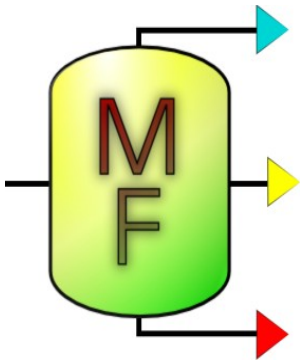
Multiflash Programmers guide

Infochem/KBC Advanced Technologies plc

Version 6.1

June 2016



Infochem/KBC Advanced Technologies plc
42-50 Hersham Road
Walton on Thames
Surrey
KT12 1RZ
UK
T +44 (0) 1932 242 424
F +44 (0) 1932 224 214

e-mail: software@kbcat.com

NOTICE

This User Guide and the information contained within is the copyright of KBC Advanced Technologies plc

Infochem/KBC Advanced Technologies plc
42-50 Hersham Road
Walton on Thames
Surrey
KT12 1RZ
UK
T +44 (0) 1932 242 424
F +44 (0) 1932 224 214
e-mail: software@kbcat.com

Disclaimer

While every effort has been made to ensure that the information contained in this document is correct and that the software and data to which it relates are free from errors, no guarantee is given or implied as to their correctness or accuracy. Neither KBC Advanced Technologies plc nor any of its employees, contractors or agents shall be liable for direct, indirect or consequential losses, damages, costs, expenses, claims or fee of any kind resulting from any deficiency, defect or error in this document, the software or the data.

Contents

Changes to Multiflash in Versions 6.0....8
Models....8
1.2 Model tuning....8
1.3 Importing....8
2 Introduction....9
3 Preliminaries....11
3.1 Initialisation sequence....11
3.2 Security system....11
3.2.1 Initialising the security system....12
3.2.2 Closing the security system....12
3.3 Numerical parameters....12
3.4 Include files....13
3.4.1 Numerical parameters....13
3.4.2 Unique compound identifiers....14
3.5 Error reporting....14
3.5.1 Standard error reporting system....14
3.5.2 Error codes....15
3.5.3 Error-handling utilities....15
3.6 Multiflash version....17
4 Defining pure components and data sources....18
4.1 Initialisation sequence....18
4.2 Initialisation of the pure component data software....18
4.3 Databank files used by Multiflash....19
4.4 Opening pure component databanks....19
4.5 Unloading databanks....21
4.6 Loading pure component data....21
4.7 Loading components from a databank....22
4.8 Defining a component by entering data directly....23
4.8.1 Entering numerical constants....23
4.8.2 Setting numerical constants to undefined....25
4.8.3 Entering character constants....25
4.8.4 Entering correlations for temperature-dependent properties....26
4.8.5 Entering Unifac subgroup structures....28
4.9 Petroleum fractions....28
4.9.1 Correlations for petroleum fraction properties....29
4.9.2 Defining petroleum fractions....29
4.10 Searching for compounds on a databank....31
4.10.1 AYPDBI: Search for a compound name:....31
4.10.2 AYPDBS: Search for compound names containing specified string:....32
4.10.3 AYPDBY: Return synonyms for compound name:....33
4.10.4 AYPDBF: Search for compounds with specified formula:....34
4.11 Reading data directly from a databank....35
4.11.1 AYPDBK: Pure component character data:....35
4.11.2 AYPDBV: Pure component real-valued constant data:....35
4.12 Other utilities....36
4.12.1 AYPNC1: Number of components defined:....36
4.12.2 AYPDEL: Delete data for a component:....36
4.12.3 AYSALN: Equivalent salt component/ions concentration:....37
5 Evaluating pure component properties....40
5.1 Summary of routines....40

