Working with Liquid/Vapor Fluid Models

David G. Goodwin

October 17, 2004

California Institute of Technology
Email: dgoodwin@caltech.edu

Contents

1 Introduction

Cantera contains built-in accurate equations of state for several pure fluids, including water, nitrogen, oxygen, methane, and hydrogen. These equations of state may be used to calculate accurate thermodynamic properties for these fluids over the full range in density and temperature for which the equations of state are valid, including pure liquid, pure vapor, saturated liquid/vapor, and supercritical regions in the phase diagram. And because Cantera integrates with popular general-purpose problem-solving environments like MATLAB and Python, it is possible not only to compute fluid properties but to construct complete simulations of any process involving these fluids, such as vapor power cycles.

In this document, we will look at how to use these capabilities provided by Cantera in MATLAB. The differences in syntax between MATLAB and Python will be described in Appendix A. It will be assumed here that you have MATLAB on your system, and have installed the Cantera MATLAB Toolbox. For instructions on how to do this, see Appendix B.

2 Fluid Objects

The first step is to create an *object* in MATLAB representing the fluid of interest. Here's how to create an object representing water:

```
>> h2o = Water

       temperature                   300  K
       pressure                   101325  Pa
       density                   996.633  kg/m^3
       mean mol. weight           18.016  amu

                            X                Y
                   -------------     ------------
               H2O   1.000000e+00     1.000000e+00

>>
```

After executing this statement, MATLAB variable `h2o` is an object representing water. A summary of the initial state of the object is printed when it is created; to suppress this, follow the statement with a semicolon:

```
>> h2o = Water;
```

The right-hand-side of this statement invokes a function `Water` in the Cantera MATLAB Toolbox that returns a new object representing water. It takes no arguments, which is why the parentheses may be omitted (i.e. `Water` is equivalent to `Water()` in MATLAB). This function may be called multiple times if it is desired to construct multiple objects, each representing water:

```
>> state1 = Water;
>> state2 = Water;
>> state3 = Water;
```

These three objects are all independent, and may each be set to a different state.

Objects representing the other fluids may be created by calling the appropriate function:

```
>> n2 = Nitrogen;
>> o2 = Oxygen;
>> h2 = Hydrogen;
```

```
>> ch4 = Methane;
```

As with the `Water` function, the functions take no arguments, and they may be called as many times as desired, to create multiple independent fluid objects of the same type.

3 Setting the State

After an object representing the fluid has been created, the next step is to set its state. These objects are designed to behave like the fluids they represent, so the state is set in the same way the thermodynamic state of a pure, single-component fluid is set — by specifying the values of two independent thermodynamic properties.

Various combinations of properties may be used to set the state. For conditions for which only a single phase is present, the temperature and pressure may be independently specified. However, for saturated states in which liquid and vapor are both present, the pressure is a function $P_{sat}(T)$ of temperature, and is therefore not independent. Temperature and specific volume (or density) are always independent, and can be therefore be used to specify the state uniquely whether one phase is present, or two are. For saturation states only, the vapor fraction may also be used as an independent property to use in specifying the state.

The `set` method is used to set the state of a fluid object. Here's an example of how it is used:

```
>> set(h2o,'T',400,'Rho',0.01);
```

The object whose properties are to be set is always the first argument, followed by two string/value pairs that specify which property is to be set, and the value to which it should be set.

The properties may either be specified by full name, or by a symbol or abbreviation. The `set` statement above could also be written in any of these forms:

```
>> set(h2o,'Temperature',400,'Rho',0.01);
>> set(h2o,'Temperature',400,'Density',0.01);
>> set(h2o,'T',400,'Density',0.01);
```

The names and abbreviations that are recognized are listed below.

