Prode Properties Test File

This file may be used to insure that the Mathcad prode.dll file is in the right directory and that the functions are working properly. Prode Properties may be tested independent of Mathcad using the Excel version in the Prode directory. A few examples are provided to demonstrate how the functions may be used. The process examples (compressor, nozzles, etc.) were based on the Excel examples included in the Prode Properties installation.

Directions for using this worksheet

The default archive from Prode, def.ppp, will be used as the starting file. This worksheet will write changes to a new archive, test.ppp. This file will be placed in the same directory as def.ppp, but the user may also later archive to other directories.

Operations that set a variable will show a result of 1 if successful or 0 if not. Results that retrieve values will show the retrieved value, or 0 if no value is available. The exceptions to this convention will be noted.

Automatic calculation has been turned off so the new user may read these instructions before starting the computations.

The Prode function names are the same as Prode with the addition of "mc_" to the name.

Procedure:

1. Calculate the entire worksheet, ctrl-F9
2. Select dep.ppp archive when first window appears, and click Open
3. When second popup window, the Prode archive, appears, it should show stream 2. Click OK.
4. Scroll through the worksheet to check for errors. See "Errors" below.
5. If the Prode window pops up, it is allowing you to view a recently created stream. Click OK to close.

Errors:

- Mathcad will show errors in red as usual. Typical errors might be caused by syntax in the argument list, or the function may not be found in the library delivered by Prode, ppp.lib.
- If the result is zero, then the Prode function had an error or could not return a value. Frequently, this may be caused by the lack of a particular phase needed for an operation. For example, solid properties can't be returned when a stream has no solid phase, or when the temperature is above the melting point for a pure compound property.
- Rarely, a ppp.dll error window may appear with "Error accessing component's data archive". This appears to be caused by a lack of data in the chem.dat file for that particular property. If this window appears, it must be closed to proceed with the computations. See also the mc_defErrMsg function for a way to prevent these windows from stopping the calculations. The program is set to prevent these error windows.

File open commands

mc_AFOpen("C:\ProgramData\prode\def.ppp") =

disabled

This command sets the path to the archive file and directory. The path shown is the default set during the Prode Properties installation. This command will affect all subsequent

uses of Prode on this computer until the path is changed again by another AFOpen command. Therefore, this command should be used with caution.

`mc_AOpen("dummy") = 1` — browse for an archive in the default directory

The next two functions do not obey the normal Prode convention regarding the result returned if successful. A result of 0 means these were successful. They must be evaluated (followed by "=") in order for the operation to take place.

`mc_setErrFlag(0) = 0` — set to 0 at start of calc's to clear flag

`wopt := 0`

`mc_defErrMsg(wopt) = 0` — wopt = 0 turns off the Window Dialog messages
wopt = 1 turns on the Window Dialog messages

Open Properties window to view edit streams

`stream := 2`

`mc_edS(stream) = 1` — edit the given stream in the current active archive using the Prode window

`mc_edSS("dummy") =` — open to the first stream (disabled for the test to reduce popups)

Chemical file operations

`mc_getFCNr("dummy") = 1.635×10^3` — number of components in data file, should be 1635 or greater. If less, then you are using the free version of Prode and not all of the routines below will work.

`id := 7732185` — CAS number of water (use internet search to find values for compounds). Note that hyphens are not included.

compcode is an integer from 1 to number of components in the data file

`compcode := mc_CompCID(id) = 1631` — given id=CAS#, return compcode from database

Note: The above statement shows that the functions may be used to define a variable in addition to merely showing a result.

mc_CompF(compcode) = "H2O"	given a component code, returns component formula string
mc_CompN(compcode) = "WATER"	component name
mc_CompID(compcode) = 7732185	CAS number of component, compare to id above

Note: The units are not returned by the Prode commands. Operations that show which units are being used are shown later.

mc_CompMw(compcode) = 18.015	molecular weight
$T_c := mc\_CompTc(compcode) \cdot K = 647.096\,K$	critical temperature
$T_c = 1.165 \times 10^3 \cdot R$	multiply the function by the current Prode units for the result to use the unit features of Mathcad
$mc\_CompPc(compcode) = 2.206 \times 10^7$	critical pressure
$mc\_CompVc(compcode) = 3.106 \times 10^{-3}$	critical volume
mc_CompAc(compcode) = 0.344	acentric factor
mc_CompDm(compcode) = 0	dipole moment
$mc\_CompRg(compcode) = 6.15 \times 10^{-11}$	radius of gyration
$mc\_CompSol(compcode) = 1.512 \times 10^3$	solubility parameter
$mc\_CompHf(compcode) = -1.342 \times 10^4$	heat of formation
$mc\_CompGf(compcode) = -1.27 \times 10^4$	Gibbs energy of formation
mc_CompSf(compcode) = 333.474	enthalpy of fusion
mc_CompNb(compcode) = 373.15	normal boiling point
mc_CompMp(compcode) = 273.15	melting point temperature

The following provide non zero values only if the phase of interest is present at the temperature requested.

tgl := 300	temperature for gas/liquids (above freezing for water)
ts := 260	temperature for solids (below freezing)
$mc\_CompVP(compcode, tgl) = 3.548 \times 10^3$	saturation pressure at temp tgl
$mc\_CompHV(compcode, tgl) = 2.436 \times 10^3$	heat of vaporization at tgl
$mc\_CompLV(compcode, tgl) = 8.563 \times 10^{-4}$	liquid viscosity at tgl