5.2 Use of NCO and IC arguments....41
5.3 Constant properties....41
5.3.1 AYPCNK: character valued constants....41
5.3.2 AYPCNR: real valued constants....42
5.3.3 AYPCNS: status of constants....42
5.3.4 AYPHC: test for hydrocarbons....43
5.4 Temperature-dependent properties....43
5.4.1 AYPDIE: dielectric constant of liquid....44
5.4.2 AYPDNL: density of liquid on the saturation line....44
5.4.3 AYPHLV: enthalpy change on evaporation (latent heat)....45
5.4.4 AYPHSI: perfect gas enthalpy, entropy and heat capacity....46
5.4.5 AYPLCP: liquid heat capacity on the saturation line....46
5.4.6 AYPLH1: liquid enthalpy, entropy and heat capacity on the saturation line....47
5.4.7 AYPLTC: thermal conductivity of liquid on the saturation line....48
AYPLVS: viscosity of liquid on the saturation line....48
AYPSCP: solid heat capacity....49
AYPSDN: density of solid....50
AYPST: surface tension of liquid on the saturation line....50
AYPSVP: saturated liquid vapour pressure....51
AYPSVR: second virial coefficient....52
AYPVTC: vapour thermal conductivity in the zero-density limit....52
AYPVVS: vapour viscosity in the zero-density limit....53
5.5 Fetching coefficients in temperature-dependent property correlations....54
6 Petroleum fluid characterisation....56
6.1 Introduction....56
6.2 Using the characterisation procedure....57
6.3 Black Oil characterisation procedure....62
6.4 Blending petroleum fluids....65
7 Binary interaction parameters....68
7.1 Defining BIPs....68
7.2 Initialising the BIP software....68
7.3 Using BIP databanks....69
7.3.1 Opening BIP databanks....69
7.3.2 Loading BIPs from a databank....69
7.3.3 Unloading databanks....71
7.4 Entering BIPs directly....71
7.5 BIPs for oil and gas mixtures....75
7.6 Utilities....76
7.6.1 Getting values of stored BIP coefficients....76
7.6.2 Freeing BIP load numbers....78
7.6.3 Deleting BIPs for a component....79
8 Setting up mixture property models....80
8.1 Initialising the Models software....80
8.2 Loading models....80
8.3 Thermodynamic models....81
8.3.1 Cubic equations of state....82
8.3.2 Non-cubic equations of state....83
8.3.3 Free energy addition model....84
8.3.4 Liquid activity equations....85
8.3.5 User-modified Unifac groups and subgroups....87
8.3.6 Electrolyte model....87
8.3.7 Solids....88
8.3.8 Other thermodynamic models....89
8.4 Transport property models....89
8.5 Information about a loaded model....92
8.6 Setting model parameter data values....92
8.7 Freeing model load numbers....93

8.8 Reinitialising models....93
8.9 Example....94
9 Phase descriptors....95
9.1 Setting up phase descriptors....95
9.1.1 Initialise PD system....95
9.1.2 Define PDs....95
9.2 Getting PD information....98
9.2.1 PDs defined....98
9.2.2 Phase, key components and models associated with a PD....98
9.3 Removing a PD....100
10 Evaluating mixture properties....101
10.1 Thermodynamic properties: AYMIX....101
10.2 Enthalpy and entropy for activity coefficient models....103
10.3 Average molecular weight: AYAVMW....104
10.4 Viscosity: AYVISC....104
10.5 Thermal conductivity: AYTCND....105
10.6 Surface tension: AYSURF....106
10.7 Binary diffusion coefficient: AYDIFF....107
10.8 Specification of datum: AYDATM....108
11 Streams....110
11.1 Setting up streams....110
11.1.1 Initialise stream type system....110
11.1.2 Define stream types....110
11.2 Getting stream type information....111
11.2.1 Number of stream types defined....111
11.2.2 Phase descriptors and components associated with a stream type....111
11.3 Removing a stream type....112
12 Phase equilibrium calculations....113
12.1 Introduction....113
12.2 Multiflash operation....114
12.3 Thermodynamic property interface....115
12.4 Initialising the phase equilibrium module....116
12.5 Flash at fixed pressure and temperature: AXFL01....116
12.6 Temperature or pressure searches and tolerance calculations: AXFL02....119
12.7 Search for hydrate temperature or pressure: AXHYD....124
12.8 Double flashes: AXFL04....126
12.9 Stability test: AXST01....131
12.10 Setting low-level calculation options....133
12.10.1 First-order phase split....133
12.10.2 Second-order phase split....134
12.10.3 Stability test....135
12.10.4 Temperature/pressure searches....135
13 Flash sensitivity calculations....137
13.1 Flashes at fixed pressure and temperature: AXSN01....137
13.2 Flashes with a temperature or pressure search: AXSN02....138
13.3 Double flashes: AXSN04....140
14 Phase envelope calculations....143
14.1 Introduction....143
14.2 Calling the phase envelope routines....143
14.3 Phase envelope for flow assurance....148
15 Property matching routines....150
15.1 Introduction....150
15.2 Phase equilibrium data....150
15.3 Gas/Liquid viscosity....152
15.4 Volumetric data....154
15.5 GOR and Bubble Point Matching....155

