This worksheet demonstrates the functions defined in the CoolProp user function (CoolPropMathcadWrapper.dll). CoolProp is an open-source, cross-platform, thermophysical fluid properties library developed by Ian Bell et al. at the University of Liege, Liege, Belgium. CoolProp documentation can be found at: **http://www.coolprop.org**

Revision History

2013-10-04: Creation and initial release of the CoolProp Mathcad Wrapper library. S. Polak
2013-11-01: Incorporated CoolProp 4.0.0 pre-release code . S. Polak
2013-11-21: Corrected typos and units. S. Polak
2016-04-30: Props Functions name changes for CoolProp 5.x - J.P. Henning
2016-05-07: Added low-level functions for get_fluid_param_string and set reference_state - J. P. Henning
2017-01-16: Added support for IF97 backward and transport functions - J. P. Henning
2017-02-10: Added Mathcad error message support for common CoolProp errors - J. P. Henning

Function Explanation & Examples

List of Fluids in the CoolProp Library (and Other Library Parameters)

The following low-level and high-level CoolProp functions are implemented in this Mathcad wrapper:

get_global_param_string("*Property*")	Extracts global CoolProp information
get_fluid_param_string("*Fluid*", "*Property*")	Extracts fluid specific information
set_reference_state("Fluid", "StateString")	Sets the reference state for a fluid
PropsSI("*OutProp*", "*InProp1*", *Val1*, "*InProp2*", *Val2*, "*Fluid*")	State-dependent Properties
Props1SI("*Fluid*", "*PropName*")	State-independent Properties
HAPropsSI("*OutProp*", "*InProp1*", *Val1*, "*InProp2*", *Val2*, , "*InProp3*", *Val3*)	Humid Air Properties

These functions will be explained in detail below and demonstrated with examples.

Low Level Information Functions

The function ***get_global_param_string*** is a wrapper for the CoolProp low level function *get_global_param_string*(). It can be used to extract parameters from the CoolProp library, such as the CoolProp version, and the list of fluids in the library using their acceptable name strings. Functions to parse the output of ***get_global_param_string*** are below (hidden for clarity - double click the arrow symbol in the left margin to expand the hidden/collapsed area).

The parameter "version" will output a single string identifying the version of CoolProp currently installed in the Mathcad installation directory.

get_global_param_string("version") = "6.1.1dev" Current version of the CoolProp library

The function parameter "FluidsList" will output a single string representing all the fluids available in the CoolProp library. For clarity, the string parsing functions defined above (hidden by default) splits that single string into a vector of strings, providing the scrollable list of fluids as shown below:

strSplit(get_global_param_string("FluidsList") , ",") =

"1-Butene"
"Acetone"
"Air"
"Ammonia"
...

Click on the output vector (table) to the left to enable scrolling through the entire list

The parameter "predefined_mixtures" will output a single string representing all the predefined fluid mixtures available.

strSplit(get_global_param_string("predefined_mixtures"), ",") =

"Air.mix"
"Amarillo.mix"
"Ekofisk.mix"
"GulfCoast.mix"
...

Click on the output vector (table) to the left to enable scrolling through the entire list

The function ***get_fluid_param_string*** is a wrapper for the CoolProp low level function get_fluid_param_string(). It can be used to extract information from the CoolProp library about a specific to a fluid such as the aliases, Chemical Abstract Service (CAS) registry number for that fluid, the (ASHRAE) standard 34 safety rating, the equivalent REFPROP_name, or a "pure" flag identifying if the fluid is a pure substance (true) or a mixture (false). Examples of the usage of ***get_fluid_param_string*** are shown below:

Aliases using the parameter "aliases"

get_fluid_param_string("1-Butene", "aliases") = "1Butene, 1BUTENE, 1-BUTENE, Butene"

get_fluid_param_string("CO2", "aliases") = "R744, co2, CO2, carbondioxide, CARBONDIOXIDE"

CAS numbers using the parameter "CAS"

get_fluid_param_string("1-Butene", "CAS") = "106-98-9"

get_fluid_param_string("CO2", "CAS") = "124-38-9"

