

Chemical 2.0

(Free open-source Modelica library)

Marek Matejak^{1,2}

¹Institute for Clinical and Experimental Medicine, Czech Republic

²First Faculty of Medicine, Charles University, Czech Republic

marek@matfyz.cz

Abstract

Free open-source Modelica library called Chemical 2.0 (<https://github.com/MarekMatejak/Chemical>) provides expressions between chemical substances and processes. These robust and unified definitions allow users to choose whether define processes or substances in their dynamic (electro-)chemical models. Propagation of substance definition and chemical solution through connected components simplify configuration. Chemical pathways can start even with unknown substances. Chemical kinetics was rewritten.

The possibilities and performance of chemical pathways modeling are increased using a new type of connectors based on inertial electro-chemical potential. Chemical processes can be directly connected without need to add insignificant states. Parameterization of chemical reactions is also streamlined, e.g. using forward rate and dissociation coefficient.

Keywords: Chemical 2.0, inertial electro-chemical potential, Modelica library, physical chemistry, thermodynamics equilibria, electrochemical potential, electrochemical cell, internal energy, semipermeable membrane, chemical kinetics, chemical pathways

1 Introduction

Motivation to improve Chemical library from version 1.4 (Matejak et al. 2015) to version 2.0 comes from following Modelica libraries:

- A) Media.IdealGases from Modelica Standard Library (Casella et al. 2006): Chemical substances and chemical processes can be described by the selected extensive physical quantities.
- B) ThermoFluidStream (Zimmer 2020): Inertial connectors can be generalized for chemical domain.

As selection of physical quantities for each substance and process definition we chose a similar approach used in Modelica.Media.IdealGases, which comes from (McBride, Zehe, and Sanford 2002). Having this

definition as a record of constant coefficients, there is a possibility to calculate molar heat capacity, free molar formation energies, free molar entropy and many other thermodynamic properties in dependence on given substance activity and state of chemical solution (temperature, pressure, electric potential, etc.). Moreover, unlike the previous version, there is no need to have separate calculations for different states of matters (phases), because phase is also included in extensive physical quantities of the definition records. Therefore, every chemical equilibrium of each process can be expressed only using the definitions of included substances and processes (Peter Atkins, De Paula, and James Keeler 2018). And because we chose extensive quantities, they can be easily added or subtracted as in common chemical processes notations (e.g. mass of substrates = mass of products; charge of substrates = charge of products; enthalpy of substrates + enthalpy of reaction = enthalpy of products, entropy of substrates + entropy of reaction = entropy of products, etc.).

The connectors from ThermoFluidStream library are revolutionary (Zimmer, Bender, and Pollok 2018). This library still allows user to define very complex thermodynamic behavior of predefined media in the same level of detail as in Modelica.Fluid and Modelica.Media does (Zimmer, Meißner, and Weber 2022). However, ThermoFluidStream library permits connections between processes without unnecessary accumulations of media, simultaneously eliminating a number of non-linear systems in the model (Meißner and Zimmer 2022). By eliminating numerous middle-state components accumulating and mixing medium the model becomes more readable. And during the simulation run there is no longer need to solve implicit equations for many non-linear systems. All this results in significant improvement of the performance and readability, with minimal compromise in the model details (Zimmer, Meißner, and Weber 2021).

Using the inertial approach of ThermoFluidStream in chemical domain promises the analogical improvements for Chemical 2.0. As a result, the processes can be connected directly together to define complex chemical

pathways such as in biochemistry or physiology. Additionally, allowing to define better chemical kinetics (Sauro 2009), complex processes and more detailed calculations of any part of the model.

The inertial approach is connected with following basic composition rules:

- Electro-chemical pathways shall start and end at a boundary component. Enabled substance ports must be connected.
- Connecting more than two substance ports together should use splitters and junctions from package Topology. Splitting directly using the connector is only recommended for the use of sensors.
- Each cyclic flow must be broken by at least one Substance component.