16 Chemical equilibrium calculations....158
16.1 Introduction....158
16.2 Chemreact operation....158
16.3 Initialising Chemreact....159
16.4 Chemical equilibrium at fixed pressure and temperature: CREA01....159
16.5 Dew and bubble point searches at chemical equilibrium: CREA02....161
17 Configuring Multiflash with a command file....164
17.1 Introduction....164
17.2 Reading a command file....164
17.3 Reading a command file without loading it....167
17.4 Carrying out Calculations....168
17.5 Writing a command file....169
17.6 Ending the Multiflash session....169
17.7 Converting a PVTSim CHC/CTM file into an MFL file....169
18 Unit conversion utilities....172
18.1 UNF: Amount/composition/flowrate....172
18.2 UND: Density/volume....173
18.3 UNI: Diffusion coefficient....174
18.4 UNH: Energy (molar/specific)....175
18.5 UNS: Entropy/heat capacity (molar/specific)....175
18.6 UNP: Pressure....176
18.7 UNR: Surface tension....177
18.8 UNT: Temperature....177
18.9 UNU: Thermal conductivity....178
18.10 UNV: Viscosity....178
18.11 UNVOL: Volume....179
Appendices....181
Infochem data banks....181
Infodata....181
Data bank structure....181
Creating the data bank....186
Infocond....187
Data bank structure....187
Binary interaction parameter databanks....187
Input specification....187
BIP data line....188
Component Names....188
Model/variant identifiers....188
BIP T-dependent correlation functions....189
Entering BIP correlation coefficients....189
Examples....190
Identification codes for pure component properties....190
C.1 Numerical values....190
Character values....192
Temperature-dependent property correlations....192
Pure component properties....193
Constant properties....193
Temperature dependent properties....194
Perfect gas molar heat capacity/ enthalpy/ entropy....195
Gas viscosity at zero density....197
Gas thermal conductivity at zero density....198
Second virial coefficient....199
Saturated liquid vapour pressure....200
Enthalpy change on evaporation for saturated liquid (latent heat)....201
Saturated liquid density....202
Saturated liquid viscosity....204
Saturated liquid thermal conductivity....205

Surface tension of saturated liquid....206
D.13 Saturated liquid Cp....206
Dielectric constant of liquid....207
Solid density....207
Solid heat capacity Cp....208
Nomenclature....209
Examples of use....210
Pure component properties....210
Low-level initialisation....210
Command file initialisation....210
Thermodynamic and transport properties of a mixture....210
Low-level initialisation....210
Command file initialisation....210
Simple flash with an equation of state....211
Simple flash with an activity model....211
Flash with hydrates....211
Flash and phase properties from LKP model....211
Low-level initialisation....212
Command file initialisation....212

Changes to Multiflash in Versions 6.0

This appendix only lists changes made in the current release to the Multiflash programming interface. For a full list of changes and new features in Multiflash, please refer to the *Multiflash for Windows* user manual.

Models

Descriptions and references detailing the models are provided in the User Guide for Models and Physical Properties.

- Viscosity models The model tuning parameter for the SuperTRAPP, Pedersen and LBC viscosity model is no longer the reference viscosity. Model specific component parameters are now stored direclty: VC for LBC, MW for both SuperTRAPP and Pedersen models.

1.2 Model tuning

Matching The API routines for model tuning AXMT** are still valid, but the model tuning methods used are not the same as the ones used in the GUI. They are provided while API entry points for the methods used in the GUI are not available.

1.3 Importing

PVTSim files The PVTSim files import tool is now capable of importing CTM files. This will produce Multiflash files that define multiple streams. For each stream a composition is defined acording to the contents of the CTM file.

2 Introduction

Multiflash is Infochem/KBC's physical property software. It is a comprehensive and flexible system that returns the thermophysical properties of pure substances and mixtures and carries out phase and chemical equilibrium calculations for fluid and solid phases. Multiflash consists of a number of distinct software modules that may be used individually or together. These modules are: databanks; databank access software; pure component property evaluation; binary interaction parameter handling; mixture models for thermodynamic properties; mixture models for transport properties; phase equilibrium calculations; and chemical equilibrium calculations. This document describes the use of Multiflash from the point of view of the programmer incorporating part or all of the system in other software. It does not attempt to explain how to use Multiflash to solve problems in applied thermodynamics.

Multiflash is written in standard Fortran 77/95/2003 and all the example code shown in this document is Fortran. The descriptions of the argument lists of the various routines are intended to be sufficiently detailed to allow the programmer to set up and use the Multiflash software. In addition some example programs are included in Appendix . It should be noted that only the software interfaces and data structures described in this manual are supported. The use of any undocumented features may result in code that fails to work or that produces incorrect results. A summary and index to all the Multiflash routines in the interface is provided in Appendix . In all subroutines mentioned in this API, all the units for any input or output argument that requires them are SI units, unless otherwise stated. A brief description of new features in the current version of Multiflash is provided in Appendix .

