HoBoil Limited

Reservoir Engineering
Consultancy & Training

HoBoil Limited
Banchory, UK
dev.pvtfree@gmail.com

PVTfree

The Free Equation of State PVT Program

Steve Furnival
dev.pvtfree@gmail.com

June 2019

This documentation describes the *PVTfree* Equation of State PVT Program.

Acknowledgements

ECLIPSE is a service mark of Schlumberger.

TEMPEST is a service mark of Emerson Roxar.

VIP and NEXUS are registered trademarks of Landmark Graphics Corporation.

IMEX and GEM are programs from CMG.

Contents

Figures

Tables

1 Introduction

Welcome to *PVTfree*, the free Equation of State (EoS) PVT program. Why '*PVTfree*'? Well it's a PVT program, and its free[1]. Simple really.

It was 'originally' written by Steve Furnival of HoBoil Limited, Banchory, UK, as a means of teaching himself Python programming. Hopefully by the time you get here, others will have contributed to the program. Maybe you will contribute something to its development soon.

Steve is an Independent Reservoir Engineering (RE) consultant[2] and HoBoil (the Hill of Banchory, some 25 km west of Aberdeen in Scotland is where he has lived with his family since 2006) is his limited company, employing just himself[3]. During one of the many lulls in business which has afflicted the oil & gas industry since the collapse in prices in 2015, rather than just sit around moaning, he decided to do something useful and write software he could use when things picked up. The coding got out of hand, and *PVTfree* is the result.

Now I'm sure you know the old adage that there's nothing in life for free! And this software does not disappoint in that respect.

- It is NOT commercial grade software. There will be bugs! If you find one, or more, insert some print statements, find out the cause, and fix it – you have the code.
- But, I here you cry, I can't program. Or, I don't understand the maths. Or, I don't understand EoS and PVT modelling. In which case, this software is not for you! That said, if you consider yourself a professional engineer, then I'd expect you learn. That is, of course, what engineers do.
- What I'm really saying here of course is I don't routinely offer support for *PVTfree*. You are on your own. Now if you want to pay me for support, either on a consultancy basis or perhaps as a training opportunity, drop me a line and let's talk.
- There are NO warranties given for its use. If you subsequently rely on predictions from the software which turn out to be wrong – that's your problem! If you are using this software, I presume you are a fellow RE (or Petroleum Engineer) and you have sufficient knowledge to recognise bad output (probably resulting from bad input) when you see it.
- There is no fancy Graphical User Interface (GUI) for the software. Python has, I understand, all the widgets you need to build such an interface so if you have the time, inclination and skill set, drop me a line and I can give you some pointers towards what I think might make a nice front-end.
- It only supports two EoS, Peng-Robinson (PR) and Soave-Redlich-Kwong (SRK), because in my experience, they are the only ones that matter, i.e. they are the only EoS used in the commercial reservoir simulators and well modelling software.
- It only considers two-phase hydrocarbon flash problems. If you want to study hydrates, waxes, asphaltenes or multiple-liquid systems like low temperature miscible flooding with CO_2, then this isn't your software, unless of course, you are prepared to code it yourself. In which case, don't be shy, send your developments back to me so I can share it with the community of *PVTfree* users.

Still with me? Still want to proceed despite all the caveats described above? Ok, turn the page and I'll try to help you get started.

[1] Within the rules of the GNU General Public License (GPL).
[2] As of June 2019, Steve has gone back to being a "staffer", so support & development of *PVTfree* may well be limited.
[3] See note above. I don't even employ myself anymore so please don't ask for a job!

2 Theoretical Background

PVTfree relies algorithmically heavily on three main references, namely:

- Whitson and Brule, Reference **1**, here given the shorthand W&B
- Soreide, Reference **2**, shorthand SOR, and
- Petitfrere, Reference **3**, shorthand PET

For instance, properties of the non-hydrocarbons N2, CO2 & H2S along with the pure hydrocarbons C1, C2, C3, iC4, nC4, iC5 and nC6 are listed in Table A-1A of W&B. Properties of the SCN groups are listed in Table 5.2 of W&B. The plus fraction 'Whitson Splitting' algorithm is discussed in Section 5.3.2 of W&B and in more detail of Section 3.2.6 of SOR.

There are three main algorithms used in *PVTfree,* being:

- The Stability Test (Stab)
- The Two-Phase Flash Calculation (Flash), and
- The Saturation Pressure Calculation (Psat)

These are discussed in detail in W&B and SOR. These algorithms use Successive Substitution (SS) accelerated by the General Dominant Eigenvalue Method (GDEM). Occasionally, these algorithms must be given a boost by the application of the Broyden Fletcher Goldfarb Shanno (BFGS) method which is discussed extensively in PET.

The Flash calculation requires that the vapour fraction be solved at each iteration in which the K-values are being updated. This is referred to as the Rachford Rice (RR) equation and is described in W&B and SOR. Despite it being a 1-dimensional monotonic equation, it can be extremely difficult to solve. We have adopted the scheme developed by Nichita and Leibovici, Reference **4**.

Regression is performed using the Rotational Discrimination Method (RD) described in detail in Chapter 4 of SOR.

The algorithmic aspects of grouping of components follow the procedure outlined in Section 5.7.2 of W&B, in particular the so-called 'Coats' method. The grouped system can then be exported for use the compositional simulators , Eclipse 300 of Schlumberger, VIP or Nexus of Landmark, GEM of CMG and Tempest-MORE of Emerson Roxar.

Black-oil tables suitable for use in the Schlumberger Eclipse 100, Landmark VIP or Nexus, CMG IMEX and Emerson Roxar Tempest-MORE reservoir simulators[4] are modified for pressures greater than the saturation pressure using the scheme developed by Singh et al. described in Reference **5**.

[4] All trademarks and copyrights acknowledged, see Page 2.

3 Getting Started

If you want to run *PVTfree*, you will need to have an up to date installation of Python on your machine of choice; all my development has been done on a 64-bit PC running Windows 10. Does it work on Linux or one of those fancy Mac-things? I don't know. I believe it should do. I'm sure someone will tell me if it doesn't!

3.1 Python

I'm an old Fortran developer who has worked on some of the best-known commercial reservoir simulation software. Various people have tried over the years to rid me of this dinosaur language and I have acquired some knowledge of C, C++, Java and Visual Basic.

I have been working with Visual Basic (VB), or strictly the cut-down version of VB that sits round the back of *Excel* called Visual Basic for Applications (VBA), for some years since becoming a consultant[5] RE as *Excel* is the go-to software of choice. Using Excel and VBA you can do pretty much anything you want and my first attempt at a *PVTfree* was written this way. However, I was never happy with what I wrote despite the obvious advantage that most laboratory PVT reports now come as *Excel* files. However, if the data is in Excel, then turning that into an ASCII[6] file is trivial given a decent file editor[7].

So why Python? I'd heard great things about it! It has a library of utilities to die for. There are masses of blogs available to find out how you do stuff. And my eldest daughter, who studied Geology and has no programming experience, was told she might need to learn how to write simple applications in Python; so, Dad thought he'd teach himself, so he can help his daughter in due course – aren't I nice?

3.1.1 What Version

I used Python 3.6.5 when I started, so I'd download that, or a later version; at the time of writing I see version 3.7.3 is the option on the "button". The official home of this release is:

https://www.python.org/downloads/

Scroll to the bottom of the page and you'll several alternative options to download. Being inherently lazy, I took the Windows x86-64 executable installer, but you might be OK working with a Gzipped source tarball, whatever that is.

3.1.2 Libraries

As well as basic Python commands, I have used utilities from three add-ons:

- NumPy
- SciPy
- Matplotlib

Even if unfamiliar with Python, you should be able to guess that NumPy is - Numerical Python, a suite of mathematical routines. For someone who has sweated blood trying to write various numerical algorithms

[5] Thinking of becoming an independent consultant? Then get proficient at writing your own 'tools' as you'll no longer have 'free' access to those (very) expensive commercial tools you take for granted in your staff job.
[6] Don't know what ASCII means? Google it!
[7] For what it's worth I have a (paid for!) copy of TextPad, although there are other excellent file editors I have used and would recommend which are great for preparing the input file for *PVTfree* such as Crisp and Notepad++. The key consideration is the ability to perform column editing. Notepad++ will probably be attractive to potential *PVTfree* users as it is itself free!

over the years, NumPy is amazing! Roots of the cubic EoS – one-line call. Diagonalizing a symmetric matrix – one-line call. And there's more as you'll see when you start browsing the code.

SciPy? Yep, Scientific Python. Matplotlib? Yes, it produces plots. See, it's often obvious.

I've provided a (Windows) script called LibsInstall.bat to setup the required libraries; I got this script from the following website, https://www.scipy.org/install.html. Whilst I was googling for this link, I noticed the following YouTube movie which may also be of help https://www.youtube.com/watch?v=Ju6zw83PoKo; its 15 minutes long but very clear.

If you are still confused about installing Python and activating these libraries, one could be uncharitable and suggest that *PVTfree* might not be for you, or I could suggest you look at one of the "total" packages which not only gives you a Python installation, but all the libraries under the sun. I myself have downloaded Anaconda3 from here:

https://www.anaconda.com/distribution/

This is a big package. My installation comprises 114,860 files spread over 13,511 folders and occupies just over 5 GB of disk space! But it has got everything a Python developer (and R[8] developer) could ever want.

3.1.3 Installing the *PVTfree* Software

Installing is a bit fancy to be honest. *PVTfree* consists of about 20 modules (at the time of first writing, now over 30 modules after some tweaking) containing the Python code where each module has the file extension 'py' along with a couple of example datasets (file extension 'dat', see Section 3.2 below) which initially I tried to 'ship' in a single compressed file. This rarely worked as most mail servers wouldn't (quite rightly!) allow file attachments with '.py' and '.bat' extensions – too much risk. I also ran into the problem of version management which is when a 'yuff' told me about GitHub.

Create a suitable directory to store the compressed file and uncompress it. This directory will now become the directory in which you will work subsequently. For what it's worth, I work in:

D:\steve\pvtdata\

As I can't be bothered doing much typing when I'm in the 'DOS' box, see Section 3.1.5

To ensure everything is installed OK, it is usually best to perform a restart to ensure all the appropriate environment variables, etc. are set.

3.1.4 Python – An Interpreted Language

Python is an interpreted language; that is there is no need to compile and link your routines to produce an executable program. The downside is it can be slow compared with software written in a compiled language like Fortran or C/C++ where the compiler writer can do clever things to make executable much faster. Using routines from NumPy can and does help as these are often written in a 'fast' language that is hidden away.