2 Definition

The previous version of Chemical library has various definitions of substances. In version 2.0, all these structures for all type of substances and even for all type of chemical processes are unified into one type of operator record called `Chemical.Interfaces.Definition`. This record can define each substance and process used in the previous version. In addition, the user can set their values independently on internal representation using suitable constructors or the function `processData`. This function creates process definition based on dissociation constant (molar-based) and consumed heat of the process (free molar enthalpy change) (**Figure 1**). Using relation between this process definition and its substances definitions, the user can easily evaluate new definition of the new chemical substance or new chemical process using expressions with `*`, `+` and `-` operators.

For example, a definition of aqueous O₂ can be set from gaseous O₂ and water dissolution process with tabulated Henry's coefficient (0.0013, 1500K). For another example, a definition of H₂O formation reaction from H₂ and O₂. Each new definition can be set as an algebraic equation between reaction, products and substrates (**Listing 1**).

Listing 1. Example of definitions

```
import Chemical.Interfaces.Definition;
import Chemical.Substances.Gas;
import Chemical.Substances.Liquid;
import Chemical.Interfaces.processData;
constant Real R = Modelica.Constants.R;

constant Definition O2_aq =
  Gas.O2 + processData(
    K = 0.0013,
    dH = -1500*R);
```

```
constant Definition H2O_formation =
  Gas.H2O - (Gas.H2 + 0.5*Gas.O2);

constant Definition Hemoglobin =
  Liquid.Unknown;
```

Each chemical pathway can start with Unknown substance. And next substance can be defined relatively by process definition. The default definition of Unknown substance is molar mass 1kg/mol and molar heat capacity 1 J/(mol.K), with zero free formation Gibbs energy and zero free formation enthalpy. These values are used if the user does not specify the substance. Because free formation energies are always defined in a relative manner, unknown substances can be used, such as any base substance with unknown formation process. If they play a role in the solution properties, at least definition data in relation with the changing solution properties should be set. E.g. molar mass and molar volume must be set if they play a role in solution composition; molar heat capacity must be set if it plays a role in heat accumulation in the chemical solution during simulation.

Figure 1. Inputs of processData function

| Inputs | | |
|--------|--------------------------------|---|
| K | <input type="text" value="1"/> | Process dissociation constant (mole-fraction based) at 25°C, 1bar |
| dH | <input type="text" value="0"/> | Process molar enthalpy change at 25°C, 1bar |
| dCp | <input type="text" value="0"/> | Process molar heat capacity change at 25°C, 1bar |
| dVm | <input type="text" value="0"/> | Process molar volume change at 25°C, 1bar |

3 Chemical solution

There are two options for how to define a chemical solution. The first involves the `Chemical.Solution` component and connecting all substances through its solution connector. This was the only option in the previous version, which was retained in this version. If a chemical solution is defined this manner, then all its properties are dynamically calculated during the simulation.

Listing 2. Example of fixed solution state

```
import Chemical.Interfaces.SolutionState;
import Chemical.Interfaces.Phase;

constant SolutionState SATP =
  SolutionState(phase = Phase.Gas,
    T = 298.15);

SolutionState heatingSolution =
  SolutionState(phase = Phase.Gas,
    T = 273.15+time);
```

However, there is not always a need to change the solution properties. If solution properties, such as temperature, pressure, total mass, total volume, total amount of particles, and electric potential, are assumed as constant during simulation, then the solution state (**Listing 2**) can

be set as a parameter. If the user did not set a solution property, then its default value (Figure 2) will be used.