Although this document describes the Fortran interface to Multiflash it is straightforward on many computer systems to make the equivalent calls from software written in other languages. Multiflash is available as a Win32 and 64 Dynamic Link Library (DLL). A Linux 32bit version is also available on request although that is not distributed by default. All the routines described in this document are exported from the Multiflash DLL and an import library is provided with the DLL. Additional information about how to call the Multiflash DLL from C++ or Visual Basic applications is provided in the Programmer's Guide for Multiflash C++/C Interface and the Programmer's Guide for the

5

Multiflash Visual Basic Interface. All Multiflash implementations can use the Multiflash command language which is described in the Multiflash Command Reference. Information contained in the Multiflash Command Reference and th User Guide for Models and Physical Properties may also be useful as background material for the programmer.

3 Preliminaries

3.1 Initialisation sequence

The Multiflash software is made up of a number of different modules that must be initialised in the correct order. This is:

- security system (see below);
- pure component data system (section and Appendix) ;
- binary interaction parameter data system (section);
- mixture models for thermodynamic and transport properties (sections , and);
- phase/chemical equilibrium calculations (sections to).

It is only necessary to follow the initialisation sequence up to the level at which properties or calculations are required. For example, if mixture viscosities are required it is necessary to initialise levels 1 to 4 before making a call to the viscosity routine but if pure component viscosities are required it is only necessary to initialise level 2. If any alteration is made to the initialisation at a later stage in running a program, then all steps in the initialisation sequence after the alteration must be repeated, e.g. if the binary interaction parameters are changed the models must be reinitialised.

3.2 Security system

The security system allows the Multiflash DLL to be used in accordance with KBC's commercial license agreement. It may operate in conjunction with a hardware device (KBC SLM dongle) or a license manager software system (KBC SLM). The license manager, RLM, from previous versions can stil be used with Multiflash 6.0. See the installation manual regarding licensing systems.

3.2.1 Initialising the security system

The security system must be initialised before Multiflash can access databanks, set up models or perform calculations. This is done by calling the routine MFSINI. The calling specification is:

```
      INTEGER, INTENT(OUT) :: IERR, IERRLS
      CHARACTER*(*), INTENT(IN) :: PSWORD
      CHARACTER*(*), INTENT(OUT) :: ERRMES

      SUBROUTINE MFSINI( PSWORD, IERR, IERRLS, ERRMES )
C
C     Initialise security system and check user and licence information.
C
C     Input arguments:
C     PSWORD   password - may be required depending on implementation.
C              Must be set to a blank string if no password is supplied.
C
C     Output arguments:
C     IERR     Multiflash error flag
C              0    OK
C              684  cannot initialise licensing system
C              685  consistency check failed
C              686  license expired
C              687  MF serial number not valid
C              688  version limit in license < MF version being used
C              689  other licensing system error
C              690  no more licenses available
C              691  server not running on network
C     IERRLS   error code returned by licensing system
C              0    OK
C              >0   error returned by licensing interface routine
C              <0   error returned by license system
C     ERRMES   error message returned by license system
```

3.2.2 Closing the security system

The security system should be closed when the calling application has finished using the Multiflash DLL . This will release any locked licenses. To close the security system call MFSCLO. The routine has no arguments. Calling MFEND (see p.) will also release any licenses.

3.3 Numerical parameters

A set of machine-dependent numerical parameters is used internally by Multiflash in numerical algorithms and to mark special values, e.g. ‘value undefined’. Only one value DLARGE is used as part of the Multiflash external programming interface. It is generally used to signal missing or undefined data values.

The machine-dependent numerical parameters must be initialised before Multiflash routines are used. This is usually achieved by calling any of the initialisation routines such as AYPINI.

5

Alternatively, the user may initialise the values directly by calling AYNUMI. The routine has no arguments.

To obtain the value of DLARGE call the utility routine AYNUMV.

```
SUBROUTINE AYNUMV( DLARGE )
REAL*8, INTENT(OUT) :: DLARGE
```

3.4 Include files

3.4.1 Numerical parameters

A number of dimensioning limits, identification codes, numerical constants and physical constants are used by Multiflash and may be required by the programmer. These values are provided in Fortran include files that contain parameters. The following is a list of include files that may be used by the programmer. Details of the parameters or variables are given in comments in the files.

