## Recommended MATLAB Interface

requestedPropertyValue = getFluidProperty(libraryLocation, requestedProperty,…   
 inputProperty1, inputProperty1Value,…  
 inputProperty2, inputProperty2Value, fluid,…  
 fluidComposition, massOrMolar, desiredUnits)

Where:

[OUTPUT]:  
requestedPropertyValue = (double) value of the requested thermodynamic property as calculated by the library

[INPUTS]:  
libraryLocation = (string) the location of the REFPROP or CoolProp library files (dll, exe, etc.)  
requestedProperty = (string) the thermodynamic property name for which the value will be returned  
inputProperty1 = (string) name of the 1st property used as the state point  
inputProperty1Value = (double) value of the 1st property used as the state point in the library’s expected units  
inputProperty2 = (string) name of the 2nd property used as the state point  
inputProperty2Value = (double) value of the 2nd property used as the state point in the library’s expected units  
fluid = (string) indicating the fluid for which the requested property should be calculated  
fluidComposition = (double array of size 1xP) species fraction where 1 <= P <= 20, whose values must sum to 1, and P must match the number of species in the fluid  
massOrMolar = [REFPROP only] (int) value to determine input composition units: 0 🡪 Molar, 1 🡪 Mass  
desiredUnits = [REFPROP only] (char) enum as expected by refprop.dll to determine the units to use (e.g., MKS, MASS BASE SI, etc.)

Requirements for this MATLAB Function:

1. This function will guarantee support for REFPROP 10 and CoolProp 6.6.0
   1. The user will be expected to obtain REFPROP 10 and CoolProp 6.6.0
   2. The user will be expected to have the proper compilers for using mex with REFPROP or CoolProp
   3. The user will be expected to mex the REFPROP and CoolProp C++ files for use on their machines (perhaps we can provide a script that will set this up for the user the first time?)
   4. The function documentation will offer guidance to the user for these things
2. This function will only support the High-level function call PropsSI for the CoolProp library. This may result in a slight performance degradation if the user is trying to create a LUT.
3. For REFPROP, inputProperty1 and inputProperty2 will have to be combined into a single input pair.
   1. After combining, the function can directly call MLrefprop – no changes needed
4. For CoolProp, the massOrMole determination is explicit in the inputProperty\* and requestedProperty strings.
5. For user-defined fluids, Coolprop combines the individual species and their mole fractions in the fluid string
6. The user should be able to input strings or chars for the string variables and doubles for the int variable. The MATLAB function should appropriately cast the variables before calling the library functions.
7. The function should check the validity of the inputs
   1. Does the library location exist
   2. Does the library location have the expected files
   3. Do the thermodynamic properties conform to the selected library’s expectations
   4. Are the expected inputs for REFPROP included
   5. For the fluid value
      1. .MIX files for REFPROP only
      2. User-defined fluids need to have the correct syntax for REFPROP or CoolProp
8. The function should output useful error messages, especially error messages that come from the library itself
9. TO CONSIDER: Should the user be able to specify an array of RequestedProperties, input property values, and/or fluids?
   1. If input property values can be sent in as arrays
      1. REPFROP handles input property values as arrays and returns an array of output values of size numel(inputProperty1Value) by numel(inputProperty2Value)
      2. CoolProp PropsSI function does not handle array inputs – the function would have to call PropsSI iteratively and collect the output values so that they imitate REFPROP’s output
   2. If output values can be sent in as arrays, for both REFPROP and CoolProp the function will have to
      1. Loop over the output values
      2. Collect the outputs in the same organized way – suggestion: a struct whose field names are the output names and whose fields contain the output values as expected from a function call with only one output value
   3. If the fluid variable can be an array, for both REFPROP and CoolProp the function will have to
      1. Loop over the fluids
      2. Collect the outputs in the same organized way – suggestion: a struct whose field names are the fluid names and whose fields contain the output values as expected from a function call with only one output value. If there are also multiple output values, the struct should be organized as fluid.property.outputValues
10. For consistency
    1. Rename hiLevelMexC.cpp to REFPROP\_HiLevelMexC.cpp
    2. Rename CoolPropHiLevelMexC.cpp to CoolProp\_HiLevelMexC.cpp
    3. Rename MLrefprop.m to getValueWithREFPROP.m
    4. Rename MLCoolProp.m to getValueWithCoolProp.m
    5. Put these files in the utilities directory
    6. Have a script that will create the mex files from the cpp files in the utilities directory
    7. Set up the MATLAB project to put everything on the path as necessary
    8. Expose only the getFluidProperty.m file in the project base directory
    9. Expose a user guide in the project base directory
11. Include test files much like exist for MLrefprop