$mc\_CompGV(compcode, tgl) = 9.925 \times 10^{-6}$	gas viscosity at tgl
$mc\_CompLD(compcode, tgl) = 995.476$	liquid density at tgl
$mc\_CompSD(compcode, ts) = 918.631$	solid density at ts
$mc\_CompLC(compcode, tgl) = 0.616$	liquid thermal conductivity at tgl
$mc\_CompGC(compcode, tgl) = 0.019$	gas thermal conductivity at tgl
$mc\_CompSC(compcode, ts) = 0$	solid thermal conductivity at ts (appears to be missing for water)
$mc\_CompST(compcode, tgl) = 0.072$	surface tension at tgl

integrated changes between two temperatures, t0 and t1 for pure components

$t0 := 280$ $t1 := 290$

$mc\_CompHG(compcode, t0, t1) = 18.611$	ideal gas enthalpy change
$mc\_CompSG(compcode, t0, t1) = 0.065$	ideal gas entropy change
$mc\_CompHL(compcode, t0, t1) = 42.018$	ideal liquid enthalpy change
$mc\_CompSL(compcode, t0, t1) = 0.147$	ideal liquid entropy change
$ts0 := 260$ $ts1 := 270$	lower the temperature range < freezing pt
$mc\_CompHS(compcode, ts0, ts1) = 20.524$	ideal solid enthalpy change
$mc\_CompSS(compcode, ts0, ts1) = 0.077$	ideal solid entropy change

Units commands

See "Units of Measurement" section in Prode manual for a list of the units and their numerical codes.

$UM := 15$	pressure is used for an example
$n\_press := mc\_getUMN(UM) = 20$	no. of units avail. for UM
$mc\_getUMC(UM) = 1$	present units code for UM
$mc\_getSUMS(UM) = \text{"Pa.a"}$	present units string for UM

sel := 5 — select unit 5

mc_getUMS(UM, sel) = "KPa.a" — units string for (UM, sel)

list all of the units for pressure

i := 1 .. n_press

P_units_i := mc_getUMS(UM, i)

P_units =

	0
0	0
1	"Pa.a"
2	"Pa.g"
3	"mbar.a"
4	"mbar.g"
5	"KPa.a"
6	"KPa.g"
7	"bar.a"
8	"bar.g"
9	"kgf/cmq.a"
10	"kgf/cmq.g"
11	"psi.a"
12	"psi.g"
13	...

mc_getP(stream)·Pa = 14.696·psi — multiply by current Prode pressure unit, then request any unit in the result

sel := 11 — select a new pressure unit

mc_setUMC(UM, sel) = 1 — change to the 11th unit for pressure

mc_getSUMS(UM) = "psi.a" — show current unit name for UM

mc_getP(stream)·psi = 14.697·psi — now pressure results must be multiplied by psi

mc_setUMC(UM, 1) = 1 — reset to original unit for remainder of worksheet

Routines UMCR, UMCS, and UMAU are not fully documented in the Prode manual so they have been left out of the dll.

mc_UMRAU(UM) = 1 — removes all added units for (property no.)

Error message flags

mc_ErrMsg("") = "Error accessing component's data archive" — last error message, maybe from a previous run

dum := "dummy"

errflag := mc_getErrFlag(dum) — 0 = no errors, 1 = errors found

errflag = 1 — This flag only works if the Window Dialog messages are turned off. Otherwise, the Dialog messages are themselves the indication of errors. See mc_defErrMsg. Errors that Mathcad detects (i.e. red indication) are not included for this flag.

At the time this test file was created, the thermal conductivity of solid water was not available in the database, causing an error and a value of 1 for errflag.

mc_setErrFlag(0) = 0 — set to 0 at start of next calc's to clear flag

mc_defErrMsg(0) = — 0 = turns off the Window Dialog messages
1 = turns on the Window Dialog messages

This function was demonstrated at start of worksheet. Turning off the Window Dialog messages allows the computations to continue without pausing to close the Dialog window when an error occurs. The error may still be visible if a 0 value is returned where a real number is expected.

Atmospheric pressure

patm := mc_getPatm("mc") = 1.013×10^5 — the pressure should be $1.013 \cdot 10^5$

Read/write stream properties

If a write operation exists, it will appear under the read operation, using the value from the read operation. This simplifies the testing process.

The write operations are in blue highlight.

stream := 1

phase := 2 — the phase position (not the phase type)

cpos := 2 — cpos is the component's numerical position in the composition vector for the stream, starting with 1

mc_isSDef(stream) = 1 — given a stream returns TRUE (integer = 1) if stream has been defined, otherwise returns FALSE (0)

name := mc_StrN(stream) = "Test Case 1" — stream name

mc_putN(stream, name) = 1

mc_setOp(stream, 150, patm) = 1 — This is an edit operation to lower the temperature so liquid will be present for the functions below.

t := mc_getT(stream) = 150 — temperature

mc_putT(stream, t) = 1

p := mc_getP(stream) = 1.013×10^5 — pressure

mc_putP(stream, p) = 1

pnr := mc_getPNr("mc") = 5 — returns the maximum number of phases that procedure can detect in the archive for all streams (may include phases at other temperatures)

mc_StrPt(stream, phase) = 1 — given a stream and phase # in range 1-getPNr() returns the phase type (0=vapor,1=liquid,2=solid)

i := 0 .. pnr − 1

$phases_i$:= mc_StrPts(stream, i + 1) — given a stream and phase # in range 1-getPNr() returns a ANSI C string with the description of type for detected phase

$$phases = \begin{pmatrix} \text{"Vapor"} \\ \text{"Liquid"} \\ \text{"Liquid"} \\ \text{"Not present"} \\ \text{"Not present"} \end{pmatrix}$$

mc_StrLf(stream) = 0.26	given a stream returns the total liquid fraction (molar basis) in stream
mc_StrPf(stream, phase) = 0.162	given a stream and phase phase # in range 1- getPNr() returns the phase fraction
w := mc_getW(stream, phase, cpos) = 0.276	mole fraction of component (cpos #) in a phase
mc_putW(stream, phase, cpos, w) = 1	mole fraction w of component cpos in a phase
stream := 5	
rate := mc_getWm(stream) = 1	stream flow rate, mass/time
mc_setWm(stream, rate) = 1	
stream := 1	
zi := mc_getZ(stream, cpos) = 0.15	mole fraction of component cpos in total stream
mc_putZ(stream, cpos, zi) = 1	
mc_getCNr(stream) = 3	number of components in stream
mc_StrZv(stream) = 0.985	returns the relevant compressibility factor (gas phase)
mc_StrMw(stream) = 22.944	molecular weight of total stream
mc_StrGMw(stream) = 17.565	molecular weight of gas phase
mc_StrLMw(stream) = 38.266	molecular weight of liquid phase
mc_StrV(stream) = 0.391	specific volume as sum of specific volumes of all phases

