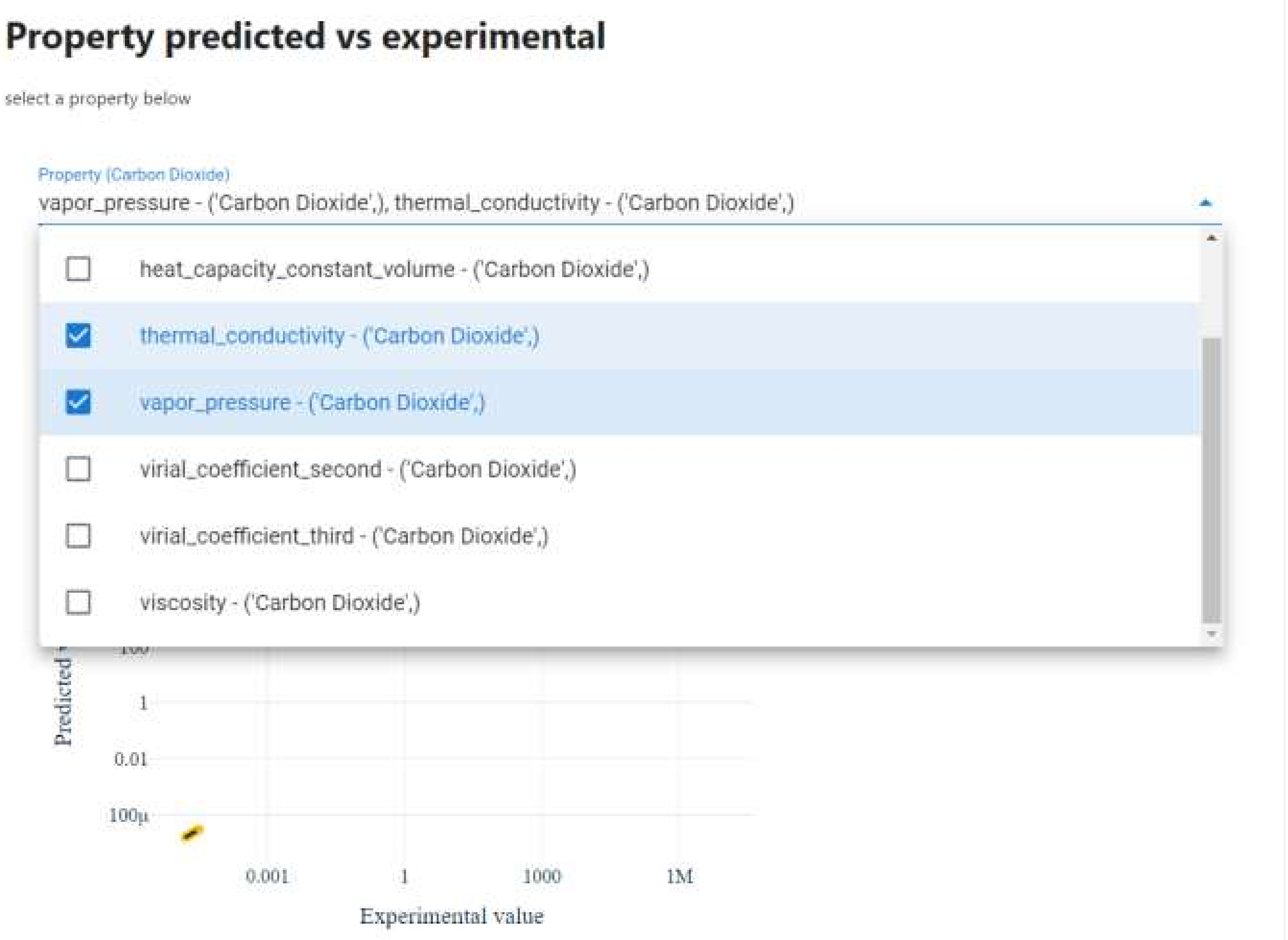
select a property below

Property (Carbon Dioxide)
vapor_pressure - ('Carbon Dioxide',)

□ Show discarded datasets

Predicted value
9
8
7
6
5
4
3
2
1M
9
8
7
6
5
4
4
5
6
7
8
9
1M
2
3
4
5
6
7
8
9
Experimental value
vapor_pressure, ('Carbon Dioxide',) [spung]
vapor_pressure, ('Carbon Dioxide',) [Reference]

# Validation examples for CO2



## Property predicted vs experimental

select a property below

Property (Carbon Dioxide)

vapor_pressure - ('Carbon Dioxide',), thermal_conductivity - ('Carbon Dioxide',)

☐ heat_capacity_constant_volume - ('Carbon Dioxide',)

☑ thermal_conductivity - ('Carbon Dioxide',)

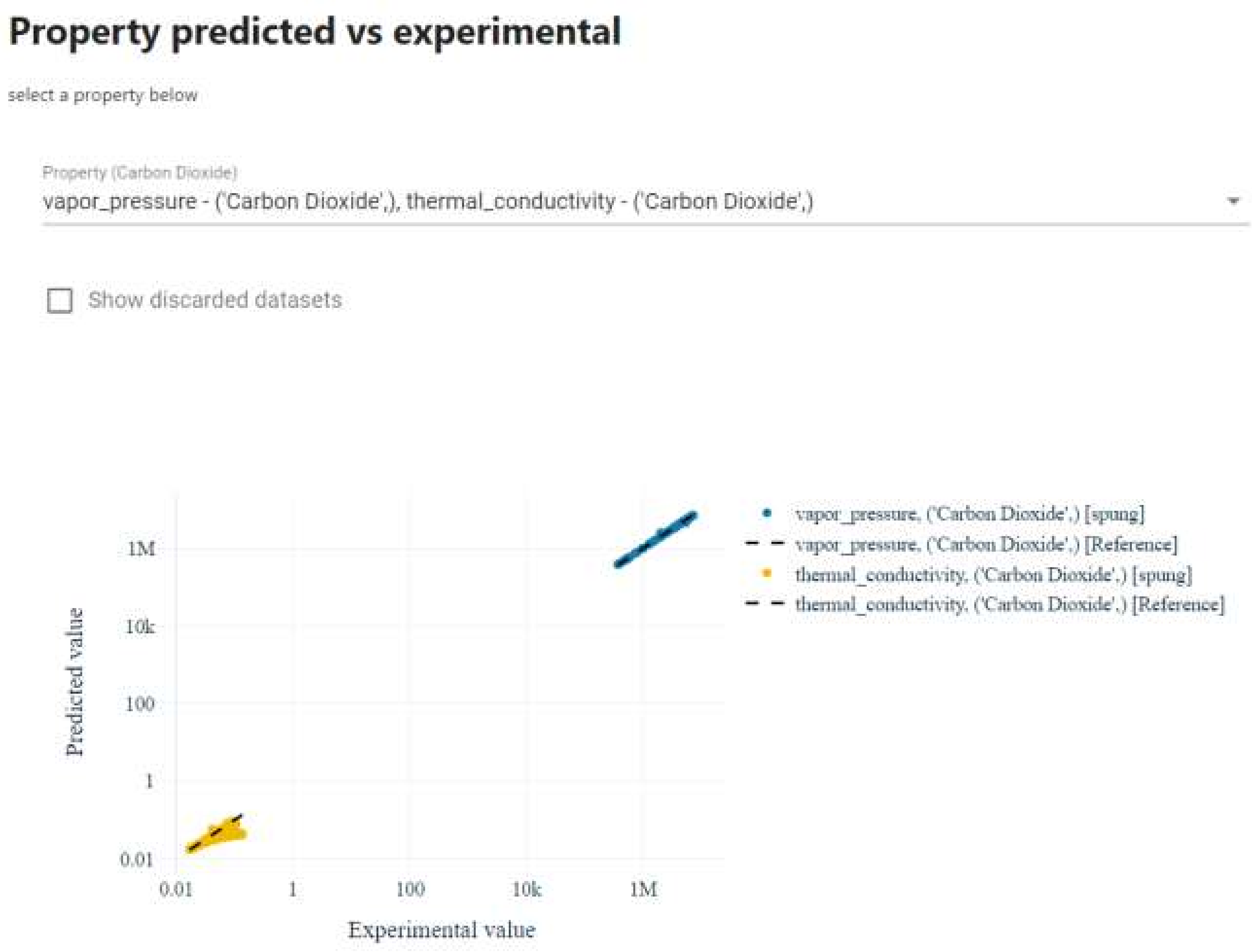
☑ vapor_pressure - ('Carbon Dioxide',)

☐ virial_coefficient_second - ('Carbon Dioxide',)

☐ virial_coefficient_third - ('Carbon Dioxide',)

☐ viscosity - ('Carbon Dioxide',)

# Validation examples for CO2



## Property predicted vs experimental

select a property below

Property (Carbon Dioxide)
vapor_pressure - ('Carbon Dioxide',), thermal_conductivity - ('Carbon Dioxide',)

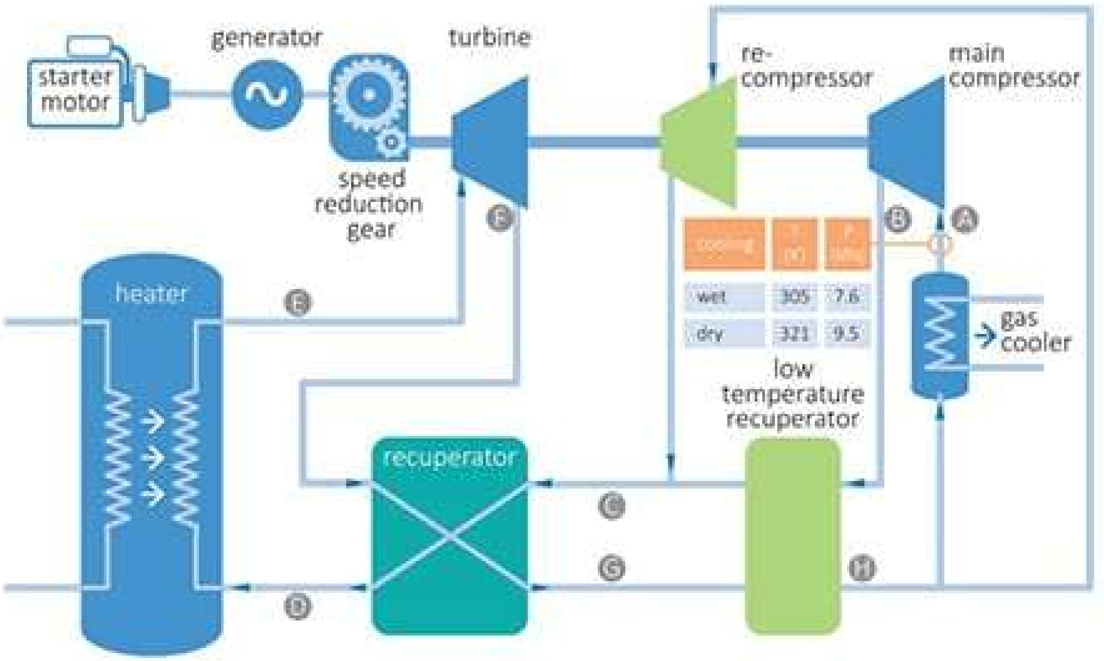
□ Show discarded datasets

Predicted value
1M
10k
100
1
0.01
0.01
1
100
10k
1M
Experimental value
● vapor_pressure, ('Carbon Dioxide',) [spung]
– – vapor_pressure, ('Carbon Dioxide',) [Reference]
● thermal_conductivity, ('Carbon Dioxide',) [spung]
– – thermal_conductivity, ('Carbon Dioxide',) [Reference]

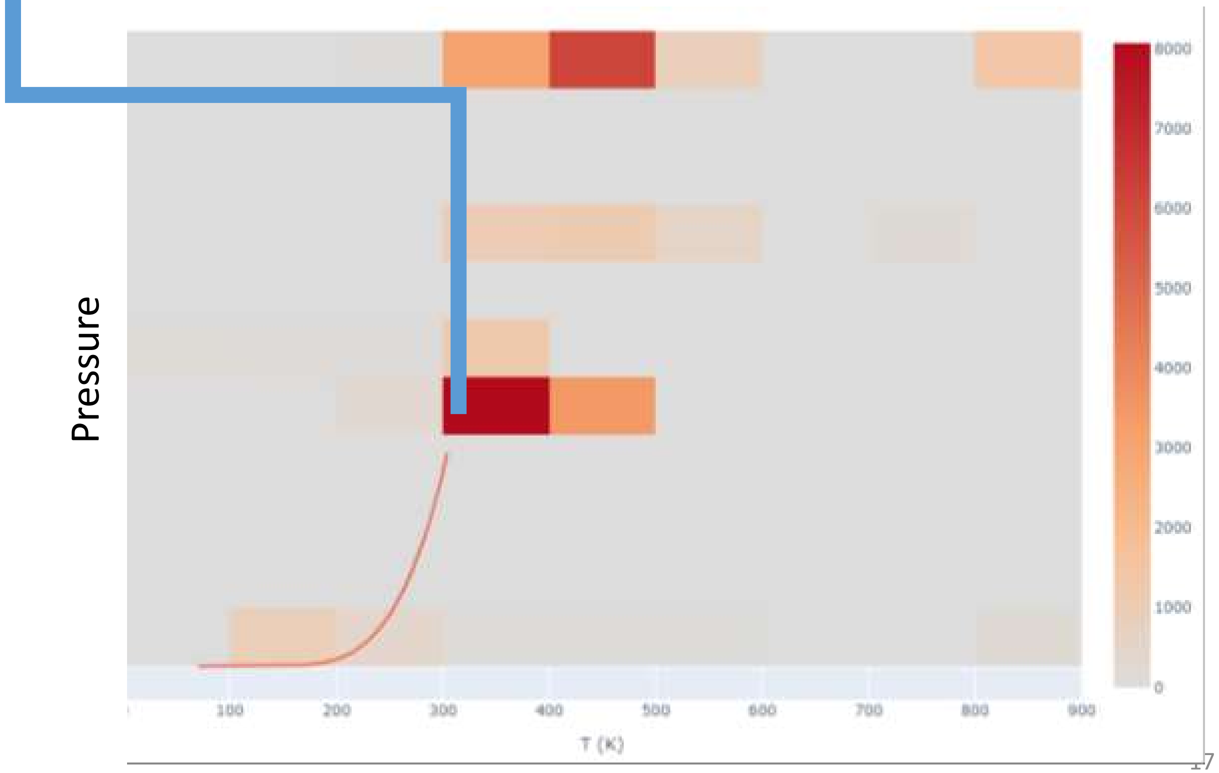
# Validating and demonstrating Q-props property package

Critical region essential for compressor performance

**1) Conceptual flowsheet steady-state flowsheet**



**2) Get all conditions from the flowsheet**



# Making Q-props a viable alternative to built-in Span-Wagner

### Improving speed of dynamic simulation

Initial testing revealed that

- Q-props PR @ 45x slower than native PR
- Q-props Span-Wagner @ **250x** slower

$\Rightarrow$ Infeasible to perform dynamic simulation

Where was the bottleneck?

$$\tau = \tau_{sim} + \tau_{interface} + \tau_{C++/Julia} + \tau_{model}$$

Fixes:

- $\tau_{C++/Julia}$ improved by 10-100x
- $\tau_{model}$ improved by ~10x
- **CAPE-OPEN** allows using our internal flash algos
- ***ExtnPropertyPackage*** uses simulator flash algos

