ProcessModelling

User Manual

Version 1.1.0

ADG Energy Services Ltd
www.adg-esl.co.uk

1. About This Tool

ProcessModelling grew out of work on gas pipeline criticality and thermodynamic stability — the question of whether a given gas composition, at given conditions, sits safely clear of the pressure/temperature region where it would separate into two phases inside a pipeline it was never designed to carry liquid in. That original, narrow focus grew over time into a general-purpose steady-state process simulation tool: equations of state, a full set of common unit operations, multiphase pipe flow, petroleum-fraction (pseudo-component) characterization, and the iterative machinery needed to solve flowsheets containing recycle loops and control targets.

The engineering core of the tool — the thermodynamics, the unit operation models, the convergence logic — was developed and proven first, then rebuilt as a native application for macOS, iPadOS, and iPhone, so it could be carried and used directly in the field rather than only at a desk.

A deliberate design choice

ProcessModelling is deliberately not a drag-and-drop flowsheet editor. A model here is written as a deck — a plain-text description of every stream and piece of equipment, in the language described in Section 3. That is a considered choice, not a limitation waiting to be fixed.

It is easy, in a purely graphical simulator, to stitch blocks together by eye until a flowsheet converges without ever having to state — in so many words — what each stream's conditions actually are, or why a given piece of equipment belongs where it does. Writing FEED=, PRESSURE=, TEMPERATURE= explicitly, in order, forces that thinking to happen. The keyword assistance built into the Model tab (Section 2) is deliberately limited to the same idea: it offers the shape of a valid block — which keywords it takes, which are optional, and what kind of value belongs in each — and never guesses or fills in an actual value on the user's behalf. The tool is meant to help an engineer think through their own process, not to think through it for them.

2. Getting Started

The app is organised into a small number of tabs:

- Model — the deck itself, as plain text.
- Diagram — an automatically generated block diagram of the current deck, for a visual check of the flowsheet's shape.
- Flowsheet / Equipment — results tables, populated after running a calculation.
- Help / About — in-app reference and version information.

To begin, either open an existing .mod file, or use the “Load Example” button shown when the Model tab is empty — this loads a simple, deliberately uncluttered example deck (a feed stream, a heater, and a three-phase separator) as a starting point to explore the deck language from.

With a deck loaded, “Run Calculation” solves the model and populates the Flowsheet and Equipment tabs. A convergence warning banner appears if any part of the model didn't settle cleanly — worth checking before trusting the results.

Inserting a block

Above the Model tab's text editor is an “Insert Block” menu, listing every block type this deck language has — STREAM, VALVE, HEATER, and so on through REPORT. Choosing one appends that block's full skeleton to the end of the deck: its header line, followed by every keyword it takes, each with a placeholder showing what kind of value is expected (e.g. FEED=<stream name>, PRESSURE=<value>) rather than an actual guessed value. This is a reference aid, not an autocomplete in the conventional sense — see “A deliberate design choice” above for why it stops there.

Every keyword line that isn't required for that block to work is marked with a trailing “! optional” comment, and states its default where one exists (e.g. “FLOW=<value> ! optional, defaults to 1.0”). A line without that marker is required — the block won't build without it. The “!” marks are genuine deck comments, not just notes for this manual: leaving one in place, even after filling in a real value on the same line, has no effect on how the block runs.

A few block types have keywords that are genuine alternatives rather than all used together — for example HEATER's TEMPERATURE= versus DUTY=, STREAM's TEMPERATURE=/ENTHALPY=/ENTROPY= (exactly one of the three), or PUMP/COMPRESSOR/EXPANDER's flat-efficiency mode versus a full performance curve. Where this applies, the inserted skeleton includes a plain comment line marking the choice, rather than leaving it to guesswork.

A three-phase feed (containing water) needs PHASES=3 on its STREAM block — forcing PHASES=2 on a wet feed will produce persistent phase-balance errors.