enthalpy

h := mc_StrH(stream) = 4.996×10^3	total stream enthalpy
mc_StrGH(stream) = 2.991×10^3	gas phase enthalpy
mc_StrSGH(stream) = 4.04×10^3	gas specific enthalpy
mc_StrLH(stream) = 2.006×10^3	liquid enthalpy
mc_StrSLH(stream) = 7.72×10^3	liquid specific enthalpy
mc_StrSH(stream) = 0	solid enthalpy
mc_StrSSH(stream) = 0	solid specific enthalpy

entropy

entropy := mc_StrS(stream) = 56.01	total stream entropy
mc_StrGS(stream) = 33.672	gas phase entropy
mc_StrSGS(stream) = 45.493	gas specific entropy
mc_StrLS(stream) = 22.338	liquid entropy
mc_StrSLS(stream) = 85.967	liquid specific entropy
mc_StrSS(stream) = 0	solid entropy
mc_StrSSS(stream) = 0	solid specific entropy

heat capacity, mass basis

mc_StrGICp(stream) = 1.887	ideal gas heat capacity
mc_StrGCp(stream) = 1.914	gas constant pressure heat capacity
mc_StrGCv(stream) = 1.416	gas constant volume heat capacity
mc_StrLCp(stream) = 1.732	liquid constant pressure heat capacity
mc_StrLCv(stream) = 1.209	liquid constant volume heat capacity
mc_StrSCp(stream) = 0	solid constant pressure heat capacity

speed of sound

mc_StrMSS(stream) = NaN	mixed phase speed of sound HEM model
mc_StrGSS(stream) = 266.87	gas phase
mc_StrLSS(stream) = 1.76×10^{3}	liquid phase

Joule Thomson coefficient

mc_StrGJT(stream) = 1.524×10^{-5}	gas phase
mc_StrLJT(stream) = -3.636×10^{-7}	liquid phase

compressibility, expansivity

mc_StrGIC(stream) = 1.003×10^{-5}	gas isothermal compressibility $\frac{1}{V} \cdot \left(\frac{d}{dP} V \right)$

$\mathrm{mc\_StrLIC(stream)} = 6.436 \times 10^{-10}$ liquid isothermal compressibility

$\mathrm{mc\_StrGVE(stream)} = -6.949 \times 10^{-3}$ gas volumetric expansivity $\frac{1}{V} \cdot \left(\frac{d}{dT} V \right)$

$\mathrm{mc\_StrLVE(stream)} = -1.639 \times 10^{-3}$ gas volumetric expansivity

density

$\mathrm{mc\_StrGD(stream)} = 1.449 \times 10^{0}$ gas density

$\mathrm{mc\_StrLD(stream)} = 1.198 \times 10^{3}$ liquid density

thermal conductivity

$\mathrm{mc\_StrGC(stream)} = 0.015$ gas conductivity

$\mathrm{mc\_StrLC(stream)} = 0.285$ liquid conductivity

viscosity

$\mathrm{mc\_StrGV(stream)} = 6.261 \times 10^{-6}$ gas viscosity

$\mathrm{mc\_StrLV(stream)} = 1.636 \times 10^{-4}$ liquid viscosity

surface tension

$\mathrm{mc\_StrST(stream)} = 0.035$ liquid/gas

flammability

$\mathrm{mc\_StrFML(stream)} = 4.993$ gas phase lean limit

$\mathrm{mc\_StrFMH(stream)} = 15.082$ gas phase rich limit

other stream properties

$\mathrm{mc\_StrHC(stream)} = 4.332 \times 10^{4}$ gas phase heat of combustion

$\mathrm{compcode} := \mathrm{mc\_getCC(stream, cpos)} = 594$ component number for component = cpos

$\mathrm{mc\_putCC(stream, cpos, compcode)} = 1$

$\mathrm{mc\_getMCNr("dummy")} = 50$ maximum number of components in a stream

interactions

int_pos := 1 — the interaction number

mc_getMBPNr("dummy") = 250 — maximum number of binary pairs in a stream

ci := mc_getCi(stream, int_pos) = 1 — component index i in interaction list

mc_putCi(stream, int_pos, ci) = 1

cj := mc_getCj(stream, int_pos) = 2 — component index j in interaction list

mc_putCj(stream, int_pos, cj) = 1

model := mc_getMod(stream, int_pos) = 51 — returns model number for interaction int_pos

mc_putMod(stream, int_pos, model)

Kji := mc_getKji(stream, int_pos) = 0.1 — Kji interaction coefficient

mc_putKji(stream, int_pos, Kji) = 1

Gji := mc_getGji(stream, int_pos) = 0 — Gji interaction coefficient

mc_putGji(stream, int_pos, Gji) = 1

Gij := mc_getGij(stream, int_pos) = 0 — Gij interaction coefficient

mc_putGij(stream, int_pos, Gij) = 1

Aji := mc_getAji(stream, int_pos) = 0 — Aji interaction coefficient

mc_putAji(stream, int_pos, Aji) = 1

thermodynamic models for streams

state := 0 stream = 1

fmodel := mc_getMFg(stream, state) = 51 — fugacity model

mc_setMFg(stream, fmodel, state) = 1

hmodel := mc_getMH(stream, state) = 51 — enthalpy model for state

mc_setMH(stream, hmodel, state) = 1

smodel := mc_getMS(stream, state) = 51 — entropy model

mc_setMS(stream, smodel, state) = 1

vmodel := mc_getMV(stream, state) = 51 volume model

mc_setMV(stream, vmodel, state) = 1

see also the methods **here** for probing properties of a stream at different operating conditions