Property	**Full Name**	**Abbreviation**	**Units**
Temperature	`Temperature`	`T`	K
Mass Density	`Density`	`Rho`	kg/m^3
Specific Volume	`Volume`	`V`	m^3/kg
Pressure	`Pressure`	`P`	Pa
Specific Enthalpy	`Enthalpy`	`H`	J/kg
Specific Entropy	`Entropy`	`S`	J/kg/K
Vapor Fraction	`Vapor`		-
Liquid Fraction	`Liquid`		-

Note that it is possible to set the state using properties other than just the temperature, density, and pressure. For example, the pressure and the specific entropy may be used to set the state. This makes it very easy to find the state resulting from an isentropic compression or expansion. For example, suppose we need to know the temperature that results if nitrogen initially at 300 K and 1 atm pressure is isentropically compressed to two atmospheres; all we need to do to solve this problem is show below.

```
>> set(n2,'T',300,'P',OneAtm);
>> s0 = entropy_mass(n2);
>> set(n2,'P',2.0*OneAtm,'S',s0);
```

```
>> temperature(n2)

ans =

  365.6988
```

Note that for the extensive properties (V, H, U, S), the values must be given *per unit mass* – that is, it is the *specific* value, not the molar or total value, that must be specified.

Also, not all property pairs are implemented. 'H' can only be set in combination with 'P', 'U' can only be set in combination with 'V' or 'Rho', and 'S' must be set in conjunction with 'P', 'V', or 'Rho'.

Saturation states may be set by specifying T or P and the liquid or vapor fraction. Specifying a vapor fraction x is precisely equivalent to specifying a liquid fraction $1 - x$. Attempting to specify a saturated state with $T > T_c$ or $P > P_c$ will result in an error.

Examples:

```
  set(h2o,'Temperature',600.0);
  set(h2o,'T',600.0);
  set(h2o,'H',0.5*enthalpy_mass(h2o),'P',pressure(h2o));
  set(h2o,'S',entropy_mass(h2o),'P',0.5*pressure(h2o));
  set(h2o,'T',500.0,'Vapor',0.8);
  set(h2o,'T',500.0,'Liquid',0.2);
  set(h2o,'P',0.5*critPressure(h2o),'Liquid',0.2);
```

4 Units

Cantera always uses SI units, with quantity expressed in kmol. The units of some common quantities are listed in the table below.

Property	**Symbol**	**Units**
Temperature	T	K
Pressure	P	Pa
Density	ρ	kg/m^3
Specific Volume	v	m^3/kg
Specific Enthalpy	h	J/kg
Specific Entropy	s	J/kg-K
Specific Heat	c_p or c_v	J/kg-K
Molar Volume	$\hat{v}$	m^3/kmol
Molar Enthalpy	$\hat{h}$	J/kmol
Molar Entropy	$\hat{s}$	J/kmol-K
Molar Heat Capacity	$\hat{c}_p$ or $\hat{c}_v$	J/kmol-K

5 Critical State Properties

The critical temperature, pressure, and density may be evaluated using the methods `critTemperature`, `critPressure`, and `critDensity`, respectively.

```
>> tcrit = critTemperature(h2o)

tcrit =
```

```
  647.2860

>> pcrit = critPressure(h2o)

pcrit =

    22089000

>> rhocrit = critDensity(h2o)

rhocrit =

   317
```

6 Saturation Properties

6.1 The Saturation Pressure

The saturation pressure $P_{\rm sat}(T)$ is the pressure at which liquid and vapor may co-exist in equilibrium at temperature T. Method `satPressure` returns the saturation pressure for a specified temperature.

```
>> satPressure(h2o,300)

ans =

   3.5282e+03
```

The temperature must be less than T_c.

