Prode Properties Test File

for Prode Properties ver. 1.2b

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. 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.

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") = 58

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)

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

compcode := mc_CompCID(id) = 21 — 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) = 2.206×10^7 — critical pressure

mc_CompVc(compcode) = 3.106×10^{-3} — critical volume

mc_CompAc(compcode) = 0.344 — acentric factor

mc_CompDm(compcode) = 0 — dipole moment

mc_CompRg(compcode) = 6.15×10^{-11} — radius of gyration

mc_CompSol(compcode) = 1.512×10^3 — solubility parameter

mc_CompHf(compcode) = -1.342×10^4 — heat of formation

mc_CompGf(compcode) = -1.27×10^4 — 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
$mc\_CompLV(compcode, tgl) = 8.562 \times 10^{-4}$	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) = "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 $1.013 \cdot 10^5$

Base values for enthalpy and entropy

This section is new for this version

The default values for the base temperature, enthalpy, and entropy may be found in the config>settings Prode window. The functions below may be used to change the settings. The settings apply only to the current archive. The settings for the archive are saved when the archive is saved.

Code Procedure
1 = initial values specified by user (values of tref and val)
2 = initial values are enthalpy of formation (or entropy of formation) and temperature 25 C

If code = 2, the tref and val inputs are ignored.

code := 1 tref := 298 val := 0

mc_setHB(code, tref, val) = 1 enthalpy references

mc_setSB(code, tref, val) = 1 entropy references

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 in this section 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

$p := mc\_getP(stream) = 1.013 \times 10^5$ 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

$$phases = \begin{pmatrix} "Vapor" \\ "Liquid" \\ "Liquid" \\ "Not\ present" \\ "Not\ present" \end{pmatrix}$$ only one liquid phase is present later, the flash routine code will be reset to obtain two liquid phases

$mc\_StrLf(stream) = 0.26$ given a stream returns the total liquid fraction (molar basis) in stream

$mc\_StrPf(stream, phase) = 0.161$ given a stream and phase phase # in range 1- getPNr() returns the phase fraction

$w := mc\_getW(stream, phase, cpos) = 0.272$ mole fraction of component (cpos #) in a phase

$mc\_putW(stream, phase, cpos, w) = 1$ mole fraction w of component cpos in a phase

$rate := mc\_getWm(stream) = 1$ stream flow rate, mass/time

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

$zi := mc\_getZ(stream, cpos) = 0.15$ 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) = 0.984	returns the relevant compressibility factor (gas phase)
mc_StrMw(stream) = 22.944	molecular weight of total stream
mc_StrGMw(stream) = 17.563	molecular weight of gas phase
mc_StrLMw(stream) = 38.272	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) = −434.413	total stream enthalpy
mc_StrGH(stream) = −163.905	gas phase enthalpy
mc_StrSGH(stream) = −289.294	gas specific enthalpy
mc_StrLH(stream) = −270.508	liquid enthalpy
mc_StrSLH(stream) = −624.11	liquid specific enthalpy
mc_StrSH(stream) = 0	solid enthalpy
mc_StrSSH(stream) = 0	solid specific enthalpy

entropy

entropy := mc_StrS(stream) = −2.137	total stream entropy
mc_StrGS(stream) = −0.756	gas phase entropy
mc_StrSGS(stream) = −1.335	gas specific entropy
mc_StrLS(stream) = −1.38	liquid entropy
mc_StrSLS(stream) = −3.185	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