ASHRAE standard 34 safety rating using the parameter "ASHRAE34"

get_fluid_param_string("H2", "ASHRAE34") = "A3"

get_fluid_param_string("CO2", "ASHRAE34") = "A1"

REFPROP equivalent name using the parameter "REFPROPName"

get_fluid_param_string("H2", "REFPROPName") = "HYDROGEN"

get_fluid_param_string("CO2", "REFPROPName") = "CO2"

Pure fluid flag using the parameter "pure"

get_fluid_param_string("air", "pure") = "false" *Indicates that "air" is **not** a pure fluid, but a mixture.*

get_fluid_param_string("CO2", "pure") = "true" *Indicates that "CO2" is a pure fluid, **not** a mixture.*

State Dependent Fluid Properties

The function **PropsSI** is used to extract state-dependent fluid properties. The calling structure of the function is as follows:

PropsSI(Output Property <string>,
Input1 Property <string>,
Input1 Value <scalar>,
Input2 Property <string>,
Input2 Value <scalar>,
Fluid <string>)

Where "Fluid" is one of the fluid names from the FluidList table extracted above, and output and input properties are taken from the partial list of most common parameters below:

Parameter	Units	Input/ Output	Trivial	Description [units]
D, DMASS, Dmass	kg/m^3	IO	False	Mass density
H, HMASS, Hmass	J/kg	IO	False	Mass specific enthalpy
P	Pa	IO	False	Pressure
Q	mol/mol	IO	False	Mass vapor quality
S, SMASS, Smass	J/kg/K	IO	False	Mass specific entropy
T	K	IO	False	Temperature
U, UMASS, Umass	J/kg	IO	False	Mass specific internal energy
ACENTRIC, acentric		O	True	Acentric factor
A, speed_of_sound	m/s	O	False	Speed of sound
CONDUCTIVITY, L,	W/m/K	O	False	Thermal conductivity
CPMASS, Cpmass, C	J/kg/K	O	False	Mass specific constant pressure specific heat
CVMASS, Cvmass, O	J/kg/K	O	False	Mass specific constant volume specific heat
GAS_CONSTANT	J/mol/K	O	True	Molar gas constant
I, SURFACE_TENSION	N/m	O	False	Surface tension
M, MOLARMASS	kg/mol	O	True	Molar mass
PCRIT, P_CRITICAL, Pcrit, p_critical, pcrit	Pa	O	True	Pressure at the critical point
PMAX, P_MAX,	Pa	O	True	Maximum pressure limit
PMIN, P_MIN,	Pa	O	True	Minimum pressure limit
PRANDTL, Prandtl	[-]	O	False	Prandtl number
PTRIPLE, P_TRIPLE	Pa	O	True	Pressure at the triple point (pure only)
RHOCRIT, rhocrit	kg/m^3	O	True	Mass density at critical point
TCRIT, Tcrit	K	O	True	Temperature at the critical point
TMAX, T_MAX, T_max, Tmax	K	O	True	Maximum temperature limit
TMIN, T_MIN, T_min, Tmin	K	O	True	Minimum temperature limit
TTRIPLE, T_TRIPLE, T_triple, Ttriple	K	O	True	Temperature at the triple point
T_FREEZE, T_freeze	K	O	True	Freezing temperature for incompressible solutions
V, viscosity	Pa s	O	False	Viscosity
Z	[-]	O	False	Compressibility factor (= *PV/RT*)

For a full list of valid Property parameters, please see
http://www.coolprop.org/coolprop/HighLevelAPI.html#table-of-string-inputs-to-propssi-function

Examples, PropsSI

$h := \text{PropsSI}(\text{"H"}, \text{"T"}, 300, \text{"P"}, 500, \text{"Helium"}) = 1563094.4$

Enthalpy of Helium [J/kg] at a temperature of 300 K and a Pressure of 500 Pa

$\text{PropsSI}(\text{"T"}, \text{"H"}, h, \text{"P"}, 500, \text{"Helium"}) = 300$

Using *h* and 500 Pa as the input pair, the input temperature above should be returned in [K].