Figure 2. Default chemical solution properties

| | | | |
|---|--------------------------------------|-----|--|
| T | <input type="text" value="20"/> | °C | Temperature of the solution |
| p | <input type="text" value="1.01325"/> | bar | Pressure of the solution |
| m | <input type="text" value="1"/> | kg | Mass of the solution |
| V | <input type="text" value="1"/> | l | Volume of the solution |
| n | <input type="text" value="1"/> | mol | Amount of the solution |
| v | <input type="text" value="0"/> | V | Electric potential in the solution |
| G | <input type="text" value="0"/> | J | Free Gibbs energy of the solution |
| Q | <input type="text" value="0"/> | C | Electric charge of the solution |
| I | <input type="text" value="0"/> | 1 | Mole fraction based ionic strength of the solution |

4 Properties of chemical processes

Each process has different properties (e.g. dissociation coefficient, free Gibbs energy or free enthalpy) in dependence on its definition and on the current state of chemical solution (e.g. temperature, pressure, etc.). The user needs to connect the substance definition and solution to model `Chemical.Interfaces.ProcessProperties` to evaluate these properties.

Listing 3. Example of process properties

```
import Interfaces.ProcessProperties;

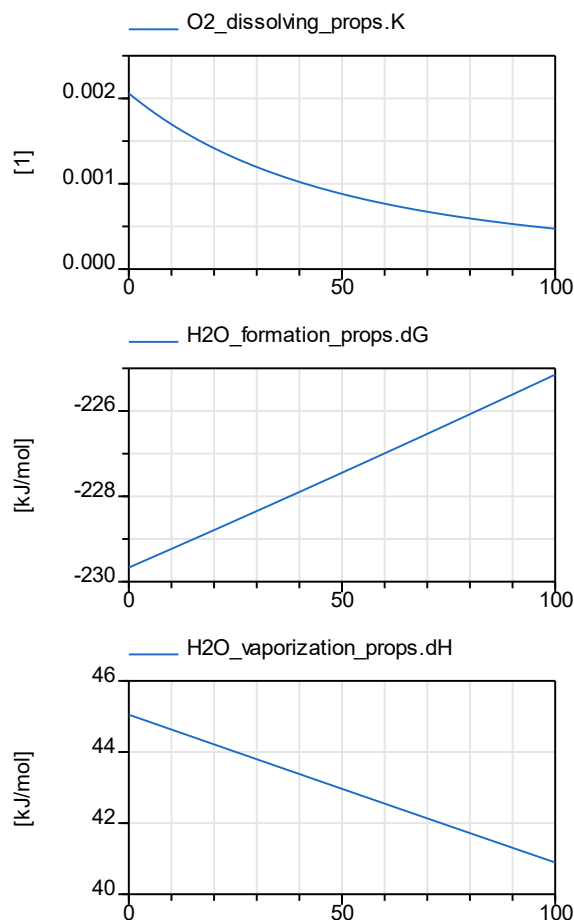
ProcessProperties O2_dissolving_props
(definition = processData(0.0013, -1500*R),
 solutionState = heatingSolution);

ProcessProperties H2O_formation_props
(definition = H2O_formation,
 solutionState = heatingSolution);

ProcessProperties H2O_vaporization_props
(definition = Gas.H2O - Liquid.H2O,
 solutionState = heatingSolution);
```

After the simulation of code (Listing 1-3) for 100 seconds, the temperature of the solution changed from 0°C to 100°C, and we can see the recalculated properties that resulted from temperature change such as dissociation constants (Henry's coefficient), free Gibbs energy change of reaction or consumed heat by process (free enthalpy change) – e.g. Figure 3.

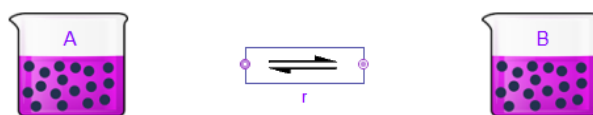
Figure 3. Results of simulation (Listing 1-3) per temperature [°C]



5 Chemical process

To create a model of a chemical reaction using Chemical 2.0, drag and drop component Substance from package `Chemical.Boundaries` as substance **A** and substance **B**. In the same way, put into the model Reaction as process **r** from package `Chemical.Processes` (Figure 4).