Q-props now performed on-par with built-in Span-Wagner when testing for a simple pair of streams

RTF: Real time factor – simulated minutes/minutes

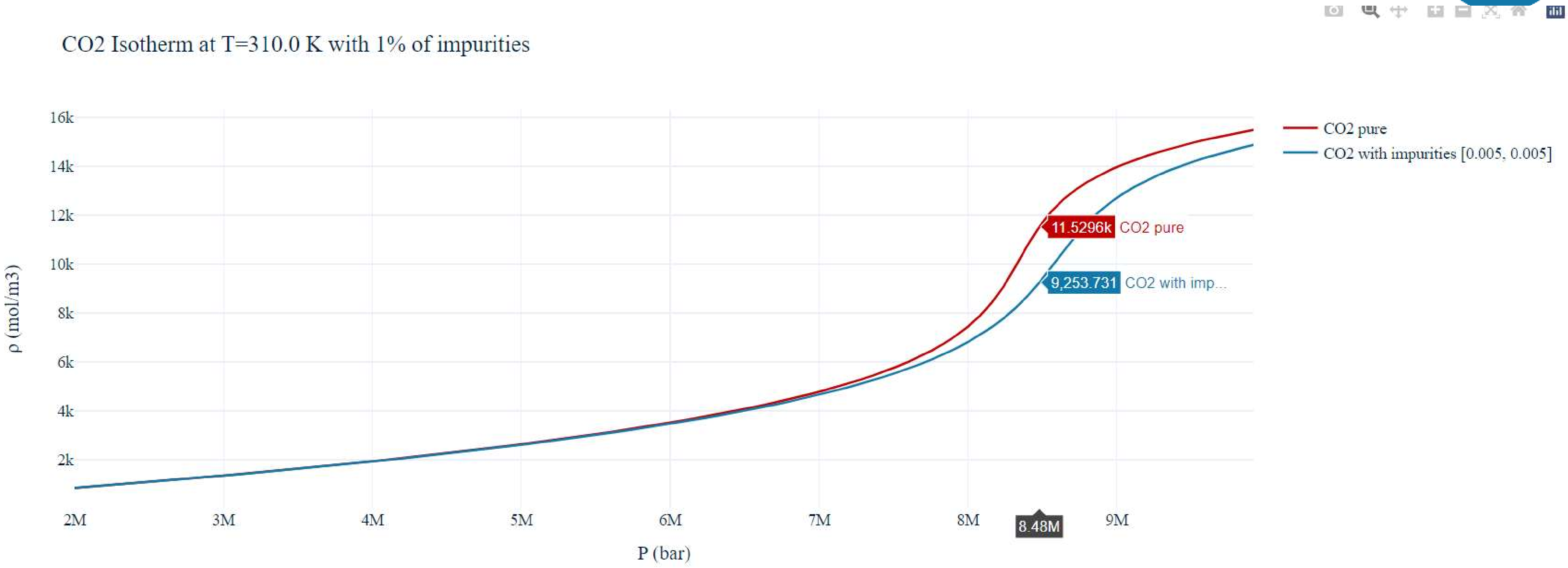
### After improving Q-props speed

| RTF / Scenario | Startup | Load change |
|---|---|---|
| Built-in Span-Wagner | ~0.1 | ~0.1 |
| ***ExtnPropertyPackage*** | **~0.25*** | **~0.8*** |
| **CAPE-OPEN** | **~0.20*** | **~0.4*** |

*) Increased robustness allows increasing the time step, leading to higher simulation real-time factors

CAPE-OPEN was about 20-50% slower than native, but we expect it to perform better for mixtures

# Effect of impurities on properties of $CO_2$-streams

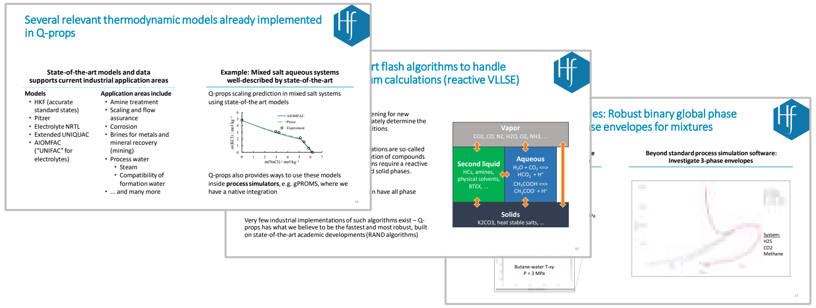


CO2 Isotherm at T=310.0 K with 1% of impurities

# Examples: Simulation of electrolyte systems

# Status of ongoing work to make Q-props a leading modeling tool for amine-based acid gas treatment and carbon capture

## 1. Q-props electrolyte thermodynamic models and general high-performance reactive flash algorithms



Several relevant thermodynamic models already implemented in Q-props

State-of-the-art models and data supports current industrial application areas

**Models**
- HKF (accurate standard states)
- Pitzer
- Electrolyte NRTL
- Extended UNIQUAC
- AIOMFAC ("UNIFAC" for electrolytes)

**Application areas include**
- Amine treatment
- Scaling and flow assurance
- Corrosion
- Brines for metals and mineral recovery (mining)
- Process water
  - Steam
  - Compatibility of formation water
- … and many more

Example: Mixed salt aqueous systems well-described by state-of-the-art

Q-props scaling prediction in mixed salt systems using state-of-the art models

Q-props also provides ways to use these models inside **process simulators**, e.g. gPROMS, where we have a native integration

…rt flash algorithms to handle …m calculations (reactive VLLSE)

…ening for new …ately determine the …itions

…ations are so-called …tion of compounds …ns require a reactive …d solid phases.

…n have all phase

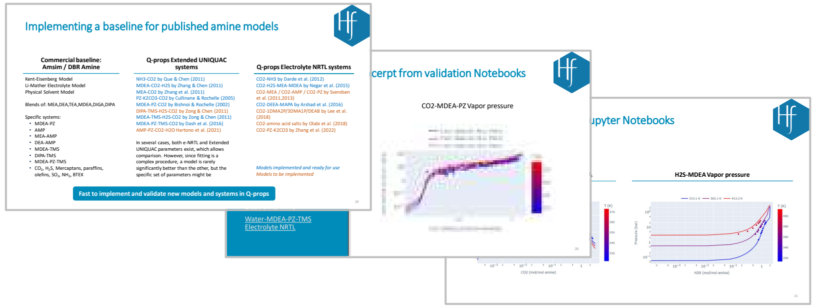
Very few industrial implementations of such algorithms exist – Q-props has what we believe to be the fastest and most robust, built on state-of-the-art academic developments (RAND algorithms)

Vapor
CO2, CO, N2, H2O, O2, NH3, …
Second liquid
HCs, amines, physical solvents, BTEX, …
Aqueous
$H_2O + CO_2 <=> HCO_3^- + H^+$
$CH_3COOH <=> CH_3COO^- + H^+$
Solids
K2CO3, heat stable salts, …

…es: Robust binary global phase …se envelopes for mixtures

Beyond standard process simulation software: Investigate 3-phase envelopes

## 2. Implementing published amine models and establishing a model performance baseline



Implementing a baseline for published amine models

| Commercial baseline: Amsim / DBR Amine | Q-props Extended UNIQUAC systems | Q-props Electrolyte NRTL systems |
|---|---|---|
| Kent-Eisenberg Model<br>Li-Mather Electrolyte Model<br>Physical Solvent Model<br><br>Blends of: MEA,DEA,TEA,MDEA,DIGA,DIPA<br><br>Specific systems:<br>• MDEA-PZ<br>• AMP<br>• MEA-AMP<br>• DEA-AMP<br>• MDEA-TMS<br>• DIPA-TMS<br>• MDEA-PZ-TMS<br>• $CO_2$, $H_2S$, Mercaptans, paraffins, olefins, $SO_2$, $NH_3$, BTEX | NH3-CO2 by Que & Chen (2011)<br>MDEA-CO2-H2S by Zhang & Chen (2011)<br>MEA-CO2 by Zhang et al. (2011)<br>PZ-K2CO3-CO2 by Cullinane & Rochelle (2005)<br>MDEA-PZ-CO2 by Bishnoi & Rochelle (2002)<br>DIPA-TMS-H2S-CO2 by Zong & Chen (2011)<br>MDEA-TMS-H2S-CO2 by Zong & Chen (2011)<br>MDEA-PZ-TMS-CO2 by Dash et al. (2016)<br>AMP-PZ-CO2-H2O Hartono et al. (2021)<br><br>In several cases, both e-NRTL and Extended UNIQUAC parameters exist, which allows comparison. However, since fitting is a complex procedure, a model is rarely significantly better than the other, but the specific set of parameters might be | CO2-NH3 by Darde et al. (2012)<br>CO2-H2S-MEA-MDEA by Negar et al. (2015)<br>CO2-MEA / CO2-AMP / CO2-PZ by Svendsen et al. (2011,2013)<br>CO2-DEEA-MAPA by Arshad et al. (2016)<br>CO2-1DMA2P/3DMA1P/DEAB by Lee et al. (2018)<br>CO2-amino acid salts by Olabi et al. (2018)<br>CO2-PZ-K2CO3 by Zhang et al. (2022)<br><br>*Models implemented and ready for use*<br>*Models to be implemented* |

Fast to implement and validate new models and systems in Q-props

…cerpt from validation Notebooks

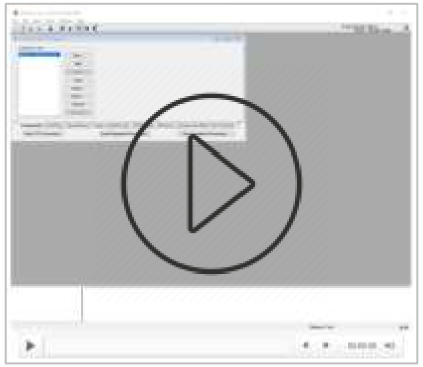
CO2-MDEA-PZ Vapor pressure

Water-MDEA-PZ-TMS Electrolyte NRTL

…upyter Notebooks

H2S-MDEA Vapor pressure

## 3. Integration with process simulators



## 4. Short-term development targets

Improving model performance
- Targeted parameter estimation for improving models
- Improved standard states with HKF
- Electrolyte equations of state (e.g. e-CPA)
- Extension of flash to liquid-liquid equilibrium with two electrolyte phases

Property prediction for novel amines and additives

# Implementing a baseline for published amine models

## Commercial baseline: Amsim / DBR Amine

Kent-Eisenberg Model
Li-Mather Electrolyte Model
Physical Solvent Model

Blends of: MEA,DEA,TEA,MDEA,DIGA,DIPA

Specific systems:

- MDEA-PZ
- AMP
- MEA-AMP
- DEA-AMP
- MDEA-TMS
- DIPA-TMS
- MDEA-PZ-TMS
- $CO_2$, $H_2S$, Mercaptans, paraffins, olefins, $SO_2$, $NH_3$, BTEX

## Q-props Extended UNIQUAC systems

NH3-CO2 by Que & Chen (2011)
MDEA-CO2-H2S by Zhang & Chen (2011)
MEA-CO2 by Zhang et al. (2011)
PZ-K2CO3-CO2 by Cullinane & Rochelle (2005)
MDEA-PZ-CO2 by Bishnoi & Rochelle (2002)
DIPA-TMS-H2S-CO2 by Zong & Chen (2011)
MDEA-TMS-H2S-CO2 by Zong & Chen (2011)
MDEA-PZ-TMS-CO2 by Dash et al. (2016)
AMP-PZ-CO2-H2O Hartono et al. (2021)

In several cases, both e-NRTL and Extended UNIQUAC parameters exist, which allows comparison. However, since fitting is a complex procedure, a model is rarely significantly better than the other, but the specific set of parameters might be

## Q-props Electrolyte NRTL systems

CO2-NH3 by Darde et al. (2012)
CO2-H2S-MEA-MDEA by Negar et al. (2015)
CO2-MEA / CO2-AMP / CO2-PZ by Svendsen et al. (2011,2013)
CO2-DEEA-MAPA by Arshad et al. (2016)
CO2-1DMA2P/3DMA1P/DEAB by Lee et al. (2018)
CO2-amino acid salts by Olabi et al. (2018)
CO2-PZ-K2CO3 by Zhang et al. (2022)

*Models implemented and ready for use*
*Models to be implemented*

**Fast to implement and validate new models and systems in Q-props**

# Model performance

Q-props is integrated with Jupyter Lab, which allows for detailed analysis and interactive plotting using Python libraries in Jupyter Notebooks

Validation Notebook examples:

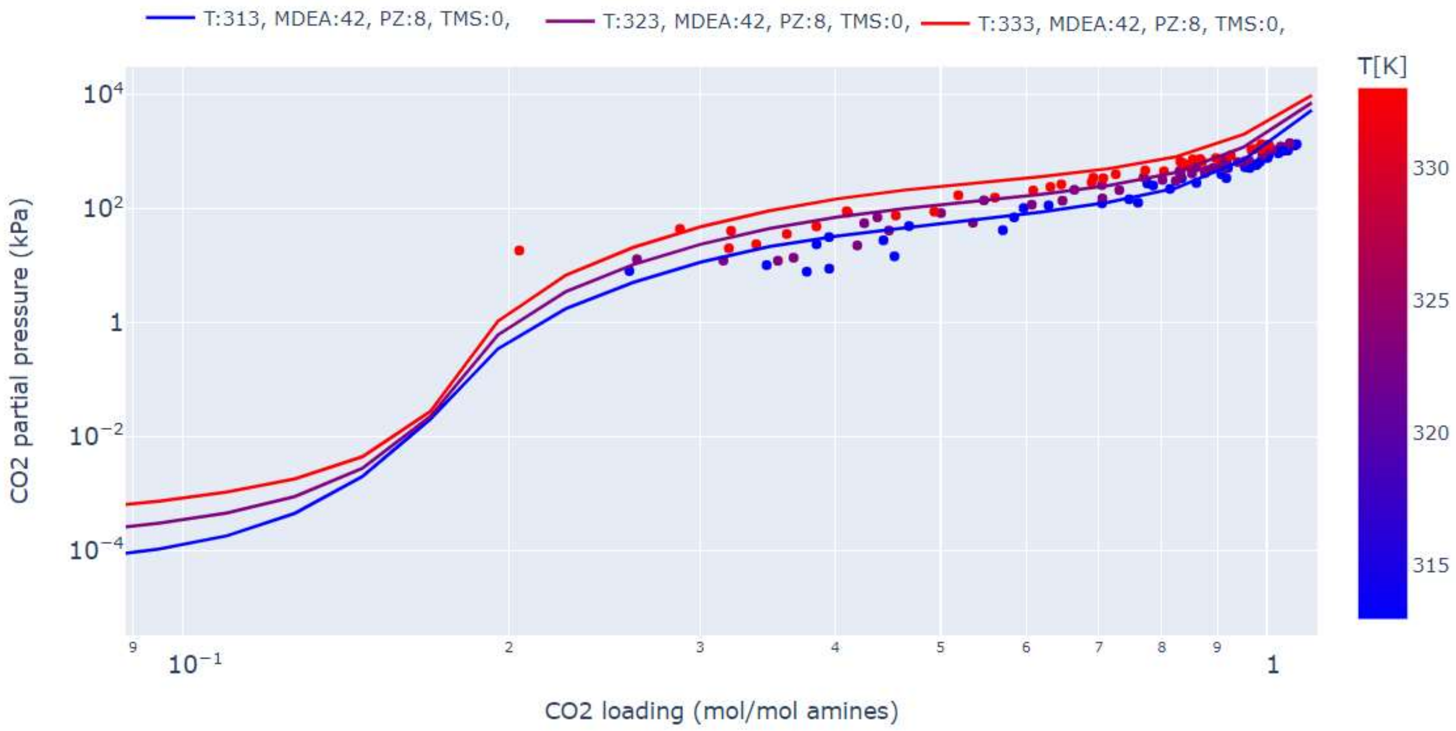
MDEA-CO$_2$-H$_2$S Electrolyte NRTL

MDEA-CO$_2$-H$_2$S Extended UNIQUAC

MDEA-PZ-TMS-CO$_2$ Electrolyte NRTL

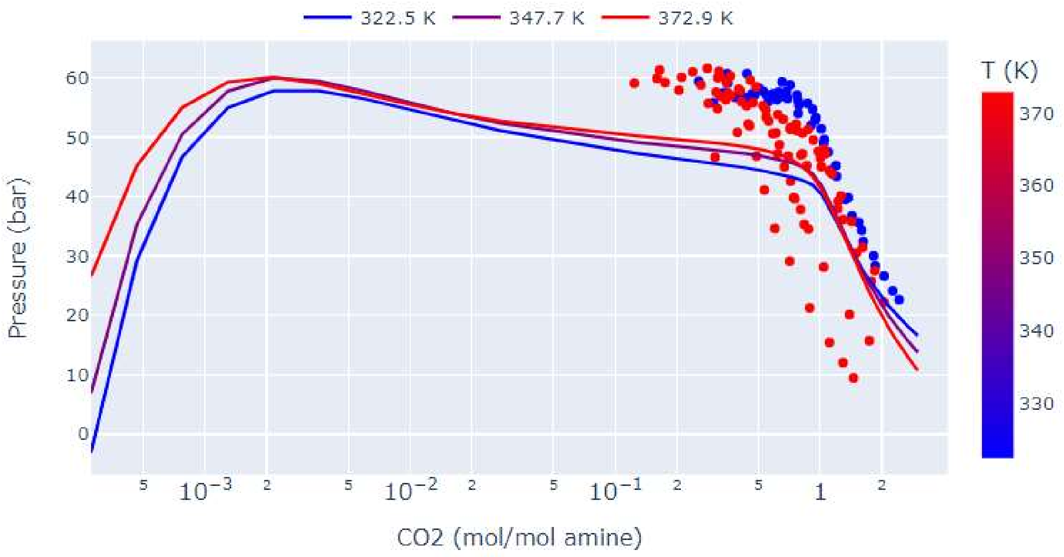
## Excerpt from validation Notebooks

CO2-MDEA-PZ loading curve

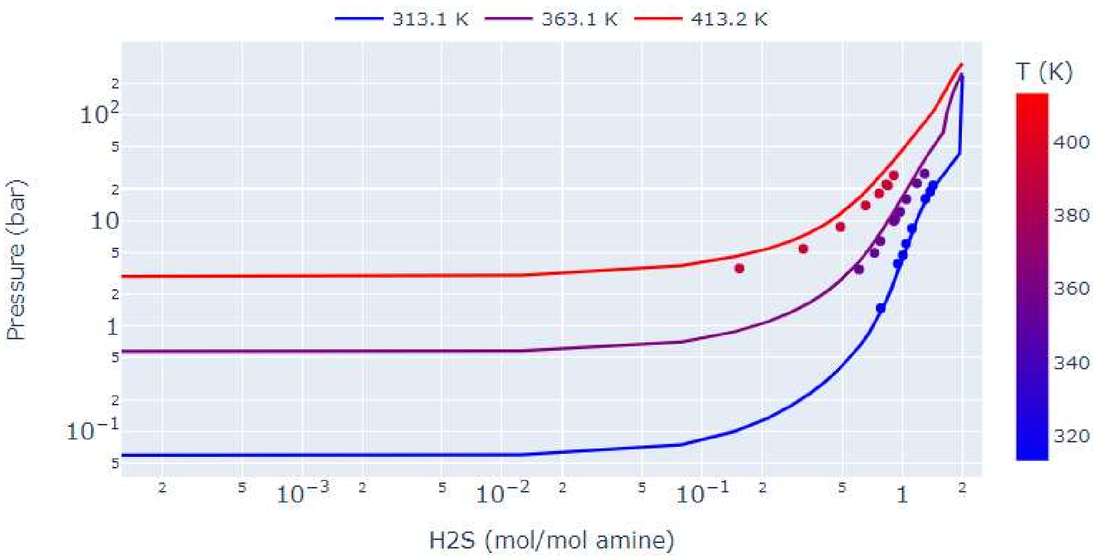


# Additional excerpts from Jupyter Notebooks

**Enthalpy of adsorption 30 wt% MDEA**

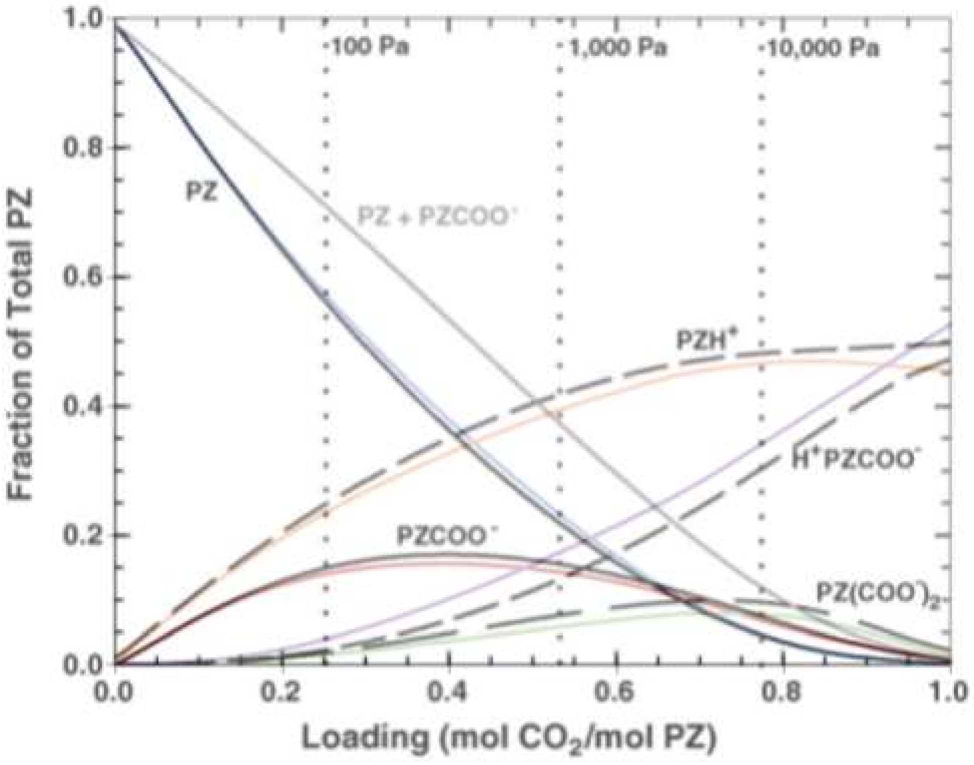


**H2S-MDEA loading curve**

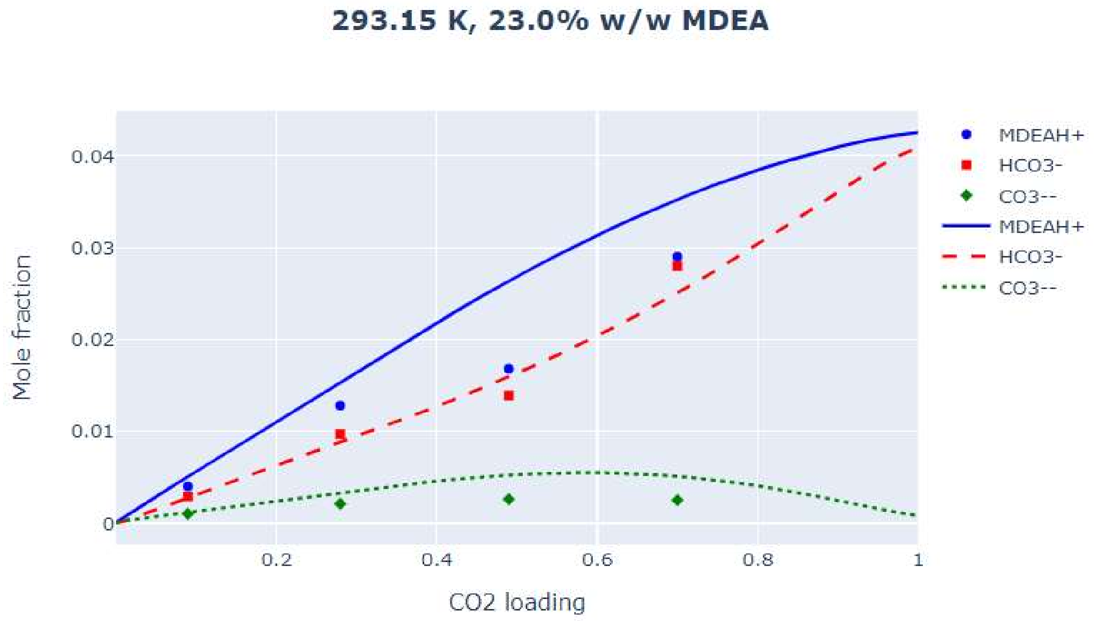


# Speciation validation

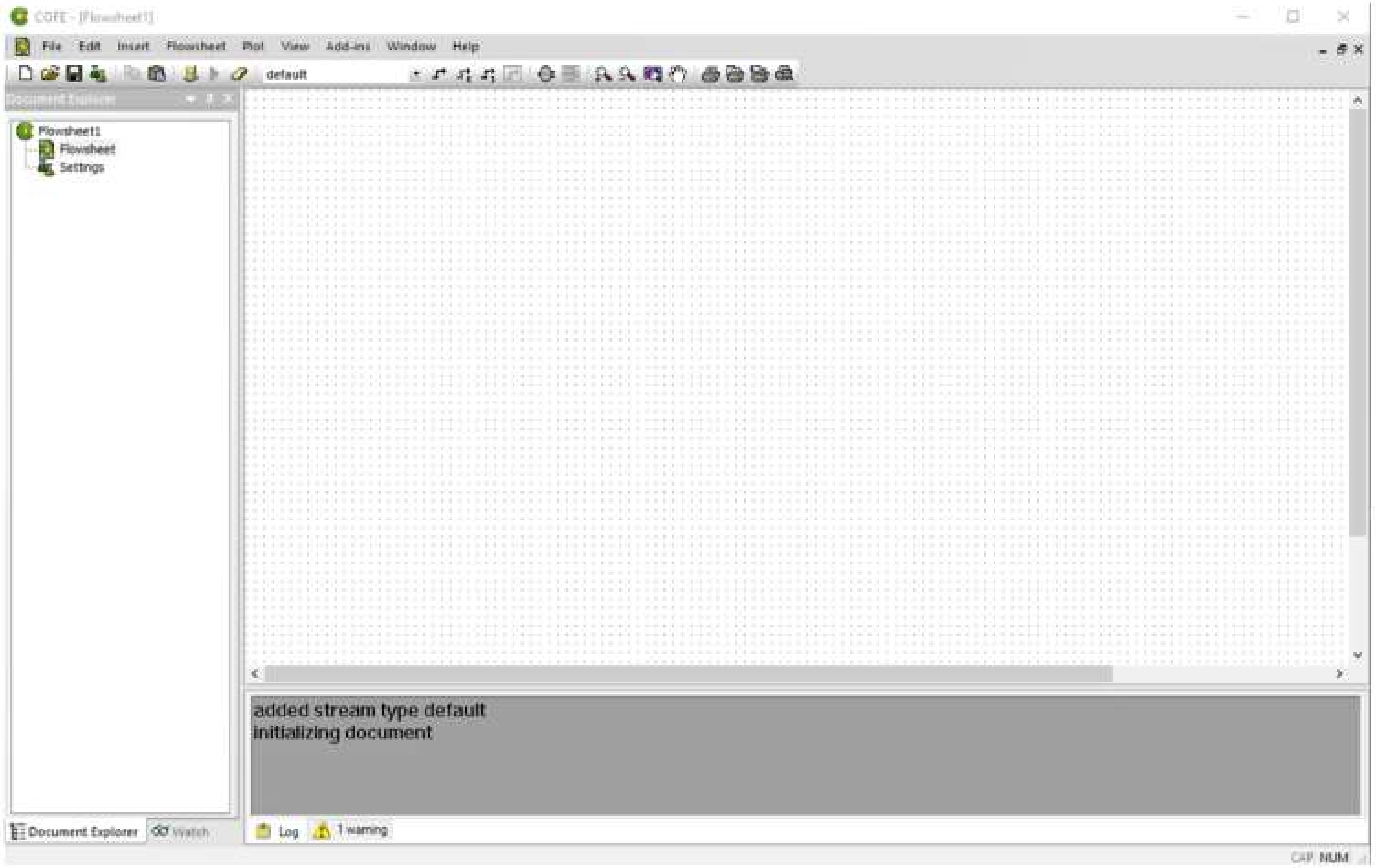
**Speciation in 1.8 molal PZ at 60°C with CO2 (Q-props prediction against original paper)**