$\text{PropsSI}\left(\text{"V"}, \text{"T"}, \frac{21\ °\text{C}}{\text{K}}, \text{"P"}, \frac{14.7\text{psi}}{\text{Pa}}, \text{"Water"}\right) = 0.000978$

Dynamic viscosity of Water [Pa*s] at a temperature of 21 °C (converted to the CoolProp temperature units, K) and a Pressure of 14.7 psi (converted to the CoolProp pressure units, Pa)

Saturated Liquid / Vapor examples

Properties along the saturation curve can be evaluated using just one of the state variables (i.e. T, P, etc.) and the vapor quality, Q, with a value of 0 for liquid, or 1 for vapor.

$\text{PropsSI}\left(\text{"T"}, \text{"P"}, \frac{10 \cdot \text{MPa}}{\text{Pa}}, \text{"Q"}, 0, \text{"Water"}\right) = 584.147$

Saturation temperature [K] of Water at a pressure of 10 MPa and a Quality of 0 (which is arbitrary for temperature output).

$\text{PropsSI}\left(\text{"D"}, \text{"P"}, \frac{10 \cdot \text{MPa}}{\text{Pa}}, \text{"Q"}, 0, \text{"Water"}\right) = 688.424$

Saturated liquid density [kg/m³] of Water at a pressure of 10 MPa.

$\text{PropsSI}\left(\text{"D"}, \text{"P"}, \frac{10 \cdot \text{MPa}}{\text{Pa}}, \text{"Q"}, 1, \text{"Water"}\right) = 55.463$

Saturated vapor density [kg/m³] of Water at a pressure of 10 MPa.

Mixture examples

Properties can be determined for a number of pre-defined mixtures; taken from the list returned by the low-level function get_global_param_string("predefiend_mixtures") as demonstrated above.

$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"Air"}) = 1.1963$

Density [kg/m³] of Air at room temperature and pressure.

User-defined mixtures can be specified in the last argument of PropsSI(), The mixture is specified by specifying each fluid[mole_fraction] separated by & between each fluid and mole fraction in a single string. For example, Air can be user defined using 21% O2 and 79% N2 as such:

$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"O2[0.21]\&N2[0.79]"}) = 1.1916$

Density [kg/m³] of Air at room temperature and pressure using the user defined mixture string.

Tertiary mixtures can be specified as well by appending another pure fluid and mole fraction to the last parameter string. To improve the mixture density calculation, Argon (Ar), the next largest component in air, can be included at the following mole fraction:

$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"O2[0.20948]\&N2[0.78084]\&Ar[0.00934]"}) = 1.1957$

CO_2 can be added to create a quaternary mixture and further improve the accuracy of the result:

$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"O2[0.20948]\&N2[0.78084]\&Ar[0.00934]\&CO2[0.000314]"}) = 1.1963$

Partial Derivatives for Single-Phase States

For some applications, it can be useful to have access to partial derivatives of thermodynamic properties. A generalized first partial derivative has been implemented into CoolProp, which can be obtained using the **PropsSI** function by encoding the desired derivative as a string. The format of the string is "d(OF)/d(WRT)|CONSTANT" .

This example uses the following partial derivative of enthalpy as an alternate method for calculation the constant pressure specific heat.

$$C_p = \left.\frac{\partial h}{\partial T}\right|_P$$

Using the above encoding for the Output Property string in PropsSI:

$$\text{PropsSI}(\text{"d(H)/d(T)|P"}, \text{"P"}, 101325, \text{"T"}, 300, \text{"Water"}) = 4180.635776556 \quad \text{[J/kg/K]}$$

As a check, using the built-in Property Name for constant pressure specific heat, we get a value of:

$$\text{PropsSI}(\text{"C"}, \text{"P"}, 101325, \text{"T"}, 300, \text{"Water"}) = 4180.635776556 \quad \text{[J/kg/K]}$$

Second partial derivatives can be encoded into **PropsSI** as well. This is the second derivative of the above quantity, Cp, with respect to enthalpy, H, at constant pressure:

$$\text{PropsSI}(\text{"d(d(H)/d(T)|P)/d(H)|P"}, \text{"P"}, 101325, \text{"T"}, 300, \text{"Water"}) = -7.7679894687 \times 10^{-5} \quad \text{[1/K]}$$

Warning: *This second derivative formulation is currently only valid for homogeneous (single-phase) states. Two phase derivatives are not defined, and are for many combinations, invalid.*

Backends

The CoolProp formulations use the Helmholtz Equation of State (HEOS). This formulation is the assumed default when specifying a fluid or mixture name. However, there are other backends that can be used within the CoolProp interface (if available) and they are specified by prefixing the fluid or mixture name with the Backend name followed by a double colon, i.e. "HEOS::".