3.1.5 Windows Users - The Command Prompt

The Command Prompt (or DOS for those old enough to remember, and how Bill Gates started on his path to becoming a Multi-Billionaire) is the command line operating system that sits below Windows. You can usual get to the Command Prompt by typing DOS or CMD in the Windows search box, bottom left if using Windows 10. The window shown in Figure 1 will then appear. And yes, you type commands by hand rather than clicking

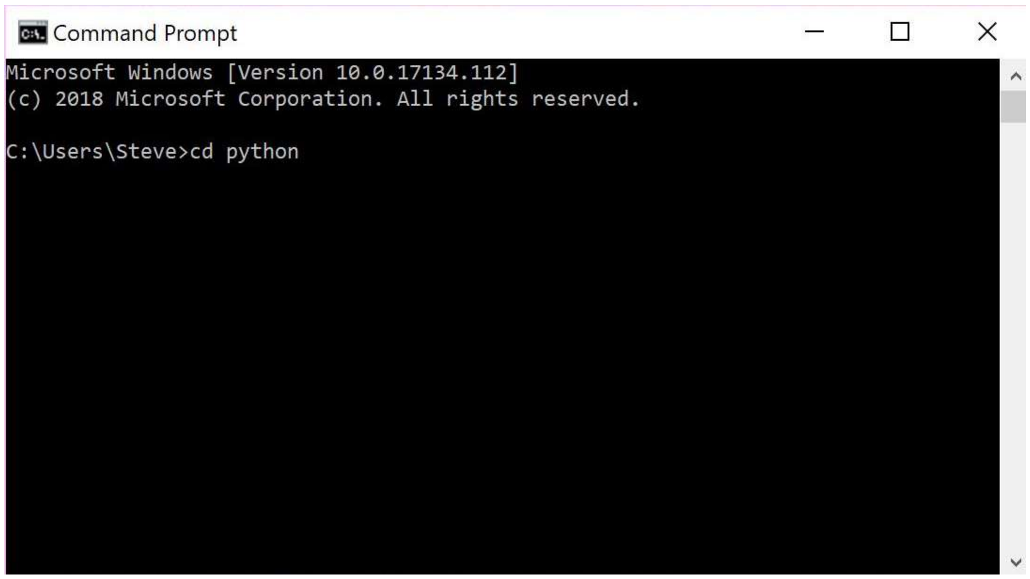
[8] R is a rival to Python, much used in the rapidly expanding "digital engineering" workspace. Too hard for me!

or double-clicking. For example, if you want to drop down into a sub-directory called 'python' where you've installed the *PVTfree* modules, you change directory or 'cd' to the required directory[9], i.e. python, and then hit carriage return or ↵. Aye, life were tough at the dawn of personal computing.

What's the alternative to the DOS box for Linux and Mac users? I don't know; well I think I know with Linux, but I don't want to embarrass myself in case I'm wrong. Hopefully, one of you clever chaps or chapesses will probably work it out and tell me.

3.2 Input File

As we said previously, *PVTfree* does not currently have a fancy GUI. Rather, the user must prepare an input file, much like the preparation one must do if writing an input file for one of the commercial reservoir simulators. In fact, we assume the input file follows the style used by the ECLIPSE reservoir simulator, now owned and developed by Schlumberger.



```
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Steve>cd python
```

Figure 1: The Command Prompt, aka the DOS box

So, the input file will consist of a number of sections interspersed with blank lines and comment lines to improve readability. Comment lines begin in the first two columns of the line with '—', i.e. dash-dash.

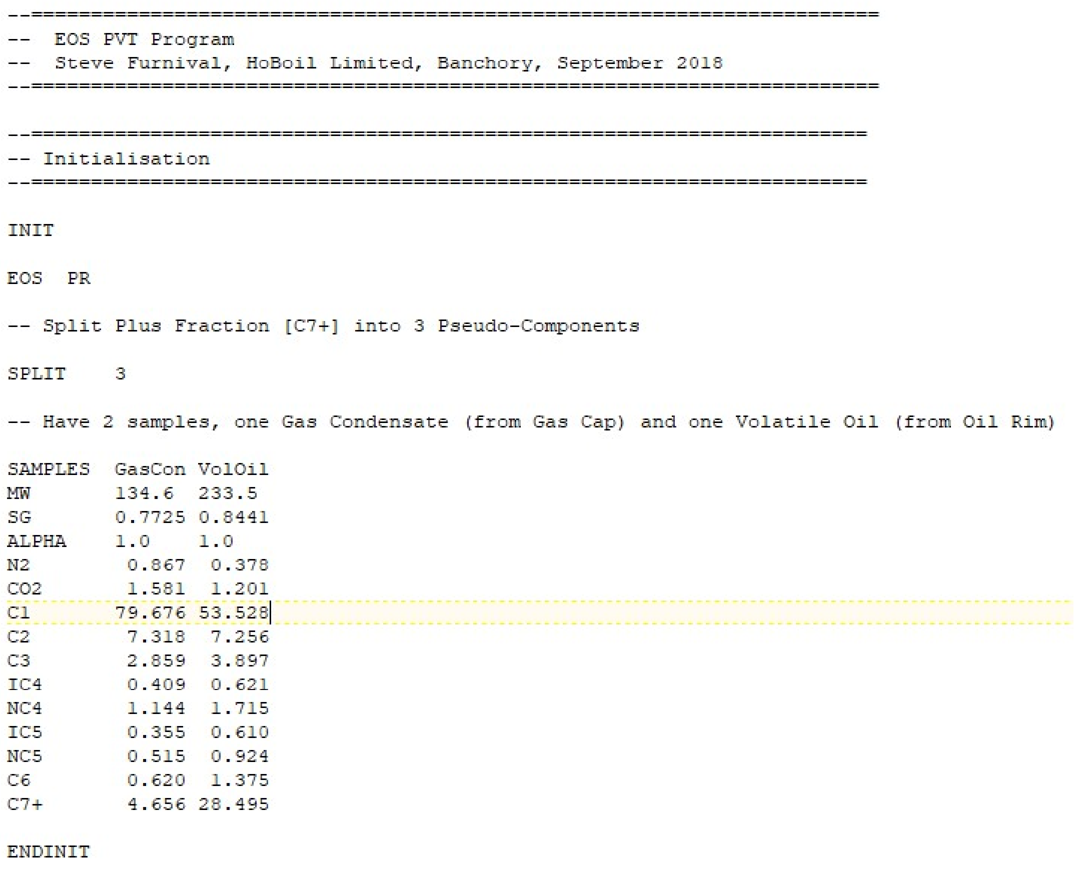
As someone who is often asked to 'debug' reservoir simulation models created by others, I cannot stress how important comments are, not only for other users, but even for yourself. It's not uncommon to work on a project, get called away to work on something else for days, weeks or even months and then have to come back to the original project only to ask oneself, "what was I doing here?". Maybe you have a better memory? I'd still suggest comments are essential, if only to help others. Remember, others may want to audit your work.

An example of (the start of) one of my input files is shown in Figure 2Figure 2: Example of Input File Showing Keywords, Comments and Data. Note the comments preceding the INIT keyword to say who constructed this example (me!), when and where. Subsequent there's a (too brief) comment to describe where this data came from.

[9] Here's one for free, to change directory up one level, type 'cd ..' (without the quotes, of course).

3.2.1 Sections

PVTfree consists of a number of sections, each of which is book-ended in the input file by a Keyword End-Keyword pair. In Figure 2 you can see the INIT … ENDINIT pair of keywords to say, obviously, we are starting to Initialise the fluid description and then that we have finished that initialisation. The current list of sections and their purpose is shown in Table 1.



```
--=======================================================================
--  EOS PVT Program
--  Steve Furnival, HoBoil Limited, Banchory, September 2018
--=======================================================================

--======================================================================
-- Initialisation
--======================================================================

INIT

EOS  PR

-- Split Plus Fraction [C7+] into 3 Pseudo-Components

SPLIT    3

-- Have 2 samples, one Gas Condensate (from Gas Cap) and one Volatile Oil (from Oil Rim)

SAMPLES  GasCon VolOil
MW       134.6  233.5
SG       0.7725 0.8441
ALPHA    1.0    1.0
N2        0.867  0.378
CO2       1.581  1.201
C1       79.676 53.528
C2        7.318  7.256
C3        2.859  3.897
IC4       0.409  0.621
NC4       1.144  1.715
IC5       0.355  0.610
NC5       0.515  0.924
C6        0.620  1.375
C7+       4.656 28.495

ENDINIT
```

Figure 2: Example of Input File Showing Keywords, Comments and Data

Start	End	Function
INIT	ENDINIT	Initialise the Fluid Description
DEFINE	ENDDEFINE	Define the Fluid Description from Previous Model
EXP	ENDEXP	Define and perform PVT Experiments
REG	ENDREG	Regress the EoS to PVT Experiments Data
BLACKOIL	ENDBLACKOIL	Create a Blackoil Description for a Simulator
GROUP	ENDGROUP	Group or Pseudoise the Number of Components
COMP	ENDCOMP	Create a Compositional Description for a Simulator

Table 1: Section Start/End Keyword and its Function

3.2.1.1 Formatting of the Input File

In the sections below, the following formatting rules apply. The keyword and required sub-keywords are shown in bold. Optional sub-keywords are shown in italic. User supplied data is shown between the angle brackets <…>; the type of the data (integer, float or string) should be clear from the specific description.

Note text strings can be specified in upper, lower or mixed case, i.e. INIT, init or Init.

3.2.1.2 INIT: Initialise the Fluid Description

This section, or the equivalent DEF section described in 3.2.1.6, should be the first section in the input file.

This section expects three keywords and their associated arguments and data, these being shown in Table 2.

Keyword	Purpose
EOS	Define the EOS
SPLIT	Split the Plus Fraction into N Pseudo-Components
SAMPLES	Define the Fluid Sample(s)

Table 2: INIT Section Keywords

3.2.1.3 EOS

This keyword takes a single argument (on the same line following the EOS keyword) which should be the string PR (Peng-Robinson) or SRK (Soave-Redlich-Kwong) being the only EoS supported by *PVTfree*; see Appendix A for a discussion of the allowed EoS and how they are implemented (principally in calcEOS.py).

EOS <PR|SRK>

3.2.1.4 SPLIT

This keyword takes a single argument (on the same line following the SPLIT keyword) which should be an integer between Nsplt ≥ 2; this is the number of pseudo components into which the plus fraction defined in the subsequent SAMPLES keyword will be split into.

SPLIT <Nsplt>

The best description of the (Gauss-Laguerre) Quadrature scheme used to define the pseudo-components can be found in Section 3.26.6 of SOR (see Reference **2**).

3.2.1.5 SAMPLES

This keyword is used to define the set of samples to be initialised in this run of *PVTfree*. The SAMPLES keyword should be followed by N strings where each string is the name given to each sample – the sample name should not contain any blanks. In Figure 2 two samples have been defined with the names GasCon and VolOil. These names will be used subsequently by the experiments, etc. so care should be given to their choice.

The keyword and its associated data take the form shown in Table 3.