axjnum.inc	Symbolic codes for Multiflash module, e.g. codes for variables, type of solution in fixed phase fraction flashes, etc.
axpara.inc	Dimensioning limits for Multiflash module
aygcon.inc	Gas constant
aybfid.inc	Identification codes for BIP temperature functions (Section)
ayjdbk.inc	Identification codes for databanks (Section)
ayjmid.inc	Identification codes for mixture models and options (Section)
ayjpid.inc	Identification codes for pure component properties (Appendix)
ayjuid.inc	Symbolic constants for status of pure component properties or BIPs.
ayjxid.inc	Symbolic constants for calculation options for AYMIX (Section)
aykcar.inc	CAS registry numbers for popular components
aymbco.inc	Maximum no. of BIP arrays that can be defined
aymbin.inc	Maximum no. of BIP load numbers that can be defined
aymlda.inc	Maximum no. of (integer) items of model identifiers/options
aymlno.inc	Maximum no. of model load numbers that can be defined
aymncs.inc	Maximum no. of components for Models module
aympd.inc	Limits for phase descriptors
aympdk.inc	Size of stored character-valued data items
ayptid.inc	Phase type identification codes
crpara.inc	Dimensioning limits for Chemreact module
unjerr.inc	Error codes for unit-conversion software
mfvbjn.inc	Codes used by the VB/VBA interface

3.4.2 Unique compound identifiers

Multiflash treats certain compounds in a special way. For example, the hydrates model (Appendix) needs to recognise water and hydrate-forming compounds. The file aykcar.inc contains a list of character constants that may be used to identify compounds in a unique way by comparing with the Chemical Abstracts Registry Number (CARN).

The character string parameters in aykcar.inc may be compared with values returned by AYPCNK (see section). For example the following code fragment searches for KWATER which has the CARN value for water.

```
      INCLUDE 'aymncs.inc'
      INCLUDE 'aykcar.inc'
      INCLUDE 'ayjpid.inc'
      INTEGER MXERR
      PARAMETER (MXERR=10)
      INTEGER NCO, IC(MAXNCS), NERR, IERR(MXERR), I, IWATER
      CHARACTER*12 CARN(MAXNCS)
C
C     Get chemical abstracts registry numbers
C
      CALL AYPCNK( NCO, IC, JKCARN, CARN, NERR, MXERR, IERR )
C
C     Find water in compound list
C
      IWATER = 0
      DO I = 1, NCO
        IF ( CARN(I).EQ.KWATER ) THEN
          IWATER = I
          EXIT
        ENDIF
      END DO
```

3.5 Error reporting

There are two types of error reporting systems used by Multiflash. The standard system that returns an error vector is used by all high-level external interfaces. A simplified system is used by some low-level utility routines such as the unit conversion routines described in section .

3.5.1 Standard error reporting system

The Multiflash error reporting system returns a vector of error codes that may trace the error condition back through a number of subroutines. Each subroutine returns unique identification numbers for any error condition or warning. A typical calling sequence might be:

```
C     Input arguments:
C     MXERR    maximum number of error codes to return in IERR
```

```
 5
C
C     Output arguments:
C     NERR     number of error codes returned in IERR
C     IERR     vector of error codes.
C
      INTEGER MXERR
      PARAMETER (MXERR=10)
      INTEGER NERR, IERR(MXERR)
      CALL AYZZZZ( NERR, MXERR, IERR )
```

3.5.2 Error codes

If no error condition was detected NERR will be returned as zero. If there were any errors or warnings NERR will be set to a positive number. A maximum of MXERR error codes may be returned in the array IERR. Positive values denote errors and negative values are warnings. NERR gives the total number of errors detected by the routine called and any lower level routines and may exceed MXERR. The first error code is returned in IERR(1) and the most recent in IERR(NERR) if NERR is less than or equal to MXERR. If NERR is equal to 0 any values in the IERR array have no significance and should be ignored.

The error codes in the standard system are unique. The meaning of each code is listed in error text files:

Module	Error codes	Error text file
(diagnostic codes)	0 - 200	
Command processor	201 - 9999	MFI.ETX
Models	10000 - 19999	MODELS.ETX
Multiflash	20000 - 29999	MULTIFLA.ETX
Chemreact	30000 - 39999	CREACT.ETX
Numerical methods	40000 - 49999	NUMETH.ETX

3.5.3 Error-handling utilities

The following utility routines are provided for handling the error codes.

