Instructions for implementing new working pair isotherms into SorpPropLib

Prepared by Zhiyao Yang, 9/14/2017

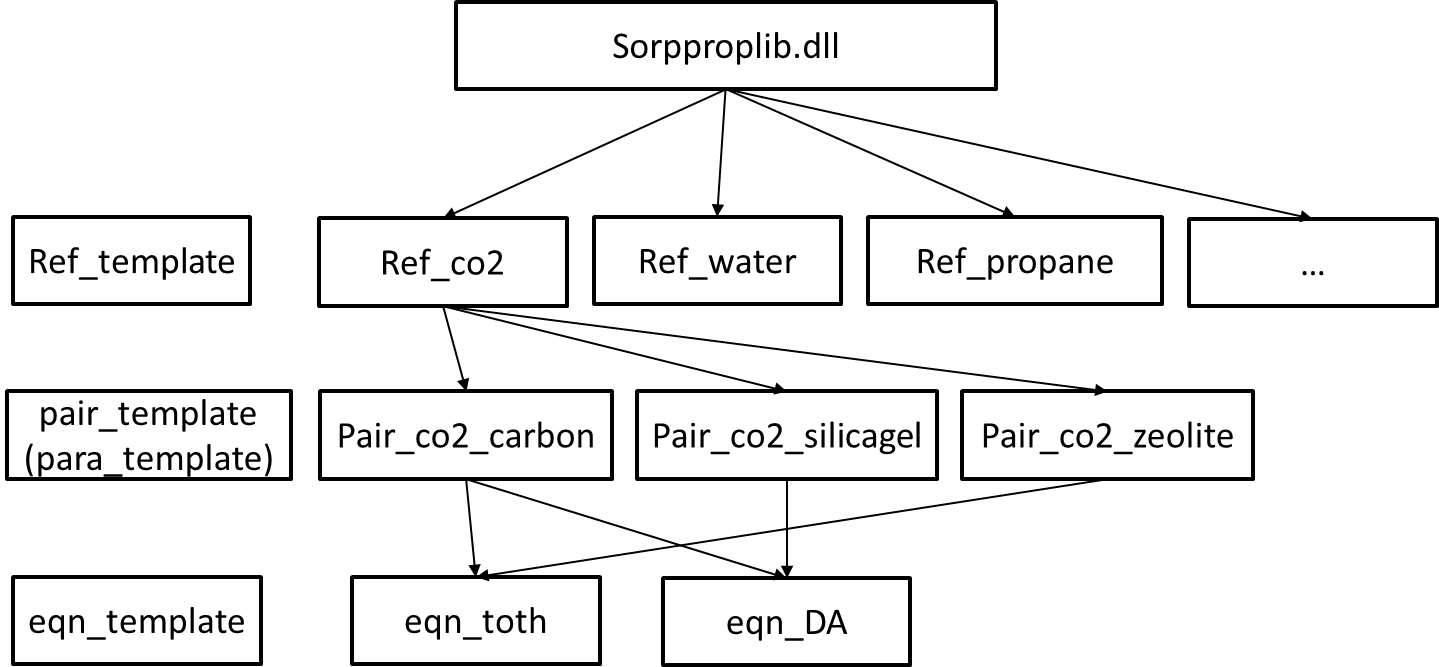
1. **General information**

SorpPropLib is the library of sorption isotherm correlations, and it was developed using only standard C++ libraries, therefore it can be linked with other program without any supporting files from Qt. However, the current version is compiled in Qt, therefore if further development is carried out also in Qt, the current Qt-project setting can directly be used.

GitHub repo: [www.github.com/zhiyaoyang/sorpproplib](http://www.github.com/zhiyaoyang/sorpproplib)

1. **Program structure and program flow**

There are many classes defined for layers of the database following the refrigerant-working pair order, and they are structured as in the following figure:



Sorppropplib.dll is the entry point of the database, which takes two std::string for identification of the working pair, and two double numbers as the temperature and the mass ratio input.

Based on the refrigerant input (the first std::string input), different refrigerant classes are called. They are the first layer of the database. The refrigerant class further directs the program flow into working pair classes based on the sorbent information (the second std::string input).