SAMPLES	<samNam1>	<samNam2>	…	<samNamN>
MW	<Mw1>	<Mw2>	…	<MwN>
SG	<SG1>	<SG2>	…	<SGN>
ALPHA	*<Alpha1>*	*<Alpha2>*	…	*<AlphaN>*
<comNam1>	<Z11>	<Z12>	…	<ZN1>
<comNam2>	<Z21>	<Z22>	…	<ZN2>
:	:	:	…	:
<comNamM>	<ZM1>	<ZM2>	…	<ZMN>

Table 3: Format of the SAMPLES Keyword

The sub-keywords MW and SG which define the mole weight and specific gravity of the N samples are compulsory, the sub-keyword ALPHA which defines the shape-factor used in the modified Whitson splitting algorithm is optional as this parameter defaults to unity for all samples.

Table 3 assumes M components are defined. Note all samples must have the same plus fraction Carbon number, i.e. C7+, C12+, etc. where the MW and SG values apply to the Mth component. Note if the components are not ordered in terms of decreasing volatility, they will be re-ordered by *PVTfree* as this is a requirement of the Rachford-Rice solver, see Reference **4**.

The component mnemonics entered in column-1 should come from the set of allowed non-hydrocarbons, pure hydrocarbons, Single Carbon Number (SCN) groups and a single plus fraction (which can be optionally split using the preceding SPLIT keyword).

It is assumed that samples should contain some of the non-hydrocarbons:

- N2, CO2 and/or H2S

Note in Figure 2 we have defined N2 and CO2 but not H2S. The pure hydrocarbons:

- C1, C2, C3, iC4, nC4, iC5 and/or nC5

Most modern (post-1990) PVT reports may specify moles (often zero) of Hydrogen and Helium; if present we suggest their moles be added to those of Nitrogen. Similarly, if neo-Pentane is present (often at the 0.01 mol% concentration, so why bother?), add its moles to those of i-Pentane (iC5).

Next, we'd expect to see members of the Single Carbon Number (SCN) groups:

- C6, C7, …, C45

In Figure 2 we have only defined C6 from the SCN's as the last component should be the plus fraction, here taken to be C7+. Again, modern reports will often 'break-up' the C7, C8, C9 and even C10 SCN's into their respective PNA's (Paraffins, Napthenes and Aromatics). If your report(s) do this, rec-combine the moles and enter these components as SCN's.

Properties of the non-hydrocarbons and pure hydrocarbons are taken from an internal library. Properties of the SCN components are generated using correlations described by W&B and SOR.

If two or more samples are specified, the SPLIT keyword must have defined before SAMPLES where Nsplt ≥ 2.

If only one sample is defined, and no SPLIT keyword is entered, the properties of the plus fraction are calculated using the same scheme employed for the SCN's.

If the plus fraction is split, the quadrature-based Whitson splitting algorithm is used as described in Section 5.3.2 of W&B and in more detail of Section 3.2.6 of SOR.

The molar distributions of the samples <Zji>, j = 1, …, M, i = 1, …, N, should sum to unity (one) if entered as mole fractions, or one hundred if entered as mole percentages – a small leeway will be given but shouldn't be necessary if you check your work in Excel prior to copying the data to the input file.

$$(1) \qquad \sum_{j=1}^{M} Z_{ji} = 1 \; or \; 100 \quad for \quad i = 1, \ldots, N$$

3.2.1.6 DEF: Define the Fluid Description

This section, or the equivalent INIT section described in 0, should be the first section in the input file. The keywords and data in this section are built automatically by *PVTfree* and written to the Save file, see Section 3.3.3. As such, it is expected that this data will be "cut and paste" from a previous run of the software.

3.2.1.7 EXP: Experiments

The currently allowed set of experiments in *PVTfree* is shown in Table 4.

The general form of the data entry for the experiments is shown in Table 5; users are recommended to study the GASANDOIL.dat input file supplied with *PVTfree* as this includes examples of all eight experiments current supported.

Mnemonic	Long Description
CCE	Constant Composition Expansion
CVD	Constant Volume Depletion
DLE	Differential Liberation Experiment
SEP	Separator Test
PSAT	Saturation Pressure Calculation
FLASH	Two-Phase Flash Calculations
SWELL	Swelling Test
GRAD	Compositional Grading (Composition versus Depth)

Table 4: Currently Supported Experiments in *PVTfree*

<**Type**>					
SAMPLE	<samNam>		<other options>		
WEIGHT	*<obsNam1>*	*<WT1>*	*<obsNam2>*	*<WT2>*	…
PLOT		*<Yes>*	*<Yes>*	…	
	Mnem1	*<Mnem2*	*<Mnem3*	…	
	Unit1	*<Unit2>*	*<Unit3>*	…	
	Val11	*<Val12>*	*<Val13>*	…	
<PSAT|DREF>	Val21	*<Val22>*	*<Val23>*	…	
	:	*:*	*:*	…	
	ValM1	*<ValM2>*	*<ValM3>*	…	

Table 5: General Form of EXP Data Entry

Type must be one of CCE, CVD, DLE, SEP, PSAT, FLASH, SWELL or GRAD.

The SAMPLE sub-keyword should be on the next line after the type and should be followed by the name of one of the sample names defined in the INIT or DEFINE section. Depending on experiment type, other data should follow.

For the CCE, CVD, DLE, SWELL and GRAD experiments a triplet should be provided to define the reservoir temperature. The triplet takes the form:

```
Tres    <temp>        <tUnit>
```

Where Tres is a string of that name, temp is a float and tUnit is one of the allowed temperature units shown in Table 7.

When a CCE is performed on a gas condensate sample, some laboratories will measure and report a liquid dropout (mnemonic SLIQ) which can be defined as liquid volume at current pressure per total volume at current pressure – TOT option, or liquid volume at current pressure per dew point volume – DEW option. If SLIQ is being supplied for the CCE, the option:

```
SLIQ <TOT|DEW>
```

Should be used to select the correct definition.

The swelling test consists of the injection of a second fluid, usually a gas, into the primary fluid, usually an oil already defined with the SAMPLE sub-keyword. The injectant should be defined with:

```
SINJ    <gasSamNam>
```

Where gasSamNam is the name of the required gas sample already defined in the INIT or DEFINE sections.

The WEIGHT sub-keyword is optional and allows the user to set non-default weight values (not equal to unity) for measured or *observed* quantities to use in regression. The possible list of 'Obs' mnemonics is shown by experiment type in Table 6. Each 'Obs' mnemonic should be followed by a float, being the required weighting factor.

If the optional PLOT sub-keyword is supplied, a YES or NO string should be specified for each column of observed data; by default, all items of observed data will be plotted against the values calculated by the EoS, i.e. all PLOT values are assumed to be YES.

The next two lines should be the mnemonics and corresponding units for the set of Independent and Observed data for each experiment; the currently allowed set is shown in Table 6.

The independent data is that needed to define the stages of each experiment. For the CCE, CVD and DLE this is a column of pressure data. For the SEP and FLASH experiments this is two columns of pressure and temperature data – the order is not important. For the PSAT calculation, a column of temperatures is required. The SWELL test requires a column of moles (of injection gas) added whilst the GRAD needs a column of heights. These moles should be incremental, i.e. 0.0, 0.2, 0.4, 0.6, etc.

The list of permissible units for input and output are shown in Table 7. As the *PVTfree* default units are British or Imperial, FIELD units in the Eclipse simulator, the units in the first column are the internal units.

The CCE, CVD, DLE and SEP experiments require the saturation pressure stage to be identified by putting the mnemonic PSAT as the first entry on the appropriate row of data entry; for the CVD, DLE and SEP experiments this is likely to be the first stage. The GRAD experiment requires the reference depth be marked by placing the mnemonic DREF as the first item on the appropriate row of HEIGHT information.

Type	CCE	CVD	DLE	SEP	PSAT	FLASH	SWELL	GRAD
BG			Obs					
BO			Obs	Obs				
BT			Obs					
DENO	Obs		Obs	Obs	Obs	Obs		
DENS								Obs
GGRV			Obs	Obs				
GOR			Obs	Obs				
HEIG								Ind
MOLE							Ind	
MREM		Obs						
PRES	Ind	Ind	Ind	Ind		Ind		
PSAT	Obs	Obs	Obs	Obs	Obs		Obs	
SLIQ	Obs	Obs						
TEMP				Ind	Ind	Ind		
VISG	Obs							
VISO	Obs		Obs					
VSWL							Obs	
ZC1								Obs
ZC7+								Obs
ZFAC	Obs	Obs	Obs		Obs			

Table 6: Experiment Mnemonics/Types

Note, the first row of data for the SEP experiment should be the reservoir temperature and (measured) saturation pressure as this allows the total GOR to be calculated.

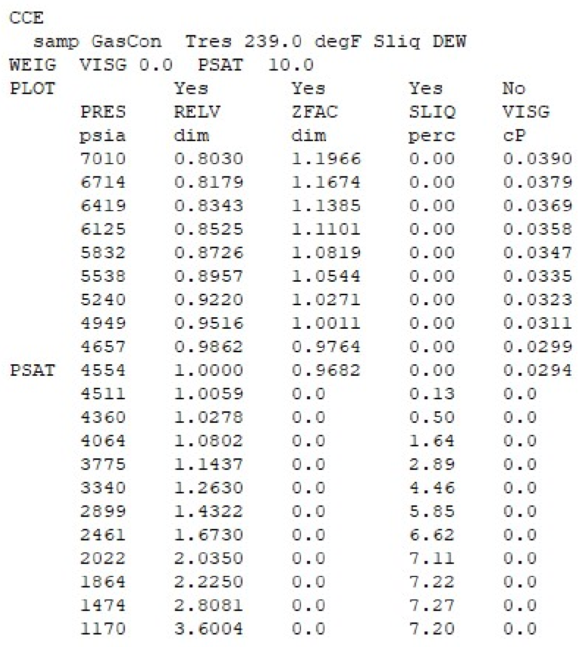
CGR	scf/scf	stb/scf	stb/Mscf	sm3/sm3	sm3/ksm3
Compressibility	1/psi	1/bar			
Density	lb/ft3	kg/m3	gm/cm3	degAPI	
Dimensionless	dim				
Gas FVF	rcf/scf	rb/scf	rb/Mscf	rm3/sm3	rm3/ksm3
GOR	scf/scf	scf/stb	Mscf/stb	sm3/sm3	ksm3/sm3
Height	ft	m			
Moles	frac	perc			
Mole Volume	ft3/lbmol	m3/kgmol	gm/gmol		
Mole Weight	lb/lbmol	kg/kgmol	gm/gmol		
Null					
Pressure	psia	psig	bara	barg	kPa
Salinity	mfrac	ppm	wt%		
Temperature	degR	degF	degC	Kelv	
Viscosity	cP				

Table 7: Allowed Unit Sets

An example of an experiment definition is the first CCE experiment in the GASANDOIL.dat input file shown in Figure 3. Following the CCE type declaration, the next line specifies the samp(le) is the GasCon fluid, the reservoir temperature is 239.0 ^{o}F (degF) and that liquid dropout data will be specified as the liquid volume per dew point volume (Sliq DEW).

The (gas) viscosity data is given a weighting factor of zero (VISG 0.0) whilst the saturation pressure is given a weight to ten (PSAT 10.0).