## Appendix 1: Installing CoolProp

For Windows users:  
Grab the installer from [CoolProp - Browse /CoolProp (at SourceForge.net)](https://sourceforge.net/projects/coolprop/files/CoolProp/). Navigate to the latest version 🡪 Installers 🡪 Windows (e.g., [CoolProp/6.6.0/Installers/Windows](https://sourceforge.net/projects/coolprop/files/CoolProp/6.6.0/Installers/Windows/)). This will install the CoolProp.dll and any required shared libraries in C:\Users\<username>\AppData\Roaming\CoolProp.

For Linux and Mac users, or Windows users who wish to compile CoolProp themselves:  
Follow the instructions give here: [Available Wrappers — CoolProp 6.6.0 documentation](http://www.coolprop.org/coolprop/wrappers/index.html#common-wrapper-prerequisites)

## Appendix 2: MLRefprop

output = MLrefprop(propReq, spec, Value1, Value2, fluid, MassOrMole, DesiredUnits, Path2Refprop, DebugOutput)

Where (see: https://refprop-docs.readthedocs.io/en/latest/DLL/high\_level.html)  
output = DOUBLE (array of size MxN or scalar) output from RefProp for the desired Property from propReq. M is the size of Value1 and N is the size of Value2  
propReq = CHAR value accepted by REFPROP as 'hOut' values   
spec = CHAR value accepted by REFPROP as 'hIn' values  
Value1 = DOUBLE (array of size 1xM or scalar) of values related to the 1st character in spec  
Value2 = DOUBLE (array of size 1XN or scalar) of values related to the 2nd character in spec  
fluid = CHAR value accepted by REFPROP as 'hFld' values for multi-species, list species 1 to P (where P is specified by the Composition variable) separated by semicolons (;)  
MassOrMolar = INT value to determine input composition units: 0 -> Molar, 1 -> Mass  
Composition = DOUBLE (array of size 1xP or scalar) of species fractions where (1 < P <= 20) and values must sum to 1  
DesiredUnits = CHAR value to determine units to use (enum as expected by refprop.dll)  
Path2Refprop = CHAR path to Refprop directory (e.g. C:\\ProgramFiles (x86)\\REFPROP)  
DebugOutput = DOUBLE value (0 to suppress, 1 to show) debug output in MATLAB console

## Appendix 3: Hi-Level CoolProp call

PropsSI(output, Input1, Input1Value, Input2, Input2Value, fluid)  
e.g., PropsSI('T', 'P', 101325, 'Q', 0, 'Water')

Where (see: [High-Level Interface — CoolProp 6.6.0 documentation](http://www.coolprop.org/coolprop/HighLevelAPI.html#propssi-function) and [High-Level Interface input table](http://www.coolprop.org/coolprop/HighLevelAPI.html#parameter-table))   
output = CHAR value indicating the property values to be returned from CoolProp given the inputs (e.g. ‘T’ will return Temperature in Kelvin)  
Input1 = CHAR value indicating the first property used to determine the state point (e.g., ‘P’ for pressure given in Pa)  
Input1Value = DOUBLE value indicating the state point of the first input (e.g., 101325 for pressure given in Pa)  
Input2 = CHAR value indicating the second property used to determine the state point (e.g., Q for the quality)  
Input2Value = DOUBLE value indicating the state point for the second input (e.g., 0 for quality)  
fluid = CHAR value indicating the fluid for which the output value is to be calculated given the two input values (e.g., ‘Water’ – see: [List of CoolProp Fluids](http://www.coolprop.org/fluid_properties/PurePseudoPure.html#list-of-fluids) and [User Defined Fluids](http://www.coolprop.org/coolprop/HighLevelAPI.html#user-defined-mixtures))

## Appendix 4: Low-level CoolProp Call

The Low-level CoolProp call looks somewhat more like the REFPROP function call we created. Instead of the two input chars used by the high-level call, the user can specify a single compound input given here: [CoolProp: CoolProp Namespace Reference Input Pairs](http://www.coolprop.org/_static/doxygen/html/namespace_cool_prop.html#a58e7d98861406dedb48e07f551a61efb) and two individual input values (Input1Value and Input2Value). The Output parameter may be any one to five of the following: [CoolProp: CoolProp Namespace Reference Parameters](http://www.coolprop.org/_static/doxygen/html/namespace_cool_prop.html#a4b49eeb37210a720b188f493955d8364).