In the working pair class, specific sub-type of sorbent is used to finally point to the data entry of that pair: a data object that contains the type of correlation equation (std::string myEqn), reference, and the coefficient values. The equation form and the coefficients are passed back to the refrigerant class to calculate the vapor pressure.

Then the type of equation form, the reference, and the calculated vapor pressure are passed back to sorpproplib.dll to return to whatever program it is linked to.

All the refrigerant classes are developed based on the ref\_template class with data struct for passing result, as well as basic variables and functions for a refrigerant class.

All working pair classes are developed based on the pair\_template class with the template for coefficient parameter values (para\_template) is also included. The pair\_template also defines basic variables and function for working pair classes.

All equation classes are developed based on the eqn\_template class, which currently only declares a calculation function. Each equation class has its own parameter data struct defined based on the para\_template to contain equation-type-specific data from working pair class.

1. **Implementation instruction**
   1. A new refrigerant

* Create a new refrigerant class inheriting the ref\_template, and following the example of existing refrigerant classes;
* Once the related pair classes are implemented (see below “a new working pair”), include the working pair classes in the header;
* In the calc() function, for each sub-type, extract the sub-type string and substantiate a corresponding working pair class with the sub-type string as the argument;
* In calc() following the branching of sub-type, the calculation part is the same as existing examples;
  1. A new working pair
* Create a new working pair class inheriting the pair\_template, and following the examples of existing working pair classes;
* If the involved equation form has been implemented, directly include the equation’s header in the new working pair header. Otherwise, see below “a new equation” to create a new equation form;
* In the init() function, follow the example of “pair\_water\_libr.cpp”:
  + Get sub-type from “mySubType” and branch to each sub-type;
  + Define the object “myEqn” with (1) equation form, (2) reference;
  + Create a new equation parameter object (para\_equationName);
  + Insert corresponding values from literature to the data object;
  + Assign the new data object to “myPara”;
  + If the equation form is “custom”, then only define the “myEqn” object and move to next step;
* In the calc() function:
  + Get equation form from “eqn”;
  + Branch to each equation form, then create a corresponding new equation object;
  + Assign the new equation object to “mEqn”;
  + If the equation form is “custom”, follow the example of “pair\_water\_cacl2.cpp”:
    - Branch the subtype using “mySubType”;
    - Define the parameters according to the literature;
    - Directly calculate the vapor pressure following the literature;
    - Return the result pressure;
  1. A new equation
* Create a new equation class inheriting the eqn\_template class, and following the examples of existing equation classes;
* Create a new parameter struct inheriting para\_template, list all input parameters related to this equation form inside;
* If the original literature has vapor pressure as the final result, follow the example of “eqn\_toth.cpp”:
  + implement calcY() function using both the input parameters in the parameter struct as well as the temperature/mass ratio input from user, following the procedure in the literature, and return the Y at the end;
  + implement calc() function as a search routine to find the pressure with the user input and the calcY() function;
* If the original literature has mass ratio as the final result, follow the example of “eqn\_antoine.cpp” and only implement calc() according to the literature.

1. **Notes:**

Currently the framework work with simple equation forms, for those absorption working pairs using “EOS”, “activity coefficient” or “mixing rules”, the framework is not complete yet. Therefore please first focus on implementing the other working pairs while I work out the framework for those types of equations.

1. **Example of implementation:**

Refrigerant: “CO2”

Adsorbent: “Zeolite:5A”

Equation name: “Toth”

Equation form:

Inputs from database:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Refrigerant** | **adsorbent** | | **q\_s** | **b\_0** | **q\*/R** | **n\_0** | **C** | **m** | **r** | **Literature** |
| CO2 | Zeolite | 5A | 642.4 | 6.761e-8 | 5.625e3 | 2.7e-1 | -2.002e1 | 1 | n | Wang, Y. and M.D. LeVan, Adsorption equilibrium of carbon dioxide and water vapor on zeolites 5A and 13X and silica gel: pure components. Journal of Chemical & Engineering Data, 2009. 54(10): p. 2839-2844. |

Inputs from user: T, Y.

Output to user: P.