Plots of relative volume (RELV), Z-factor (ZFAC) and liquid dropout (SLIQ) have been requested, but not the gas viscosity (VISG).



```
CCE
  samp GasCon  Tres 239.0 degF Sliq DEW
WEIG  VISG 0.0  PSAT  10.0
PLOT          Yes       Yes       Yes     No
      PRES    RELV      ZFAC      SLIQ    VISG
      psia    dim       dim       perc    cP
      7010    0.8030    1.1966    0.00    0.0390
      6714    0.8179    1.1674    0.00    0.0379
      6419    0.8343    1.1385    0.00    0.0369
      6125    0.8525    1.1101    0.00    0.0358
      5832    0.8726    1.0819    0.00    0.0347
      5538    0.8957    1.0544    0.00    0.0335
      5240    0.9220    1.0271    0.00    0.0323
      4949    0.9516    1.0011    0.00    0.0311
      4657    0.9862    0.9764    0.00    0.0299
PSAT  4554    1.0000    0.9682    0.00    0.0294
      4511    1.0059    0.0       0.13    0.0
      4360    1.0278    0.0       0.50    0.0
      4064    1.0802    0.0       1.64    0.0
      3775    1.1437    0.0       2.89    0.0
      3340    1.2630    0.0       4.46    0.0
      2899    1.4322    0.0       5.85    0.0
      2461    1.6730    0.0       6.62    0.0
      2022    2.0350    0.0       7.11    0.0
      1864    2.2250    0.0       7.22    0.0
      1474    2.8081    0.0       7.27    0.0
      1170    3.6004    0.0       7.20    0.0
```

Figure 3: Example of a CCE Experiment

Pressure (PRES) in psia defines the one column of Independent data needed to define the CCE experiment. Units of RELV, ZFAC, SLIQ and VISG are dimensionless (dim), dim, percentage (perc) and cP, respectively.

If values of a quantity are not known or reported across the range of the Independent variable, zero (0.0) should be entered. If no value is given, the *PVTfree* parser (which is limited by my programming ability) will get into a mess!

As this is a CCE experiment, the row which defines the saturation pressure (PSAT) should be marked by placing that string as the first item on that row.

Now you don't have to be as fussy as me when it comes to formatting your data file as *PVTfree* only really cares about spaces between contiguous collections of characters or strings[10]. That said, I find the format shown above easier on the eye and hence better for finding mistakes.

The SEP experiment can optionally contain two additional columns of data with column headers LSEP and VSEP; to be consistent with the observed quantities, in the following row of data the user is recommended to define 'units' with the string 'Stg#'. The purpose of LSEP and VSEP is to allow the user to define a non-default routing for the liquid and vapour output of any given stage of the experiment. If required, these two quantities (neither or both should be defined) should be the last two columns of the experiment definition.

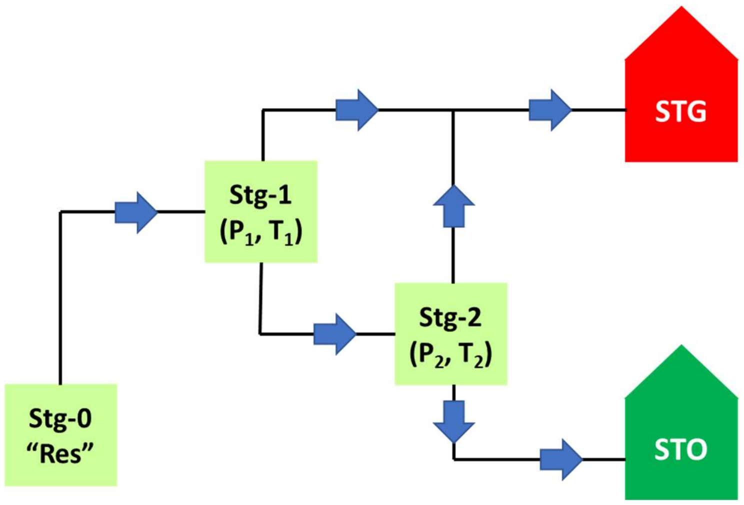


Figure 4: Default Separator Train Configuration (All Gas -> STG & Last-Stg Oil -> STO)

Remember the first row of the SEP must define reservoir temperature and saturation pressure which is deemed to be stage 0 (zero), therefore both the liquid and vapour output from this stage must go to stage 1 (one). Thereafter, by default (in which case LSEP and VSEP are not required), the liquid output of this stage (k) goes to the next stage (k+1) whereas all vapour is collected in the stock tank, designated stage zero (0). An example of this (trivial) default setup is shown in Figure 5.

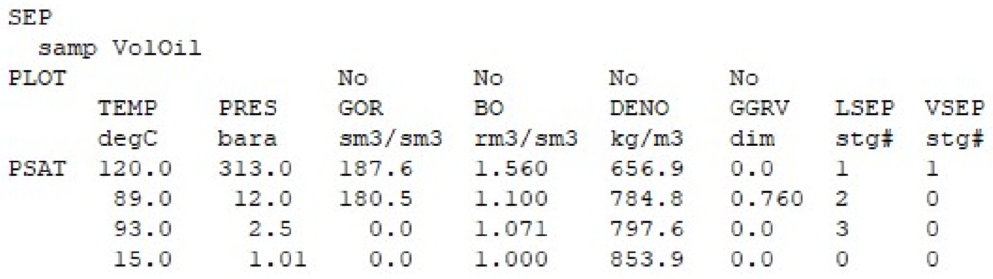

```
SEP
  samp VolOil
PLOT                    No        No        No        No
      TEMP    PRES      GOR       BO        DENO      GGRV    LSEP   VSEP
      degC    bara      sm3/sm3   rm3/sm3   kg/m3     dim     stg#   stg#
PSAT  120.0   313.0     187.6     1.560     656.9     0.0     1      1
       89.0    12.0     180.5     1.100     784.8     0.760   2      0
       93.0     2.5       0.0     1.071     797.6     0.0     3      0
       15.0     1.01      0.0     1.000     853.9     0.0     0      0
```

Figure 5: Example of a SEP experiment with default LSEP/VSEP routing

[10] The code reads the data file line by line and then uses the split() function to split it into strings, or tokens as I call them in the code. Be careful if copying data from Excel as (invisible) tab-characters may delimit the strings. Using a proper editor, you can replace the tab-characters (or other non-displaying characters) by multiple space characters.

An example of a non-standard Separator Train configuration which must use the LSEP & VSEP columns is shown in Figure 6.

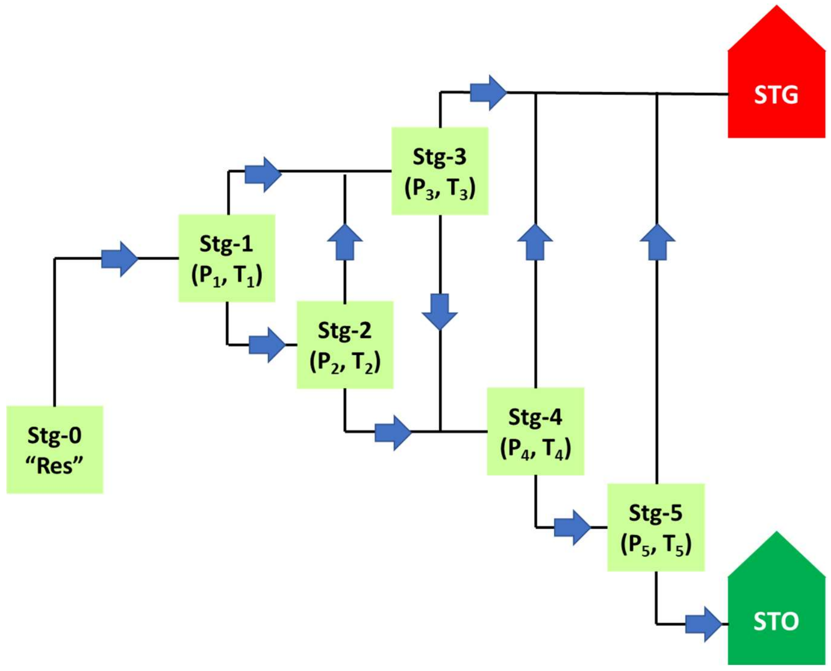


Figure 6: Non-Standard Separator Train Configuration that Requires LSEP & VSEP definitions

3.2.1.8 REG: Regression

This keyword must be preceded by the EXP/ENDEXP keywords, see Section 3.2.1.7, in which the possible set of experiments and their associated data is defined ready for use in regression. Four keywords are expected in this section as shown in Table 8.

Keyword	Purpose
EXP	Select Experiments from those currently define
VAR	Start of variable selection
ENDVAR	End of variable selection
NITER	Maximum number of iterations

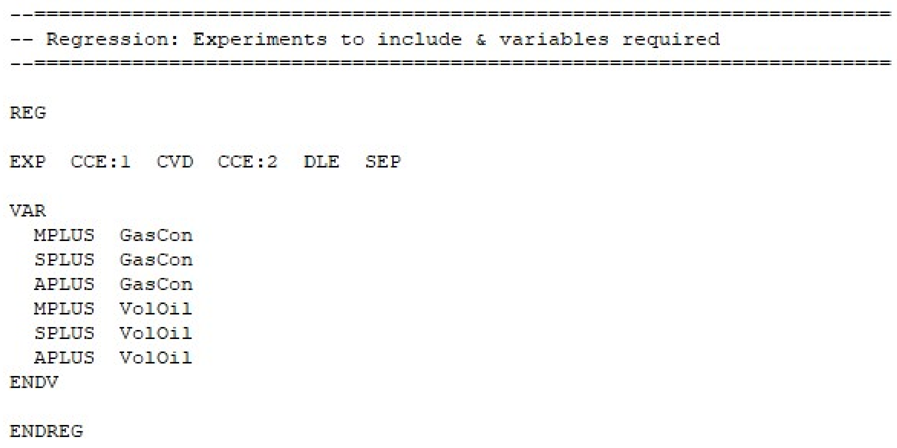
Table 8: REG Section Keywords

As stated, all experiments to be considered for regression must have been defined in one or more preceding EXP sections. Therefore, the EXP keyword in this section is used to select the set of experiments required for this particular regression. It takes the form:

```
EXP     <type:I>        <type:J>        <type:K>        …
```

Type should be one of the strings shown in Table 4. The colon ':' and integer I, J, K, … is optional and depends on whether there are two or experiments of the same type under consideration. Figure 7 gives an example of this setup.

In GASANDOIL.dat, there are two CCE experiments defined, one for the gas condensate sample GasCon and a second for the volatile oil sample VolOil, hence the two experiments are referenced CCE:1 (for GasCon) and CCE:2 (for VolOil). There is only a single instance of the CVD, DLE and SEP experiments so these strings do not need to be augmented (and should not be!).



```
--=====================================================================
-- Regression: Experiments to include & variables required
--=====================================================================

REG

EXP  CCE:1  CVD  CCE:2  DLE  SEP

VAR
  MPLUS  GasCon
  SPLUS  GasCon
  APLUS  GasCon
  MPLUS  VolOil
  SPLUS  VolOil
  APLUS  VolOil
ENDV

ENDREG
```

Figure 7: Example of REG section keywords

The NITER parameter is used to define the maximum number of iterations to be tried during the non-linear regression. The keyword takes the form:

NITER <Niter>

The default value for the integer Niter is five (5); this rarely needs increasing so this keyword is considered optional.