Thermodynamic calculations

stream := 5 use second stream for examples below

t = 150 $p = 1.013 \times 10^5$

state := 1 state (0=vapor, 1=liquid, 2=solid)

phase equilibria

n := 1 pf := .3 see below

mc_PfPF(stream, p, pf, state, n) = 292.316 n th equilibrium temp at p, pf (phase fraction), state (0=vapor, 1=liquid, 2=solid)

mc_PfTF(stream, t, pf, state, n) = 0 n th equil. press at t, pf, state

lf := .2 set liquid fraction

mc_LfPF(stream, p, lf) = 300.736 first equil. temp at liquid fraction, lf

mc_LfTF(stream, t, lf) = 0 first equil. pressure at liquid fraction, lf

mc_StrCPnr(stream) = 1 number of critical points found

cpn := 1

mc_StrPc(stream, cpn) = 4.708×10^6 critical pressure for critical point #, cpn

mc_StrCBp(stream) = 0 cricondenBar pressure

mc_StrCBt(stream) = 0 cricondenBar temperature

mc_StrCTp(stream) = 4.474×10^6 cricondenTherm pressure

mc_StrCTt(stream) = 449.239 cricondenTherm temperature

mc_StrAc(stream) = 0.208 acentric factor (mole fraction average)

phase diagrams

stream := 5

lnr := mc_PELNr(stream)

Given a stream calculates the phase diagram and returns the number of equilibrium lines available

lnr = 2

line types

line := 2

ltype := mc_PELT(stream, line)

Given a stream and line number, returns the line type:
1. bubble line
2. dew line
3. three phase line
4. fractional phase

ltype = 2

line properties

lprop := mc_PELP(stream, line)

Given a stream and line, returns the line properties:
1. vapor-liquid
2. vapor-liquid-liquid
3. vapor-solid
4. liquid-solid
5. fractional phase

lprop = 1

equilibrium lines

The prode.dll has assumed a maximum number of points of 50 for the equilibrium lines. This dimension cannot be changed dynamically for the variables passed to and from Mathcad. Therefore, the mc_PELine routine leaves out the maxpt variable that is shown in the Prode corresponding routine.

The mc_PELine function (see the first line in the program below) produces a matrix result. Although this matrix can be used "as is" the Mathcad program, "PELine" below calls mc_PELine and splits the matrix into the separate variables.

PELine(stream, line) :=
- M ← mc_PELine(stream, line)
- npts ← $M_{0,2}$
- T ← submatrix(M, 0, npts − 1, 0, 0)
- P ← submatrix(M, 0, npts − 1, 1, 1)
- (T P npts)

$(T1 \quad P1 \quad npts1) := PELine(stream, line)$

Given stream and equilibrium line number, the temperature and pressure vectors and the total number of points are computed and returned.

The output is shown below.

$npts1 = 33$

$T1 =$

	0
0	311.865
1	316.865
2	321.865
3	326.865
4	331.865
5	336.865
6	341.865
7	346.865
8	351.865
9	356.865
10	361.865
11	366.865
12	371.865
13	...

$P1 =$

	0
0	$1.013 \cdot 10^5$
1	$1.225 \cdot 10^5$
2	$1.47 \cdot 10^5$
3	$1.754 \cdot 10^5$
4	$2.079 \cdot 10^5$
5	$2.451 \cdot 10^5$
6	$2.874 \cdot 10^5$
7	$3.353 \cdot 10^5$
8	$3.893 \cdot 10^5$
9	$4.5 \cdot 10^5$
10	$5.18 \cdot 10^5$
11	$5.94 \cdot 10^5$
12	$6.786 \cdot 10^5$
13	...

phase fraction lines

$stream := 5$

$state := 0$

$fraction := .5$

$$PFLine(stream, state, fraction) := \begin{array}{l} M \leftarrow mc\_PFLine(stream, state, fraction) \\ npts \leftarrow M_{0,2} \\ T \leftarrow submatrix(M, 0, npts - 1, 0, 0) \\ P \leftarrow submatrix(M, 0, npts - 1, 1, 1) \\ (T \quad P \quad npts) \end{array}$$

(Tf Pf nf) := PFLine(stream, state, fraction)

nf = 35

Given stream, state, and fraction of that state, computes the temperature and pressure vectors along that phase fraction, plus the number of points on the curve.

Tf =

	0
0	273.111
1	278.111
2	283.111
3	288.111
4	293.111
5	298.111
6	303.111
7	308.111
8	313.111
9	318.111
10	323.111
11	328.111
12	333.111
13	...

Pf =

	0
0	$1.013 \cdot 10^5$
1	$1.212 \cdot 10^5$
2	$1.44 \cdot 10^5$
3	$1.7 \cdot 10^5$
4	$1.995 \cdot 10^5$
5	$2.328 \cdot 10^5$
6	$2.702 \cdot 10^5$
7	$3.121 \cdot 10^5$
8	$3.587 \cdot 10^5$
9	$4.104 \cdot 10^5$
10	$4.675 \cdot 10^5$
11	$5.304 \cdot 10^5$
12	$5.995 \cdot 10^5$
13	...

The Mathcad program, "PhaseEnv" below obtains all of the equilibrium curves which can then be plotted.