Figure 4. Components of simple chemical reaction



The default setting for the substances does not provide any connector, so it is necessary to setup which connectors will be used. This is done in parameter dialog by selecting the *useFore* checkbox for substance **A** and *useRear* for substance **B** (Figure 5).

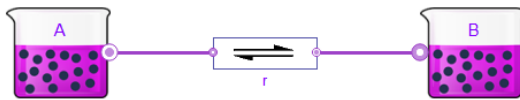
Figure 5. Configuration of substance connectors

Conditional inputs

| | |
|-------------|-------------------------------------|
| useRear | <input checked="" type="checkbox"/> |
| useFore | <input type="checkbox"/> |
| useSolution | <input type="checkbox"/> |

Having all necessary connectors, it is possible to connect substances with chemical reaction r (Figure 6).

Figure 6. Connections



This model could be simulated, but everything remains constant because the dissociation coefficient is equal to one and both substances are the same (defined as Liquid.Unknown). However, if we change the dissociation coefficient K to 2 (as shown in Listing 4) then we can see dynamic behavior in the default chemical solution. And after 5s of simulation time, the reaction reaches almost the equilibrium where the amount of product B is two times higher than the amount of substrate A . We can change reaction speed by setting forward reaction rate coefficient $r.k_{forward}$ lower than its default value. Backward reaction rate is not possible to set as a reaction parameter because it is defined as $k_{forward}/K$. This reaction can have as many substrates (nS) and as many products (nP) as needed, and its index is defined by the order of drawn connections. Note, that default chemical solution in this example is set to Figure 2 and remains constant during simulation.

Listing 4. Generated code for a reaction example

```
model SimpleReaction
  extends Modelica.Icons.Example;

  import Chemical.Boundaries.Substance;
  import Chemical.Processes.Reaction;
  import Chemical.Interfaces.processData;

  Substance A( useFore = true );
  Substance B( useRear = true );

  Reaction r( nP = 1, nS = 1,
    process = processData(2) );

equation
  connect( A.fore , r.substrates[1] );
  connect( r.products[1] , B.rear );
end SimpleReaction;
```

If the user prefers products definition before definition of process, then it is possible to set process option *firstProductFrom* into “Substance” selection. This enables the parameter *firstProduct* (the first product definition) in the same dialog as the process setting. With knowledge of all products and all substrate definitions the process definition can be evaluated, so the process definition parameter is disabled during this selection. Please note, that each product is by default defined by process, so it is not necessary to define it again in following process or boundary component.

Figure 7. Process defined by substances

firstProductFrom

Process - First product is defined by process
 Substance - First product definition is a parameter

firstProduct

dropOfCommons.DefaultSubstance

HCL - HCL(g)
 H2 - H2(g)
 H2O - H2O(g)
 H2O2 - H2O2(g)
 He - He(g)
 NO - NO(g)
 NO2 - NO2(g)
 N2 - N2(g)
 O2 - O2(g)
 Unknown - Unknown incompressible substance

In addition, stoichiometric coefficients can be set as parameter s for substrates and parameter p for products as in previous version. The new extension for processes allows to change chemical kinetics as redefinition of relation between electro-chemical gradient and process molar rate called $uLoss$. By default, $uLoss$ is set to traditional chemical kinetics based on forward rate coefficient, but the user can define their own type of chemical kinetics constrained by proposed interface for potential loss functions.

Listing 4. Example of reaction defined by product with uncommon chemical kinetics

```
import Chemical.Utilities.Types.
  FirstProductChoice;
import Chemical.Processes.Internal.Kinetics;

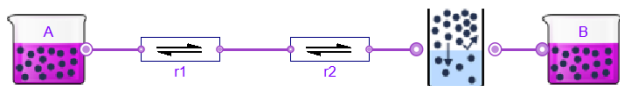
Chemical.Processes.Reaction H2_burning (
  firstProductFrom =
    FirstProductChoice.Substance,
  firstProduct = Gas.H2O,
  nS=2, s = {2,1},
  nP=1, p = {2},

  redeclare function uLoss =
    Kinetics.fastPotentialLoss,
)
```