$mc\_StrGCp(stream) = 1.915$ gas constant pressure heat capacity

$mc\_StrGCv(stream) = 1.416$ gas constant volume heat capacity

$mc\_StrLCp(stream) = 1.737$ liquid constant pressure heat capacity

$mc\_StrLCv(stream) = 1.215$ liquid constant volume heat capacity

$mc\_StrSCp(stream) = 0$ solid constant pressure heat capacity

speed of sound

$mc\_StrMSS(stream) = 0$ mixed phase speed of sound HEM model

$mc\_StrGSS(stream) = 266.864$ gas phase

$mc\_StrLSS(stream) = 1.894 \times 10^{3}$ liquid phase

Joule Thomson coefficient

$mc\_StrGJT(stream) = 1.528 \times 10^{-5}$ gas phase

$mc\_StrLJT(stream) = -3.992 \times 10^{-7}$ liquid phase

compressibility, expansivity

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

$mc\_StrLIC(stream) = 5.977 \times 10^{-10}$ liquid isothermal compressibility

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

$mc\_StrLVE(stream) = -1.522 \times 10^{-3}$ gas volumetric expansivity

density

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

$mc\_StrLD(stream) = 1.113 \times 10^{3}$ liquid density

thermal conductivity

$mc\_StrGC(stream) = 0.015$ gas conductivity

$mc\_StrLC(stream) = 0.285$ liquid conductivity

viscosity

$mc\_StrGV(stream) = 6.261 \times 10^{-6}$	gas viscosity
$mc\_StrLV(stream) = 1.649 \times 10^{-4}$	liquid viscosity

surface tension

$mc\_StrST(stream) = 0.034$	liquid/gas

flammability

$mc\_StrFML(stream) = 4.993$	gas phase lean limit
$mc\_StrFMH(stream) = 15.082$	gas phase rich limit

other stream properties

$mc\_StrHC(stream) = 4.332 \times 10^{4}$	gas phase heat of combustion
$compcode := mc\_getCC(stream, cpos) = 2$	component number for component = cpos
$mc\_putCC(stream, cpos, compcode) = 1$	
$mc\_getMCNr("dummy") = 50$	maximum number of components in a stream

interactions

This section was changed for this version.

$int\_pos := 1$	the interaction number (i.e. the row in the BIP window for the stream)
$mc\_getMBPNr("dummy") = 250$	maximum number of binary pairs in a stream (for all streams)
$ci := mc\_getCi(stream, int\_pos) = 1$	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
$mc\_putCj(stream, int\_pos, cj) = 1$	
$model := mc\_getMB(stream, int\_pos) = 50$	returns model number for interaction # int_pos in given stream
$mc\_putMB(stream, int\_pos, model) = 1$	sets the model number for a given stream and interaction number, int_pos

```
id := 0
```
This id is the BIP column (starting with 0 for BIP-1) shown in the PPP window under BIPs for the stream.

```
Kji := mc_getBIP(stream, int_pos, id) = 0.08
```
value of the interaction coefficient

```
mc_putBIP(stream, int_pos, id, Kji) = 1
```
specifies a value for an interaction coefficient

thermodynamic models for streams

This section was changed for this version.

```
stream = 1
```

```
Kcode := 50
```
code for SRK standard model package, see manual for other package numbers

```
mc_setKM(stream, Kcode) = 1
```

mp codes for functions below

1 fugacity
2 enthalpy
3 entropy
4 molar volume
5 viscosity

Examples for fugacity and enthalpy models below:

```
state := 0
```
vapor state

```
mp := 1
```
fugacity model for stream, state

```
fmodel := mc_getMP(stream, mp, state) = 50
```
retrieve model number

```
mc_setMP(stream, mp, fmodel, state) = 1
```
set model number

```
mp := 2
```
enthalpy model for stream, state

```
hmodel := mc_getMP(stream, mp, state) = 50
```
retrieve model number

```
mc_setMP(stream, mp, hmodel, state) = 1
```
set model number

Thermodynamic calculations

stream := 5 — use stream 5 for examples below

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

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

phase equilibria

n := 1 $\quad$ pf := .3 — see below

mc_PfPF(stream, p, pf, state, n) = 0 — 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) = 300.674 — 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 — selected critical point

$mc\_StrPc(stream, cpn) = 5.242 \times 10^6$ — critical pressure for critical point #, cpn

mc_StrCBp(stream) = 0 — cricondenBar pressure

mc_StrCBt(stream) = 0 — cricondenBar temperature

$mc\_StrCTp(stream) = 4.982 \times 10^6$ — cricondenTherm pressure

mc_StrCTt(stream) = 457.544 — 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.

$$\begin{aligned} &PELine(stream, line) := \\ &\quad M \leftarrow mc\_PELine(stream, line) \\ &\quad npts \leftarrow M_{0,2} \\ &\quad T \leftarrow submatrix(M, 0, npts - 1, 0, 0) \\ &\quad P \leftarrow submatrix(M, 0, npts - 1, 1, 1) \\ &\quad (T \quad P \quad npts) \end{aligned}$$

(T1 P1 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	311.885
1	320.94
2	325.94
3	330.94
4	335.94
5	340.94
6	345.94
7	350.94
8	355.94
9	360.94
10	365.94
11	370.94
12	375.94
13	...