ERTEST: testing for a particular error code

```
      LOGICAL FUNCTION ERTEST( NERR, MXERR, IERR, IERNO )

      INTEGER, INTENT(IN) :: MXERR, NERR, IERR(MXERR), IERNO
C
C     Tests a set of error numbers to see if any is equal to IERNO.
C
C     Input arguments:
C     NERR     Number of errors to be tested
C     MXERR    Dimension of array IERR
C     IERR     Array of error numbers to be tested
```

```
 5
C      IERNO    Error number to be tested for
C
C      Function value:
C      .TRUE. if error number found, otherwise .FALSE.
```

ERFATL: testing for a serious error This routine scans the error array IERR and checks for a serious error, i.e. a positive error code, as opposed to a negative warning code. If ERFATL is called with a value of NERR greater than MXERR it will return a result of .TRUE. irrespective of the error codes present because not all errors reported have been recorded.

```
      LOGICAL FUNCTION ERFATL( NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NERR, MXERR, IERR(MXERR)
C
C      Tests a set of error numbers to see if any is a fatal error.
C
C      Input arguments:
C      NERR     Number of errors to be tested
C      MXERR    Dimension of array IERR
C      IERR     Array of error numbers to be tested
C
C      Function value:
C      .TRUE. if serious error found, otherwise .FALSE.
```

ERLAST: getting the last error

```
      SUBROUTINE ERLAST( NERR, MXERR, IERR, NERR0, LASTER, FATAL )

      INTEGER, INTENT(IN) :: NERR, MXERR, IERR(MXERR)
      INTEGER, INTENT(OUT) :: NERR0, LASTER
      LOGICAL, INTENT(OUT) :: FATAL
C
C      Returns the last error code stored in IERR.
C
C      Input arguments:
C      NERR     Number of errors to be tested
C      MXERR    Dimension of array IERR
C      IERR     Array of error numbers to be tested
C
C      Output arguments:
C      NERR0    number of error flags stored in IERR
C      LASTER   most recent error code in IERR. Zero if none.
C      FATAL    .TRUE. if most recent code was positive (serios error),
C               otherwise .FALSE.
```

MFDEMS: getting the error message for a given error number

```
      INTEGER FUNCTION MFDEMS( IERNO, EMS )
```

```
5
      INTEGER, INTENT(IN) :: IERNO
      CHARACTER*80, INTENT(OUT) :: EMS
C
C     Returns an error message associated with an error number.
C
C     Input arguments:
C     IERNO    Error number
C
C     Output values returned:
C     0        OK
C     1        error getting message - see EMS for details
C
C     Output arguments:
C     EMS      Error message
```

3.6 Multiflash version

The MFVERS routine returns the software version number and modification date for Multiflash. It may be called at any time and does not require that any initialisation has been performed. The calling specification is as follows:

```
      SUBROUTINE MFVERS( VERSN, DATE )

      CHARACTER*(*), INTENT(OUT) :: VERSN, DATE
C
C     Output arguments:
C     VERSN    version number (character string)
C     DATE     revision date (character string)
```

To retrieve all the information the VERSN string should be at least 8 characters long and the DATE string should be at least 20 characters long.

4 Defining pure components and data sources

4.1 Initialisation sequence

The pure component data section of the software must be initialised before any other. Typically, the following initialisation sequence would be followed:

- Initialise the pure component data software.
- Set location of standard databank files (optional).
- Open pure component data banks - if any.
- Load pure component data for particular components.
- Retrieve pure component data as required.
- Unload databank if not necessary. (optional but recommended)

Steps 1 - 5 are described in this section whilst the procedures for retrieving data are detailed in section starting on p..

4.2 Initialisation of the pure component data software

The pure component data software is initialised by calling AYPINI. This erases all loaded components if any. The calling specification is:

```
      SUBROUTINE AYPINI( NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(OUT):: NERR, IERR(*)
C
C     Input arguments:
C     MXERR    maximum number of error codes to return in IERR
C
```

```
 5
C      Output arguments:
C      NERR     number of error codes returned in IERR
C      IERR     error codes. Values for this routine:
C               10001  - 10004  licensing errors.
```

4.3 Databank files used by Multiflash

Multiflash has interfaces to a number of databanks. These include the internal INFODATA and INFOCOND as well as DIPPR databanks. A separate license is needed for the DIPPR databank. The standard file names used for the Infochem and DIPPR databanks are listed in the following table:

infodata.bin	Infochem pure component databank for fluids
infodata.idx	component names index for Infodata
infobips.bin	Infochem VLE BIP databank
infobips.idx	Index for Infochem VLE BIP databank
infollbips.bin	Infochem LLE BIP databank
infollbips.idx	Index for Infochem LLE BIP databank
infocond.bin	Infochem databank for condensed components
infocond.idx	component names index for Infocond
dippr.bin	DIPPR databank
dippr.idx	DIPPR databank names index

For convenience, the location of standard data files which Multiflash uses may be set by calling AYFLFL (see below). It is not necessary to set the location in this way but if AYFLFL is not called the programmer is responsible for supplying complete file access strings to the databank opening routines described in section .