6.2 The Saturation Temperature

It is also possible to compute the saturation temperature given a pressure $P < P_c$:

```
>> satTemperature(h2o,oneatm)

ans =

  373.1772
```

6.3 Vapor Fraction

Saturated states have both liquid and vapor present. The fraction of vapor is returned by method `vaporFraction`.

```
>> set(h2o,'T',critTemperature(h2o),'P',critPressure(h2o))

       temperature             647.286  K
       pressure            2.2089e+07  Pa
       density                 307.021  kg/m^3
```

```
       mean mol. weight          18.016  amu

                            X                Y
                   -------------     ------------
               H2O   1.000000e+00     1.000000e+00

>> set(h2o,'T',600,'Rho',density(h2o))

       temperature                  600  K
       pressure             1.23294e+07  Pa
       density                  307.021  kg/m^3
       mean mol. weight          18.016  amu

                            X                Y
                   -------------     ------------
               H2O   1.000000e+00     1.000000e+00

>> vaporFraction(h2o)

ans =

    0.1406
```

7 Property Relationships

The properties are computed from self-consistent thermodynamic equations of state, not from approximate correlations. The equations of state are taken largely from Reynolds [1979], which is in turn a compilation of data from other sources (e.g. Jacobsen et al. [1972]).

All relationships among properties and their derivatives (the Clapeyron Equation, the Maxwell Relations, etc.) should be satisfied to good accuracy by the numerical values Cantera returns. Saturation states have equal values of the Gibbs function in the liquid and vapor states to high accuracy.

The script shown below tests the extent to which the results satisfy the Clapeyron Equation

$$\frac{dP_{\rm sat}}{dT} = \frac{s_g - s_f}{v_g - v_f} \tag{1}$$

```
function rel_error = test_clapeyron(sub, T)

dt = 0.01;
psat1 = satPressure(sub, T);
psat2 = satPressure(sub, T + dt);
dPsatdT = (psat2 - psat1)/dt;

set(sub, 'T', T, 'Liquid', 1.0);
sf = entropy_mass(sub);
vf = 1.0/density(sub);

set(sub, 'T', T, 'Vapor', 1.0);
sg = entropy_mass(sub);
vg = 1.0/density(sub);
```

```
delta_s_over_delta_v = (sg - sf)/(vg - vf);
rel_error = ((delta_s_over_delta_v/dPsatdT) - 1.0);
```

```
>> ch4 = Methane;
>> rel_error = test_clapeyron(ch4, 180)

rel_error =

   -1.2553e-04

>> o2 = Oxygen;
>> rel_error = test_clapeyron(o2, 120)

rel_error =

   -2.0924e-04
```

8 Reference

The methods that can be used with objects representing pure fluids are listed below. These methods also may be used for objects representing multicomponent chemical solutions (not described in this document). Some of them return arrays of properties for the species in the solution; for pure fluids, there is only one species. The units for these properties are (kmol, m, s, K).

```
% atomic masses of the elements
atomicMasses(fluid)

% chemical potential of the (one) species
chemPotentials(fluid)

% specific heat at constant pressure
cp_mass(fluid)

% molar heat capacity at constant pressure
cp_mole(fluid)

% critical density
critDensity(fluid)

% critical pressure
critPressure(fluid)

% critical temperature
critTemperature(fluid)

% specific heat at constant volume
cv_mass(fluid)

% molar heat capacity at constant volume
cv_mole(fluid)
```

```
% mass  density
density(fluid)

% index of element with name 'name'
elementIndex(fluid,name)

% name of element with integer index 'index'
elementName(fluid,index)

% specific enthalpy
enthalpy_mass(fluid)

% molar enthalpy
enthalpy_mole(fluid)

% specific entropy
entropy_mass(fluid)

% molar entropy
entropy_mole(fluid)

% specific Gibbs function
gibbs_mass(fluid)

% molar Gibbs function
gibbs_mole(fluid)

% specific internal energy
intEnergy_mass(fluid)

% molar internal energy
intEnergy_mole(fluid)

% maximum temperature for which equation of state is valid
maxTemp(fluid)

% molar mass
meanMolarMass(fluid)

% molecular weight (molar mass)
meanMolecularWeight(fluid)

% minimum temperature for which equation of state is valid
minTemp(fluid)

% molar density
molarDensity(fluid)

% number of atoms of an element in the (one) species
nAtoms(s,species,element)

% number of elements
nElements(fluid)
```

```
% pressure
pressure(fluid)

% standard-state pressure
refPressure(fluid)

% saturation pressure at temperature T
satPressure(fluid, T)

% saturation temperature at pressure P
satTemperature(fluid, P)

% set the thermodynamic state
set(fluid,prop1, value1, prop2, value2)

% speed of sound
soundspeed(fluid)

% temperature
temperature(fluid)

% vapor fraction
vaporFraction(fluid)
```