The keywords VAR … ENDVAR are used to define the set of variables to be used in this particular regression. Three sets are supported, these being:

• Regular	M_A, M_B, T_c, P_c, ω, s, K_{ij}
• Plus Fraction	$M_w(C_{N+})$, $\gamma(C_{N+})$, $\alpha(C_{N+})$
• LBC Viscosity	V_c

Where (M_A, M_B) are multipliers of the component EoS (Ω_A, Ω_B), i.e. initial values of (M_A, M_B) are unity, (T_c, P_c) are the component critical Temperature and critical Pressure, (ω, s) are the component acentric factor and dimensionless volume shifts and K_{ij} are the Binary Interaction Parameters. These parameters are defined using the strings:

MULTA, MULTB, TCRIT, PCRIT, ACENF, SHIFT & KIJ

To request the adjustment of any of these parameters for a given component, the use enters on a line-by-line basis something like:

PARM <comName1> *<comName2>* …

Entering more than one component name, comName, per line tells *PVTfree* to treat the set of components as a 'group' so that the selected parameter of each member of the group is adjusted by the same multiplier.

Note that if PARM = KIJ, then pairs of component names are required for the (i, j) components. Here, the first component name must be an individual component, but it can link to a group, i.e.:

KIJ C1 C7+(1) C7+(2) C7+(3)

Where the BIP between C1 and the three pseudo components split from C7+ are treated as a single variable.

The second set of parameters relate to samples created with a plus fraction like the C7+ fraction shown in Figure 2 The laboratory provided data for this 'component', namely the molecular weight and specific gravity are clearly uncertain, as is the distribution of molecules within it, as described by the shape-factor or alpha parameter. These parameters are defined using the strings:

```
MPLUS, SPLUS, APLUS
```

To represent adjustment of the plus fraction mole weight, specific gravity and shape-factor respectively. Each of these parameters should be entered on a line-by-line basis followed the name of a currently defined fluid sample, i.e.

```
MPLUS        GasCon
```

Finally, viscosity is calculated in *PVTfree* using the Lohrenz-Bray-Clark (LBC) correlation, see Section 3.4.10 of W&B. This correlation depends strongly on the reduced density $\rho_r = \rho/\rho_c$ where ρ is the fluid density and ρ_c is the critical density calculated from $\rho_c = 1/V_c$ where V_c is the mixture critical volume. Because of this strong dependency, W&B recommend the critical volumes be adjusted using 'light' and 'heavy' components. This can be achieved using the grouped variables concept, i.e.

```
VCRIT   N2   CO2  C1  C2      C3      IC4     NC4
VCRIT   IC5  NC5  C6  C7+(1)  C7+(2)  C7+(3)
```

In fact this 'default' adjustment for viscosity matching can be defined by the single argument LBCSTD within the VAR … ENDVAR pair of keywords; all components with a mole weight less than 60.0 are added to the 'light' group, otherwise they are added to the 'heavy' group.

As no other calculation in *PVTfree* depends on critical volume, it is recommended that any adjustment of viscosity takes place in a second (or later) regression run, after the phase behaviour modification has taken place. In the GASANDOIL.dat example, a sixth experiment (third CCE) as added to the set of existing experiments after the regression using plus fraction variables, see Figure 7, has completed. This experiment has only the viscosity data as observed. A second REG … ENDREG section is then defined in which only the third CCE is requested and the two VCRIT 'groups' shown above are requested.

Now, what is the best set of variables to chosen for a given dataset? That is a matter of personal choice, experience, trial-and-error, etc. If you don't have a personal choice (or have one imposed on you by company dictate) or much experience, I would strongly recommend a trial-and-error approach. Additional information is written to the reg-file which may help, and this is discussed briefly in Section 3.3.4.

3.2.1.9 BLACK: Request Blackoil Output

This section is used to request black oil table output suitable for inclusion in one of the supported commercial reservoir simulators. It should be relatively straight forward to modify the code used here (readBlk.py, blkDrive.py and writeOut.py) to generate data suitable for use in other simulators.

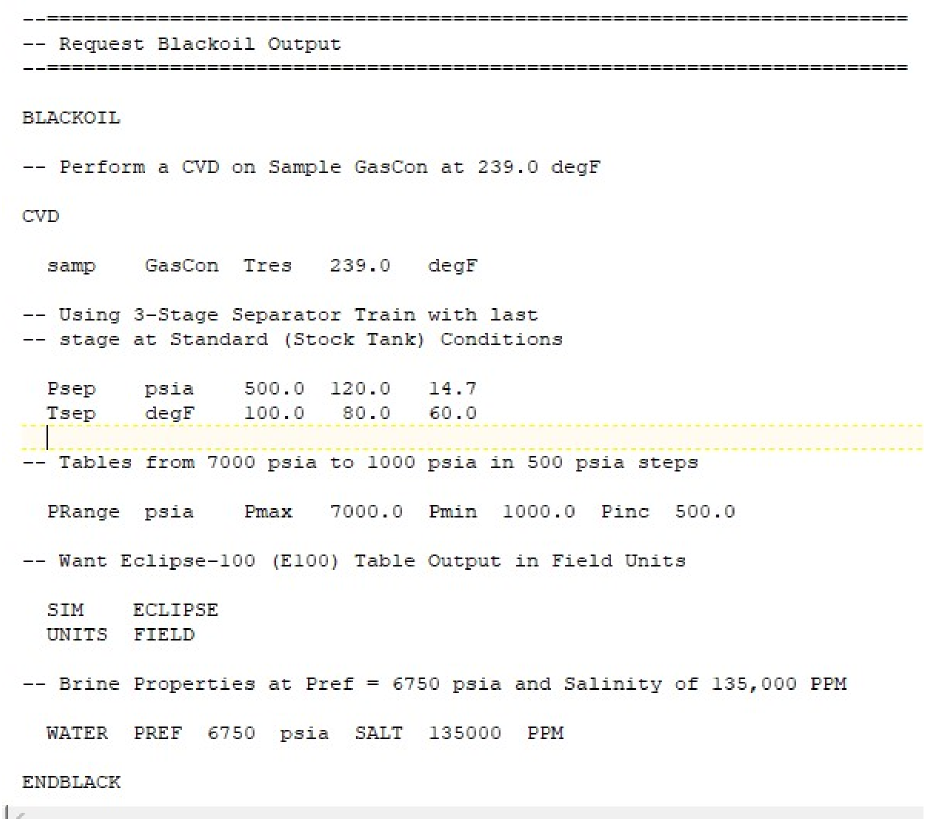
The depletion of the reservoir is simulated using one of CCE, CVD or DLE. It is expected that the single keyword CVD (with no arguments) will be selected for a gas condensate sample (or maybe for a volatile) oil whilst DLE is normally used for a crude oil sample.

The sample to be used and the reservoir temperature are defined using the following combination:

```
SAMPLE   <samName>   TRES   <tRes>   <tUnit>
```

Where samName is one of the currently defined fluid samples and tRes is a float of the reservoir temperature in the allowed temperature unit string tUnit.

The separator train is defined using the PSEP, TSEP and optionally LSEP and VSEP keywords. PSEP and TSEP are followed by the appropriate pressure or temperature units, see Table 7 for the allowed choices, and then Nsep floats which are the pressures and temperatures for the Nsep stages of the separator train. LSEP and VSEP should be specified if the routing of the liquid and vapour output from any given stage (k) differs from the default of liquid to the next stage (k+1) and vapour accumulated in the stock tank (stage 0); therefore VSEP and LSEP should be followed by Nsep integers. Note the last stage of the train should be at or very close to standard conditions (60 °F & 14.7 psia or 15.6 °C & 1.01 bars).



```
--========================================================================
-- Request Blackoil Output
--========================================================================

BLACKOIL

-- Perform a CVD on Sample GasCon at 239.0 degF

CVD

  samp    GasCon  Tres   239.0   degF

-- Using 3-Stage Separator Train with last
-- stage at Standard (Stock Tank) Conditions

  Psep    psia    500.0  120.0   14.7
  Tsep    degF    100.0   80.0   60.0

-- Tables from 7000 psia to 1000 psia in 500 psia steps

  PRange  psia    Pmax   7000.0  Pmin  1000.0  Pinc  500.0

-- Want Eclipse-100 (E100) Table Output in Field Units

  SIM    ECLIPSE
  UNITS  FIELD

-- Brine Properties at Pref = 6750 psia and Salinity of 135,000 PPM

  WATER  PREF  6750  psia  SALT  135000  PPM

ENDBLACK
```

Figure 8: Example of BLACKOIL section keywords & data

The pressure range and increment for the tables is defined with the following keyword:

PRANGE <pUnit> PMAX <pMax> PMIN <pMin> PINC <pInc>

The pressure units (again see Table 7) are defined by the string pUnit. The maximum, minimum and increment for the tables are defined by the three floats pMax, pMin and pInc.

To select the simulator from ECLIPSE, MORE, IMEX or VIP/Nexus, the user must define the following:

SIM <ECL|MORE|CMG|VIP|NEX> <Required_Tables>

Required_Tables should be one (for CMG) or two (for the rest) strings which are the keywords used to define the gas and/or oil PVT tables in the respective simulators. The allowed list is shown in Table 9; note if no values are provided for this/these strings, the default tables will be selected.

Simulator	OIL Tables	Gas Tables
ECL	PVCO/PVDO/**<u>PVTO</u>**	PVDG/**<u>PVTG</u>**
MORE	**<u>OPVT</u>**/OPVD/PVTO	**<u>GPVT</u>**/PVTG
VIP/Nexus	**<u>BOOTAB</u>**/BODTAB	**<u>BOGTAB</u>**/BDGTAB
CMG	PVT/PVTG/**<u>PVTVO</u>**/PVTCOND	

Table 9: Allowed Blackoil Tables by Simulator [Defaults in Bold/Underline]

The units adopted in the outputted tables are requested with the UNITS keyword:

```
UNITS   <FIELD|METRIC>
```

If water properties are required (via DENSITY and PVTW) then this is done via the WATER keyword:

```
WATER    PREF  <pRef>    <pUnit>    SALT    <sCon>    <sUnit>
```

Where pRef is a float of the reference pressure in units of pUnit and sCon is a float of the salt concentration in units of sUnit.

The output from this section is written to the appropriate file, depending on the simulator selected, see Section 3.3.5. An example from the GASANDOIL.dat input file is shown in Figure 8.

3.2.1.10 COMP: Request Compositional Output

This section is used to request compositional output suitable for inclusion in the in one of the supported compositional reservoir simulators. It should be relatively straight forward to modify the code used here (readComp.py) to generate data suitable for use in other simulators.