$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"HEOS::Water"}) = 997.7712$	Default HEOS backend for CoolProp
$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"IF97::Water"}) = 997.7706$	**IAPWS-IF97** backend for water and steam.
$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"INCOMP::DowQ"}) = 964.1091$	**INCOMPressible** backend - using *DowthermQ* pure fluid.
$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"INCOMP::MITSW[0.035]"}) = 1024.35$	**INCOMPressible** backend - using *MIT Seawater* as an aqueous solution at 3.5% salinity [gm/kg_w].
$\text{PropsSI}(\text{"D"}, \text{"T"}, 295.16, \text{"P"}, 101325, \text{"INCOMP::MITSW[0.0]"}) = 997.5942$	**INCOMPressible** backend - using *MIT Seawater* again. A mass fraction of zero for the solute indicates 100% water.

For more information on the backends for CoolProp, please see http://www.coolprop.org. *Note that support functions for Tabular Interpolation are not implemented in the Mathcad wrapper at this time.*

Fluid Properties Not Dependent on State

The function **Props1SI** is used to extract fluid-specific parameters that are not dependent on the state. The calling structure of the function is as follows:

Props1SI(Fluid <string>, PropertyName <string>)

Where "Fluid" is one of the fluid names from the FluidList table above, and "PropertyName" is one of the property name abbreviations from the table below:

Tcrit	*Critical temperature [K]*
pcrit	*Critical pressure [Pa]*
rhocrit	*Critical density [kg/m^3]*
molemass	*Molecular mass [kg/kmol]*
Ttriple	*Triple-point temperature [K]*
Tmin	*Minimum temperature [K]*
ptriple	*Triple-point pressure [Pa]*
acentric	*Acentric factor [-]*

For a full list, see the CoolProp Table of String Inputs at this address:
http://coolprop.sourceforge.net/coolprop/HighLevelAPI.html#table-of-string-inputs-to-propssi-function
All items in the table marked as Trivial (True) can be entered as a "PropertyName" into the PropsSI1 function.

Examples, Props1SI

Props1SI("Argon", "rhocrit") = 535.6	Extracts the critical density of Argon, in [kg/m^3]
Props1SI("Water", "Tcrit") = 647.096	Extracts the critical temperature of Water, in [K]
Props1SI("CO2", "ptriple") = 517964	Extracts the critical pressure of CO2, in [Pa]
Props1SI("N2", "acentric") = 0.0372	Extracts the acentric factor for N2 [*unitless*]
Props1SI("1-Butene", "molemass") = 0.056	Extracts the molecular mass of 1-Butene, in [kg/kmol]

***NOTE: Props1SI(**"FluidName","PropertyName"**)** is just a shortcut call to **PropsSI(**"PropertyName","",0,"",0,"FluidName"**)** where the two input parameters are not specified and given a value of zero. The same state-independent values can be found using **PropsSI()** as shown here:*

PropsSI("rhocrit", "", 0, "", 0, "Argon") = 535.6	Extracts the critical density of Argon, in [kg/m^3]
PropsSI("Tcrit", "", 0, "", 0, "Water") = 647.096	Extracts the critical temperature of Water, in [K]
PropsSI("ptriple", "", 0, "", 0, "CO2") = 517964	Extracts the critical pressure of CO2, in [Pa]
PropsSI("acentric", "", 0, "", 0, "N2") = 0.0372	Extracts the acentric factor for N2 [*unitless*]
PropsSI("molemass", "", 0, "", 0, "1-Butene") = 0.056	Extracts the molecular mass of 1-Butene, in [kg/kmol]

Humid Air Fluid Properties

The function **HAPropsSI** is used to find fluid properties of humid air. The physics behind the HAPropsSI function is based on the analysis in ASHRAE RP-1845, which is available online:

http://rp.ashrae.biz/page/ASHRAE-D-RP-1485-20091216.pdf.