9 Examples

9.1 PVT Surfaces

A MATLAB script to generate the surface $P(v,T)$ for a supplied fluid object is shown below, and a surface plot generated from the data produced by this function for water is shown in Fig. 1.

```
function [logv t logp] = pvt(sub)
%
% PVT - pressure/volume/temperature data for substance 'sub'
%       This function returns two one-dimensional arrays
%       containing log10(v) and T, respectively, and a 2D
%       array containing log10[p(v,T)].
%
tmin = minTemp(sub) + 0.01;
tmax = maxTemp(sub) - 0.01;

set(sub, 'T',tmin,'Liquid',1.0);
vmin = 0.5/density(sub);
set(sub, 'T',tmin,'Vapor',1.0);
vmax = 10.0/density(sub);

nt = 100;
dt = (tmax - tmin)/nt;
nv = 100;
dlogv = log10(vmax/vmin)/nv;
logvmin = log10(vmin);
v = zeros(nv,1);
```

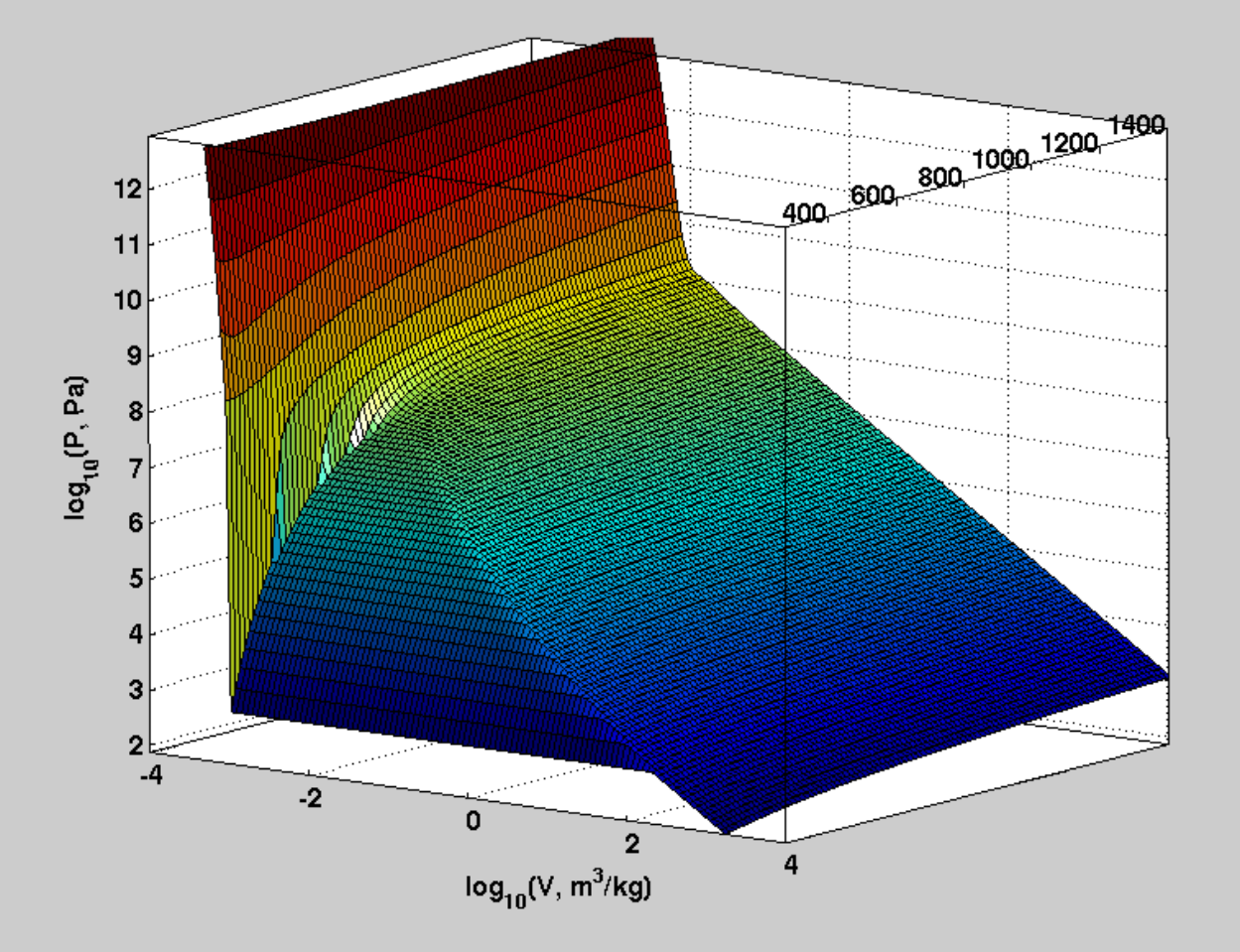

Figure 1: P-V-T surface for water.

```
t = zeros(nt,1);
p = zeros(nt,nv);
for n = 1:nv
  logv(n) = logvmin + (n-1)*dlogv;
  v = 10.0^logv(n);
  for m = 1:nt
    t(m) = tmin + (m-1)*dt;
    set(sub, 'T', t(m), 'V', v);
    logp(m,n) = log10(pressure(sub));
  end
end
```

9.2 A Rankine Cycle

A MATLAB function to simulate a simple Rankine cycle is shown below. The function arguments are:

1. T_1. The temperature at the inlet to the pump.
2. P_2. The pressure at the outlet of the pump, which is taken to be constant to the turbine inlet.

3. The isentropic efficiency of the pump.
4. The isentropic efficiency of the turbine.