Analyzing a stream: property tables and phase envelopes

Once a model has run successfully, the Flowsheet tab's own column headers (each one a stream name) offer further analysis for that specific stream, beyond what the standard results table shows. On iPad and Mac, click the small chevron next to a stream's name; on iPhone, open that stream's own detail view and use the toolbar button there. Two options are offered:

- Generate Property Table… — sweeps a chosen physical or transport property (density, viscosity, Cp, and everything else the app itself computes for a stream) across a temperature range, and optionally a pressure range too, for that stream's own composition and flow. Shown as a table, with CSV export.
- Generate Phase Envelope… — traces the dew and bubble curves for that stream's own composition, plus its critical point, and displays it as a P-T, P-S, or P-H chart alongside the underlying raw data. Opens as a genuine, resizable, movable window on macOS; as a sheet on iPad and iPhone.

Both are generated on demand, not automatically as part of a normal calculation run — a phase envelope trace in particular can take a genuinely noticeable moment to compute, and running it silently as part of every “Run Calculation” would make the app feel like it had hung. Both results persist for as long as the underlying model stays unchanged, and are cleared automatically the moment the deck text is edited or a different model is opened — re-running an unmodified deck does not lose a previously generated table or envelope.

Quality lines

On a phase envelope's own P-T view, up to three constant vapor-fraction (“quality”) contours can be drawn inside the envelope, alongside the dew and bubble lines. Enter a vapor fraction as a whole-number percentage (e.g. 90) into one, two, or three of the fields below the chart, then “Plot”. Each is traced independently: from just above the bubble line, up through its own turning point in the retrograde region near the top of the envelope, and back down toward the critical point — the same overall shape the dew and bubble lines themselves have, just for a fixed vapor fraction other than 0% or 100% rather than the boundary cases. “Clear Quality Lines” removes all of them at once.

Reliable across roughly 90-99.5% vapor fraction. Very low quality (below roughly 15-20%) is outside this feature's own validated range and may fail to trace or behave unpredictably; if a requested value can't be traced, the chart states which one and why, rather than silently showing nothing for it.

The Data tab's own picker switches between the traced envelope's dew/bubble/critical points and the quality lines' own points, once at least one has been plotted. “Export CSV…” always writes both sections into a single file — labeled “# Phase Envelope” and “# Quality Lines” — regardless of which one the picker currently shows, so a single export can't accidentally omit one half by forgetting to switch the picker first.

3. The Deck Language

A deck is read from top to bottom. TITLE, COMPONENTS, THERMO, and UNITS (in that order) normally come first and apply to the whole model; a PSEUDOCOMPONENT block, if used, must also appear before the first STREAM block. A stream referenced by name (in a FEED=, FEEDS=, or similar keyword) must already have been defined earlier in the deck, either as a STREAM block or as an unnamed default outlet from an earlier piece of equipment. REPORT blocks are placed wherever their output is wanted, usually at the end.

Every block below is introduced by its own line (the block type, then a name — e.g. “HEATER H101”), followed by its own indented keyword lines. A keyword marked (optional) can be left out entirely with no effect beyond falling back to its stated default, if it has one.

TITLE

A single free-text line naming the model. Optional, but a good habit for keeping multiple decks straight.

COMPONENTS

Lists every chemical component the model will use, by name (e.g. N2, CO2, C1, C2, C3, nC4, H2O). Must appear before the first STREAM block, and after any PSEUDOCOMPONENT block whose own name it references. Component names must match entries in the critical properties data file, or a name characterized by an earlier PSEUDOCOMPONENT block in this same deck.

THERMO

Chooses the equation of state and the property data files to use for the whole model.

`EOS=<PR|SRK>`	Which equation of state to use — Peng-Robinson or Soave-Redlich-Kwong. See the Documentary Reference section for the original papers behind each. (optional, defaults to PR)
`CRITCSV=<filename.csv>`	Critical-properties data file to use instead of the bundled default. A bare filename is looked for alongside the deck file itself on macOS, if a file actually exists there — falling back to the bundled default otherwise. On iPadOS, a bare filename can't reliably resolve this way at all (the document picker used to open a model doesn't grant access to files sitting next to it), so it always falls back to the bundled default there; use the app's own Data Files picker to select a specific file on that platform instead. An absolute path is used as given, on either platform. (optional)
`KIJCSV=<filename.csv>`	Binary interaction parameter file, same resolution rules as CRITCSV= above. (optional)

UNITS

Sets which unit system the deck's own numbers are written in, and which unit system results are reported in. These can differ — e.g. type the deck in FIELD units but see results in SI.