The calling specification for AYFLFL is:

```
      SUBROUTINE AYFLFL( APPDIR )

      CHARACTER*(*), INTENT(IN) :: APPDIR
C
C      Input arguments:
C      APPDIR   complete pathname (directory) for location of
C               Multiflash files.
```

4.4 Opening pure component databanks

Before information stored on a databank can be accessed the databank must be opened. The routine AYPDBO is used to open all databanks for Multiflash. The databanks may contain pure component data for defined components, binary interaction parameters (BIPs) or data for the properties of petroleum

fractions. The ‘databanks’ may be external data files, data stored internally within the code or collections of methods or correlations that are used to estimate the required data. This section deals only with data banks for defined pure components. For petroleum fraction correlations see section and for BIP databanks see section . The calling specification for AYPDBO is:

```
      SUBROUTINE AYPDBO( NBANK, JBANK, NFIL, NAMFIL, LNODB, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NFIL, NBANK, JBANK(*), MXERR
      INTEGER, INTENT(OUT):: LNODB, NERR, IERR(*)
      CHARACTER*(*), INTENT(IN) :: NAMFIL(*)

C
C     Input arguments:
C     NBANK    Dimension of the array JBANK.
C     JBANK    code for databank to be opened or for databank structure.
C              If NFIL is zero JBANK refers to a particular databank
C              that is known to Multiflash. Otherwise it is used to
C              identify the file structure.
C              The available codes for JBANK are listed in the include
C              file AYJDBK.INC.
C     NFIL     Number of filenames passed in NAMFIL. If NFIL is set
C              to zero the default filenames will be used for each
C              databank. To open a databank with the same structure as
C              one of the standard databanks it is necessary to set NFIL
C              and NAMFIL correctly (see below).
C     NAMFIL   Array of NFIL file names. The following information
C              is required for each databank structure:
C              structure    JBANK    NFIL     NAMFIL
C              INFODATA     JDOINF   2       1) index file
C                                            2) databank file
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     LNODB    Data bank load number (used to refer to databank in
C              all subsequent operations)
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10401  Unrecognised data bank
C              10402  Run out of data bank load numbers
C              10403  Run out of space in data bank load information array
C              10404  Data files not appropriate for this data source
C              10405  Data files required for this data source
```

The databank identity or type of databank structure is specified by the argument JBANK which must be one of the parameters in the file ayjdbk.inc. The standard databanks are opened by setting NFIL to zero. In this case the location of the files used is determined by the databank path set by calling AYFLFL. To open other databanks with one of the standard structures the appropriate databank structure code (JDOINF or JDOINC) must be set the required file names must be specified. The databank load number returned by AYPDBO must be stored and passed without alteration to any routine that subsequently accesses the databank. For an example of using AYPDBO see section .

4.5 Unloading databanks

Although there is no logical limit to the number of databanks that may be opened simultaneously by Multiflash, there is a limit to the internal storage allocated to databank information. Unloading (closing) a databank frees the internal storage associated with an open bank. The databank load number is then no longer valid.

The normal pattern for using databanks is to open them, load data for selected compounds (section) and then to unload the databank(s). However, if a databank is to be accessed frequently then unloading is not usually desirable because there is an overhead associated with re-opening the bank.

The routine AYPDBU is used to unload pure component data banks. The calling specification is:

```
      SUBROUTINE AYPDBU( LNODB )

      INTEGER, INTENT(INOUT):: LNODB
C
C     Input/output arguments:
C     LNODB    Data bank load number (set to zero on return)
C
```

If AYPDBU is called with an invalid load number no action is taken and no error is reported.

4.6 Loading pure component data

Components used by Multiflash can be defined in a number of ways:

- data may be loaded from a databank (this is the usual method);
- the data can be directly loaded by making subroutine calls (section);
- the component may be specified as a petroleum fraction (section);
- or a set of components may be defined by entering a petroleum fluid analysis (section).

Components may be loaded in any order and each component may be loaded from a distinct databank or data source. All data loading subroutines return a component load number for each component loaded. This number is used subsequently by all other Multiflash routines that need to refer to a particular component. The maximum number of components that may be loaded simultaneously is set by the parameter MAXNCS in file aymncs.inc.

4.7 Loading components from a databank

Pure component data for the required components may be loaded from any (previously opened) databanks. Components are identified by name. A valid name, stored on the databank, must be specified to the loading routine. Note that there may be several alternative names for any compound. The valid names depend on the databank that is specified and it is necessary to consult the documentation provided with the databank to establish the compounds that are available. A set of routines is provided to facilitate searching for compounds on a databank. These routines are described in section .