```
function [work, efficiency] = rankine(t1, p2, eta_pump, ...
                                      eta_turbine)

% create an object representing water
w = Water;

% start with saturated liquid water at t1
set(w,'T',t1,'Liquid',1.0);

% pump it to p2
pump_work = pump(w, p2, eta_pump);
h2 = enthalpy_mass(w);

% heat to saturated vapor
set(w,'P',p2,'Vapor',1.0);
h3 = enthalpy_mass(w);

heat_added = h3 - h2;

% expand adiabatically back to the initial pressure
turbine_work = expand(w, p1, eta_turbine);
work = turbine_work - pump_work;

% compute the efficiency
efficiency = (work - pump_work)/heat_added;



function w = pump(fluid, pfinal, eta)
% PUMP - Adiabatically pump a fluid to pressure pfinal, using a pump
% with isentropic efficiency eta.
%
h0 = enthalpy_mass(fluid);
s0 = entropy_mass(fluid);
set(fluid, 'S', s0, 'P', pfinal);
h1s = enthalpy_mass(fluid);
isentropic_work = h1s - h0;
actual_work = isentropic_work / eta;
h1 = h0 + actual_work;
set(fluid, 'H',h1, 'P',pfinal);
w = actual_work;


function w = expand(fluid, pfinal, eta)
% EXPAND - Adiabatically expand a fluid to pressure pfinal, using a
% turbine with isentropic efficiency eta.
%
h0 = enthalpy_mass(fluid);
s0 = entropy_mass(fluid);
set(fluid, 'S', s0, 'P', pfinal);
```

```
h1s = enthalpy_mass(fluid);
isentropic_work = h0 - h1s;
actual_work = isentropic_work * eta;
h1 = h0 - actual_work;
set(fluid, 'H',h1, 'P',pfinal);
w = actual_work;
```