PhaseEnv(stream) :=
- lnr ← mc_PELNr(stream)
- for line ∈ 1 .. lnr
  - $\left(t^{\langle line-1 \rangle} \quad p^{\langle line-1 \rangle} \quad n_{line-1}\right) \leftarrow$ PELine(stream, line)
  - $ltype_{line-1} \leftarrow$ mc_PELT(stream, line)
  - $lprop_{line-1} \leftarrow$ mc_PELP(stream, line)
- (t p lnr n ltype lprop)

stream := 5

(Tj Pj lnr nc type prop) := PhaseEnv(stream)

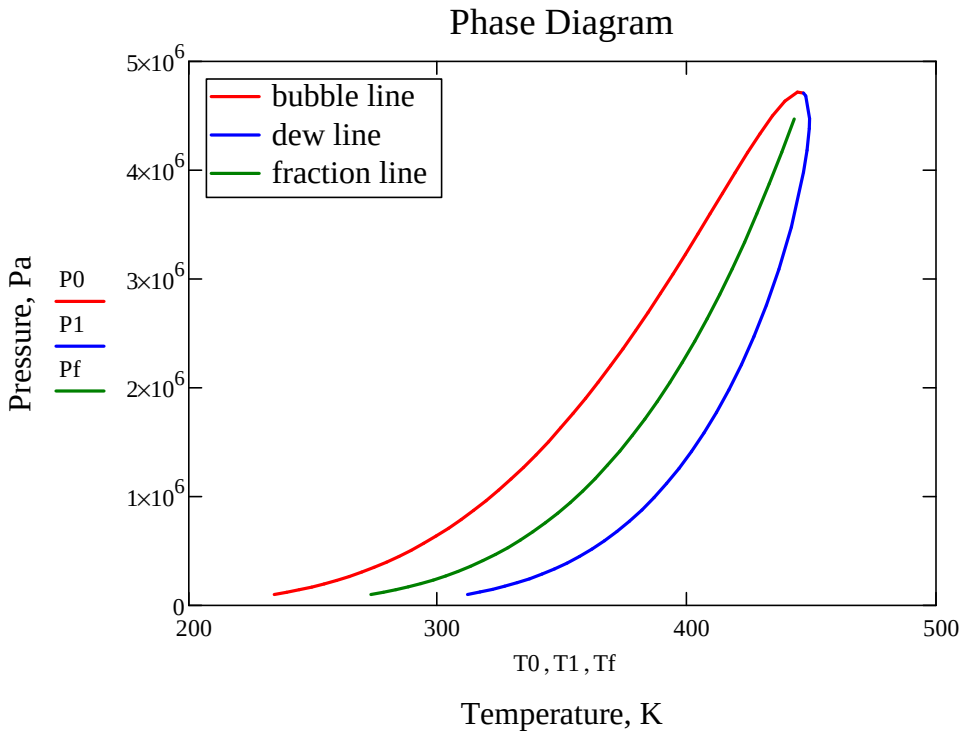
lnr = 2

$$type = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad nc = \begin{pmatrix} 44 \\ 33 \end{pmatrix}$$

As shown in the nc vector, the lines may have different number of points. In order to prevent curves returning to the origin, extract the data from Tj and Pj.

$T0 := \text{submatrix}(Tj, 0, nc_0 - 1, 0, 0)$ $\quad$ $P0 := \text{submatrix}(Pj, 0, nc_0 - 1, 0, 0)$

$T1 := \text{submatrix}(Tj, 0, nc_1 - 1, 1, 1)$ $\quad$ $P1 := \text{submatrix}(Pj, 0, nc_1 - 1, 1, 1)$



The legend labels in the above plot were input manually supplied based on the values of the "type" results. Also, the number of curves is determined manually using lnr as guidance. Some streams have more equilibrium curves due to multiple liquid and/or solids.

The PELine, PFLine, and PhaseEnv programs may be copied into or referenced by other programs.

hydrates

hydmodel := 1 thyd := 260

str_hyd := 2 stream for hydrate function below

hydmodel =
1 = assume free water present, this option produces conservative but safe values
2 = calculate amount of water in liquid phase
3 = solve as multiphase equilibria, solve phase equilibria including solids as ice

Since water is not present in the stream chosen for testing, the 2 and 3 hydmodels will return 0.

phyd := mc_HPFORM(str_hyd, thyd, hydmodel) = 1.822×10^{6} returns the pressure that hydrates form at temperature = thyd

The HTFORM Prode function is not available in the Basic version, but the HPFORM function should suffice.

flashes

stream = 5

mc_setSOp(stream) = 1 flash at standard conditions

et := 0 estimated temperature set to 0 for automatic

ep := 0 estimated pressure set to 0 for automatic

mc_setOp(stream, 150, patm) = 1 set new operating conditions and flash

t := mc_getT(stream) = 150 temperature

p := mc_getP(stream) = 1.013×10^{5} pressure

h := mc_StrH(stream) = 4.681×10^{3} enthalpy obtained above

entropy := mc_StrS(stream) = 53.18 entropy obtained above

sv := mc_StrV(stream) = 1.379×10^{-3} volume obtained above

find temperature

mc_VPF(stream, p, sv, et) = 150 volume-pressure flash, et=temp guess

mc_HPF(stream, p, h, et) = 150 enthalpy-pressure flash, et=temp guess

mc_SPF(stream, p, entropy, et) = 150 entropy-pressure flash, et=temp guess

find pressure

$mc\_VTF(stream, t, sv, ep) = 0$	volume-temp flash, ep=press guess
$mc\_HTF(stream, t, h, ep) = 0$	enthalpy-temp flash, ep=press guess
$mc\_STF(stream, t, entropy, ep) = 0$	entropy-temp flash, ep=press guess

The flashes that determine pressure have some difficulty converging for multiphase (liquids and solids) problems. Select another flash routine and iterate if needed.

Additional flashes for mixing and dividing streams are found at this **section**

Extended methods for accessing stream properties

These functions allow simultaneous setting of temperature and pressure followed by an isothermal flash before the desired property is returned. These methods should be used with care because of the change in the stream conditions.