P1 =

	0
0	$1.013 \cdot 10^5$
1	$1.415 \cdot 10^5$
2	$1.686 \cdot 10^5$
3	$1.997 \cdot 10^5$
4	$2.351 \cdot 10^5$
5	$2.754 \cdot 10^5$
6	$3.21 \cdot 10^5$
7	$3.723 \cdot 10^5$
8	$4.299 \cdot 10^5$
9	$4.944 \cdot 10^5$
10	$5.663 \cdot 10^5$
11	$6.462 \cdot 10^5$
12	$7.348 \cdot 10^5$
13	...

phase fraction lines

stream := 5

state := 0

fraction := .5

PFLine(stream, state, fraction) :=
M ← mc_PFLine(stream, state, fraction)
npts ← $M_{0,2}$
T ← submatrix(M, 0, npts − 1, 0, 0)
P ← submatrix(M, 0, npts − 1, 1, 1)
(T P npts)

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

nf = 37

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	272.895
1	277.895
2	282.895
3	287.895
4	292.895
5	297.895
6	302.895
7	307.895
8	312.895
9	317.895
10	322.895
11	327.895
12	332.895
13	...

Pf =

	0
0	$1.013 \cdot 10^5$
1	$1.21 \cdot 10^5$
2	$1.436 \cdot 10^5$
3	$1.693 \cdot 10^5$
4	$1.985 \cdot 10^5$
5	$2.313 \cdot 10^5$
6	$2.683 \cdot 10^5$
7	$3.096 \cdot 10^5$
8	$3.556 \cdot 10^5$
9	$4.066 \cdot 10^5$
10	$4.63 \cdot 10^5$
11	$5.251 \cdot 10^5$
12	$5.933 \cdot 10^5$
13	...

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

$$\begin{aligned}
&\text{PhaseEnv}(\text{stream}) := \left|\begin{array}{l}
\text{lnr} \leftarrow \text{mc_PELNr}(\text{stream}) \\
\text{for } \text{line} \in 1 .. \text{lnr} \\
\quad \left|\begin{array}{l}
\left(t^{\langle \text{line}-1 \rangle} \quad p^{\langle \text{line}-1 \rangle} \quad n_{\text{line}-1}\right) \leftarrow \text{PELine}(\text{stream}, \text{line}) \\
\text{ltype}_{\text{line}-1} \leftarrow \text{mc_PELT}(\text{stream}, \text{line}) \\
\text{lprop}_{\text{line}-1} \leftarrow \text{mc_PELP}(\text{stream}, \text{line})
\end{array}\right. \\
(t \quad p \quad \text{lnr} \quad n \quad \text{ltype} \quad \text{lprop})
\end{array}\right.
\end{aligned}$$

$\text{stream} := 5$

$(Tj \quad Pj \quad \text{lnr} \quad \text{nc} \quad \text{type} \quad \text{prop}) := \text{PhaseEnv}(\text{stream})$