A Using Python

Everything described here for MATLAB can be done in Python too. The syntax is a little different, since Python uses the "dot" syntax to invoke methods on an object, rather than the function-like syntax used by MATLAB. That is, the `temperature` method would be called by writing

```
T = fluid.temperature()
```

instead of

```
T = temperature(fluid)
```

Note that in Python, parentheses are required in function or method calls, even if there are no arguments.

Also the syntax of the `set` method is different:

```
fluid.set(T = 400.0, Rho = 0.01)
```

instead of

```
set(fluid, 'T', 400.0, 'Rho', 0.01)
```

Finally, the following two lines should be put at the top of the script:

```
from Cantera import *
from Cantera.liquidvapor import *
```

To get help on the methods that can be used on an object, do this:

```
>>> w = Water()
>>> help(w.__class__)
```

Alternatively, you can use the `pydoc` program to view documentation, which can be started from the command line ot from the Start menu (Windows).

The Rankine cycle example is shown below in Python.

```
from Cantera import *
from Cantera.liquidvapor import Water

def pump(fluid, pfinal, eta):
```

```
    """Adiabatically pump a fluid to pressure pfinal, using
    a pump with isentropic efficiency eta."""
    h0 = fluid.enthalpy_mass()
    s0 = fluid.entropy_mass()
    fluid.set(S = s0, P = pfinal)
    h1s = fluid.enthalpy_mass()
    isentropic_work = h1s - h0
    actual_work = isentropic_work / eta
    h1 = h0 + actual_work
    fluid.set(H = h1, P = pfinal)
    return actual_work


def expand(fluid, pfinal, eta):
    """Adiabatically expand a fluid to pressure pfinal, using
    a turbine with isentropic efficiency eta."""
    h0 = fluid.enthalpy_mass()
    s0 = fluid.entropy_mass()
    fluid.set(S = s0, P = pfinal)
    h1s = fluid.enthalpy_mass()
    isentropic_work = h0 - h1s
    actual_work = isentropic_work * eta
    h1 = h0 - actual_work
    fluid.set(H = h1, P = pfinal)
    return actual_work


def rankine(t1, p2, eta_pump, eta_turbine):
   # create an object representing water
   w = Water()

   # start with saturated liquid water at t1
   w.set(T = t1, Vapor = 0.0)
   h1 = w.enthalpy_mass()

   # pump it adiabatically to p2
   pump_work = pump(w, p2, eta_pump)
   h2 = w.enthalpy_mass()

   # heat to saturated vapor
   w.set(P = p2, Vapor = 1.0)
   h3 = w.enthalpy_mass()
   heat_added = h3 - h2

   # expand back to p1
   turbine_work = expand(w, p1, eta_turbine)

   work = turbine_work - pump_work
   # compute the efficiency
   efficiency = (turbine_work - pump_work)/heat_added

   return [work, efficiency]
```

B Installing Cantera

Cantera can be downloaded from http://sourceforge.net/projects/cantera. Select "Files" then select the appropriate installer for your machine type.

B.1 Installing on a PC running Windows XP or 2000

From the same sourceforge site, download file 'cantera154-install.pdf' in the "Cantera Documentation" package under "building and installing." Follow the instructions in this document to install Cantera and configure MATLAB to use the Cantera Toolbox.

B.2 Installing on a linux workstation

To use Cantera under linux, download the unix/linux source distribution, unpack it, go into the top-level Cantera directory, and type at a shell prompt:

```
./configure
make
make install
```

You may want to edit the 'configure' script before running it, to set various options. Further information is in the files 'README', 'INSTALLING', and 'configure'.