`INPUT=<SI|FIELD|METRIC|name>`	Unit system the values you type into this deck are in — a built-in preset, or the name of a UNITSET block defined earlier in this deck. (optional, defaults to SI)
`OUTPUT=<SI|FIELD|METRIC|name>`	Unit system results are reported in, same choices as INPUT= above. (optional, defaults to SI)

UNITSET

Defines a custom, named unit system (an alternative to the built-in SI/FIELD/METRIC presets) for use in a later UNITS block. Give it a name, then set only the fields you want to differ from SI — every field below is independently optional.

`PRESSURE=<unit>`	e.g. psia, bara, kPa (optional)
`TEMPERATURE=<unit>`	e.g. F, C, K (optional)
`MOLARFLOW=<unit>`	e.g. lbmol/h, kmol/h (optional)
`MASSFLOW=<unit>`	e.g. lb/h, kg/h (optional)
`VOLUMEFLOW=<unit>`	e.g. bbl/d, m3/h (optional)
`GASSTDFLOW=<unit>`	e.g. MMSCFD, Sm3/d (optional)
`DENSITY=<unit>`	e.g. lb/ft3, kg/m3 (optional)
`MOLARDENSITY=<unit>`	e.g. lbmol/ft3, kmol/m3 (optional)
`VISCOSITY=<unit>`	e.g. cP, Pa.s (optional)
`THERMALCONDUCTIVITY=<unit>`	e.g. Btu/(h.ft.F), W/(m.K) (optional)
`CPMOLAR=<unit>`	molar heat capacity, e.g. Btu/(lbmol.F), J/(mol.K) (optional)
`CPMASS=<unit>`	mass heat capacity, e.g. Btu/(lb.F), J/(kg.K) (optional)
`SURFACETENSION=<unit>`	e.g. dyn/cm, N/m (optional)
`ENTHALPYMOLAR=<unit>`	e.g. Btu/lbmol, J/mol (optional)
`ENTROPYMOLAR=<unit>`	e.g. Btu/(lbmol.F), J/(mol.K) (optional)
`MOLECULARWEIGHT=<unit>`	e.g. lb/lbmol, g/mol, kg/kmol (optional)

STREAM

Defines a feed stream — a starting point for the model. Give it a name, a composition (each component name followed by its amount), and its conditions.

`PRESSURE=<value>`	Stream pressure.

`PUNIT=<unit>`	Overrides the deck's own pressure unit just for this PRESSURE= value. (optional)

EXACTLY ONE of TEMPERATURE=, ENTHALPY=, or ENTROPY= — a normal T-P flash, or a P-H/P-S flash instead.

`TEMPERATURE=<value>`	Stream temperature.
`TUNIT=<unit>`	Overrides the deck's own temperature unit just for this value. Only applies alongside TEMPERATURE=. (optional)
`ENTHALPY=<value>`	Stream molar enthalpy, for a pressure/enthalpy flash instead of pressure/temperature.
`HUNIT=<unit>`	Overrides the deck's own enthalpy unit just for this value. Only applies alongside ENTHALPY=. (optional)
`ENTROPY=<value>`	Stream molar entropy, for a pressure/entropy flash instead of pressure/temperature.
`SUNIT=<unit>`	Overrides the deck's own entropy unit just for this value. Only applies alongside ENTROPY=. (optional)
`FLOW=<value>`	Stream flow rate, interpreted per BASIS= below. (optional, defaults to 1.0)
`FUNIT=<unit>`	Overrides the deck's own flow unit just for this value. (optional)
`BASIS=<GASSTD|MOLAR|MASS|LIQUID>`	What kind of quantity FLOW= represents — a standard gas volume rate, a molar rate, a mass rate, or a liquid volume rate. (optional, defaults to MOLAR)
`PHASES=<AUTO|2|3>`	Whether to allow gas/liquid (2) or gas/liquid/aqueous (3) phase splitting, or detect automatically. Use 3 explicitly for any feed containing water — see the Getting Started section. (optional, defaults to AUTO)
`<component>=<value>`	Mole fraction for each component named in COMPONENTS, e.g. N2=0.02 CO2=0.03 C1=0.75 …
`WATERSPLIT=<YES|NO>`	Pre-splits water into its own phase before running a cheaper 2-phase flash on the remaining hydrocarbon system, rather than a full 3-phase flash. (optional, defaults to NO)
`TVP=<YES|NO>`	Whether to also compute the stream's true vapor pressure. (optional, defaults to NO — inexpensive to turn on)
`RVP=<YES|NO>`	Whether to also compute the stream's Reid vapor pressure. (optional, defaults to NO — noticeably more expensive than TVP=, since it needs its own separate calculation)