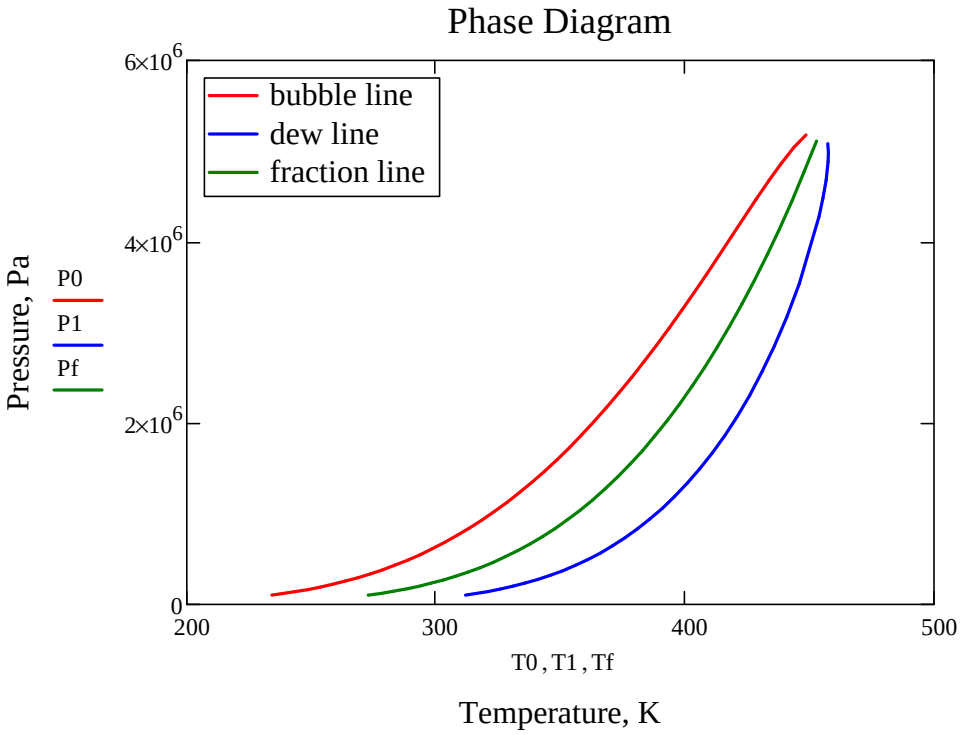
$\text{lnr} = 2$

$$\text{type} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad \text{nc} = \begin{pmatrix} 43 \\ 33 \end{pmatrix}$$

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.

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

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



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.

$phyd := mc\_HPFORM(str\_hyd, thyd, hydmodel) = 1.821 \times 10^{6}$ 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
$p := mc\_getP(stream) = 1.013 \times 10^{5}$	pressure
$h := mc\_StrH(stream) = -697.466$	enthalpy obtained above
$entropy := mc\_StrS(stream) = -2.866$	entropy obtained above
$sv := mc\_StrV(stream) = 1.405 \times 10^{-3}$	volume obtained above

find temperature

$mc\_VPF(stream, p, sv, et) = 150$	volume-pressure flash, et=temp guess
$mc\_HPF(stream, p, h, et) = 0$	enthalpy-pressure flash, et=temp guess
$mc\_SPF(stream, p, entropy, et) = 0$	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.

$stream := 2$

$mc\_EStrGMw(stream, t, p) = 16.047$	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
$phase := 1$	position of phase, not the state code. positions are usually vapor=1, liquid=2, solid=3 but extra liquid and solid phases may be present
$mc\_EStrPf(stream, phase, t, p) = 0.999$	molar phase fraction of phase
$mc\_EStrZv(stream, t, p) = 0.984$	gas (vapor) compressibility factor
$mc\_EStrH(stream, t, p) = -317.84$	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
$mc\_EStrLIC(stream, t, p) = 5.211 \times 10^{-10}$	liquid isothermal compressibility
$mc\_EStrMSS(stream, t, p) = 0$	mixture speed of sound
$mc\_EStrGSS(stream, t, p) = 0$	gas speed of sound
$mc\_EStrLSS(stream, t, p) = 3.325 \times 10^{3}$	liquid speed of sound

$mc\_EStrGJT(stream,t,p) = 1.466\times 10^{-5}$	gas Joule Thomson coefficient
$mc\_EStrLJT(stream,t,p) = -6.958\times 10^{-7}$	liquid Joule Thomson coefficient
$mc\_EStrGVE(stream,t,p) = -6.939\times 10^{-3}$	gas volumetric expansivity coefficient
$mc\_EStrLVE(stream,t,p) = -8.86\times 10^{-4}$	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
$mc\_EStrS(stream,t,p) = -1.466$	total entropy
$mc\_EStrGD(stream,t,p) = 1.325$	gas density
$mc\_EStrLD(stream,t,p) = 729.12$	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

This section was changed for this version.