**MDEA-CO2 speciation**

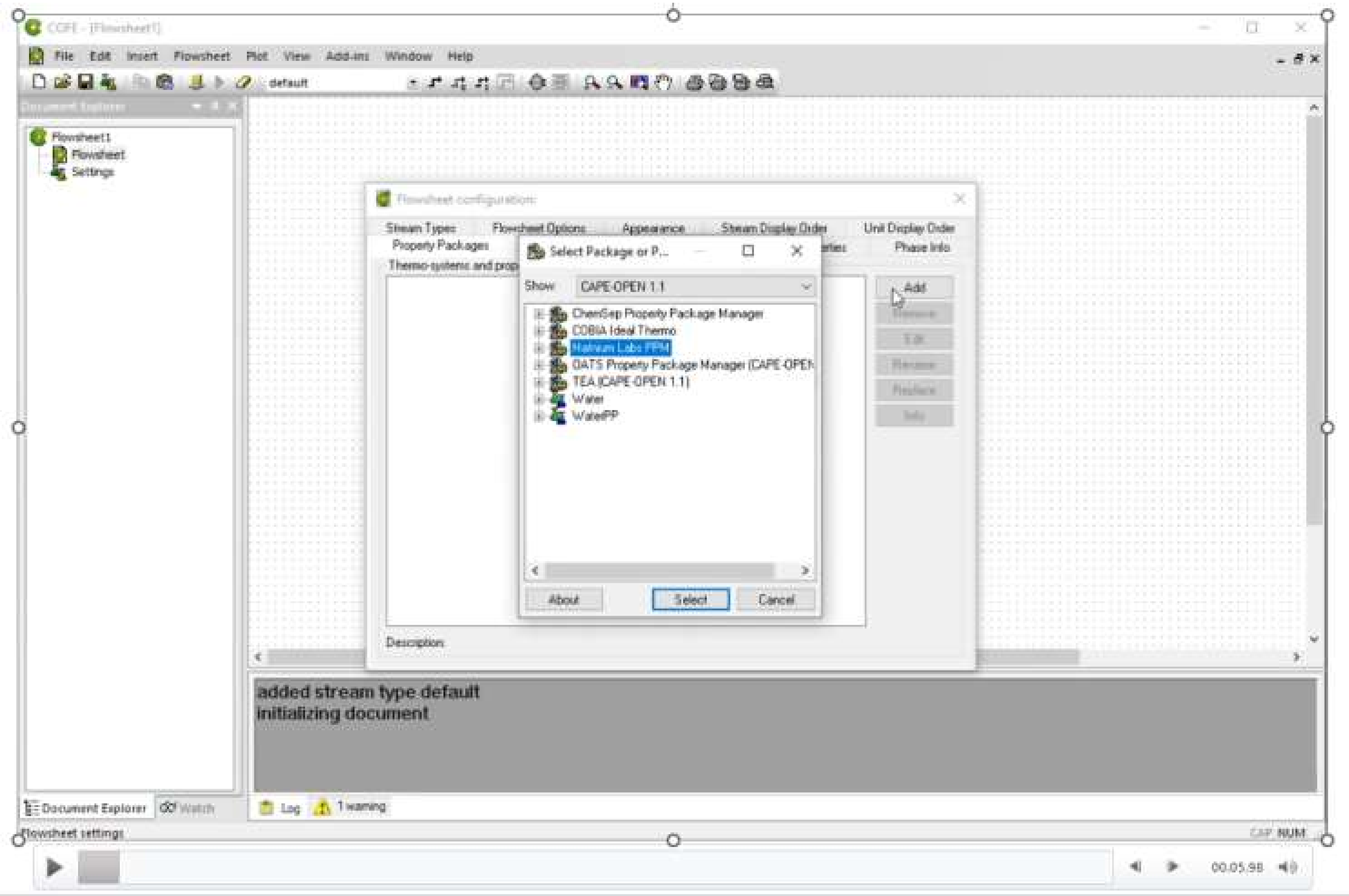


# Examples of simulation with Q-props MDEA packages



```
added stream type default
initializing document
```

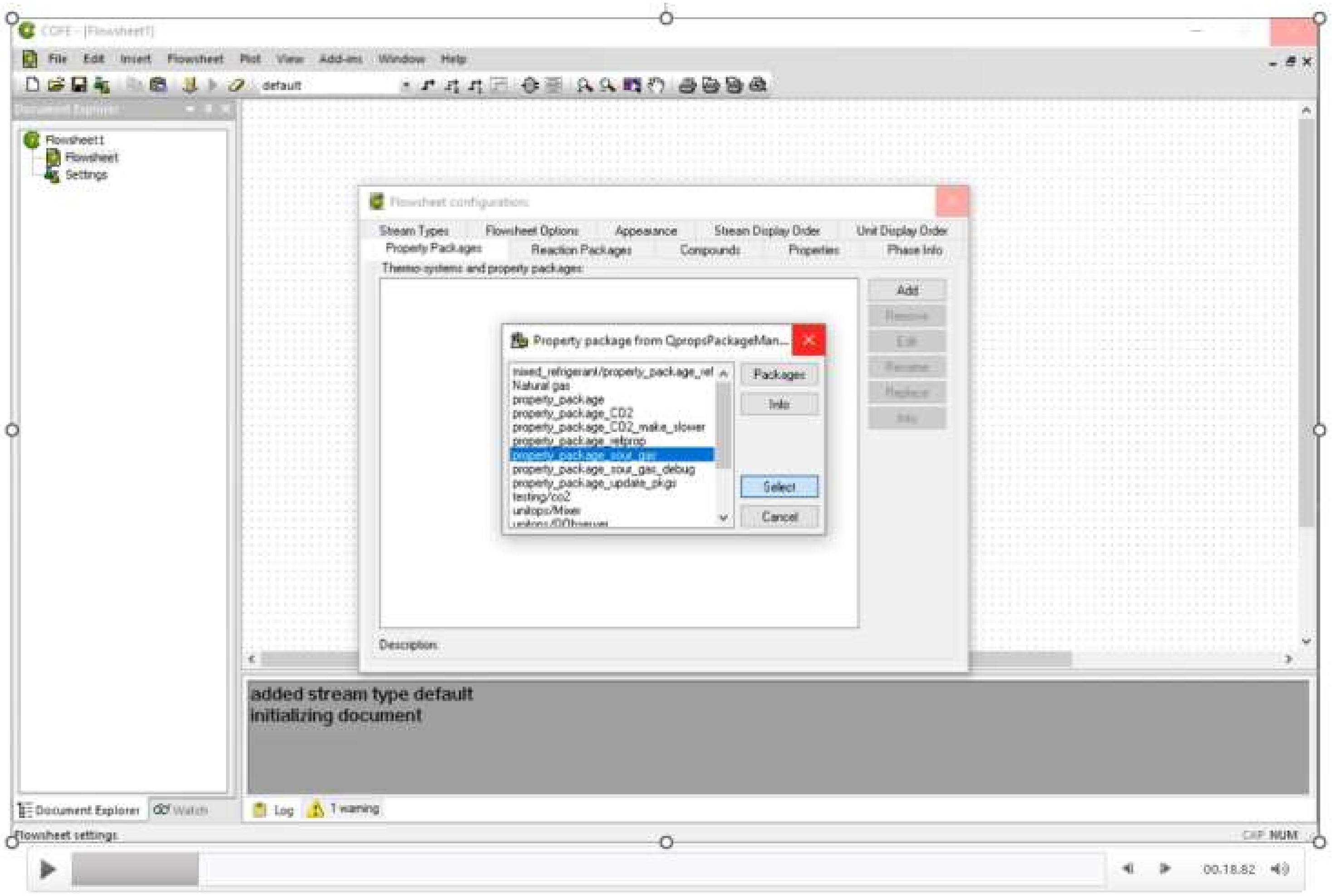
# Examples of simulation with Q-props MDEA packages



Show: CAPE-OPEN 1.1

```
added stream type default
initializing document
```

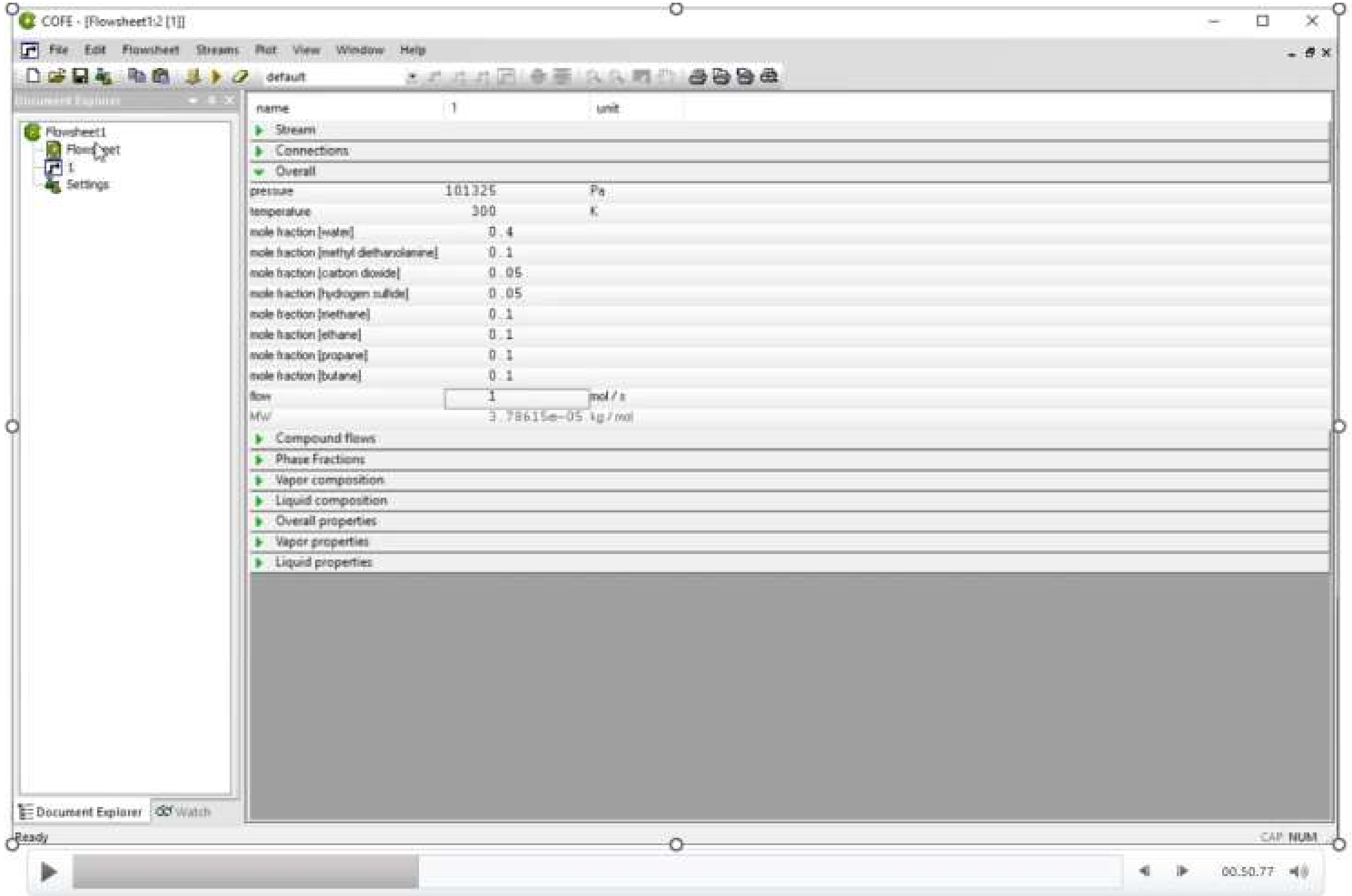
# Examples of simulation with Q-props MDEA packages



mixed_refrigerant/property_package_ref
Natural gas
property_package
property_package_CO2
property_package_CO2_make_slower
property_package_refprop
property_package_sour_gas
property_package_sour_gas_debug
property_package_update_pkgs
testing/co2
unitops/Mixer
unitops/OOObserver

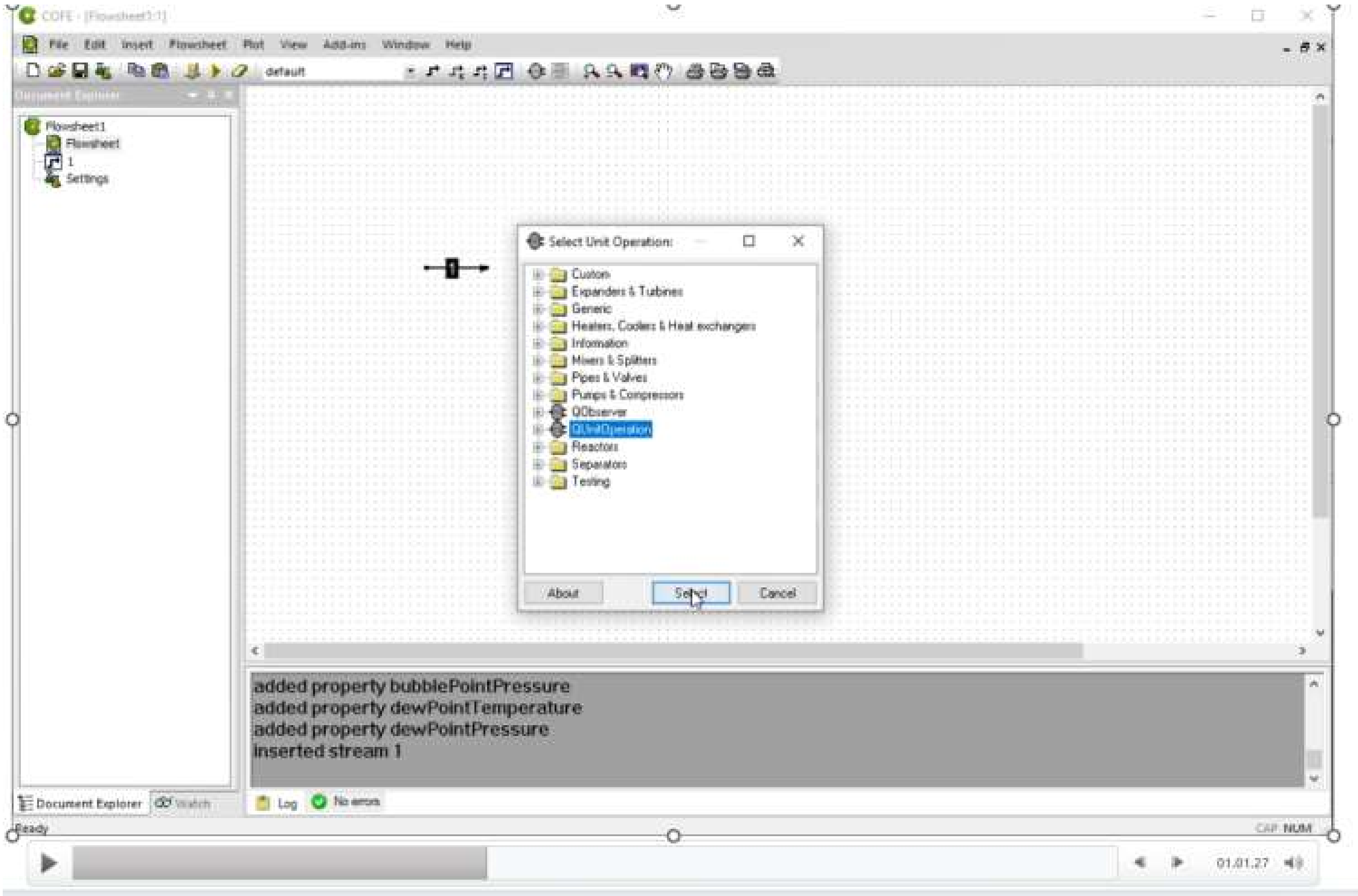
added stream type default
initializing document