It employs real gas properties for both air and water, as well as the most accurate interaction parameters and enhancement factors. RP-1845 is based largely on the IAPWS-95 formulation for the properties of water. The calling structure of the function is as follows:

HAPropsSI(OutputProperty <string>,
Input1 Property <string>,
Input1 Value <scalar>,
Input2 Property <string>,
Input2 Value <scalar>,
Input3 Property <string>,
Input3 Value <scalar>)

Where "OutputParameter" and all three Input Parameters are variables from the table below:

Parameter	Aliases	Input/Output	Description [units]
B	Twb, T_wb, WetBulb	Input/Output	Wet-Bulb Temperature [K]
C	cp	Output	Mixture specific heat [J/kg dry air/K]
Cha	cp_ha	Output	Mixture specific heat [J/kg humid air/K]
D	Tdp, T_dp, DewPoint	Input/Output	Dew-Point Temperature [K]
H	Hda, Enthalpy	Input/Output	Mixture enthalpy [J/kg dry air]
Hha		Input/Output	Mixture enthalpy [J/kg humid air]
K	k, Conductivity	Output	Mixture thermal conductivity [W/m/K]
M	Visc, mu	Output	Mixture viscosity [Pa-s]
P		Input	Pressure [Pa]
R	RH, Relhum	Input/Output	Relative humidity in (0,1) [-]
S	Sda, Entropy	Input/Output	Mixture entropy [J/kg dry air/K]
T	Tdb, T_db, DryBulb	Input/Output	Dry-Bulb Temperature [K]
V	Vda	Input/Output	Mixture volume [m^3/kg dry air]
Vha		Input/Output	Mixture volume [m^3/kg humid air]
W	Omega, HumRat	Input/Output	Humidity Ratio [kg water/kg dry air]
Z		Output	Compressibility factor (pv/RT)

Examples, HAPropsSI

$\text{HAPropsSI}(\text{"H"}, \text{"T"}, 298.15, \text{"P"}, 101325, \text{"R"}, 0.5) = 5.042 \times 10^4$

Enthalpy of humid air [J kg^-1] at a dry-bulb temperature of 298.15 K, a Pressure of 101325 Pa, and relative humidity of 0.5

$\text{HAPropsSI}(\text{"T"}, \text{"P"}, 101325, \text{"H"}, h, \text{"R"}, 0.5) = 371.24$

Temperature of saturated air (R=1.0) at the previous enthalphy value

Note that the humid air functions at a relative humidity of zero should give roughly the same values as Dry Air from the **PropsSI** function. This check can be easily demonstrated here.

$\text{HAPropsSI}(\text{"C"}, \text{"T"}, 298.15, \text{"P"}, 101325, \text{"R"}, 0) = 1006.289371$ [J/kg/K]

$\text{PropsSI}(\text{"C"}, \text{"T"}, 298.15, \text{"P"}, 101325, \text{"Air"}) = 1006.308143$ [J/kg/K]

Reference States

Enthalpy and entropy are relative properties! You should always be comparing differences in enthalpy rather than absolute values of the enthalpy or entropy. That said, if can be useful to set the reference state values for enthalpy and entropy to one of a few standard values. This is done by the use of the low-level **set_reference_state** function, which is a wrapper for the CoolProp set_reference_stateS() function.

A number of pre-defined reference states can be used:

- IIR: h = 200 kJ/kg, s=1 kJ/kg/K at 0C saturated liquid { *IIR = International Institute of Refrigeration* }
- ASHRAE: h = 0, s = 0 @ -40C saturated liquid
- NBP: h=0, s=0 for saturated liquid at 1 atmosphere { *NBP = Normal Boiling Point* }
- DEF: Go back to the default reference state for the fluid

as demonstrated below.