6 Data propagation

The version 2.0 of the Chemical library is based on a special type of substance ports. These connectors propagate the state of substance without inertial part in flow direction and generate equality of inertial electro-chemical potential, instead of direct equality of the total electro-chemical potential. The equilibria and even the dynamic simulation reach almost the same values if inertia is fast enough. As proven by (Zimmer 2020) this type of connector has physical background and better performance because it eliminates non-linear systems of equations where solvers typically spend the most of time. Moreover, this design allows users to connect more processes together without any accumulation of substance (or just a few Substance components). As a result, it is possible to define long chemical pathways as are usual in physiology.

Figure 8, Example of possible pathway connection



To have a chemical pathway such as in **Figure 8** there is a need to have each substance definition between each chemical process. This is done automatically using a propagation pattern from forward connectors (called fore) into rearwards connectors (called rear). Please note that substances definition propagation is not dependent on flow direction. Each substance definition remains constant during simulation, so the direction of definitions propagation does not take any role during simulation and vice versa. In **Figure 8** only the substance **A** can be defined by user. The definition of a product of reaction **r1** is evaluated from the definition of this reaction and its substrates. So generally, each process can define the first product from its other products, substrates, and process definition. Even the substance **B** does not have to be explicitly defined, because its definition can come from PhaseTransition process.

In the same way, the fixed state of the chemical solution can be propagated. If the state of chemical solution is constant, it remains the same for each substance in the solution during each step of the simulation. In the model shown in **Figure 8** it means that chemical solution is specified only in substance **A** and propagated through connections to each other components as the same constant values.

To model the situation when processes cross different solutions, there is an option to break the solution propagation. In some processes, such as Diffusion, Membrane or PhaseTransition, the solution states are not

propagated by default because they change the chemical solution between substrates and products. As a result, the new solution must be defined for the process and its product to propagate into next processes. Please note, that “next” here is not defined by molar flow but by fore-rear connections.

Listing 5. Example of process changing chemical solution

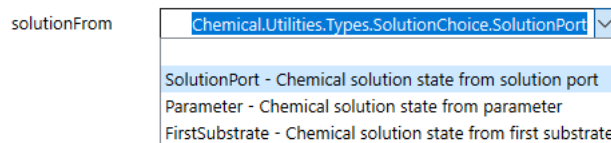
```
import Chemical.Processes.PhaseTransition;
import Chemical.Interfaces.SolutionState;
import Chemical.Interfaces.Phase;
import Chemical.Substances.Gas;

PhaseTransition PhaseTransition(
    solutionParam =
        SolutionState( phase = Phase.Gas ),
    Product = Gas.CO2 );
```

Diffusion and Membrane process have the same substance on both sides. PhaseTransition changes the substance. This allows to use it even for solvation (gas dissolution in liquids and gas volatility from liquids) or for phase transition (vaporization, condensation, sublimation, deposition, melting and freezing). Note that the definition of each chemical compound is dependent on phase. For example, gaseous water was defined differently from liquid water. Phase transition of a substance is a chemical process, which even consumes/releases heat (has non-zero free enthalpy change).

Please note, that if a chemical substance significantly changes the chemical solution properties, then it must be connected to Solution component using **SolutionPort** instead of fixed parametrized solution propagation. This situation is selected by choice **solutionFrom** (**Figure 9**).

Figure 9. If substance change the solution



7 Initialization

Substance component is initialized by initial mass or amount of base substance molecules. Recalculation from number of base molecules and mass is obtained through molar mass, which is part of the substance definition. Switching between mass-based and molar-based accumulations is based on **preferMass** checkbox (**Figure 10**). If mass is preferred, **amountOfSubstance_start** is disabled and **mass_start** is enabled as an initial value of substance.