Note that you need to compile Cantera with a version of g++ compatible with the compiler used to compile MATLAB itself. For MATLAB 7, you will need g++ version 3.x, while for MATLAB 6, version 2.95 is required.

B.3 Installing on a Mac running OS 10.3

A binary installer is available for OS 10.3 from the Cantera sourceforge site. This contains those portions of Cantera needed to access it from MATLAB and Python. Download the disk image 'Cantera.dmg', mount it, and run the 'Cantera.pkg' installer.

If you plan to use Cantera from C++ or Fortran, then you need to download the source distribution and build Cantera. You may also need to build Cantera yourself if you are running MATLAB 6 (R13), instead of 7 (R14). Follow the procedures described above for linux.

There are a few extra packages you may wish to install, in addition to those listed in Section B.4.

Tcl/Tk Aqua. Get this if you want to run Tcl/Tk applications from Python. http://tcltkaqua.sourceforge.net/. Highly recommended.

MacPython Although Python comes with OS X, there are a few add-ons that are useful to have. These may be installed by adding the MacPython package, available from http://homepages.cwi.nl/~jack/macpython/. Select the link "MacPython 2.3 for Panther addons" to download the disk image.

If you have installed MacPython, start the PackageManager, located in /Applications/MacPython. You should install at least the packages "readline," "Numeric," and "_tkinter." You may want to install some of the others too.

B.4 Extras

There are a couple of extra packages you may want to install. Neither of these is required to use Cantera from MATLAB, C++, or Fortran. If you plan to use Cantera from Python, the Numeric package is required. Graphviz is

optional.

Graphviz. Useful for displaying reaction path diagrams generated by Cantera (e.g. from MixMaster). The main download site is http://www.research.att.com/sw/tools/graphviz/download.html. Select the binary installer for your operating system.

Numeric. Adds fast MATLAB-like array operations to Python. For Windows or linux, go to http://sourceforge.net/projects/numpy/, and select the "numpy" file release (not "Numarray"). Get the appropriate binary installer or source distribution. For Mac OS X, you can install Numeric from the MacPython package manager.

B.5 Setting the MATLAB Path

Once you have installed Cantera, you need to tell MATLAB where to find it. From the MATLAB "File" menu, select "Set Path," then press the button "Add with subfolders." Navigate through the Cantera folder hierarchy to 'matlab/toolbox/cantera/cantera' and add it.

B.6 Environment Variables

These should not need to be set, but may be useful in some cases.

PYTHON_CMD If Python is not on the path, so that it cannot be found by simply typing "python," then set this to the full path to the Python interpreter.

CANTERA_DATA Cantera uses some data files, and needs to be able to find the folder these files are in. If you install Cantera in the "usual" place on your system, it will find the folder. But if you install it somewhere else, you can set this variable to the full path to the Cantera data folder. Alternatively, from your Python or MATLAB script you can add the command

```
addDirectory('/your/cantera/data/directory')
```

The places Cantera looks by default for the data folder are:

Windows. "C:\Common Program Files\Cantera\data"

Mac OS X. /Applications/Cantera/data

linux. /usr/local/cantera/data

Note that any location specified by CANTERA_DATA or addDirectory is searched first, before looking in these default locations.

B.7 The Setup Script

On the linux and Mac platforms, the installation includes a script called "setup_cantera." Depending on how you installed Cantera, it may or may not be necessary to run this script before using Cantera. If you need to run it, copy it to your home directory, and type

```
source ./setup_cantera
```

You can also add this command to your shell login script.

References

R. T. Jacobsen, R. B. Stewart, and A. F. Myers. An equation of state for oxygen and nitrogen. *Adv. Cryo. Engrg.*, 18: 248, 1972.

W. C. Reynolds. Thermodynamic properties in si. Technical report, Department of Mechanical Engineering, Stanford University, 1979.