VALVE

A pressure-reducing device with no separate energy input — an isenthalpic pressure drop.

`FEED=<stream name>`	The stream entering the valve.
`PRODUCT=<stream name>`	Name to give the outlet stream. (optional, defaults to <n>_outlet)
`PRESSURE=<value>`	Outlet pressure.
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for this value. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above — applies the same way to this block's outlet. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

HEATER

Adds or removes heat from a stream, to either a target outlet temperature or a specified duty.

`FEED=<stream name>`	The stream entering the heater.
`PRODUCT=<stream name>`	Name to give the outlet stream. (optional, defaults to <n>_outlet)

EITHER a target temperature OR a target duty, not both.

`TEMPERATURE=<value>`	Target outlet temperature.
`TUNIT=<unit>`	Overrides the deck's own temperature unit just for this value. (optional)

`DUTY=<value>`	Target heat duty instead of a target temperature, as a molar quantity (J/mol) — not a total rate; multiply by molar flow to get watts.
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

PUMP / COMPRESSOR / EXPANDER

Raises (PUMP/COMPRESSOR) or lowers (EXPANDER) a stream's pressure with a real energy input or output, at a given efficiency. Two ways to define one: a target pressure and a flat efficiency, or a full performance curve.

`FEED=<stream name>`	The stream entering the machine.
`PRODUCT=<stream name>`	Name to give the outlet stream. (optional, defaults to <n>_outlet)
`EFFTYPE=<ADIABATIC|POLYTROPIC>`	Which efficiency convention EFFICIENCY= (or an EFFCURVE=) is defined against. (optional, defaults to ADIABATIC)
`NPOLYSTEPS=<integer>`	Number of calculation steps used for a polytropic solution. (optional, defaults to 5)

EITHER flat-efficiency mode OR a full performance curve, not both.

`PRESSURE=<value>`	Target outlet pressure (flat-efficiency mode).
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for this value. (optional)
`EFFICIENCY=<0-1>`	Flat efficiency to apply (flat-efficiency mode).
`FLOWCURVE=<v1,v2,...>`	Flow points defining a performance curve (curve mode) — give this together with DPCURVE= and exactly one of EFFCURVE= or POWERCURVE=, instead of PRESSURE=/EFFICIENCY=.
`FLOWCURVEUNIT=<unit>`	Unit for the FLOWCURVE= values. (optional)
`DPCURVE=<v1,v2,...>`	Pressure-rise points matching each FLOWCURVE= point.
`DPCURVEUNIT=<unit>`	Unit for the DPCURVE= values. (optional)

Curve mode also needs EXACTLY ONE of EFFCURVE= or POWERCURVE=.

`EFFCURVE=<v1,v2,...>`	Efficiency at each FLOWCURVE= point (curve mode, efficiency-driven).
`POWERCURVE=<v1,v2,...>`	Power at each FLOWCURVE= point instead of EFFCURVE= (curve mode, power-driven).
`POWERCURVEUNIT=<unit>`	Unit for the POWERCURVE= values. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

MIXER

Combines two or more streams into one, conserving mass and energy.