$stream := 2$

$state\_pos := 1$	position of state, not the state code positions are usually vapor=1, liquid=2, solid=3
$mc\_EStrGMw(stream, t, p) = 16.047$	gas molecular weight
$mc\_EStrLMw(stream, t, p) = 55.298$	liquid molecular weight
$mc\_EStrLf(stream, t, p) = 9.572 \times 10^{-4}$	mole fraction of liquid
$mc\_EStrPf(stream, state\_pos, t, p) = 0.999$	molar phase fraction of phase number = state_pos
$mc\_EStrZv(stream, t, p) = 0.984$	gas (vapor) compressibility factor
$mc\_EStrH(stream, t, p) = 5.304 \times 10^{3}$	total enthalpy
$mc\_EStrV(stream, t, p) = 0.752$	total specific volume
$mc\_EStrGCp(stream, t, p) = 2.103$	gas constant pressure heat capacity
$mc\_EStrGCv(stream, t, p) = 1.559$	gas constant volume heat capacity
$mc\_EStrLCp(stream, t, p) = 1.709$	liquid constant pressure heat capacity
$mc\_EStrLCv(stream, t, p) = 1.399$	liquid constant volume heat capacity
$mc\_EStrGIC(stream, t, p) = 1.003 \times 10^{-5}$	gas isothermal compressibility

$mc\_EStrLIC(stream, t, p) = 4.991 \times 10^{-10}$	liquid isothermal compressibility
$mc\_EStrMSS(stream, t, p) = 317.724$	mixture speed of sound
$mc\_EStrGSS(stream, t, p) = 318.267$	gas speed of sound
$mc\_EStrLSS(stream, t, p) = 3.471 \times 10^{3}$	liquid speed of sound
$mc\_EStrGJT(stream, t, p) = 1.466 \times 10^{-5}$	gas Joule Thomson coefficient
$mc\_EStrLJT(stream, t, p) = -7.311 \times 10^{-7}$	liquid Joule Thomson coefficient
$mc\_EStrGVE(stream, t, p) = -6.939 \times 10^{-3}$	gas volumetric expansivity coefficient
$mc\_EStrLVE(stream, t, p) = -8.487 \times 10^{-4}$	liquid volumetric expansivity coefficient
$mc\_EStrHC(stream, t, p) = 5.001 \times 10^{4}$	heat of combustion
$mc\_EStrFML(stream, t, p) = 4.999$	lean flammability limit of gas
$mc\_EStrFMH(stream, t, p) = 14.999$	rich flammability limit of gas
$mc\_EStrS(stream, t, p) = 60.353$	total entropy
$mc\_EStrGD(stream, t, p) = 1.325$	gas density
$mc\_EStrLD(stream, t, p) = 698.388$	liquid density
$mc\_EStrGC(stream, t, p) = 0.016$	gas thermal conductivity
$mc\_EStrLC(stream, t, p) = 0.175$	liquid thermal conductivity
$mc\_EStrGV(stream, t, p) = 6.028 \times 10^{-6}$	gas viscosity
$mc\_EStrLV(stream, t, p) = 1.245 \times 10^{-3}$	liquid viscosity
$mc\_EStrST(stream, t, p) = 0.029$	surface tension

Fugacity and derivatives

The operations below behave like subroutines rather than functions because they return more than one result. The Mathcad system imposes some restrictions on function intput and output so the normal C++ methods of passing variables is not possible. These restrictions are needed to enforce the "non code" look of the Mathcad interface. As a result of these restrictions, the functions below have slightly different argument lists than found in Prode and all of the results are returned in a single matrix. Mathcad routines are then provided to split these results into the appropriate variables.

The prode.dll has assumed a maximum number of components of 50 for all vector and matrix routines. This dimension cannot be changed dynamically for the variables passed to and from Mathcad. For greater number of components, prode.dll must be rebuilt. The constant "maxnc" in the source code for the routines in this section must be changed to the higher number.

The "fugacity" returned by the Prode routines is actually the fugacity coefficient times the total pressure. Multiply the "fg" results below by the corresponding liquid or vapor fraction to obtain the fugacity of that component. The term "fugacity multiplyer" will be used herein.

Because Mathcad programs have been supplied to separate the results into individual arrays, the names do not require the "mc_" convention.

$stream := 1$

$NC := mc\_getCNr(stream)$

$t = 150 \qquad p = 1.013 \times 10^5$ These variables were defined above.

$mc\_setOp(stream, t, p) = 1$

$i := 0 .. NC - 1$

$phase := 2$ This is the liquid phase. Caution!. For some streams, there may be no vapor phase, so the liquid phase would be 1.

$w_i := mc\_getW(stream, phase, i + 1)$ The composition of the liquid phase.

$$w = \begin{pmatrix} 0.01 \\ 0.262 \\ 0.727 \end{pmatrix}$$

fugacity multiplyer vector

$state := 1$ The liquid state is being used.

$fg := mc\_StrFv(stream, state, t, p, w, NC) \cdot Pa$ This routine returns the fugacity multiplyer without the derivatives.

$$fg = \begin{pmatrix} 9.196 \times 10^6 \\ 1.815 \times 10^4 \\ 1.05 \times 10^3 \end{pmatrix} Pa$$

$\text{fugacity}_i := w_i \cdot fg_i$

$$\text{fugacity} = \begin{pmatrix} 9.424 \times 10^4 \\ 4.759 \times 10^3 \\ 763.817 \end{pmatrix} \text{Pa}$$

fugacity multiplyer vector plus derivatives wrt T, P, w

$$\text{StrFvd}(\text{stream}, \text{state}, t, p, w, NC) := \left| \begin{array}{l} M \leftarrow \text{mc\_StrFvd}(\text{stream}, \text{state}, t, p, w, NC) \\ fg \leftarrow \text{submatrix}(M, 0, NC - 1, 0, 0) \\ dfgt \leftarrow \text{submatrix}(M, 0, NC - 1, 1, 1) \\ dfgp \leftarrow \text{submatrix}(M, 0, NC - 1, 2, 2) \\ dfgw \leftarrow \text{submatrix}(M, 0, NC - 1, 3, 3 + NC - 1) \\ (fg \quad dfgt \quad dfgp \quad dfgw) \end{array} \right.$$