# Examples of simulation with Q-props MDEA packages



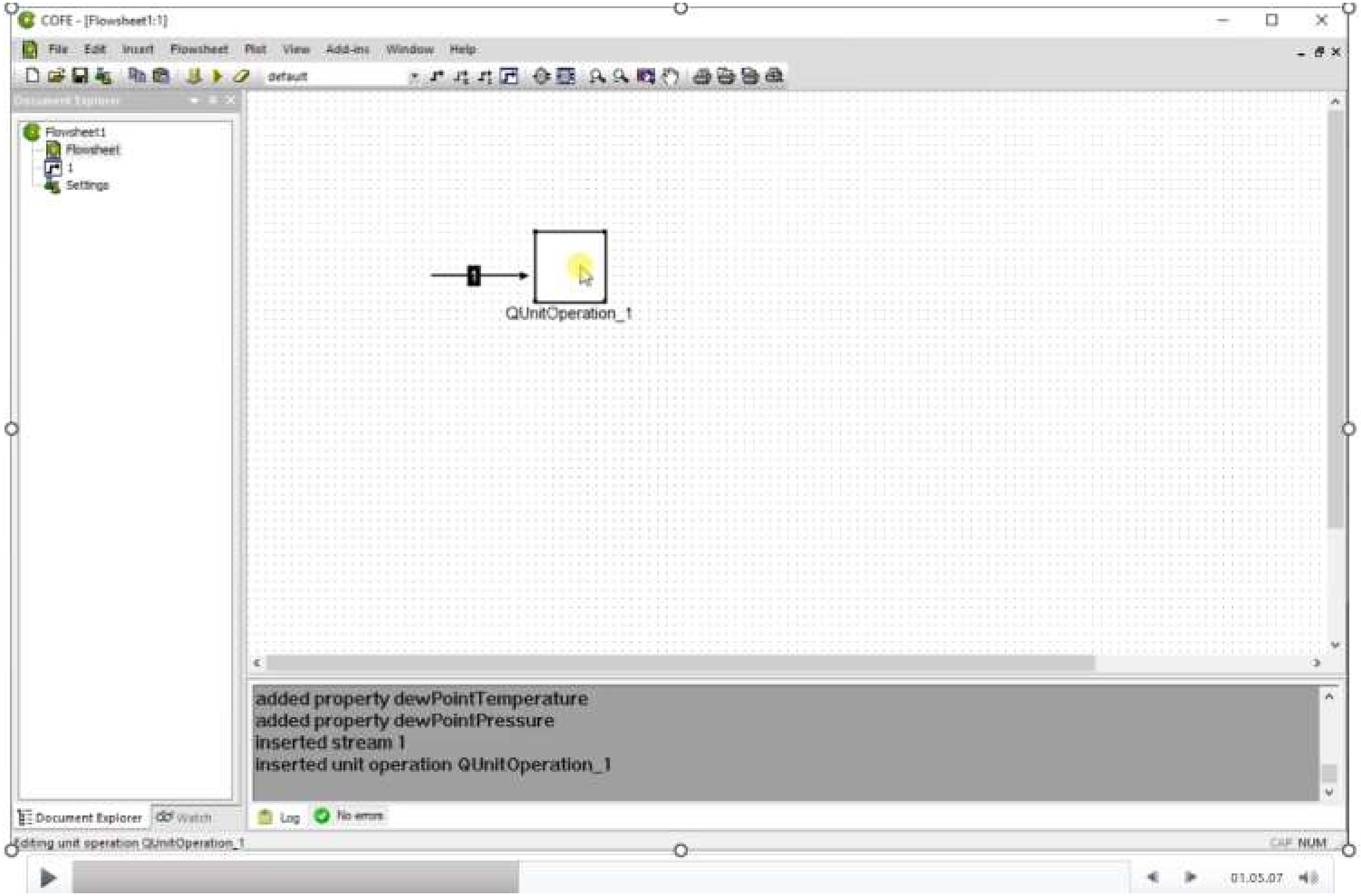
| name | 1 | unit |
|---|---|---|
| Stream | | |
| Connections | | |
| Overall | | |
| pressure | 101325 | Pa |
| temperature | 300 | K |
| mole fraction [water] | 0.4 | |
| mole fraction [methyl diethanolamine] | 0.1 | |
| mole fraction [carbon dioxide] | 0.05 | |
| mole fraction [hydrogen sulfide] | 0.05 | |
| mole fraction [methane] | 0.1 | |
| mole fraction [ethane] | 0.1 | |
| mole fraction [propane] | 0.1 | |
| mole fraction [butane] | 0.1 | |
| flow | 1 | mol / s |
| MW | 3.78615e-05 | kg / mol |
| Compound flows | | |
| Phase Fractions | | |
| Vapor composition | | |
| Liquid composition | | |
| Overall properties | | |
| Vapor properties | | |
| Liquid properties | | |

# Examples of simulation with Q-props MDEA packages



```
added property bubblePointPressure
added property dewPointTemperature
added property dewPointPressure
inserted stream 1
```

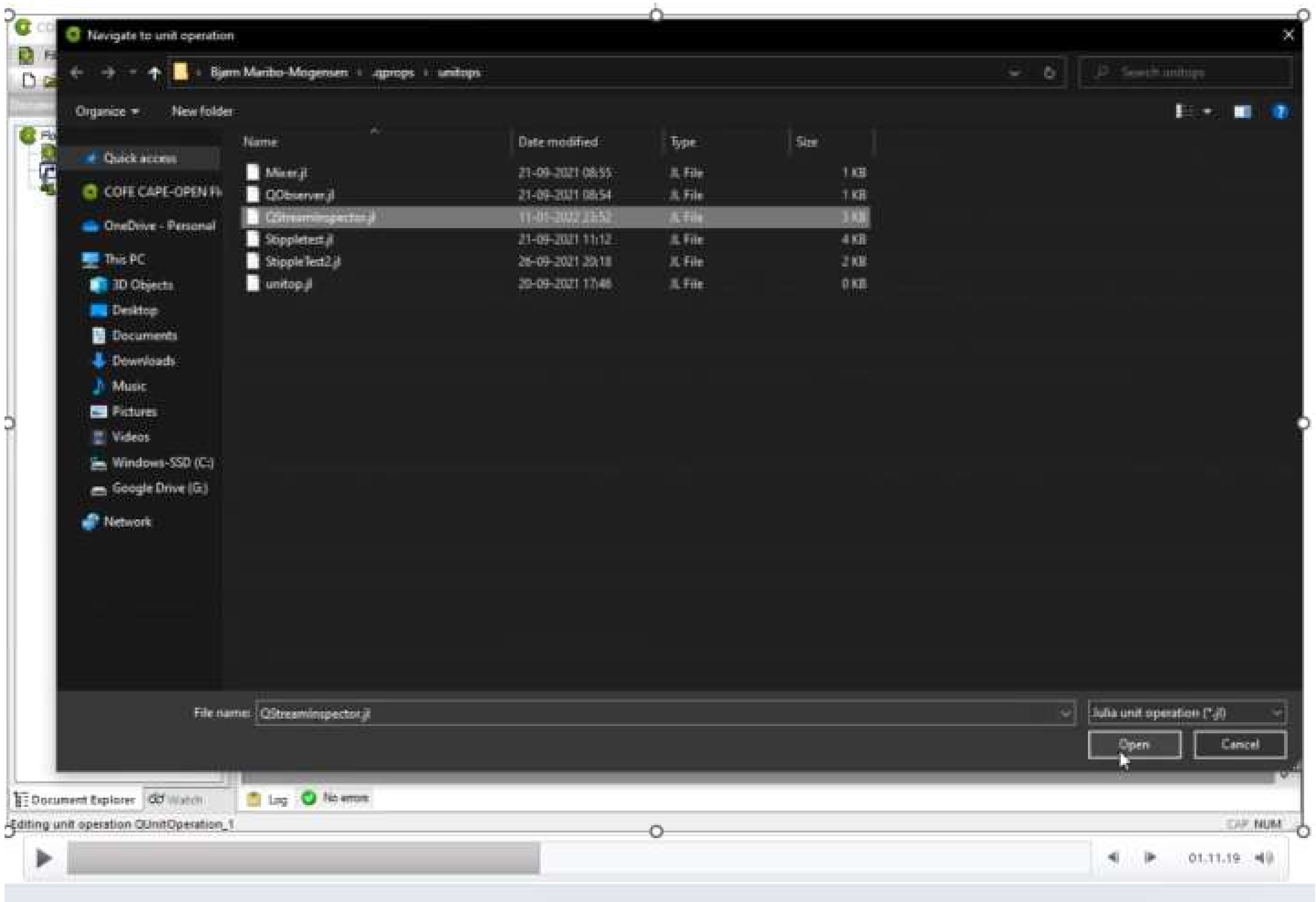
# Examples of simulation with Q-props MDEA packages



```
added property dewPointTemperature
added property dewPointPressure
inserted stream 1
inserted unit operation QUnitOperation_1
```

Editing unit operation QUnitOperation_1

# Examples of simulation with Q-props MDEA packages



Navigate to unit operation

Bjørn Maribo-Mogensen › .qprops › unitops

| Name | Date modified | Type | Size |
|---|---|---|---|
| Mixer.jl | 21-09-2021 08:55 | JL File | 1 KB |
| QObserver.jl | 21-09-2021 08:54 | JL File | 1 KB |
| QStreamInspector.jl | 11-01-2022 23:52 | JL File | 3 KB |
| Stippletest.jl | 21-09-2021 11:12 | JL File | 4 KB |
| StippleTest2.jl | 26-09-2021 20:18 | JL File | 2 KB |
| unitop.jl | 20-09-2021 17:46 | JL File | 0 KB |

File name: QStreamInspector.jl

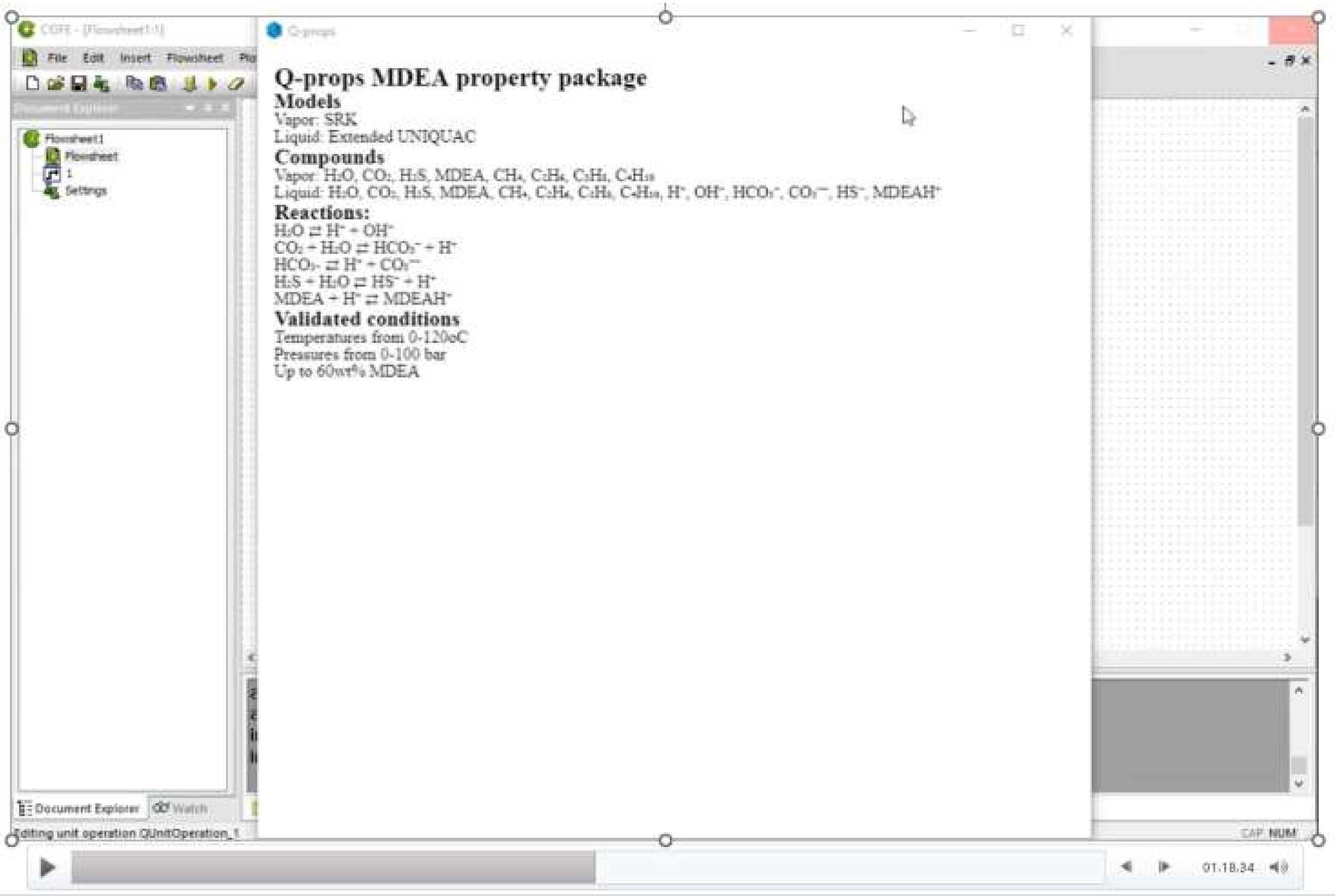
Julia unit operation (*.jl)

No errors

Editing unit operation QUnitOperation_1

Julia-script uses Q-props thermos internally, optionally circumveinting PME-PMC calls

# Examples of simulation with Q-props MDEA packages



## Q-props MDEA property package

### Models

Vapor: SRK
Liquid: Extended UNIQUAC

### Compounds

Vapor: $H_2O$, $CO_2$, $H_2S$, MDEA, $CH_4$, $C_2H_6$, $C_3H_8$, $C_4H_{10}$
Liquid: $H_2O$, $CO_2$, $H_2S$, MDEA, $CH_4$, $C_2H_6$, $C_3H_8$, $C_4H_{10}$, $H^+$, $OH^-$, $HCO_3^-$, $CO_3^{--}$, $HS^-$, $MDEAH^+$

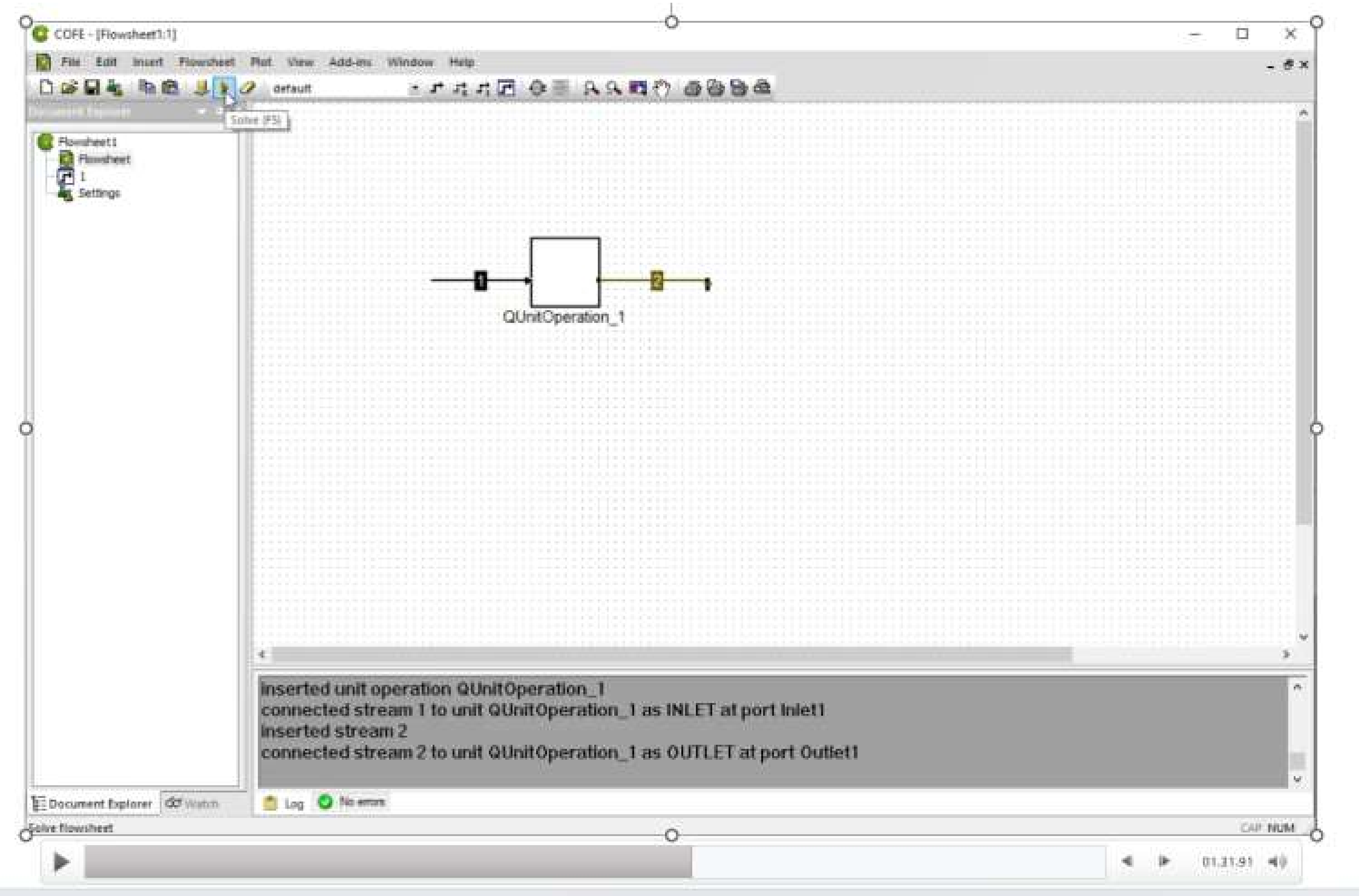
### Reactions:

$H_2O \rightleftarrows H^+ + OH^-$
$CO_2 + H_2O \rightleftarrows HCO_3^- + H^+$
$HCO_3- \rightleftarrows H^+ + CO_3^{--}$
$H_2S + H_2O \rightleftarrows HS^- + H^+$
$MDEA + H^+ \rightleftarrows MDEAH^+$

### Validated conditions

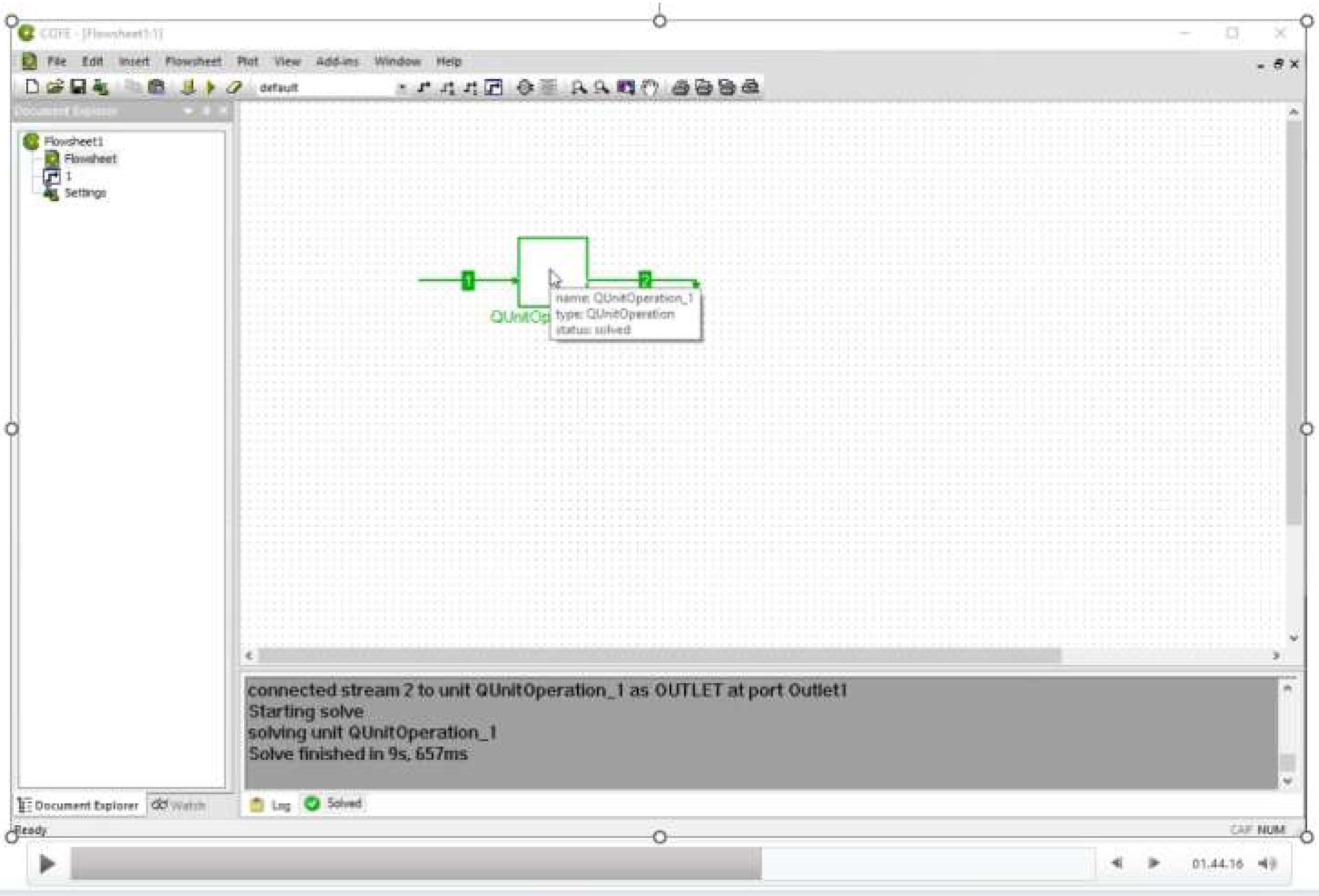
Temperatures from 0-120oC
Pressures from 0-100 bar
Up to 60wt% MDEA

# Examples of simulation with Q-props MDEA packages



```
inserted unit operation QUnitOperation_1
connected stream 1 to unit QUnitOperation_1 as INLET at port Inlet1
inserted stream 2
connected stream 2 to unit QUnitOperation_1 as OUTLET at port Outlet1
```

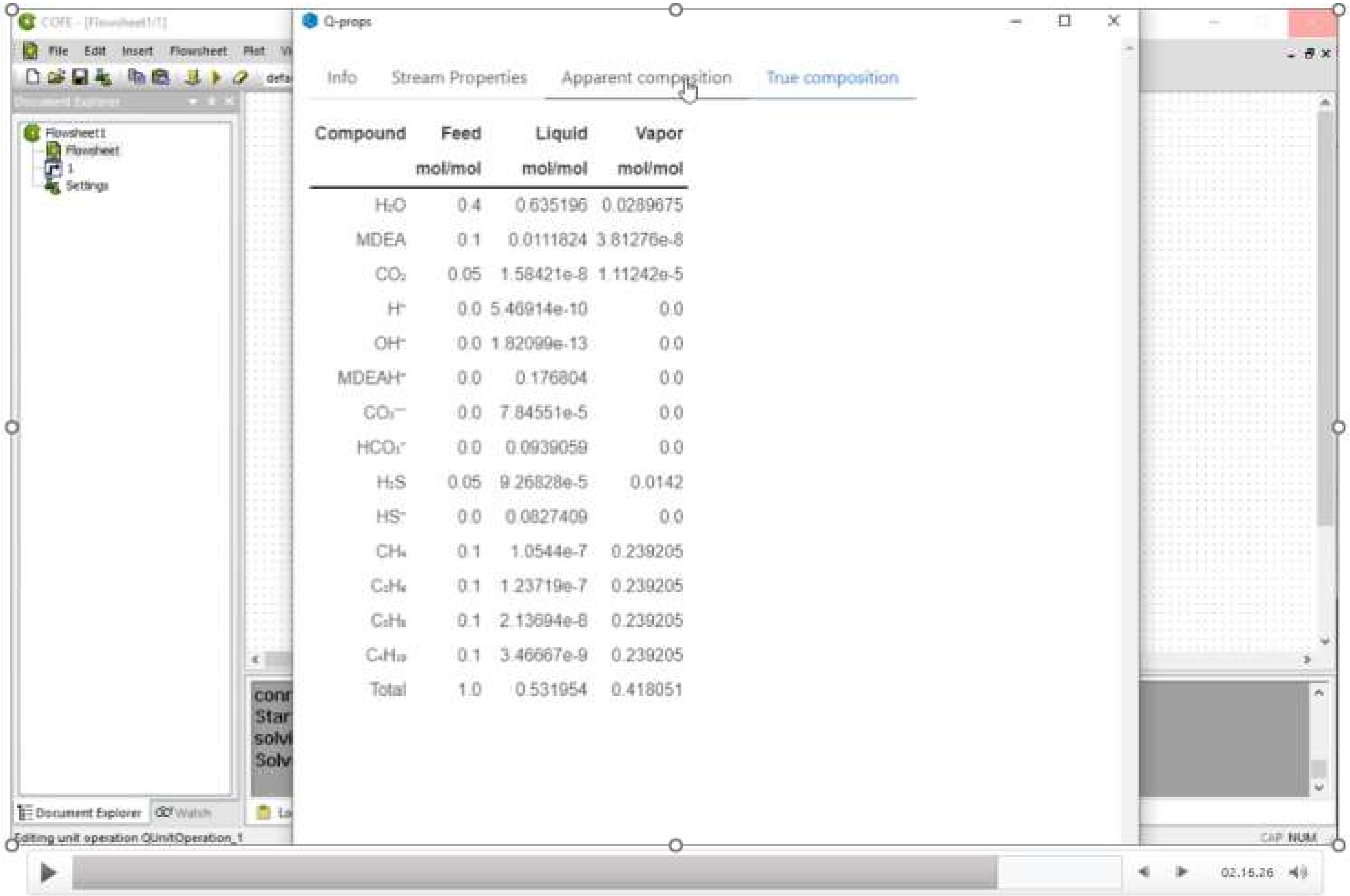
# Examples of simulation with Q-props MDEA packages



name: QUnitOperation_1
type: QUnitOperation
status: solved

```
connected stream 2 to unit QUnitOperation_1 as OUTLET at port Outlet1
Starting solve
solving unit QUnitOperation_1
Solve finished in 9s, 657ms
```

# Examples of simulation with Q-props MDEA packages



| Compound | Feed<br>mol/mol | Liquid<br>mol/mol | Vapor<br>mol/mol |
|---|---|---|---|
| $H_2O$ | 0.4 | 0.635196 | 0.0289675 |
| MDEA | 0.1 | 0.0111824 | 3.81276e-8 |
| $CO_2$ | 0.05 | 1.58421e-8 | 1.11242e-5 |
| $H^+$ | 0.0 | 5.46914e-10 | 0.0 |
| $OH^-$ | 0.0 | 1.82099e-13 | 0.0 |
| $MDEAH^+$ | 0.0 | 0.176804 | 0.0 |
| $CO_3^{--}$ | 0.0 | 7.84551e-5 | 0.0 |
| $HCO_3^-$ | 0.0 | 0.0939059 | 0.0 |
| $H_2S$ | 0.05 | 9.26828e-5 | 0.0142 |
| $HS^-$ | 0.0 | 0.0827409 | 0.0 |
| $CH_4$ | 0.1 | 1.0544e-7 | 0.239205 |
| $C_2H_6$ | 0.1 | 1.23719e-7 | 0.239205 |
| $C_3H_8$ | 0.1 | 2.13694e-8 | 0.239205 |
| $C_4H_{10}$ | 0.1 | 3.46667e-9 | 0.239205 |
| Total | 1.0 | 0.531954 | 0.418051 |

# Examples: Development of new property models

# Case study: Implementation of Polar PC-SAFT

### Motivation

- New chemistries puts new demands on predictive capabilities of thermodynamic models
- Adding new physical terms can reduce need for data to fit kijs
- We need to be able to quickly implement and evaluate models as they appear in literature
- One example of such new model is the Polar PC-SAFT, which we'll demonstrate next

### Model

Marshall and Bokis, Fluid Phase Equilibria, 489 (2019) 83-89

Model implemented in a Jupyter Notebook and included in a property package (demo)

Adds a new pure compound parameter (polarizability)

Model validation

- Cyclohexane-benzene

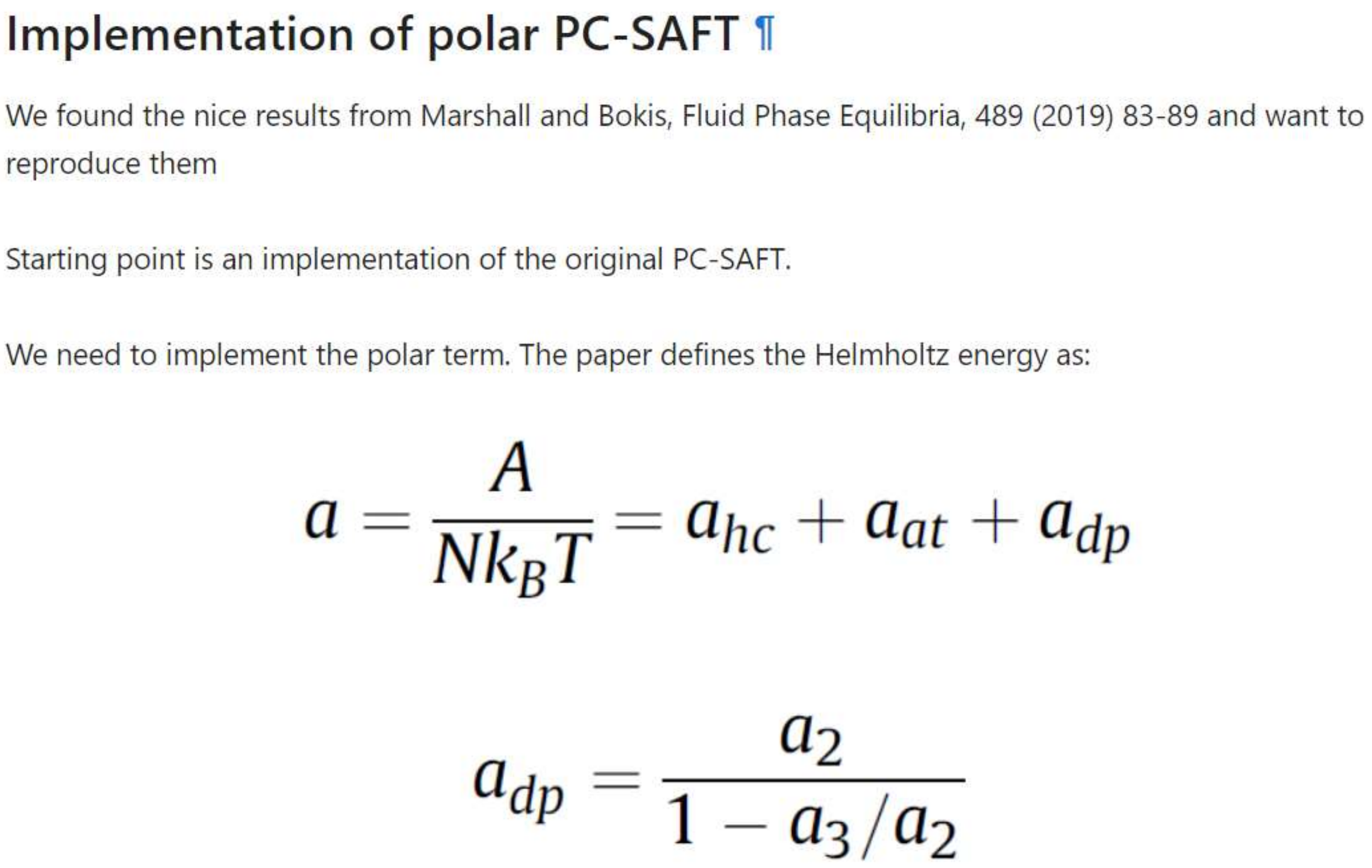
### Export

Exports to a Q-props Model JSON that contains definitions of

- Compounds
- Models in each phase
- Parameters in each phase
- Experimental data (optional)
- Fit strategy (optional)