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Propane"}) = 1.051 \times 10^{5}$ *Using the default reference state*

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Butane"}) = 1.108 \times 10^{5}$

$\text{set\_reference\_state}(\text{"n-Propane"}, \text{"ASHRAE"}) = 0$ *Using the ASHRAE reference state.*

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Propane"}) = 1.459 \times 10^{-11}$ *Value should be zero, or very close to it.*

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Butane"}) = 1.108 \times 10^{5}$ *Note that value changed for n-Propane, as requested, but not for n-Butane..*

$\text{set\_reference\_state}(\text{"n-Propane"}, \text{"DEF"}) = 0$ *Back to default (DEF) reference state.*

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Propane"}) = 1.051 \times 10^{5}$

$\text{PropsSI}(\text{"H"}, \text{"T"}, 233.15, \text{"Q"}, 0, \text{"n-Butane"}) = 1.108 \times 10^{5}$

Warning: *The changing of the reference state should **only** be done at the very beginning of your worksheet or unexpected results may occur. It is not recommended to change the reference state during the course of making calculations as done here for demonstration purposes only. Further more, because of Mathcad's top-down calculation order, switching back and forth between reference states can lead to very unexpected results (real or apparent) and is not recommended.*

Creating Specific Unit Functions & Plotting

Specialized Mathcad functions can be created that simplify input and correctly handle units. For example, the above saturation temperature call can be embedded in a user defined function:

$$T_{sat}(P, fluid) := PropsSI\left("T", "P", \frac{P}{Pa}, "Q", 0, fluid\right) \cdot K$$

This new function can then be used to easily calculate saturation temperature with units applied:

$fluid := "Water"$ $\quad P := 10 \cdot MPa$

$T_{sat}(P, fluid) = 310.997 \cdot °C$

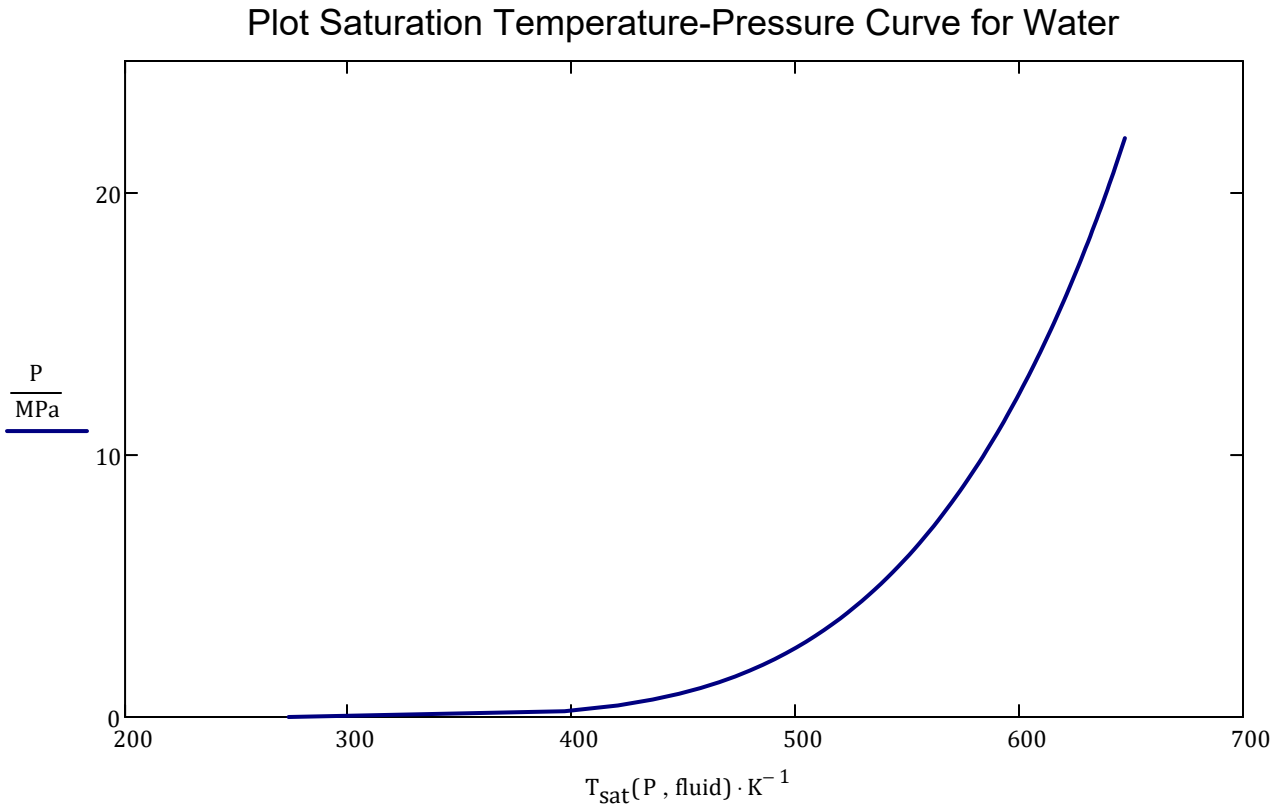
This is an extremely useful and compact way to call these functions over a range and create a plot of the results.