$(fg \quad dfgt \quad dfgp \quad dfgw) := \text{StrFvd}(\text{stream}, \text{state}, t, p, w, NC)$

given the stream, state, temp, press, composition vector, w, and the number of components, NC, return fugacity multplyer vector, fg, and derivatives of fg wrt t, p, and w.

$$fg \cdot \text{Pa} = \begin{pmatrix} 9.196 \times 10^6 \\ 1.815 \times 10^4 \\ 1.05 \times 10^3 \end{pmatrix} \text{Pa} \qquad dfgt \cdot \frac{\text{Pa}}{\text{K}} = \begin{pmatrix} 2.127 \times 10^5 \\ 1.648 \times 10^3 \\ 114.748 \end{pmatrix} \frac{\text{kg}}{\text{m} \cdot \text{K} \cdot \text{s}^2}$$

$$dfgp = \begin{pmatrix} 2.127 \times 10^5 \\ 1.648 \times 10^3 \\ 114.748 \end{pmatrix} \qquad dfgw \cdot \text{Pa} = \begin{pmatrix} -1.181 \times 10^8 & -9.559 \times 10^7 & -6.799 \times 10^7 \\ -7.563 \times 10^4 & -7.834 \times 10^4 & -2.081 \times 10^4 \\ 1.768 \times 10^3 & 1.788 \times 10^3 & 543.402 \end{pmatrix} \text{Pa}$$

The results assume the default dimensions in Prode. The displayed results have been multiplied by those dimensions.

Other stream state variables and their derivatives

Functions were provided above (eg. mc_StrH) to obtain the enthalpy (H), entropy(S), and molar volume (V) of a stream. The next routine allows the operating conditions (t, p, w) to be specified to values other than those in the stream data file. The user selects which variable, H, S, or V, is desired using the "X" argument. The program calls the appropriate mc_xxx function and then separates the variables from the output matrix.

$$\text{StrXvd}(X,\text{stream},\text{state},t,p,w,NC) := \begin{array}{l} M \leftarrow \text{mc\_StrHvd}(\text{stream},\text{state},t,p,w,NC) \quad \text{if } X = \text{"H"} \\ M \leftarrow \text{mc\_StrSvd}(\text{stream},\text{state},t,p,w,NC) \quad \text{if } X = \text{"S"} \\ M \leftarrow \text{mc\_StrVvd}(\text{stream},\text{state},t,p,w,NC) \quad \text{if } X = \text{"V"} \\ x \leftarrow M^{\langle 0 \rangle} \\ dxt \leftarrow M^{\langle 1 \rangle} \\ dxp \leftarrow M^{\langle 3 \rangle} \\ dxw \leftarrow \text{submatrix}(M,0,0,3,NC+2) \\ (x \quad dxt \quad dxp \quad dxw) \end{array}$$

$KJ := 1000 \cdot J$ $\quad$ $Kmol := 1000 \cdot mol$ $\quad$ new units for Mathcad

$(H \quad dHt \quad dHp \quad dHw) := \text{StrXvd}(\text{"H"},\text{stream},\text{state},t,p,w,NC)$

$$H \cdot \frac{KJ}{Kmol} = \left(1.679 \times 10^{5}\right) \cdot \frac{KJ}{Kmol}$$

default units have been applied to the results

$$dHt \cdot \frac{\frac{KJ}{Kmol}}{K} = (64.929) \cdot \frac{\frac{KJ}{Kmol}}{K}$$

$$dHp \cdot \frac{\frac{KJ}{Kmol}}{Pa} = \left(8.085 \times 10^{4}\right) \cdot \frac{\frac{KJ}{Kmol}}{Pa}$$

$$dHw \cdot \frac{KJ}{Kmol} = \left(8.085 \times 10^{4} \quad 2.074 \times 10^{5} \quad 1.549 \times 10^{5}\right) \cdot \frac{KJ}{Kmol}$$

$(S \quad dSt \quad dSp \quad dSw) := \text{StrXvd}(\text{"S"},\text{stream},\text{state},t,p,w,NC)$

$$S \cdot \frac{KJ}{Kmol \cdot K} = \left(1.888 \times 10^{3}\right) \cdot \frac{KJ}{Kmol \cdot K}$$

$$dSt \cdot \frac{KJ}{Kmol \cdot K^{2}} = (0.433) \cdot \frac{KJ}{Kmol \cdot K^{2}}$$

$$dSp \cdot \frac{KJ}{Kmol \cdot K \cdot Pa} = (919.223) \cdot \frac{KJ}{Kmol \cdot K \cdot Pa}$$

$$dSw \cdot \frac{KJ}{Kmol \cdot K} = \left(919.223 \quad 2.264 \times 10^{3} \quad 1.789 \times 10^{3}\right) \cdot \frac{KJ}{Kmol \cdot K}$$

$(V \quad dVt \quad dVp \quad dVw) := \mathrm{StrXvd}("V", stream, state, t, p, w, NC)$

$$V \cdot \frac{m^3}{mol} = (0.032)\frac{m^3}{mol}$$

$$dVt \cdot \frac{m^3}{mol \cdot K} = \left(4.846 \times 10^{-5}\right) \cdot \frac{m^3}{mol \cdot K}$$

$$dVp \cdot \frac{m^3}{mol \cdot Pa} = (0.072) \cdot \frac{m^3}{mol \cdot Pa}$$

$$dVw \cdot \frac{m^3}{mol} = (0.072 \quad 0.066 \quad 0.066)\frac{m^3}{mol}$$

Operations to set/retrieve the options needed for equation of state models

$option := \mathrm{mc\_getOM}(stream) = 553$ current option set

As an example, a value of 552 means:

No multi liquid phases
No solid equilibria
Standard tests are used for multiphase initiation
Peneloux corrections are used for liquid volume
Use either Gibbs free energy or isocompressibility to detect a single phase
Discard unstable solutions
Critical points are not validated, only estimated
Select EOS roots according to state
Standard EOS parameters will be used

See the Prode manual and also open the Prode drop menus for the model to view the other options available.