The operations below behave like subroutines rather than functions because they return more than one result. The Mathcad system imposes some restrictions on function input 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.

$stream := 1$

$NC := mc\_getCNr(stream)$

$t = 150$

These variables were defined above.

$p = 1.013 \times 10^{5}$

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

$i := 0 .. NC - 1$

$phase := 2$

$w_i := mc\_getW(stream, phase, i + 1)$

$$w = \begin{pmatrix} 6.329 \times 10^{-3} \\ 0.272 \\ 0.722 \end{pmatrix}$$

$state := 1$

The liquid state is being used.

New in version 1.2b: The fugacity, H, S, and V functions and their derivatives previously included the stream number as the first argument. Now the first argument is a "process code". The code is set using the mc_DPinit function which also loads the stream information into active memory. The stream (process) stays loaded, making execution of StrFv and similar routines faster. This is especially useful if the routines are repeatedly called by a program loop.

$process := 1$

Up to 5 processes may be defined in the base Prode version. Processes may be redefined with new streams.

$mc\_DPinit(process, stream) = 1$

fugacity vector

$fg := mc\_StrFv(process, state, t, p, w, NC) \cdot Pa$

This routine returns the fugacity vector alone.

$$fg = \begin{pmatrix} 1.488 \times 10^{7} \\ 1.788 \times 10^{4} \\ 1.056 \times 10^{3} \end{pmatrix} Pa$$

$fugacity_i := fg_i \cdot w_i$

The Prode routines define the "fugacity" variable as the fugacity coefficient times the

$$\text{fugacity} = \begin{pmatrix} 9.416 \times 10^4 \\ 4.86 \times 10^3 \\ 762.147 \end{pmatrix} \text{Pa}$$

total pressure. Thus, fugacity is obtained by the equation on the left.

fugacity vector plus derivatives wrt T, P, w

$$\text{StrFvd}(\text{process}, \text{state}, t, p, w, NC) := \left| \begin{array}{l} M \leftarrow \text{mc\_StrFvd}(\text{process}, \text{state}, t, p, w, NC) \\ \text{"separate the results into vectors and a matrix"} \\ fg \leftarrow \text{submatrix}(M, 0, NC - 1, 0, 0) \\ dfgt \leftarrow \text{submatrix}(M, 0, NC - 1, 1, 1) \\ dfgp \leftarrow \text{submatrix}(M, 0, NC - 1, 2, 2) \\ dfgw \leftarrow \text{submatrix}(M, 0, NC - 1, 3, 3 + NC - 1) \\ (fg \quad dfgt \quad dfgp \quad dfgw) \end{array} \right.$$

$(fg \quad dfgt \quad dfgp \quad dfgw) := \text{StrFvd}(\text{process}, \text{state}, t, p, w, NC)$ given the stream, state, temp, press, composition vector, w, and the number of components, NC, return fugacity vector, fg, and derivatives of fg wrt t, p, and w.

Add the default Prode units as needed.

$$fg \cdot \text{Pa} = \begin{pmatrix} 1.488 \times 10^7 \\ 1.788 \times 10^4 \\ 1.056 \times 10^3 \end{pmatrix} \text{Pa} \qquad dfgt \cdot \frac{\text{Pa}}{\text{K}} = \begin{pmatrix} 2.686 \times 10^5 \\ 1.627 \times 10^3 \\ 115.318 \end{pmatrix} \cdot \frac{\text{Pa}}{\text{K}} \qquad dfgp = \begin{pmatrix} 2.686 \times 10^5 \\ 1.627 \times 10^3 \\ 115.318 \end{pmatrix}$$

$$dfgw \cdot \text{Pa} = \begin{pmatrix} -2.139 \times 10^8 & -1.533 \times 10^8 & -1.202 \times 10^8 \\ -6.41 \times 10^4 & -7.652 \times 10^4 & -2.018 \times 10^4 \\ 1.557 \times 10^3 & 1.796 \times 10^3 & 523.217 \end{pmatrix} \text{Pa}$$

Other stream state variables and their derivatives

This section was changed for this version.