Procedure:

1. If there is no **[eqn\_toth]** class, create one inheriting [eqn\_template] class as parent, and follow the sub-steps below. If there is already a [eqn\_toth] class, jump to step 2>;
   1. In [eqn\_toth], create a struct [para\_toth] inheriting [para\_template];
   2. Declare data members in the struct according to the “input from database” in the table above;
   3. If the equation above has “Y” as the final outcome:
      1. Define [double calcY()] according to the equation, taking [para\_toth], [tK], [pKpa] as argument;
      2. Define [double calc()] to solve for [P] taking [para\_toth], [tK], [xMass] as argument, using the [calcY()] function;
   4. If the equation above has “P” as the final outcome, define [double calc()] according to the equation taking [para][tK][xMass] as the argument;
2. If there is no **[pair\_CO2\_Zeolite]** class, create one inheriting [pair\_template] class as parent, and follow the sub-steps from a. below. Else, jump to sub-step x. below;
   1. Add an empty constructor inheriting the constructor from [pair\_template] class;
   2. Add definition of function [init()]:
      1. Branch using string [mySubType], define branch for “5A”;
      2. Assign equation name and reference to [myEqn]: {“Toth”, reference from “input from database” in the table above};
      3. Create a new [para\_toth];
      4. Assign parameter values according to the “input from database” in the table above;
      5. Add the [myEqn] into the vector [myEqns];
   3. Add definition of function [calc()]:
      1. Branch using string [eqn];
      2. For “Toth”, create new [eqn\_toth] and point the [eqn\_template \* mEqn] to it;
      3. After all branches, return the result of [mEqn->calc()];
3. If there is no **[ref\_CO2]** class, create one inheriting [ref\_template] class as parent, and follow the sub-steps below. Else, jump to sub-step iii. below.;
   1. Add an empty constructor inheriting the constructor from [ref\_template] class;
   2. Add definition of function [double calc()]:
      1. Declare a string [sType];
      2. Create a pointer of [pair\_template] [\*p] and point to NULL;
      3. Branch using string [mySorbate], define branch for “Zeolite:5A”;
      4. Set string [sType] as the sub-type of Zeolite, i.e. “5A”;
      5. Create a new [pair\_CO2\_Zeolite] with current [sType] as argument, and have the [\*p] point to it;
      6. After all branches, if [\*p] is not NULL, call [calcEquations()] with [\*p] as argument;
4. In **[sorpproplib.cpp]**, in function [calc()], if there is no branch for [ref==”CO2”], include [ref\_CO2.h], add one branch, and in the branch point the [ref\_template \*refPtr] to a new [ref\_CO2] object.

The results of the process above can be refer to following files in existing code:

* Sorpproplib.cpp
* Ref\_co2.cpp/.h
* Pair\_co2\_zeolite.cpp/.h
* Eqn\_toth.cpp/.h

1. **Data to be implemented:**

Equation name: “Dubinin-Astakov”

Equation form:

14er[8-994.vn/.hbe refer to following files in existing code:uation taking [para

Inputs from database:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Refrigerant** | **Adsorbent** | | **W0** | **E** | **n** | **Literature** |
| Ethanol | Composite | SG-LiBr | 0.00068 | 6900 | 1.8 | Brancato, V., et al., *Ethanol adsorption onto carbonaceous and composite adsorbents for adsorptive cooling system.* Energy, 2015. **84**: p. 177-185. |
| Carbon | AC-SRD-1352/3 | 0.00082 | 8780 | 1.5 |
| AC-AP4-60 | 0.00045 | 10600 | 2 |
| AC-ATO | 0.00061 | 11200 | 1.7 |
| AC-COC-L1200 | 0.00044 | 13300 | 2 |
| ACF-FR20 | 0.00075 | 13500 | 2 |
| Methanol | Carbon | AC-G32-H | 0.000482 | 19220 | 2.59 | Henninger, S., et al., *Evaluation of methanol adsorption on activated carbons for thermally driven chillers part I: Thermophysical characterisation.* international journal of refrigeration, 2012. **35**(3): p. 543-553. |
| AC-Norit-R-1-Extra | 0.000519 | 17380 | 2.27 |
| AC-RUTGERS-CG1-3 | 0.000535 | 14260 | 1.8 |
| AC-Norit-RX-3-Extra | 0.000551 | 16890 | 2.06 |
| AC-Carbotech-C40/1 | 0.000633 | 12460 | 1.85 |
| AC-Carbotech-A35/1 | 0.000786 | 11720 | 1.76 |
| R-507A | Carbon | Charcoal-Maxsorb III | 0.001175 | 5740 | 1.47 | Saha, B.B., et al., *Adsorption of equal mass fraction near an azeotropic mixture of pentafluoroethane and 1, 1, 1-trifluoroethane on activated carbon.* Journal of Chemical & Engineering Data, 2008. **53**(8): p. 1872-1876. |
| R-32 | Carbon | AC-pellet-Maxsorb III | 0.00405 | 3939 | 1.15 | Askalany, A.A., et al., *Adsorption isotherms and heat of adsorption of difluoromethane on activated carbons.* Journal of Chemical & Engineering Data, 2013. **58**(10): p. 2828-2834. |
| ACF-A-20 | 0.00458 | 4098 | 1.09 |
| R-134a | Carbon | Charcoal-Chemviron | 0.000279 | 14870 | 1.60 | Akkimaradi, B.S., et al., *Adsorption of 1, 1, 1, 2-tetrafluoroethane on activated charcoal.* Journal of Chemical & Engineering Data, 2001. **46**(2): p. 417-422. |
| Charcoal-Fluka | 0.000449 | 8897 | 0.95 |
| Charcoal-Maxsorb | 0.001548 | 8269 | 1.50 |
| Charcoal-Maxsorb III | 0.001649 | 8460 | 1.3 | Saha, B.B., et al., *Adsorption characteristics and heat of adsorption measurements of R-134a on activated carbon.* international journal of refrigeration, 2009. **32**(7): p. 1563-1569. |
| R-410a | Carbon | AC-Maxsorb III | 0.00596 | 4327 | 1.17 | Askalany, A.A., B.B. Saha, and I.M. Ismail, *Adsorption isotherms and kinetics of HFC410A onto activated carbons.* Applied Thermal Engineering, 2014. **72**(2): p. 237-243. |
| ACF-A-20 | 0.00325 | 5263.5 | 1.43 |
| R -407C | Carbon | AC-AquaSorb2000 | 0.001139 | 6885.8 | 1.36 | El-Sharkawy, M., et al., *Adsorption isotherms and kinetics of a mixture of Pentafluoroethane, 1, 1, 1, 2-Tetrafluoroethane and Difluoromethane (HFC-407C) onto granular activated carbon.* Applied Thermal Engineering, 2016. **93**: p. 988-994. |
| R-404A | Carbon | AC-AquaSorb2000 | 0.001035 | 9579.4 | 1.03 | Ghazy, M., et al., *Adsorption isotherms and kinetics of HFC-404A onto bituminous based granular activated carbon for storage and cooling applications.* Applied Thermal Engineering, 2016. **105**: p. 639-645. |
| Difluoroethane | Carbon | AC-Maxsorb III | 0.003438 | 5947.8 | 1.3 |
| Methane | Carbon | AC-Maxsorb III | 0.002193 | 4757.3 | 1.05 | Rahman, K.A., et al., *On thermodynamics of methane+ carbonaceous materials adsorption.* International Journal of Heat and Mass Transfer, 2012. **55**(4): p. 565-573. |
| AC-AX21 | 0.00108 | 5464.1 | 1.26 |
| AC-BPL | 0.00036 | 7040 | 1.54 |
| AC -Norit-R1-Extra | 0.00043 | 7500 | 1.73 |
| AC-F30/470 | 0.000389 | 7742.9 | 1.81 |
| AC-Chemviron | 0.000407 | 8684.1 | 1.86 |
| AC-Calgon | 0.000309 | 8955.1 | 2.41 |
| ACF-A-20 | 0.000717 | 6198.4 | 1.51 |
| R-134a | Carbon | AC-SRD-1352/3 | 0.000767 | 10916.6 | 1.7 | Saha, B.B., et al., *Accurate adsorption isotherms of R134a onto activated carbons for cooling and freezing applications.* international journal of refrigeration, 2012. **35**(3): p. 499-505. |
| ACF-A-20 | 0.00101 | 8611.7 | 1.5 |

Inputs from user: T, Y.

Output to user: P.