Gets loaded by the process simulator, which can now use it internally

# Case study: Implementation of Polar PC-SAFT



## Implementation of polar PC-SAFT ¶

We found the nice results from Marshall and Bokis, Fluid Phase Equilibria, 489 (2019) 83-89 and want to reproduce them

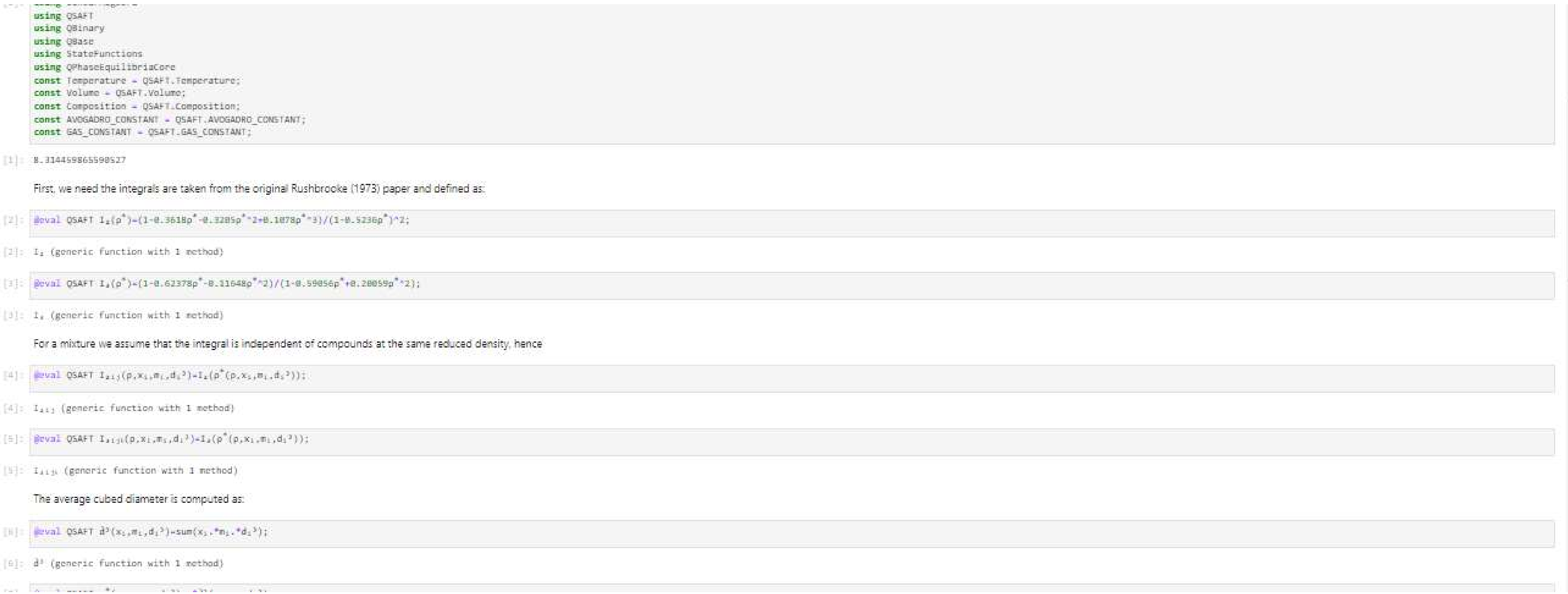
Starting point is an implementation of the original PC-SAFT.

We need to implement the polar term. The paper defines the Helmholtz energy as:

$$a = \frac{A}{Nk_BT} = a_{hc} + a_{at} + a_{dp}$$

$$a_{dp} = \frac{a_2}{1 - a_3/a_2}$$

# Case study: Implementation of Polar PC-SAFT



```
using QSAFT
using QBinary
using QBase
using StateFunctions
using QPhaseEquilibriaCore
const Temperature = QSAFT.Temperature;
const Volume = QSAFT.Volume;
const Composition = QSAFT.Composition;
const AVOGADRO_CONSTANT = QSAFT.AVOGADRO_CONSTANT;
const GAS_CONSTANT = QSAFT.GAS_CONSTANT;
```

```
[1]: 8.314459865598527
```

First, we need the integrals are taken from the original Rushbrooke (1973) paper and defined as:

```
[2]: @eval QSAFT I₂(ρ*)=(1-0.3618ρ*-0.3205ρ*^2+0.1078ρ*^3)/(1-0.5236ρ*)^2;
```

```
[2]: I₂ (generic function with 1 method)
```

```
[3]: @eval QSAFT I₃(ρ*)=(1-0.62378ρ*-0.11648ρ*^2)/(1-0.59056ρ*+0.20059ρ*^2);
```

```
[3]: I₃ (generic function with 1 method)
```

For a mixture we assume that the integral is independent of compounds at the same reduced density, hence

```
[4]: @eval QSAFT I₂ᵢⱼ(ρ,xᵢ,mᵢ,dᵢ³)=I₂(ρ*(ρ,xᵢ,mᵢ,dᵢ³));
```

```
[4]: I₂ᵢⱼ (generic function with 1 method)
```

```
[5]: @eval QSAFT I₃ᵢⱼₖ(ρ,xᵢ,mᵢ,dᵢ³)=I₃(ρ*(ρ,xᵢ,mᵢ,dᵢ³));
```

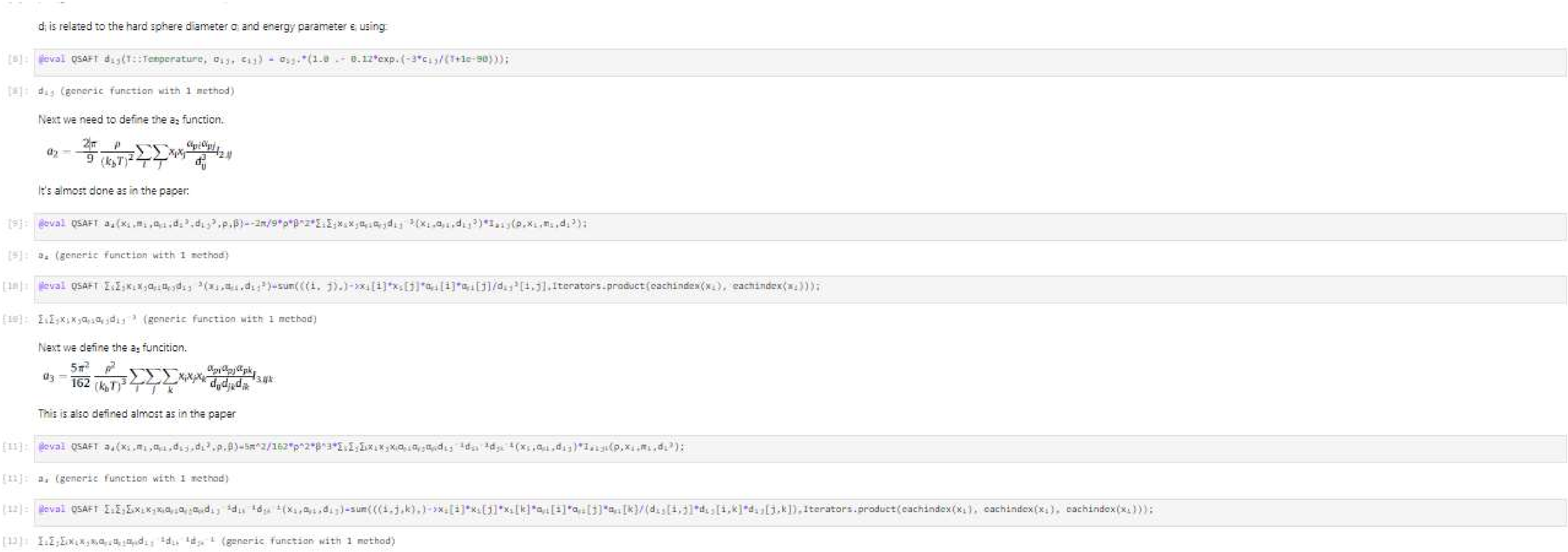
```
[5]: I₃ᵢⱼₖ (generic function with 1 method)
```

The average cubed diameter is computed as:

```
[6]: @eval QSAFT d̄³(xᵢ,mᵢ,dᵢ³)=sum(xᵢ.*mᵢ.*dᵢ³);
```

```
[6]: d̄³ (generic function with 1 method)
```

# Case study: Implementation of Polar PC-SAFT



$d_i$ is related to the hard sphere diameter $\sigma_i$ and energy parameter $\epsilon_i$ using:

```
[8]: @eval QSAFT dᵢⱼ(T::Temperature, σᵢⱼ, εᵢⱼ) = σᵢⱼ.*(1.0 .- 0.12*exp.(-3*εᵢⱼ/(T+1e-90)));
```

```
[8]: dᵢⱼ (generic function with 1 method)
```

Next we need to define the $a_2$ function.

$$a_2 = -\frac{2\pi}{9}\frac{\rho}{(k_bT)^2}\sum_i\sum_j x_ix_j\frac{\alpha_{pi}\alpha_{pj}}{d_{ij}^3}J_{2,ij}$$

It's almost done as in the paper:

```
[9]: @eval QSAFT a₂(xᵢ,mᵢ,αₚᵢ,dᵢ³,dᵢⱼ³,ρ,β)=-2π/9*ρ*β^2*ΣᵢΣⱼxᵢxⱼαₚᵢαₚⱼdᵢⱼ⁻³(xᵢ,αₚᵢ,dᵢⱼ³)*I₂ᵢⱼ(ρ,xᵢ,mᵢ,dᵢ³);
```

```
[9]: a₂ (generic function with 1 method)
```

```
[10]: @eval QSAFT ΣᵢΣⱼxᵢxⱼαₚᵢαₚⱼdᵢⱼ⁻³(xᵢ,αₚᵢ,dᵢⱼ³)=sum(((i, j),)->xᵢ[i]*xᵢ[j]*αₚᵢ[i]*αₚᵢ[j]/dᵢⱼ³[i,j],Iterators.product(eachindex(xᵢ), eachindex(xᵢ)));
```

```
[10]: ΣᵢΣⱼxᵢxⱼαₚᵢαₚⱼdᵢⱼ⁻³ (generic function with 1 method)
```

Next we define the $a_3$ funcition.

$$a_3 = \frac{5\pi^2}{162}\frac{\rho^2}{(k_bT)^3}\sum_i\sum_j\sum_k x_ix_jx_k\frac{\alpha_{pi}\alpha_{pj}\alpha_{pk}}{d_{ij}d_{jk}d_{ik}}J_{3,ijk}$$

This is also defined almost as in the paper

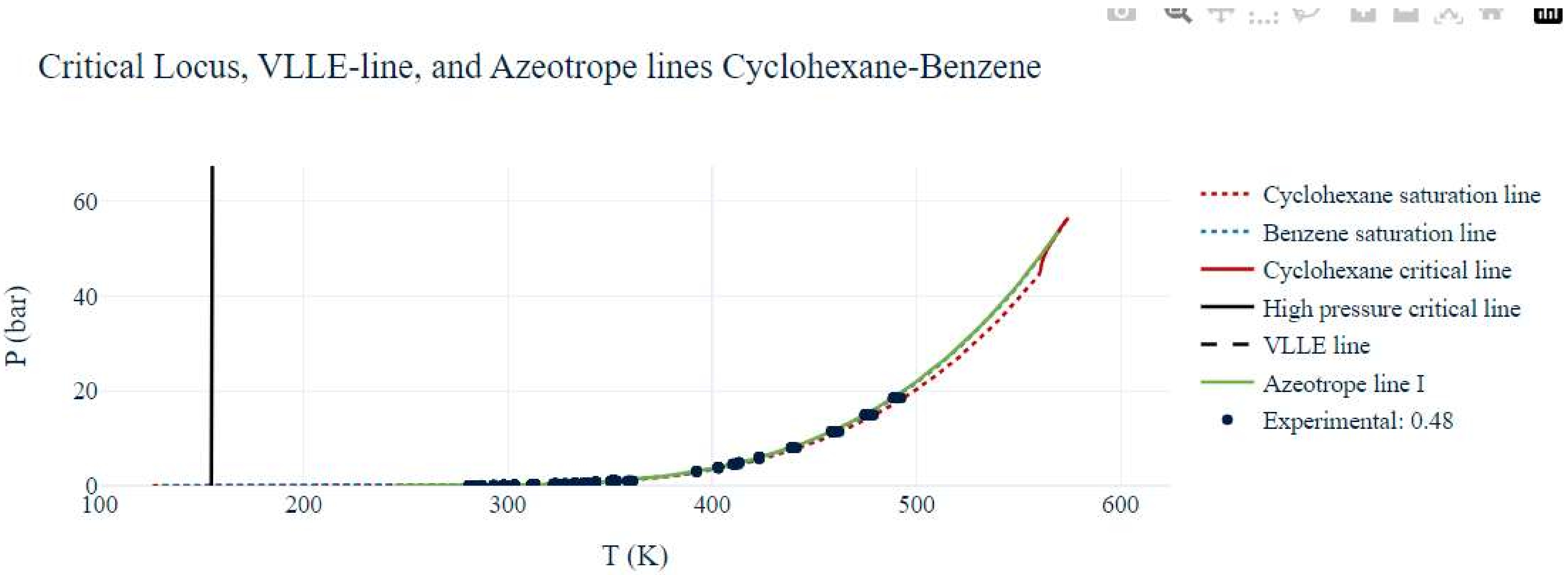
```
[11]: @eval QSAFT a₃(xᵢ,mᵢ,αₚᵢ,dᵢⱼ,dᵢ³,ρ,β)=5π^2/162*ρ^2*β^3*ΣᵢΣⱼΣₖxᵢxⱼxₖαₚᵢαₚⱼαₚₖdᵢⱼ⁻¹dᵢₖ⁻¹dⱼₖ⁻¹(xᵢ,αₚᵢ,dᵢⱼ)*I₃ᵢⱼₖ(ρ,xᵢ,mᵢ,dᵢ³);
```

```
[11]: a₃ (generic function with 1 method)
```