`FEEDS=<stream1,stream2,...>`	The streams to combine.
`PRODUCT=<stream name>`	Name to give the combined outlet stream. (optional, defaults to <n>_outlet)
`PRESSURE=<value>`	An explicit outlet pressure. Must not exceed the lowest feed pressure. (optional — if omitted entirely, the outlet is set to the LOWEST feed pressure, matching HYSYS's own “Set Outlet to Lowest Inlet”)
`PRESSURE=EQUALIZE`	Requires every feed to already be at the same pressure — this is checked, not assumed or forced. If they genuinely match, that shared value becomes the outlet pressure; if they don't, this throws a clear error rather than silently picking one. Matches HYSYS's own “Equalize All”. (optional)
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for this value. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

SPLITTER

Divides one stream into two or more, each carrying the same composition and conditions but a different share of the flow.

`FEED=<stream name>`	The stream to split.
`PRODUCTS=<name1,name2,...>`	Names to give each outlet stream.
`FRACTIONS=<f1,f2,...>`	Fraction of the feed flow going to each product, in the same order as PRODUCTS= — must sum to 1.
`PDROPS=<dp1,dp2,...>`	Pressure drop for each individual product line, same order as PRODUCTS=. (optional)
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for the PDROPS= values. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

PIPE

Models pressure drop (and, for a wet or multiphase feed, phase behaviour) along a length of pipe, given its size, route, and roughness.

`FEED=<stream name>`	The stream entering the pipe.
`PRODUCT=<stream name>`	Name to give the outlet stream. (optional, defaults to <n>_outlet)
`LENGTH=<value>`	Pipe length.
`LUNIT=<unit>`	Overrides the deck's own length unit just for LENGTH= (and ELEVATION=, if used). (optional)
EITHER a plain diameter OR nominal size + schedule, not both.	
`DIAMETER=<value>`	Internal diameter, if not describing the pipe by nominal size and schedule.
`DUNIT=<unit>`	Overrides the deck's own length unit just for DIAMETER=. (optional)
`NOMINALSIZE=<value>`	Nominal pipe size (inches), used together with SCHEDULE= instead of DIAMETER=.
`SCHEDULE=<schedule>`	Pipe schedule (e.g. 40, 80), used together with NOMINALSIZE=.
EITHER an elevation change OR an angle from horizontal — both are optional; omit both for a level run.	
`ELEVATION=<value>`	Net elevation change over the pipe's length.
`EUNIT=<unit>`	Overrides the deck's own length unit just for ELEVATION=. (optional)
`ANGLE=<degrees>`	Uphill (positive) or downhill (negative) angle from horizontal, instead of ELEVATION=.
`ROUGHNESS=<value>`	Internal pipe roughness. (optional, defaults to 4.6e-5 m — commercial steel)
`RUNIT=<unit>`	Overrides the deck's own length unit just for ROUGHNESS=. (optional)
`FITTINGS=<...>`	Equivalent-length fittings to include in the pressure-drop calculation. (optional)
`NSEGMENTS=<integer>`	Number of segments to break the pipe into for a multi-step calculation, re-checking phase behaviour partway along the run rather than once at the inlet. (optional, defaults to 1 — a single, lumped calculation)
`METHOD=<BASIC|BEGGSBRILL1973|BEGGSBRILL1979>`	Calculation method — a simple single-phase pressure-drop method, or the Beggs & Brill multiphase correlation as originally published (1973) or with the Payne et al. correction (1979, matching HYSYS's own default). The 1979 correction multiplies the fully-computed liquid holdup by 0.924 on uphill runs or 0.685 on downhill runs (horizontal flow is unaffected), correcting for overprediction found in the original correlation. A bare METHOD=BEGGSBRILL is not accepted — the year must be stated explicitly, since different simulators default to different years. (optional, defaults to BASIC)
`PRESSURE=<value>`	If given, the pipe SOLVES FOR the flow rate that produces exactly this pressure drop — the reverse of the normal calculation. If omitted, the pipe instead uses the feed's own given flow rate and calculates whatever pressure

	drop results from it, forward. (optional)
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for this value. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

SCRUBBER

A simple gas/liquid separator — two outlets, no water phase modelled separately from the liquid.