Figure 10. Substance amount initialization

Substance

| | | |
|-------------------------|-------------------------------------|-----|
| substanceDefinition | Chemical.Substances.Solid.Pb | |
| preferMass | <input checked="" type="checkbox"/> | |
| mass_start | 0.2072 | kg |
| amountOfSubstance_start | 1 | mol |

A principle of the inertial connectors affects the initialization of the model processes. Having acceleration of molar flows on each connection, each flow has a state. Therefore, the simplest process initialization is to set values of molar flows (**Figure 11**). From these values the gradients of electro-chemical potentials on process components are calculated as a relations of substance states between boundaries.

Figure 11. Molar flow initialization

| | |
|--|---|
| Chemical.Utilities.Types.InitializationMethods.steadyState | ▼ |
| none - No initialization | |
| steadyState - Steady state initialization (derivatives of states are zero) | |
| state - Initialization with initial states | |
| derivative - Initialization with initial derivatives of states | |

Alternatively, it is possible to set the initial process state to steady state. This will take inertial electrochemical potential change r to zero. If each process has this type of initialization, then the solver is looking for the non-accelerated state of whole system, where derivation of each flow is zero.

8 Discussion

ThermofluidStream is a proof of concept that even a complex models could improve usability, performance, details and interfaces. We hope that the update of Chemical library from version 1.4 to version 2.0 marks also this kind of a progression.

Despite the complexity of this code containing many physical equations and parameters, the user interface is simplified significantly. It is permitting users to start with basic models without any significant alterations to the default component settings. And it works, because common (electro-)chemical assumptions are hidden as these default values. Starting with a very simple model and extending significant details during each modeling step leads to optimal model development with minimal effort. This development strategy can be reversed if something does not work – model can be simplified and results should be also well interpretable.

Theory behind Chemical library at the level of physical chemistry is summarized in documentation found at: Chemical\Resources\Documentation\Chemical.pdf.

Acknowledgements

Special thanks to Taras (Terry) Yavorskyy and Dr. Jason Brown, who contributed by proof reading this article for English grammar and syntax.

References

- Casella, Francesco, Martin Otter, Katrin Proelss, Christoph Richter, and Hubertus Tummescheit. 2006. "The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-Fluid Pipe Networks." In *Proceedings of the 5th International Modelica Conference*, 631–40.
- Matejak, Marek, Martin Tribula, Filip Jeezek, and Jirı Kofranek. 2015. "Free Modelica Library of Chemical and Electrochemical Processes." In *11th International Modelica Conference, Versailles, France*, 118:359–66. Linkoping University Electronic Press, Linkopings universitet.
- McBride, BJ, MJ Zehe, and G Sanford. 2002. "Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species." *Report: NASA/TP 211556*.
- Meißner, Michael, and Dirk Zimmer. 2022. "Robust Modeling of Volumes for Dynamic Simulations of Thermo-Fluid Stream Networks." *IFAC-PapersOnLine* 55 (20): 265–70.
- Peter Atkins, P, J De Paula, and James Keeler. 2018. *Atkins' Physical Chemistry*. OUP Oxford.
- Sauro, Herbert M. 2009. "Network Dynamics." *Computational Systems Biology*, 269–309.
- Zimmer, Dirk. 2020. "Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks." *Mathematical and Computer Modelling of Dynamical Systems* 26 (3): 204–33.
- Zimmer, Dirk, Daniel Bender, and Alexander Pollok. 2018. "Robust Modeling of Directed Thermofluid Flows in Complex Networks." In *Proceedings of the 2nd Japanese Modelica Conference*, 39–48. Linkoping University Press.
- Zimmer, Dirk, Michael Meißner, and Niels Weber. 2021. "Robust Simulation of Stream-Dominated Thermo-FluidSystems: From Directed to Non-Directed Flows." *SNE Simulation Notes Europe* 31 (4): 177–84.
- Zimmer, Dirk. 2022. "The DLR Thermofluid Stream Library." *Electronics* 11 (22): 3790.