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 a string variable with the corresponding variable initial. The program calls the appropriate mc_xxx function and then separates the variables from the output matrix.

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

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

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

$$H \cdot \frac{KJ}{Kmol} = \left(-2.419 \times 10^{4}\right) \cdot \frac{KJ}{Kmol}$$

the default Prode units have been applied to the results

$$dHt \cdot \frac{\frac{KJ}{Kmol}}{K} = (65.204) \cdot \frac{KJ}{Kmol \cdot K}$$

$$dHp \cdot \frac{\frac{KJ}{Kmol}}{Pa} = \left(-8.027 \times 10^{3}\right) \cdot \frac{KJ}{Kmol \cdot Pa}$$

$$dHw \cdot \frac{KJ}{Kmol} = \left(-8.027 \times 10^{3} \quad -2.153 \times 10^{4} \quad -2.533 \times 10^{4}\right) \cdot \frac{KJ}{Kmol}$$

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

$$S \cdot \frac{KJ}{Kmol \cdot K} = (-121.246) \cdot \frac{KJ}{Kmol \cdot K}$$

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

$$dSp \cdot \frac{KJ}{Kmol \cdot K \cdot Pa} = (-70.635) \cdot \frac{KJ}{Kmol \cdot K \cdot Pa}$$

$$dSw \cdot \frac{KJ}{Kmol \cdot K} = (-70.635 \quad -105.28 \quad -104.679) \cdot \frac{KJ}{Kmol \cdot K}$$

$$(V \quad dVt \quad dVp \quad dVw) := StrXvd("V", process, state, t, p, w, NC)$$

$$V \cdot \frac{m^3}{mol} = (0.034) \frac{m^3}{mol}$$

$$dVt \cdot \frac{m^3}{mol \cdot K} = \left(4.845 \times 10^{-5}\right) \frac{m^3}{mol \cdot K}$$

$$dVp \cdot \frac{m^3}{mol \cdot Pa} = (0.073) \cdot \frac{m^3}{mol \cdot Pa}$$

$$dVw \cdot \frac{m^3}{mol} = (0.073 \quad 0.069 \quad 0.068) \frac{m^3}{mol}$$

Operations to set/retrieve the options needed for equation of state models and flash routine phases

See the Prode manual (see paragraph "Codes used in Prode library") and also open the Prode drop menus for the model to view the description of the options set by the OM code variable. The user will probably find it easier to set the options using the Prode window.

All even code values mean that only single liquid phases are allowed in the flash routines. For multiple liquids, the code value must be odd.

$stream := 1$

$option := mc\_getOM(stream) = 545$ — current option set. This should = 552 for stream 1 in the def.ppp archive. With this option value, only L-V flashes will result.

$mc\_setOM(stream, option + 1) = 1$ — This changes the flashes of "stream" to multiple liquids.

$mc\_setOp(stream, t, p) = 1$ — redo the flash

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

$$\text{phases} = \begin{pmatrix} \text{"Vapor"} \\ \text{"Solid"} \\ \text{"Solid"} \\ \text{"Not present"} \\ \text{"Not present"} \end{pmatrix}$$ Previously, only one liquid was present

mc_setOM(stream, option) = 1 reset the code to single liquids

Initializing a stream

This section was changed for this version.

The example will create a stream with water and methanol. The component numbers in the Prode databank can change with updates, so always use CAS numbers when initializing by program instead of manually using the Prode window.

methanol_id := 67561 CAS number of methanol

methanol_code := mc_CompCID(methanol_id) = 24

water_id := 7732185 CAS number of water

water_code := mc_CompCID(water_id) = 21

stream := 11

mc_initS(stream) = 1 initialize a new stream

model := 50 SRK standard see Prode manual for model codes

mc_setKM(stream, model) = 1 set property model package

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) = 0 view the stream then press OK

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) = 329.55 — mixed stream temperature

stream2 := 12 — 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

stream2 := 13 — 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 := 14

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 := 15

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) = 369.276

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::
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) = 0 calculated orifice area, m^2

mc_getErrFlag(" ") = 1

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") = 0` save modifications to a new archive