`FEED=<stream name>`	The stream entering the scrubber.
`GASPRODUCT=<stream name>`	Name to give the gas outlet stream. (optional, defaults to <n>_gas)
`LIQUIDPRODUCT=<stream name>`	Name to give the liquid outlet stream. (optional, defaults to <n>_liquid)
`CARRYOVER=<0-1>`	Fraction of liquid carried over into the gas outlet, modelling imperfect separation. (optional, defaults to 0)
`PDROP=<value>`	Overall pressure drop across the vessel, applied to both outlets unless overridden below. (optional, defaults to 0)
`PDROP_GAS=<value>`	Pressure drop for the gas outlet specifically, overriding PDROP= for that outlet only. (optional)
`PDROP_LIQUID=<value>`	Pressure drop for the liquid outlet specifically. (optional)
`PUNIT=<unit>`	Overrides the deck's own pressure unit for every PDROP keyword above. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

SEPARATOR

A full three-phase separator — gas, oil, and water outlets, each with its own configurable carryover into the others.

`FEED=<stream name>`	The stream entering the separator.
`GASPRODUCT=<stream name>`	Name to give the gas outlet stream. (optional, defaults to <n>_gas)
`OILPRODUCT=<stream name>`	Name to give the oil outlet stream. (optional, defaults to <n>_oil)
`WATERPRODUCT=<stream name>`	Name to give the water outlet stream. (optional, defaults to <n>_water)
`CARRYUNDER_OIL_WATER=<0-1>`	Fraction of oil carried under into the water outlet. (optional, defaults to 0)
`CARRYOVER_WATER_OIL=<0-1>`	Fraction of water carried over into the oil outlet. (optional, defaults to 0)
`CARRYOVER_OIL_GAS=<0-1>`	Fraction of oil carried over into the gas outlet. (optional, defaults to 0)
`CARRYOVER_WATER_GAS=<0-1>`	Fraction of water carried over into the gas outlet. (optional, defaults to 0)
`PDROP=<value>`	Overall pressure drop, applied to all three outlets unless overridden below. (optional, defaults to 0)
`PDROP_GAS=<value>`	Pressure drop for the gas outlet specifically. (optional)
`PDROP_OIL=<value>`	Pressure drop for the oil outlet specifically. (optional)
`PDROP_WATER=<value>`	Pressure drop for the water outlet specifically. (optional)
`PUNIT=<unit>`	Overrides the deck's own pressure unit for every PDROP keyword above. (optional)
`TVP=<YES|NO>`	See STREAM's own TVP= above. (optional, defaults to NO)
`RVP=<YES|NO>`	See STREAM's own RVP= above. (optional, defaults to NO)

PSEUDOCOMPONENT

Characterizes a petroleum fraction — a plus fraction, a distillation cut, or any hydrocarbon blend without a single well-defined chemical identity — from bulk assay data, then makes it usable exactly like any named pure component. Must appear before the COMPONENTS block that lists its name.

`NBP=<value>`	Normal boiling point.

`NBPUNIT=<unit>`	Unit for NBP=. (optional, defaults to K)
`SG=<value>`	Specific gravity (60F/60F).
`MW=<value>`	Measured molecular weight, if known. (optional — reported alongside the correlation's own estimate as a cross-check, not used in place of it)
`METHOD=<TWU|KESLERLEE>`	Which correlation to use for critical properties — Twu (1984), or Kesler-Lee (1976). (optional, defaults to TWU)
`KIJ=<component:value,...>`	Overrides the automatically synthesized binary interaction parameter for the named component pair(s). (optional — every other hydrocarbon pairing is synthesized automatically via the Chueh-Prausnitz correlation; non-hydrocarbon pairs default to 0 unless given here)
`EXPORTCSV=<path>`	Appends the characterized component's own row to a CriticalProperties.csv-shaped file at this path, creating it if it doesn't already exist — so the same characterization can be reused directly in a future deck. (optional)
`EXPORTKIJCSV=<path>`	Appends the same component's synthesized binary interaction parameters to a Kij.csv-shaped file at this path. (optional)

COMPONENTSPLIT

Removes a fraction of one component from a stream by simple mass balance — stands in for equipment such as a dehydration unit without modelling its internal physics.