Output to Eclipse 300, Tempest-MORE, GEM or VIP/Nexus is supported:

```
SIM      <E300|MORE|CMG|VIP>
```

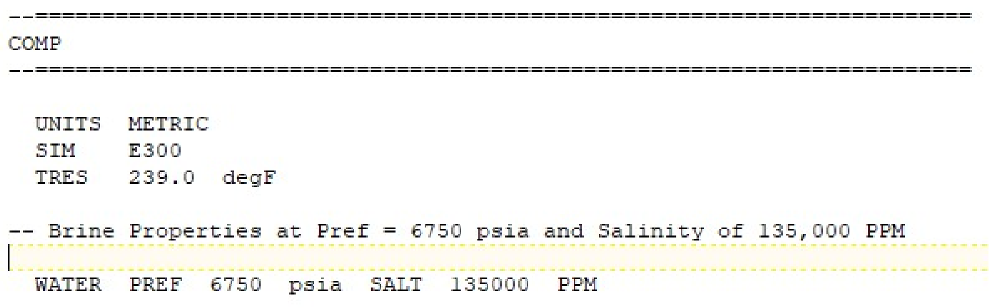
The output units should be selected from FIELD or METRIC thus:

```
OUTPUT          <FIELD|METRIC>
```

And the reservoir temperature should be defined with the TRES keyword:

```
TRES     <tRes>  <tUnit>
```

Where tRes is a float defining the temperature and tUnit is a string defining the units.



```
--========================================================================
COMP
--========================================================================

  UNITS  METRIC
  SIM    E300
  TRES   239.0  degF

-- Brine Properties at Pref = 6750 psia and Salinity of 135,000 PPM

  WATER  PREF  6750  psia  SALT  135000  PPM
```

Figure 9: Example of COMP Section Keywords and Data

Water properties can be defined by using the WATER keyword defined in the BLACKOIL Section 3.2.1.9.

The output from this section is written to the e300 file, see Section 3.3.6. An example from the GASANDOIL.dat input file is shown in Figure 9.

3.2.1.11GROUP: Perform Grouping

This section is used to reduce the number of components, ahead of a potential output for compositional simulation. The data is read and processed by the file readGrp.py.

In the recommended grouping procedure described in Section 5.7.2 of W&B, most properties of the newly created groups are weighted using the mole fractions of the components which comprise the group. Therefore, a sample has to be selected to provide these weights. This done using the SAMPLE keyword thus:

SAMPLE <samName>

Having performed the grouping as described below, the option exists to run all experiments currently defined, before and after the grouping to see the effect. This is achieved with the keyword:

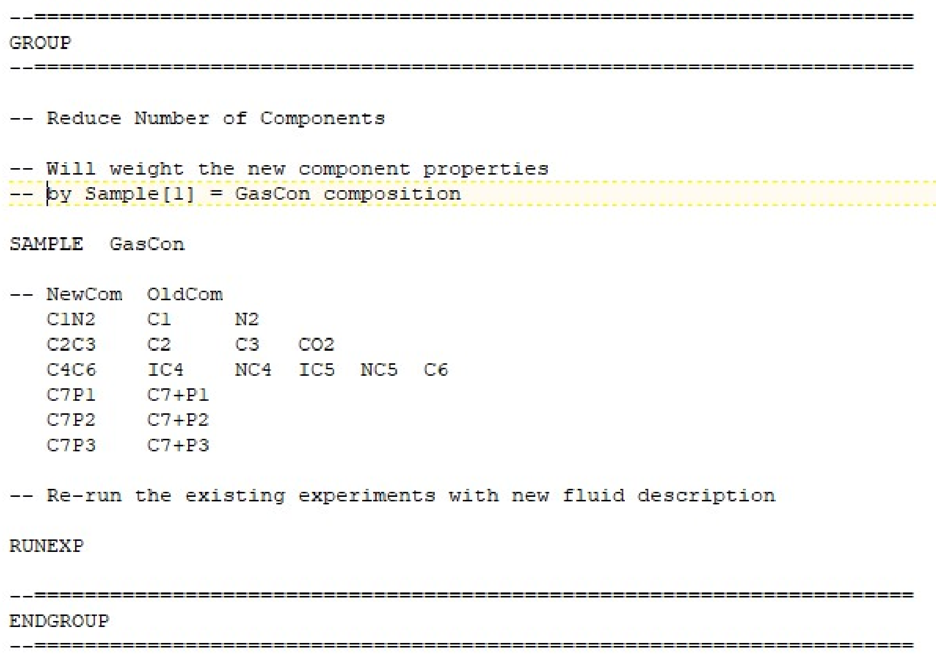
RUNEXP

Finally, it is necessary to define the names of the new groups and the 'old' components they contain. Each group must contain at least one 'old' component:

newGroup <comNam1> *<comNam2>* *<comNam3>* …

An example of the format of this section from GASANDOIL.dat is shown in Figure 10. Going from 13 to 6 components in one step is probably excessive and users are recommended to study the 'Stepwise Regression' procedure described in Section 5.7.3 of W&B ahead of performing this option.

If successful, the modified fluid description will be written to the output (.out) file and to the save (.sav) file; see Sections 3.3.1 and 3.3.3 respectively.



```
--========================================================================
GROUP
--========================================================================

-- Reduce Number of Components

-- Will weight the new component properties
-- by Sample[1] = GasCon composition

SAMPLE  GasCon

-- NewCom  OldCom
   C1N2    C1     N2
   C2C3    C2     C3    CO2
   C4C6    IC4    NC4   IC5   NC5   C6
   C7P1    C7+P1
   C7P2    C7+P2
   C7P3    C7+P3

-- Re-run the existing experiments with new fluid description

RUNEXP

--========================================================================
ENDGROUP
--========================================================================
```

Figure 10: Example of GROUP Keywords and Data

3.3 Other Files

PVTfree creates a number of other files, depending on what the user asks of the software. These files are listed in Table 10; a detailed description of the contents of the files follows below.

The files will be created in the directory in which the input file currently resides and will take the form of rootname.ext where rootname is the name of the input file (minus the '.dat' extension) and the extension is shown in the second column of Table 10.

Optionally, graphics output will be generated by *PVTfree* (in the Portable Network Graphics format with the '.png' extension) in the GraphOut sub-directory (created by *PVTfree* if it doesn't already exist) below the directory where the input file currently resides.

File	Extension	Purpose
Output	.out	Output Initialisation, Experiments, etc.
Debug	.deb	Detailed (Algorithmic) Debug
Save	.sav	Save EoS Description Post-Initialisation, Regression or Grouping
Regression	.reg	Detailed Regression Output
Eclipse 100	.e100	Schlumberger Eclipse 100 PVT Input File
Eclipse 300	.e300	Schlumberger Eclipse 300 PVT Input File
IMEX	.imex	CMG IMEX Input File
GEM	.gem	CMG GEM Input File
VIP/Nexus	.vip	Landmark VIP or Nexus Input File
MORE	.mor	Emerson Roxar Tempest-MORE Input File

Table 10: Other Files Generated by *PVTfree*

3.3.1 Output File & GraphOut Plots

The output file with the extension .out contains the results of the fluid characterisation or changes to it via regression or grouping along with the results of running the experiments. Users may find experiment results are best analysed using the plots written in the GraphOut sub-directory. Figure 11 shows an example of a before (red line) and after (green line) grouping plot of the liquid dropout from the CVD experiment along with the measured data (red squares).

3.3.2 Debug File

This file will only be generated if the user activates the DEBUG … ENDDEBUG pair of keywords with allowed arguments. It should be used to understand the internal workings of *PVTfree* and is unlikely to be of help unless trying to diagnose a potential algorithmic problem.

3.3.3 Save File

After *PVTfree* has successfully initialised the fluid description, performed a regression or a grouping of components, the new fluid description (components, component properties, BIPS and compositions) will be written to this file. Each description will be book-ended by the DEF-ENDDEF pair of keywords.

The user can cut the appropriate block of data from the '.sav' file to the start of a new input file as an alternative to performing a new initialisation. This feature can be useful if the user is experimenting with regression and/or grouping schemes.

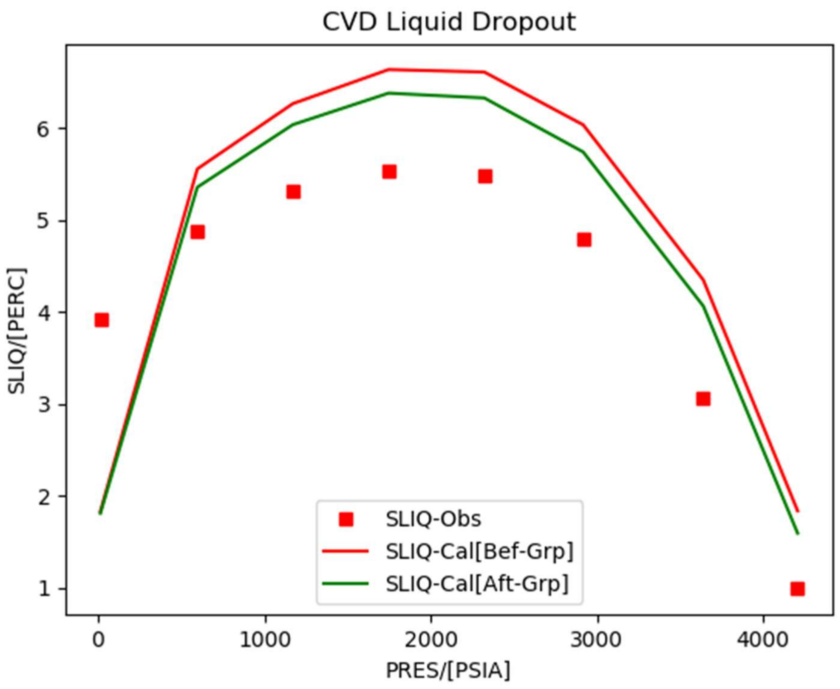


Figure 11: Example of Plot Produced by *PVTfree*

3.3.4 Regression File

This file with the extension .reg is only generated if a regression is made. It will list the initial and final (post-regression) values of the EoS calculated quantities alongside their user defined observed values. Using the calculated y_i^{cal} and observed y_i^{obs} values, a residual is defined from:

$$r_i = w_i \frac{\left(y_i^{cal} - y_i^{obs}\right)}{y_i^{obs}}$$

From which a sum of squares error *f* is calculated:

$$f = \frac{1}{2}\sum_{i=1}^{M} r_i^2$$

The Rotational Discrimination (RD) as described in Section 4.2 of SOR seeks to minimise *f* by adjusting a set of user defined variables $\underline{x} = (x_1, x_2, \dots, x_N)$ where of course $y_i^{cal} = y_i^{cal}(\underline{x})$ and hence $f = f(\underline{x})$.

As well as the measured and initial & final values of the calculated data, the reg file lists the weighting factors w_i, the residuals and the percentage contribution of each measurement to the total *f*.