$P_t := Props1SI("Water", "ptriple") \cdot Pa = 611.655\,Pa$ Start range at the Triples Point Pressure.

$P_c := Props1SI("Water", "pcrit") \cdot Pa = 22.064 \cdot MPa$ End range at the Critical Point Pressure.

$\Delta P := \frac{P_c - P_t}{100} = 0.221 \cdot MPa$ Set the range interval over 100 points.

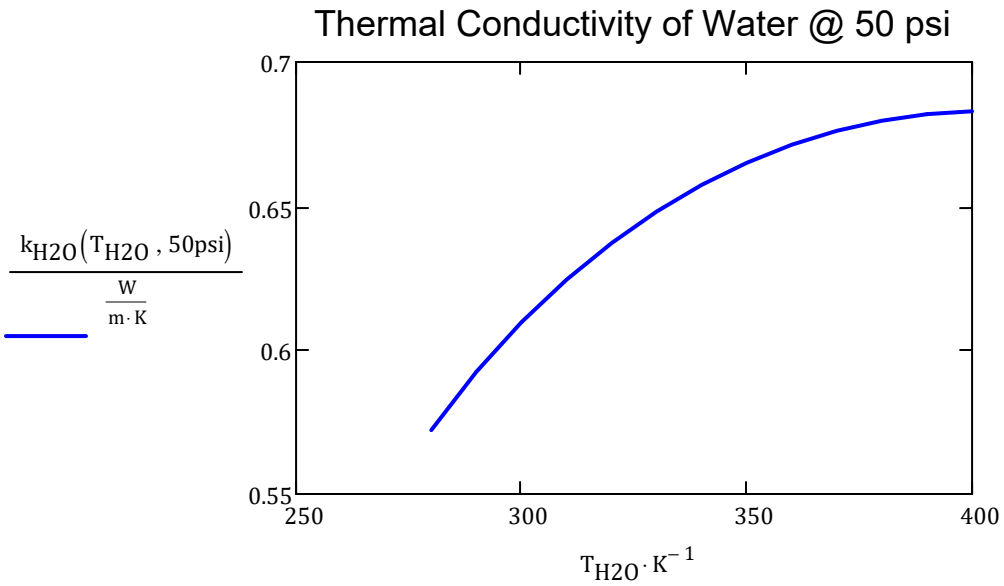
$P := P_t, P_t + \Delta P .. P_c$ Set the Range



Unit functions of other properties can be created as well.