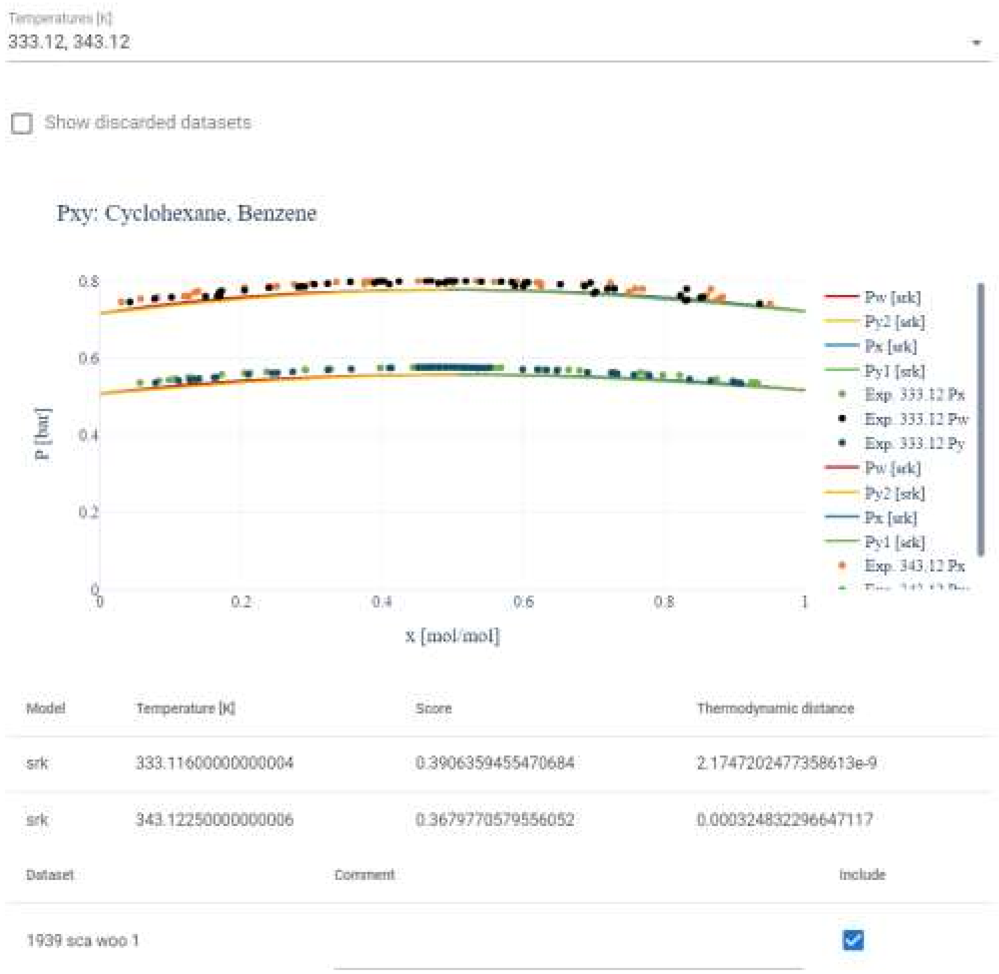
```
[12]: @eval QSAFT ΣᵢΣⱼΣₖxᵢxⱼxₖαₚᵢαₚⱼαₚₖdᵢⱼ⁻¹dᵢₖ⁻¹dⱼₖ⁻¹(xᵢ,αₚᵢ,dᵢⱼ)=sum(((i,j,k),)->xᵢ[i]*xᵢ[j]*xᵢ[k]*αₚᵢ[i]*αₚᵢ[j]*αₚᵢ[k]/(dᵢⱼ[i,j]*dᵢⱼ[i,k]*dᵢⱼ[j,k]),Iterators.product(eachindex(xᵢ), eachindex(xᵢ), eachindex(xᵢ)));
```

```
[12]: ΣᵢΣⱼΣₖxᵢxⱼxₖαₚᵢαₚⱼαₚₖdᵢⱼ⁻¹dᵢₖ⁻¹dⱼₖ⁻¹ (generic function with 1 method)
```

# Case study: Implementation of Polar PC-SAFT



# Case study: Implementation of Polar PC-SAFT

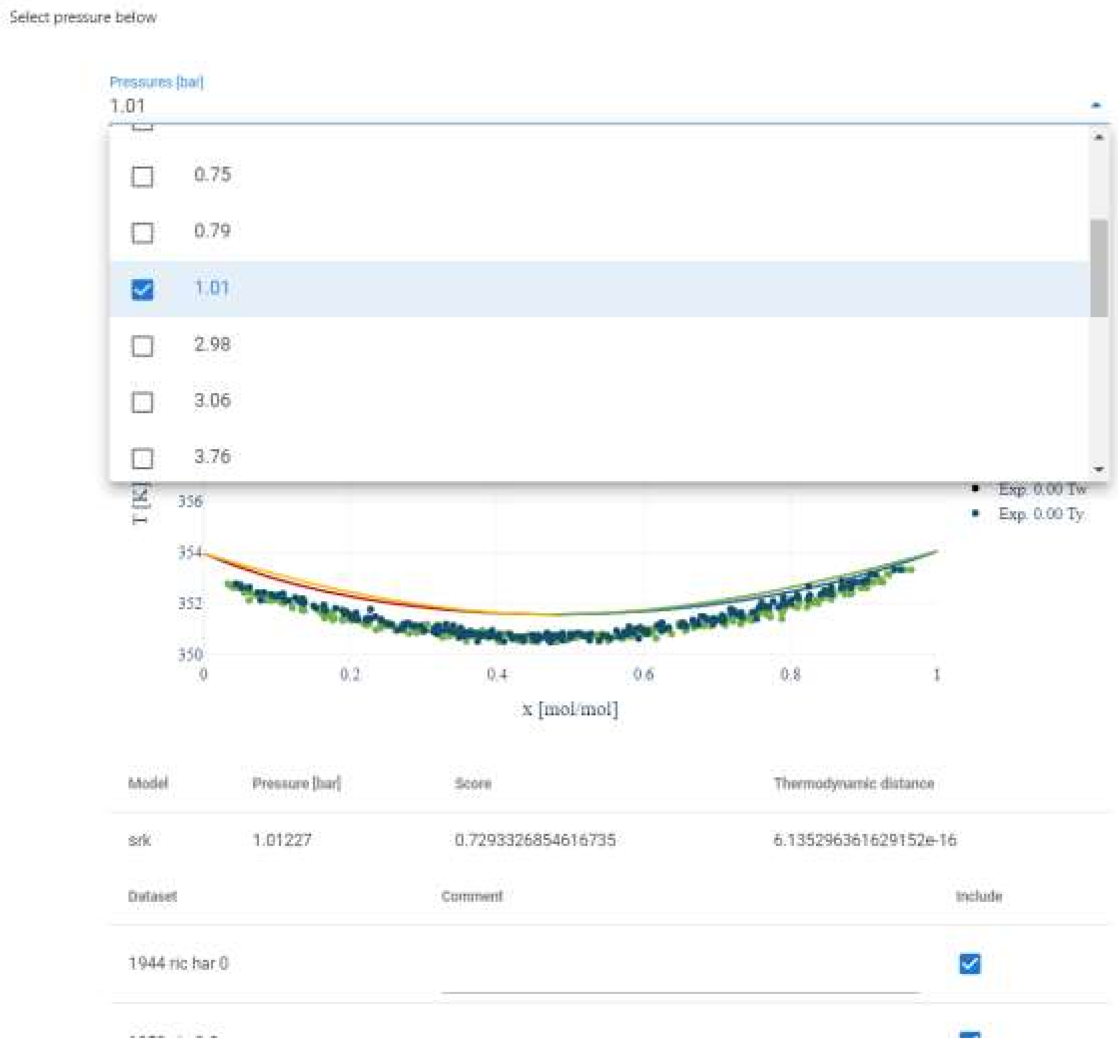


Temperatures [K]
333.12, 343.12

□ Show discarded datasets

| Model | Temperature [K] | Score | Thermodynamic distance |
|---|---|---|---|
| srk | 333.11600000000004 | 0.3906359455470684 | 2.1747202477358613e-9 |
| srk | 343.12250000000006 | 0.3679770579556052 | 0.000324832296647117 |

| Dataset | Comment | Include |
|---|---|---|
| 1939 sca woo 1 | | ☑ |



Select pressure below

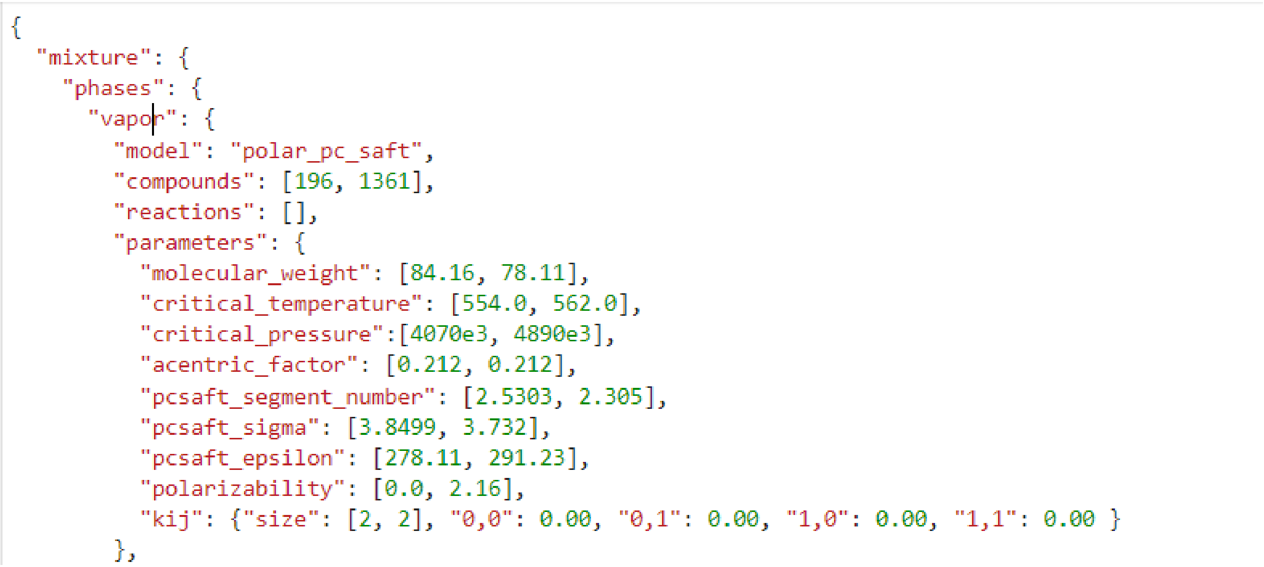
Pressures [bar]
1.01

□ 0.75
□ 0.79
☑ 1.01
□ 2.98
□ 3.06
□ 3.76

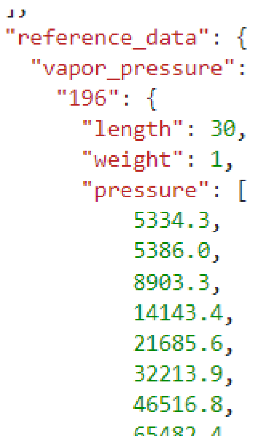
| Model | Pressure [bar] | Score | Thermodynamic distance |
|---|---|---|---|
| srk | 1.01227 | 0.7293326854616735 | 6.135296361629152e-16 |

| Dataset | Comment | Include |
|---|---|---|
| 1944 ric har 0 | | ☑ |

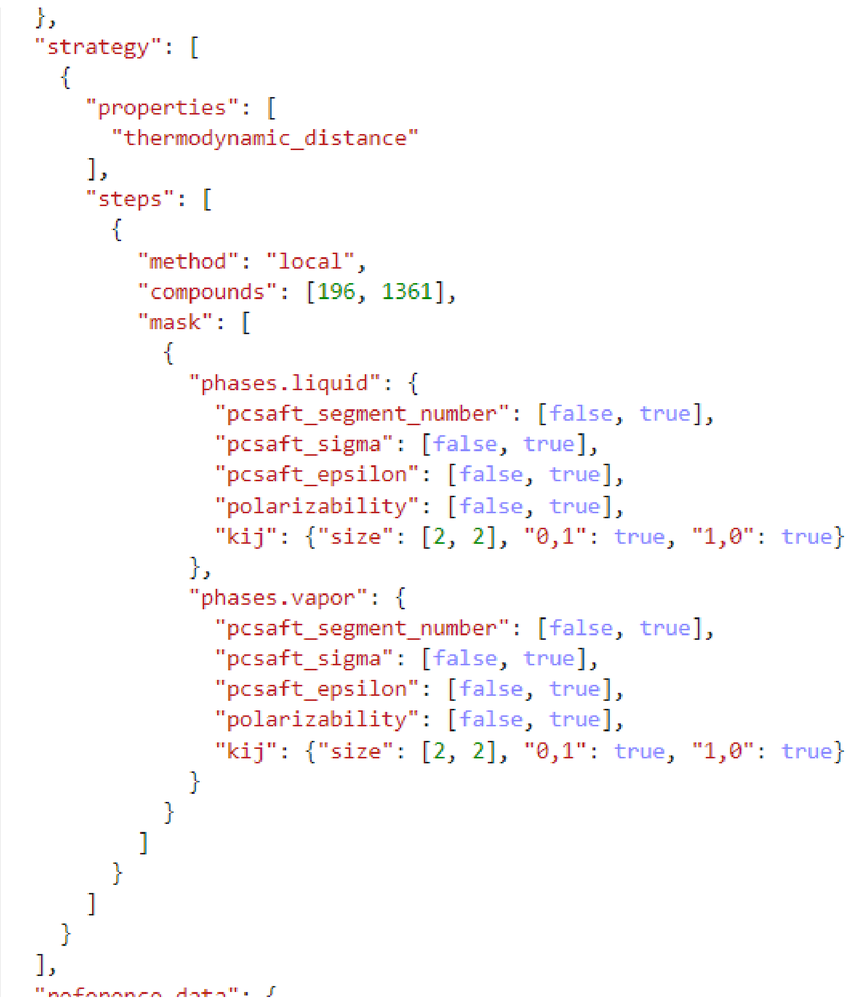
# Case study: Implementation of Polar PC-SAFT



```
{
  "mixture": {
    "phases": {
      "vapor": {
        "model": "polar_pc_saft",
        "compounds": [196, 1361],
        "reactions": [],
        "parameters": {
          "molecular_weight": [84.16, 78.11],
          "critical_temperature": [554.0, 562.0],
          "critical_pressure":[4070e3, 4890e3],
          "acentric_factor": [0.212, 0.212],
          "pcsaft_segment_number": [2.5303, 2.305],
          "pcsaft_sigma": [3.8499, 3.732],
          "pcsaft_epsilon": [278.11, 291.23],
          "polarizability": [0.0, 2.16],
          "kij": {"size": [2, 2], "0,0": 0.00, "0,1": 0.00, "1,0": 0.00, "1,1": 0.00 }
        },
```



```
  ]},
  "reference_data": {
    "vapor_pressure": {
      "196": {
        "length": 30,
        "weight": 1,
        "pressure": [
            5334.3,
            5386.0,
            8903.3,
            14143.4,
            21685.6,
            32213.9,
            46516.8,
            65482.4
```



```
},
"strategy": [
  {
    "properties": [
      "thermodynamic_distance"
    ],
    "steps": [
      {
        "method": "local",
        "compounds": [196, 1361],
        "mask": [
          {
            "phases.liquid": {
              "pcsaft_segment_number": [false, true],
              "pcsaft_sigma": [false, true],
              "pcsaft_epsilon": [false, true],
              "polarizability": [false, true],
              "kij": {"size": [2, 2], "0,1": true, "1,0": true}
            },
            "phases.vapor": {
              "pcsaft_segment_number": [false, true],
              "pcsaft_sigma": [false, true],
              "pcsaft_epsilon": [false, true],
              "polarizability": [false, true],
              "kij": {"size": [2, 2], "0,1": true, "1,0": true}
            }
          }
        ]
      }
    ]
  }
],
"reference_data": {
```

# Thank you

# Hafnium Labs

Predicting Chemistry