Following suggestions in Section 4.3 of SOR, the reg file also shows iteration by iteration the eigenvalues of the Hessian matrix and whether on this basis one or more variables should be dropped from the regression, whether any pairs of variables are linearly dependent and the value of the variable multipliers (initially all multipliers are set to unity). Limits are applied to each variable depending on its type. These are set in the classReg class at the head of file readReg.py. They may become adjustable within the input data file in a later version of *PVTfree*.

3.3.5 Blackoil Simulator File

This file is only created if the BLACKOIL option is activated, see Section 3.2.1.9. The file should contain the data needed by selected simulator to define the oil, water and gas properties for one fluid region.

It should contain saturated, under-saturated and 'extended ' data for the oil & gas phases where the 'extended' data pertains to pressures greater than the saturation pressure is calculated by the method of Singh et al., Reference **5**.

3.3.6 Compositional Simulator File

This file is only created if the COMP option is activated, see Section 3.2.1.10. The file should contain the data needed by the chosen simulator to perform its EoS calculations and the water.

It is likely that the fluid model will have been subject to grouping, see Section 3.2.1.11, before a compositional simulation file output is requested.

4 References

1 “Phase Behavior”,
Whitson, C.H., and Brule, M.R.,
SPE Monograph, Richardson, Texas, 2000.

2 Soreide, I.,
“Improved Phase Behavior Predictions of Petroleum Reservoir Fluids from a Cubic Equation of State”,
Ph.D. Thesis, University of Trondheim, Norway, April 1989, at
http://www.ipt.ntnu.no/~curtis/courses/Theses/Ingolf-Soreide-NTH-PhD-Thesis.pdf

3 Petitfrere, M.,
“EOS Based Simulations of Thermal and Compositional Flows in Porous Media”,
Ph.D. Thesis, Universite de Pau et Des Pays de l’Adour, France, September 2014, at,
https://www.theses.fr/2014PAUU3036.pdf

4 Nichita, D.V., and Leibovici, C.F.
“A Rapid and Robust Method for Solving the Rachford-Rice Equation using Convex Transformations”,
Fluid Phase Equilibria, 353, pp. 38-49, 2013.

5 Singh, K., Fevang, O. and Whitson, C.H.,
“Consistent Black-Oil PVT Table Modification”,
SPE 109596, Presented at the 2007 SPE ATCE, Anaheim, California, 11-14 Nov 2007.

6 “Numerical Recipes”, Section 5.6, Quadratic & Cubic Equations,
http://www.aip.de/groups/soe/local/numres/bookfpdf/f5-6.pdf

7 Michelsen, M.L.,
“A simple method for calculation of approximate phase boundaries”,
Fluid Phase Equilibria, 98, p. 1-11, 1994.

Appendix A – Theoretical Background

A.1 Equation of State

The most commonly used EoS in commercial PVT packages and reservoir & production simulators are the Peng-Robinson (PR) and Soave-Redlich-Kwong (SRK). Both these EoS can be written using the following formalism:

(1) $$p = \frac{RT}{(V-b)} - \frac{a}{(V+m_1 b)(V+m_2 b)}$$

I've always referred to this formalism as Martin's two-parameter EoS, but I can't find a reference to that. Regardless, the constants (*m₁*, *m₂*) are defined in Table 11.

EoS	m_1	m_2	n_1	n_2	n_3	n_4	n_5
SRK	0	1	1	2	3	2	-1
PR	1+√2	1-√2	2+√2	2-√2	4	2	1/(2√2)

Table 11: Constants in the General Two-Parameter EoS

The constants n_1 through n_5 will be defined shortly.

It should be obvious to the reader that this equation is indeed cubic in molar volume *V* but rather than working in this variable it is preferable to use the following definitions:

(2) $$A = \frac{ap}{(RT)^2} \quad B = \frac{bp}{RT} \quad Z = \frac{pV}{RT}$$

So that (1) becomes:

(3) $$1 = \frac{1}{(Z-B)} - \frac{A}{(Z+m_1 B)(Z+m_2 B)}$$

Where *Z* is the Z-factor. Now clearly for the solution of this equation to be physically realistic we require $\eta = (Z - B) > 0$. We can re-write the second term on the right-hand side of (3) using the η-parameter by noting that $(Z + m_1 B) = (\eta + n_1 B)$ and $(Z + m_2 B) = (\eta + n_2 B)$ where $n_1 = m_1 + 1$ and $n_2 = m_2 + 1$ as shown in Table 11 so that (3) becomes:

(4) $$1 = \frac{1}{\eta} - \frac{A}{(\eta + n_1 B)(\eta + n_2 B)}$$

Expanding the denominator of the second term on the right-hand side of (4) and defining $n_3 = n_1 + n_2$ and $n_4 = n_1 \times n_2$, then (4) becomes:

(5) $$1 = \frac{1}{\eta} - \frac{A}{(\eta^2 + n_3 B\eta + n_4 B^2)}$$

And multiplying (5) through gives:

(6) $$f(\eta) = \eta^3 + (n_3 B - 1)\eta^2 + (n_4 B^2 - n_3 B + A)\eta - n_4 B^2 = 0$$

For future algebraic convenience, we will define the following variables:

(7) $$E_2 = n_3 B - 1 \quad E_1 = n_4 B^2 - n_3 B + A \quad E_0 = -n_4 B^2$$

So that (6) becomes:

$$f(\eta) = \eta^3 + E_2\eta^2 + E_1\eta + E_0 \tag{8}$$

This is the form of the cubic EoS solved in *PVTfree*. Having found the one or three real (physical, $\eta > 0$) roots, the Z-Factor(s) are simply $Z = \eta + B$. Of interest here is how we solve (8).

The NumPy library contains a function called roots which solves the general Nth order polynomial; after some experiments, the method described in Numerical Recipes, see Reference **6** was used.

A.1.1 Equation of State Derivatives

Now because (8) only depends on the values of (A, B), derivatives of η with respect to pressure, temperature or composition are readily found; we will use the symbol θ to represent the generic parameter p, T or x_i. Because η is defined implicitly in (8), we must use the Implicit Function Theorem to find the required derivative, namely:

$$\frac{\partial f}{\partial \theta} + \frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial \theta} = 0 \quad \text{so that} \quad \frac{\partial \eta}{\partial \theta} = -\left(\frac{\partial f}{\partial \theta}\right)\Big/\left(\frac{\partial f}{\partial \eta}\right)$$

Where:

$$\frac{\partial f}{\partial \eta} = 3\eta^2 + 2E_2\eta + E_1 \quad \text{and} \quad \frac{\partial f}{\partial \theta} = \eta^2\frac{\partial E_2}{\partial \theta} + \eta\frac{\partial E_1}{\partial \theta} + \frac{\partial E_0}{\partial \theta}$$

With:

$$\frac{\partial E_2}{\partial \theta} = n_3\frac{\partial B}{\partial \theta} \quad \text{and} \quad \frac{\partial E_1}{\partial \theta} = (2n_4B - n_3)\frac{\partial B}{\partial \theta} + \frac{\partial A}{\partial \theta} \quad \text{and} \quad \frac{\partial E_0}{\partial \theta} = -2n_4B\frac{\partial B}{\partial \theta}$$

So that:

$$\frac{\partial f}{\partial \theta} = \eta\frac{\partial A}{\partial \theta} + [n_3\eta^2 + (2n_4B - n_3)\eta - 2n_4B]\frac{\partial B}{\partial \theta}$$

And so, it should be obvious that:

$$\frac{\partial \eta}{\partial \theta} = \eta_A\frac{\partial A}{\partial \theta} + \eta_B\frac{\partial B}{\partial \theta} \tag{9}$$

Where:

$$\eta_A = -\frac{\eta}{(\partial f/\partial \eta)} \quad \textit{and} \quad \eta_B = -\frac{[n_3\eta^2 + (2n_4B - n_3)\eta - 2n_4B]}{(\partial f/\partial \eta)}$$

A.2 The (log) Fugacity Coefficient

In the general two-parameter formalism adopted here, the log of the fugacity coefficient of the ith component can be calculated from:

$$ln\phi_i = -ln(Z - B) + (Z - 1)\frac{B_i}{B} + \frac{1}{(m_1 - m_2)}\frac{A}{B}\left(2\frac{A_i}{A} - \frac{B_i}{B}\right)ln\left(\frac{Z + m_2B}{Z + m_1B}\right)$$

This can be re-written in terms of the η-parameter as:

$$ln\phi_i = -ln(\eta) + (\eta + B - 1)\frac{B_i}{B} + n_5\frac{A}{B}\left(2\frac{A_i}{A} - \frac{B_i}{B}\right)ln\left(\frac{\eta + n_2B}{\eta + n_1B}\right) \tag{10}$$

Where $n_5 = 1/(m_1 - m_2) = 1/(n_1 - n_2)$. Values of n_5 are shown for the SRK & PR EoS in Table 11.

To facilitate the calculation of this expression, and in particular its pressure, temperature & composition derivatives, it is convenient to define a number of work variables:

$$\sigma_1 = \eta + n_1 B,\ \ \sigma_2 = \eta + n_2 B,\ \ C_1 = -ln(\eta),\ \ C_2 = \eta + B - 1,\ \ C_3 = n_5 \frac{A}{B},\ \ C_4 = ln\left(\frac{\sigma_2}{\sigma_1}\right)$$

Along with:

$$\alpha_i = \frac{A_i}{A},\ \ \beta_i = \frac{B_i}{B}$$

So that (10) becomes:

(11) $$ln\phi_i = C_1 + 2D_A\alpha_i + D_B\beta_i$$

Where $D_A = C_3C_4$ and $D_B = C_2 - C_3C_4$ The derivatives of (11) with respect to $\theta = p$, T or x_i are:

(12) $$\frac{\partial ln\phi_i}{\partial\theta} = \frac{\partial C_1}{\partial\theta} + 2D_A\frac{\partial\alpha_i}{\partial\theta} + 2\alpha_i\frac{\partial D_A}{\partial\theta} + D_B\frac{\partial\beta_i}{\partial\theta} + \beta_i\frac{\partial D_B}{\partial\theta}$$

All the derivatives in (11) ultimately depend on the derivatives of the (A, B) coefficients or their component equivalents (A_i, B_i); the derivatives of η have already been defined in Section A.1.1.

A.2.1Pressure Derivatives

The derivative with respect to pressure is particularly easy because of the dependence of the (A, B) and (A_i, B_i) coefficients with respect to pressure. It is readily shown that:

$$\frac{\partial A}{\partial p} = \frac{A}{p},\ \ \frac{\partial B}{\partial p} = \frac{B}{p},\ \ \frac{\partial A_i}{\partial p} = \frac{A_i}{p}\ \ and\ \ \frac{\partial B_i}{\partial p} = \frac{B_i}{p}$$

And therefore:

$$\frac{\partial\alpha_i}{\partial p} = \frac{\partial\beta_i}{\partial p} = \frac{\partial C_3}{\partial p} = 0$$

Thus (12) becomes:

(13) $$\frac{\partial ln\phi_i}{\partial p} = \frac{\partial C_1}{\partial p} + 2\alpha_i\frac{\partial D_A}{\partial p} + \beta_i\frac{\partial D_B}{\partial\theta}$$

Where the (D_A, D_B) derivatives are readily calculated; note the C_3-derivative is zero as stated just above!