$T_{H2O} := 280 \cdot K, 290 \cdot K .. 400 \cdot K$

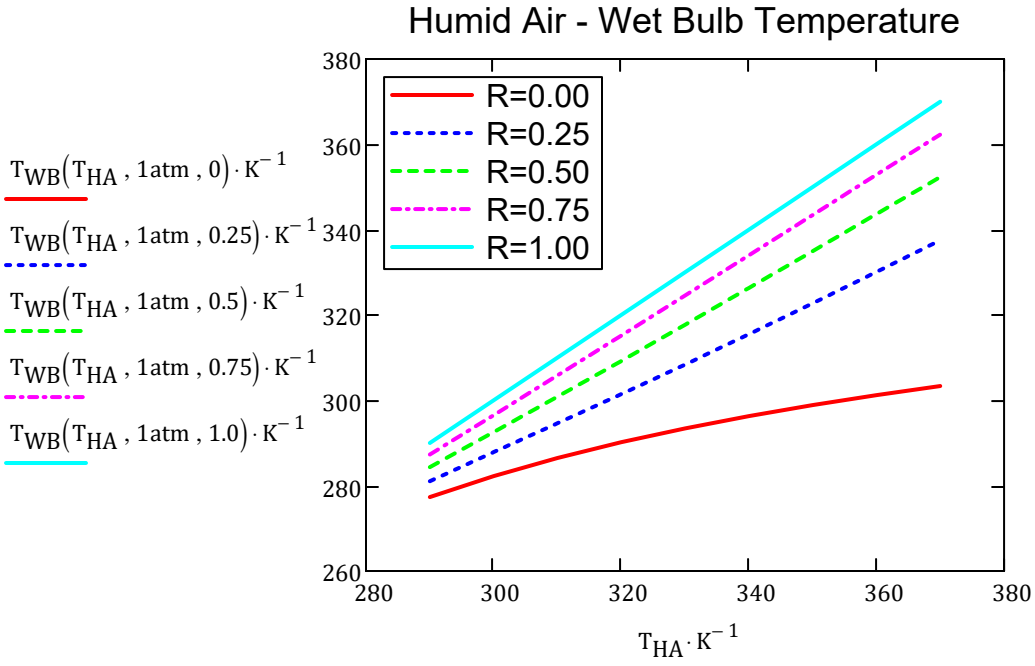
$k_{H2O}(T, P) := PropsSI\left("L", "T", \frac{T}{K}, "P", \frac{P}{Pa}, "IF97::Water"\right) \cdot \frac{W}{m \cdot K}$



This is an example of plotting the wet bulb temperature of humid air at various dry-bulb temperatures and relative humidity ratios at a pressure of 1 atm.

$T_{WB}(T, P, R) := HAPropsSI\left("B", "T", \frac{T}{K}, "P", \frac{P}{Pa}, "R", R\right) \cdot K$

$T_{HA} := 290 \cdot K, 300 \cdot K .. 370 \cdot K$



Error Messages

Efforts have been made to provide user feedback through the Mathcad error system, which highlights the CoolProp function parameter causing the error and displays a pop-up error message when clicking on the offending equation. For example:

Props1SI("Water", "Tcrit") = 647.096	*Normal Function results.*
Props1SI("Tcrit", "Water") = 647.096	*Normal Function results. Note that Props1SI will actually take the input parameters in any order.*
Props1SI("HOH", "Tcrit") = ■	*Input a bad, or undefined fluid string: the offending parameter is highlighted in red.*
Props1SI("HOH", "Tcrit") = ■ Invalid Fluid String	*Clicking on the equation will give the pop-up error message, as shown in the graphic to the left, since HOH is not a valid alias for Water.*

Getting More Detail

To view further information on the error, follow the offending equation with a call to

get_global_param_string("errstring") = "Neither input to Props1SI [Tcrit,HOH] is a valid fluid"

which will output the actual error string from CoolProp. This error string is associated with the immediately preceding equation; if a valid equation follows, the error string will be cleared. These error strings are cleared when read and ***MUST*** follow the offending equation before any other equation is typed. If the error message strings get out of sync with the preceding equations, just press <Ctrl>-F9 to recalculate the entire worksheet in the correct order.

Uncaught CoolProp Errors

While attempts have been made to capture the most common CoolProp errors, there are undoubtedly some CoolProp errors that will not be identified by the Mathcad wrapper. In these cases, the first parameter will be highlighted with the pop-up message:

PropsSI("T", "P", $2000 \cdot 10^6$, "S", $5 \cdot 10^3$, "Water") = ■ ERROR: Use get_global_param_string("errstring") for more info	*Clicking on the equation will give a generic pop-up error message with instructions for inspecting the CoolProp errString.*

In these cases, the CoolProp error string will have to be examined to get details on the exact cause of the error and the parameter that is causing it. In this case, the value of Pressure [P] or Entropy [S] is outside the valid IF97 calculation limits.

get_global_param_string("errstring") = "unable to solve 1phase PY flash with Tmin=348.401, Tmax=3000 due to error:
Inputs in Brent [53548.499562,71494.911982] do not bracket the root.
Function values are [-1.010873,-0.211041] :
PropsSI("T","P",2000000000,"S",5000,"Water")"