`FEED=<stream name>`	The stream to remove a component from.
`REMOVE=<component:fraction>`	Which component to remove, and what fraction of it — e.g. H2O:0.95 removes 95% of the water present. Any component not listed passes through entirely unchanged.
`PRODUCT=<stream name>`	Name to give the main outlet stream (what's left after removal). (optional, defaults to <n>_product)
`REMOVEDPRODUCT=<stream name>`	Name to give the stream carrying whatever was removed. (optional, defaults to <n>_removed) This is a real stream, so its molar/mass flow can be read directly — but its temperature, pressure, and phase don't correspond to any real physical stream, since this operation doesn't model where the removed material actually goes (a glycol circuit, an amine system, etc.), only that it left the main stream.
`PDROP=<value>`	Pressure drop across the unit. (optional, defaults to 0)
`PUNIT=<unit>`	Overrides the deck's own pressure unit just for PDROP=. (optional)

RECYCLE

Closes a loop in the flowsheet — a downstream stream feeds back into an earlier point in the model, and the model iterates until the guessed and calculated values agree. Needs an initial guess to start from.

`FEED=<stream name>`	The real, calculated stream at the far end of the loop — what the guess is being checked against each pass.
`PRODUCT=<stream name>`	Name to give the current guessed stream, used elsewhere in the deck as the actual feed to whatever comes next in the loop.
`INITIALGUESS=<stream name>`	An existing stream to seed the very first guess from — the closer this is to the eventual answer, the fewer iterations are needed.
`TOLT=<value>`	Convergence tolerance for temperature. (optional, defaults to 0.001)
`TOLP=<value>`	Convergence tolerance for pressure. (optional, defaults to 0.001)
`TOLFLOW=<value>`	Convergence tolerance for flow rate. (optional, defaults to 0.001)
`TOLCOMPOSITION=<value>`	Convergence tolerance for composition. (optional, defaults to 0.001)
`MAXITER=<integer>`	Maximum number of iterations to attempt before giving up. (optional, defaults to 50)
`DAMPING=<0-1>`	Slows how aggressively each new guess moves toward the previous pass's answer. (optional, defaults to 1.0 — full, undamped Wegstein) Lower values (try 0.5 first) make a stubborn loop — particularly one with a cooler and

	scrubber inside it — more likely to settle rather than oscillate; never set this above 1.0.

ADJUST

A control block, not a physical piece of equipment — automatically varies one keyword on another block until a target elsewhere in the model is met. A pure control block has nothing of its own to show in a REPORT.

`ADJUSTEDVAR=<BLOCK.KEYWORD>`	Which block and keyword to vary — e.g. K101.PRESSURE.
`TARGETVAR=<BLOCK.KEYWORD>`	Which block and keyword to watch as the target — e.g. K101.POWER, or a downstream stream's own property.
`TARGETVALUE=<value>`	The value TARGETVAR= should reach. Note that POWER and DUTY targets (TARGETVAR=<equipment>.POWER or .DUTY) are always plain watts, regardless of the deck's own units — but if ADJUSTEDVAR= is itself a HEATER's own DUTY= keyword, that one is molar (J/mol), a genuinely different quantity from the watts TARGETVAR=...DUTY reads, not just a unit conversion of it.
`MIN=<value>`	Lower bound for ADJUSTEDVAR= — must genuinely bracket the answer, since the first two passes probe MIN and MAX directly before any further step is trusted.
`MAX=<value>`	Upper bound for ADJUSTEDVAR=, same bracketing requirement as MIN= above.
`TOLERANCE=<value>`	How close TARGETVAR= needs to get to TARGETVALUE= to count as converged. (optional, defaults to 0.001)
`MAXITER=<integer>`	Maximum number of iterations to attempt before giving up. (optional, defaults to 50)

REPORT

Prints and/or writes to CSV the results for named streams and equipment. A bare name can be either a stream or a piece of equipment. ADJUST blocks are never included — they have nothing of their own to show. Every keyword in a REPORT block is optional; a block with none of them simply does nothing.