$\mathrm{mc\_setOM}(stream, option + 16) = 1$ this should change the options so that only isocompressibility is used to detect a single phase

$\mathrm{mc\_setOM}(stream, option) = 1$ reset to original value

Initializing a stream

The example will create a stream with water and methanol.

methanol_id := 67561 CAS number

methanol_code := mc_CompCID(methanol_id) = 1.047×10^{3}

water_code := 1631

stream := 11

mc_initS(stream) = 1 initialize a new stream

model := 51 see Prode manual for model codes

mc_setMFg(stream, 51, 0) = 1 set vapor VLE model

mc_setMFg(stream, 51, 1) = 1 set liquid VLE model

mc_putZ(stream, 1, .5) = 1 set total stream mole fractions

mc_putZ(stream, 2, .5) = 1

mc_putCC(stream, 1, methanol_code) = 1 define components

mc_putCC(stream, 2, water_code) = 1

mc_setS(stream) = 1 validate the stream

mc_loadSB(stream) = 1 load BIP coefficients

mc_setWm(stream, 1.3) = 1 set mass flow rate

temp := 300

pres := patm

mc_setOp(stream, temp, pres) = 1 set temp and pres and flash

mc_edS(stream) = view the stream then press OK (if enabled)

Other stream operations

stream2 := 1

stream1 := 10

mc_StrCopy(stream1, stream2) = 1 copy stream2 to stream1 (note the order!)

et := mc_getT(stream2) = 150

mc_getT(stream1) = 150

mc_MixF(stream1, stream2, et) = 1

flash at lower stream press, et=temp guess for mixed stream. The sum of the streams replaces stream1. A new stream is NOT created.

mc_getT(stream1) = 150.373

mixed stream temperature

stream2 := 11

the new stream to be created by Divi

wdiv := .7

mc_Divi(stream1, stream2, wdiv) = 1

Given one stream (stream1) and a flowrate fraction (0-1) performs a divider operation so that stream 1 is shifted into two streams (stream1, stream2) of the same composition, temperature and pressure, flowrate fractions are subdivided as
specified by wdiv (stream2 = wdiv, stream1 = 1- wdiv)

Only one new stream is created, NOT two. The starting stream gets overwritten.

phase separation

stream1 := 5

mc_setOp(stream1, 288, patm) = 1

set the temperature to provide two phases for the operations below

stream2 := 12

the new stream to be created by PSep

phase := 1

phase number to separate, NOT the phase type

mc_PSep(stream1, stream2, phase) = 1

Given a stream (stream1) performs an isothermal flash to simulate a phase separator and returns the specified phase number (not phase type) to stream2.

gasstream := 13

mc_GSep(stream1, gasstream) = 1

Given a stream (stream1) performs an isothermal flash to simulate a phase separator and returns the gas phase to gasstream

liqstream := 14

mc_LSep(stream1, liqstream) = 1

Given a stream (stream1) performs an isothermal flash to simulate a phase separator and returns the liq phase(s) to liqstream

Polytropic compressor/expander

rate compressor efficiency

$pin := 10^6$ pressure in Pa

$pout := 2 \cdot 10^6$

$tin := 300$ temperature in K

$tout := 370$

$model := 2$ for a rating, model may be the following:
2 = Huntington method
4 = Paron method

$stream := 2$

$mc\_setOp(stream, tin, pin) = 1$ set the inlet stream conditions

$mc\_PSPF(stream, pout, model, tout) = 0.743$ efficiency rating and "stream" in archive now contains the outlet conditions

design model

$eff := .75$ polytropic efficiency given

$model := 1$ for a design, model may be the following:
1 = Huntington
3 = Paron

$mc\_setOp(stream, tin, pin) = 1$ reset the inlet stream conditions

$mc\_PSPF(stream, pout, model, eff) = 369.345$ outlet temperature and "stream" now contains the outlet conditions

Isentropic expansion, nozzles

$stream := 5$

$tin := 340$ $pin := 2 \cdot 10^6$

$mc\_setWm(stream, 1.23) = 1$

$mc\_setOp(stream, tin, pin) = 1$ set stream conditions

$model := 2$

model options (more explanation needed):
1 = homogeneous, equilibrium
2 = homogeneous, non equilibrium
3 = homogeneous, non equilibrium ?
4 = nonhomogeneous, non equilibrium

$pout := patm$

$parameter := .75$

Prode manual does not explain

$mc\_ISPF(stream, pout, model, parameter) = 4.239 \times 10^{-5}$ calculated orifice area, m^2

$mc\_getErrFlag(" ") = 0$

Pipe flow

The PIPE function is only available for users with an extended Prode license.

$model := 1$

$stream := 1$

$diam := \frac{1 \cdot in}{m} = 0.025$

$rough := .00045$

$length := \frac{100 \cdot km}{m} = 1 \times 10^{5}$

$dHeight := 0$

$dHeat := 0$

$mc\_PIPE(stream, model, diam, rough, length, dHeight, dHeat) = 0$

The result above will be 0 if the user has a Basic Prode license or 1 for an Extended license. The pressure and phase changes are made in the stream databank.

Parameters :
stream (int) inlet stream
model (int) model for fluid flow and phase equilibria (see below)
diam (double) pipe internal diameter
rough (double) parameter defining relative pipe roughness
length (double) length of this segment
dHeight (double) height difference (inlet, outlet)
dHeat (double) heat added, removed
Codes for models
1 Beggs & Brill / Hazen-Williams / AGA
additional models on request to Prode

File save

`mc_AFSave("C:\ProgramData\prode\test.ppp") = 1` save modifications to a new archive