A.2.2 Compositional Derivatives

For completeness, the compositional derivatives follow. These are not currently implemented in *PVTfree* as it doesn't use any full Newtonian formulation of the saturation pressure, flash or stability test. However, who knows what the author, or you, might want to do with the software in the future[11].

Firstly, the derivatives of the (A, B) and (A_i, B_i) coefficients are:

$$\frac{\partial A}{\partial x_j} = A_j \qquad \frac{\partial A_i}{\partial x_j} = A_{ij} \qquad \frac{\partial B}{\partial x_j} = B_j \qquad \frac{\partial B_i}{\partial x_j} = 0$$

So that the derivatives of the (α_i, β_i) coefficients become:

[11] Does anyone fancy writing a Slim Tube option? Basically a 2-Phase (no Water) 1D compositional simulator. It's probably not as hard as it sounds … honest!

$$\frac{\partial \alpha_i}{\partial x_j} = \frac{A_{ij}}{A} - \alpha_i \alpha_j \qquad \frac{\partial \beta_i}{\partial x_j} = -\beta_i \beta_j$$

From Equation (9d), it should be obvious that:

$$\frac{\partial \eta}{\partial x_j} = \eta_A \frac{\partial A}{\partial x_j} + \eta_B \frac{\partial B}{\partial x_j} = \eta_A A_j + \eta_B B_j$$

But as we prefer to work with the (α_i, β_i) coefficients rather than the (A_i, B_i) coefficients, this last equation can be re-written as:

$$\frac{\partial \eta}{\partial x_j} = \eta_A A \alpha_j + \eta_B B \beta_j$$

Since $A_j = A\alpha_j$ & $B_j = B\beta_j$. Therefore, defining new coefficients:

$$\eta_A^C = A\eta_A \;\&\; \eta_B^C = B\eta_B$$

The compositional derivative of the η-parameter becomes:

$$\frac{\partial \eta}{\partial x_j} = \eta_A^C \alpha_j + \eta_B^C \beta_j$$

Since $C_1 = -\ln(\eta)$, then:

$$\frac{\partial C_1}{\partial x_j} = -\frac{1}{\eta}\frac{\partial \eta}{\partial x_j} = -\frac{1}{\eta}\left(\eta_A^C \alpha_j + \eta_B^C \beta_j\right)$$

Defining $C_1^A = -\eta_A^C/\eta$ & $C_1^B = -\eta_B^C/\eta$, then:

$$\frac{\partial C_1}{\partial x_j} = C_1^A \alpha_j + C_1^B \beta_j$$

Previously we defined $C_2 = \eta + B - 1$ so that:

$$\frac{\partial C_2}{\partial x_j} = \frac{\partial \eta}{\partial x_j} + B_j = \eta_A^C \alpha_j + \eta_B^C \beta_j + B\beta_j = C_2^A \alpha_j + C_2^B \beta_j$$

Where $C_2^A = \eta_A^C$ and $C_2^B = \left(\eta_B^C + B\right)$

Next, given we have defined $C_3 = n_5\, A/B$ then:

$$\frac{\partial C_3}{\partial x_j} = C_3\left(\frac{A_j}{A} - \frac{B_j}{B}\right) = C_3\left(\alpha_j - \beta_j\right) = C_3^A \alpha_j + C_3^B \beta_j$$

Where $C_3^A = C_3$ and $C_3^B = -C_3$.

Next, we consider $C_4 = ln(\sigma_2/\sigma_1)$, so that:

$$\frac{\partial C_4}{\partial x_j} = \frac{1}{\sigma_2}\frac{\partial \sigma_2}{\partial x_j} - \frac{1}{\sigma_1}\frac{\partial \sigma_1}{\partial x_j} = \frac{1}{\sigma_2}\left(\frac{\partial \eta}{\partial x_j} + n_2 B_j\right) - \frac{1}{\sigma_1}\left(\frac{\partial \eta}{\partial x_j} + n_1 B_j\right)$$

This then becomes:

$$\frac{\partial C_4}{\partial x_j} = \left(\frac{1}{\sigma_2} - \frac{1}{\sigma_1}\right)\eta_A^C \alpha_j + \left[\frac{\left(\eta_B^C + n_2 B\right)}{\sigma_2} - \frac{\left(\eta_B^C + n_1 B\right)}{\sigma_1}\right]\beta_j = C_4^A \alpha_j + C_4^B \beta_j$$

Where $C_4^A = \left(\frac{1}{\sigma_2} - \frac{1}{\sigma_1}\right)\eta_A^C$ and $C_4^B = \left[\frac{\left(\eta_B^C+n_2 B\right)}{\sigma_2} - \frac{\left(\eta_B^C+n_1 B\right)}{\sigma_1}\right]$.

Now generally, any compositional derivatives will be written in terms of unconstrained mole numbers X_j rather than mole fractions x_j since the latter are constrained (they must sum to unity), where:

$$x_j = \frac{X_j}{S_x} \quad where \quad S_x = \sum_{k=1}^{N} X_k$$

So that:

$$\frac{\partial ln\phi_i}{\partial X_j} = \frac{1}{S_x}\left(\frac{\partial ln\phi_i}{\partial x_j} - \sum_{k=1}^{N} x_k \frac{\partial ln\phi_i}{\partial x_k}\right)$$

The term inside the summation can be greatly simplified by noting that $\sum_{i=1}^{N} \alpha_i = \sum_{i=1}^{N} \beta_i = 1$ so that the terms $\sum_{j=1}^{N} x_j \left(\partial\, C_m / \partial x_j\right) = C_m^A + C_m^B$ for m = 1,2,3,4.

A.2.3 Temperature Derivatives

The temperature dependency of the B-coefficient is trivially calculated from:

$$\frac{\partial B}{\partial T} = -\frac{B}{T} \quad \text{and} \quad \frac{\partial B_i}{\partial T} = -\frac{B_i}{T} \quad \text{which implies} \quad \frac{\partial \beta_i}{\partial T} = 0$$

Perhaps surprisingly, the temperature derivatives of the log fugacity coefficients are the hardest to formulate because of the temperature dependency of the A-coefficient.

$$A = \frac{p}{T^2} a(T) \quad \text{which implies} \quad \frac{\partial A}{\partial T} = A\left(\frac{1}{a}\frac{\partial a}{\partial T} - \frac{1}{T}\right)$$

The temperature dependence of the (small) a-coefficient for the PR & SRK EoS equations can be written:

$$a = \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \left(a_{ci} a_{cj}\right)^{1/2} \left(\alpha_i \alpha_j\right)^{1/2} \left(1 - K_{ij}\right)$$

Where the a_{ci} and a_{cj} depend only on the component (i and j) critical temperature, critical pressure, Ω_A and Ω_B coefficients, K_{ij} are the Binary Interaction Parameters (BIPs) and it is the α_i and α_j expressions that contain the additional temperature dependency via:

$$\alpha_i^{1/2} = 1 + m(\omega_i)\left(1 - T_{ri}^{1/2}\right)$$

Where $m(\omega_i)$ is a polynomial (of quadratic or cubic order) in the acentric factor ω_i and T_{ri} is the reduced temperature, $T_{ri} = T/T_{ci}$ where T_{ci} is the critical temperature of the ith component Defining $p_i = 1 + m(\omega_i)$ and $q_i = -m(\omega_i)/T_{ci}^{1/2}$ then we have:

$$\alpha_i^{1/2} = p_i + q_i T^{1/2}$$

Now if we define:

$$\sigma_{pi} = \sum_{j=1}^{N} x_j a_{cj}^{1/2} p_j \left(1 - K_{ij}\right) \quad \text{and} \quad \sigma_{qi} = \sum_{j=1}^{N} x_j a_{cj}^{1/2} q_j \left(1 - K_{ij}\right)$$

From which we can then define:

$$\tau_{pp} = \sum_{i=1}^{N} x_i a_{ci}^{1/2}\, p_i \sigma_{pi}, \quad \tau_{pq} = \sum_{i=1}^{N} x_i a_{ci}^{1/2}\, p_i \sigma_{qi}, \quad \tau_{qp} = \sum_{i=1}^{N} x_i a_{ci}^{1/2}\, q_i \sigma_{pi} \quad and \quad \tau_{qq} = \sum_{i=1}^{N} x_i a_{ci}^{1/2}\, q_i \sigma_{qi}$$

Which are all independent of temperature to give:

$$a = \tau_{pp} + \left(\tau_{pq} + \tau_{qp}\right) T^{1/2} + \tau_{qq} T$$

Which is easily differentiable; furthermore, it can be shown that $\tau_{pq} = \tau_{qp}$. Similar expressions can be derived for the derivatives of the (A_i, B_i).

Appendix B – Python Modules

At the time of typing (June 2019), the release contains the following modules, see Table 12.

Module Name	Purpose
blkCoefs.py	Calculates the 2-Component EoS Coefficients (and their derivatives) for the Blackoil Table generation
blkDrive.py	Driver for generating Blackoil Tables
blkE100.py	Writes Eclipse 100 Tables
blkIMEX.py	Writes IMEX Tables
blkMORE.py	Writes MORE Tables
blkOther.py	Other routines used in Blackoil Table generation
blkProps.py	Calculate Blackoil properties
blkRegEOS.py	Regression of the 6-Parameter 2-Component EoS for Blackoil Tables
blkRegVis.py	Regression of the 6-Parameter 2-Component Viscosity Model for Blackoil Tables
blkVIP.py	Writes VIP or Nexus Tables
calcEOS.py	EoS Coefficients, Fugacities (and Derivatives), Cubic Solver, Phase Properties
calcExps.py	Performs the Experiments (CCE, CVD, DLE, SEP, PSAT, FLASH, SWELL & GRAD)
calcFlash.py	Two-Phase Flash
calcPhsPlt.py	Performs the Approximate Michelsen Phase Plot Calculation (see Reference **7**)
calcProps.py	Assigns Library properties, calculates SCN properties, splits Plus-Fraction
calcReg.py	Performs non-linear regression
calcSat.py	Saturation Pressure (and Temperature)
calcTest.py	Test code – not all used
calcTherm.py	Thermal properties – not yet used, but it will be – keep watching
calcWater.py	Calculate water properties for simulator output
constants.py	Constants
genPlots.py	Experiment and Post-Regression/Grouping Plots
PVTfree.py	Main routine!!
readComp.py	Read and perform E300 output
readExps.py	Read experiment definitions
readGen.py	Read general data – currently just debug settings
readGrp.py	Read and perform grouping operations
readReg.py	Read data for regression
readSamp.py	Read initial sample definition
stabTest.py	Stability Test
texttable.py	Module for creating ASCII tables, acknowledging Gerome Fournier
units.py	Defines units and internal/external unit conversion routines
writeOut.py	Writes data to output files

Table 12: List of Modules in current *PVTfree* release