`FLOWSHEET=<YES|name1,name2,...>`	YES prints one combined table covering every stream in the deck; a specific list restricts it to just those streams. (optional)
`FLOWSHEET_CSV=<filename.csv>`	Also writes the flowsheet table to this file, alongside the deck file itself. (optional — only used if FLOWSHEET= is also given; the app's own “Export CSV…” button is a more reliable way to save this on either platform)
`EQUIPMENT=<YES|name1,name2,...>`	Same as FLOWSHEET= above, but for equipment rather than streams. (optional)
`EQUIPMENT_CSV=<filename.csv>`	Also writes the equipment table to this file, alongside the deck file itself. (optional — only used if EQUIPMENT= is also given; same caveat as FLOWSHEET_CSV= above)

4. Documentary Reference

ProcessModelling's core methods are drawn from established, published chemical and petroleum engineering literature rather than novel or proprietary techniques. The primary sources for each are given below for anyone wanting to check the underlying theory directly.

Peng, D.-Y.; Robinson, D.B.
“A New Two-Constant Equation of State.” Industrial & Engineering Chemistry Fundamentals, 15(1), 59–64, 1976.
Used for: The Peng-Robinson equation of state (EOS=PR).

Soave, G.
“Equilibrium Constants from a Modified Redlich-Kwong Equation of State.” Chemical Engineering Science, 27(6), 1197–1203, 1972.

Used for: The Soave-Redlich-Kwong equation of state (EOS=SRK).

Beggs, H.D.; Brill, J.P.
"A Study of Two-Phase Flow in Inclined Pipes." Journal of Petroleum Technology, 25(5), 607–617, 1973.
Used for: The multiphase pipe flow correlation (PIPE METHOD=BEGGSBRILL1973).

Payne, G.A.; Palmer, C.M.; Brill, J.P.; Beggs, H.D.
"Evaluation of Inclined-Pipe, Two-Phase Liquid Holdup and Pressure-Loss Correlations Using Experimental Data." Journal of Petroleum Technology, 31(9), 1198–1208, 1979.
Used for: The corrected multiphase pipe flow correlation (PIPE METHOD=BEGGSBRILL1979), matching HYSYS's own default treatment of Beggs & Brill.

Wegstein, J.H.
"Accelerating Convergence of Iterative Processes." Communications of the ACM, 1(6), 9–13, 1958.
Used for: The convergence-acceleration approach behind RECYCLE.

Twu, C.H.
"An Internally Consistent Correlation for Predicting the Critical Properties and Molecular Weights of Petroleum and Coal-Tar Liquids." Fluid Phase Equilibria, 16(2), 137–150, 1984.
Used for: PSEUDOCOMPONENT's default critical-property correlation (METHOD=TWU).

Kesler, M.G.; Lee, B.I.
"Improve Prediction of Enthalpy of Fractions." Hydrocarbon Processing, 55(3), 153–158, 1976.
Used for: PSEUDOCOMPONENT's alternative critical-property and ideal-gas heat-capacity correlations (METHOD=KESLERLEE), and the acentric-factor correlation used by both methods.

Chueh, P.L.; Prausnitz, J.M.
"Vapor-Liquid Equilibria at High Pressures: Vapor-Phase Fugacity Coefficients in Nonpolar and Quantum-Gas Mixtures." Industrial & Engineering Chemistry Fundamentals, 6(4), 492–498, 1967.
Used for: The automatic binary-interaction-parameter synthesis for a characterized pseudo-component against the deck's other hydrocarbons (PSEUDOCOMPONENT's own KIJ=).

Whitson, C.H.; Brulé, M.R.
"Phase Behavior." SPE Monograph Volume 20, Society of Petroleum Engineers, 2000.
Used for: The specific equation forms and coefficients used for Twu (1984) and Kesler-Lee (1976) above, as compiled and cross-checked in this reference.

Kooijman, H.A.; Taylor, R.
The ChemSep Book (component property database and format).
Used for: The critical-properties file format (CriticalProperties.csv).