The routine AYPDBC is used to load a compound. It has the following calling specification:

```
      SUBROUTINE AYPDBC( LNODB, CNAME, ICO, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNODB, MXERR
      INTEGER, INTENT(INOUT) :: ICO
      INTEGER, INTENT(OUT) :: NERR,  IERR(*)
      CHARACTER*(*), INTENT(IN) :: CNAME
C
C     Input arguments:
C     LNODB    Data bank load number
C     CNAME    component name (must be a valid name for the databank)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     ICO      Component load number. If equal to 0 a new component
C              is set up. If an existing component load number is
C              given the entries for that component will be overwritten.
C
C     Output values:
C     ICO      New component load number created (if zero entered)
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10421   unrecognised data bank load number
C              10422   unrecognised data bank type
C              10423   component not loaded (see subsequent error codes)
```

If the required compound is not found or not loaded error information is returned in IERR from lower level routines. The loading routine attempts to find the compound in the names index for the databank and, if successful, attempts to read stored data from the databank itself. A cross-check is performed to ensure that the compound name in the databank is consistent with the entry in the databank index. If the cross-check fails this usually means that the databank has the wrong file structure or that the index and databank are out of step because of version changes.

Example:

```
      INTEGER MXERR
      PARAMETER (MXERR=10)
```

```
5
      INTEGER JBANK(1), LNODB1, LNODB2, ID, LNOC1, LNOC2,
     +        NERR, IERR(MXERR)
      CHARACTER*1 KDUM(1)
C
C     Data bank i.d. codes:
C
      INCLUDE 'ayjdbk.inc'
C
      CALL AYPINI( NERR, MXERR, IERR )
C
C     Open INFODATA data bank:
C
      JBANK(1) = JDINFO
      CALL AYPDBO( 1, JBANK, 0, KDUM, LNODB1, NERR, MXERR,
     +             IERR )
C
C     Open DIPPR data bank:
C
      JBANK(1) = JDIPPR
      CALL AYPDBO( 1, JBANK, 0, KDUM, LNODB2, NERR, MXERR,
     +             IERR )
C
C     Component with load number LNOC1 is methane loaded from INFODATA
C
      LNOC1 = 0
      CALL AYPDBC( LNODB1, 'METHANE', LNOC1, NERR, MXERR, IERR )
C
C     Component with load number LNOC2 is ethane loaded from DIPPR
C
      LNOC2 = 0
      CALL AYPDBC( LNODB2, 'ETHANE', LNOC2, NERR, MXERR, IERR )
```

4.8 Defining a component by entering data directly

Compound data may be entered directly by making subroutine calls as described below. An entirely new compound may be set up or, if required, additional pure component physical properties may be entered or the existing values overwritten for a compound that has already been loaded. Three classes of data may be entered: numerical constants, character constants and coefficients in a temperature-dependent correlation.

4.8.1 Entering numerical constants

The numerical constants are the quantities listed in section and Appendix , e.g. critical temperature. They are set by calling AYPSTR as follows:

```
      SUBROUTINE AYPSTR( ICO, INIT, NVALUE, JVALUE, RCONST, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) ::  NVALUE, JVALUE(*), MXERR
```

```
5
      INTEGER, INTENT(INOUT) :: ICO
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: RCONST(*)
      LOGICAL, INTENT(IN) :: INIT

C
C     Input arguments:
C     INIT     .TRUE. if all properties to be initialised, .FALSE. if
C              only undefined properties to be intialised
C     NVALUE   Number of properties to be entered
C     JVALUE   Property identification codes (from ayjpid.inc)
C     RCONST   Pure component constants. -DLARGE means don't overwrite
C              existing value.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     ICO      Component load number (0=set up new component)
C
C     Output values:
C     ICO      Component load number
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              -13271  the JVALUE is not a recognised property
C              13271   error in component number (<1 or > maxncs)
C              13272   component storage exhausted
```

If the component load number ICO is zero a new component is set up and its load number is returned otherwise properties are stored for an existing component. The array of property codes JVALUE should be set using the parameter values listed in Appendix and defined in file ayjpid.inc. Setting a property value in RCONST to -DLARGE (from routine AYNUMV) means the existing value for that property is not to be overwritten. Unless otherwise stated in the units are always SI units.

The following example shows how to set the critical temperature and pressure for a component that is already loaded with load number LNOC3:

```
INCLUDE 'ayjpid.inc'
INTEGER MAXVAL, MXERR
PARAMETER ( MAXVAL=2, MXERR=10 )
INTEGER LNOC3, NVALUE, JVALUE(MAXVAL), NERR, IERR(MXERR)
REAL*8 RCONST(MAXVAL)
JVALUE(1) = JTCRIT
RCONST(1) = 513.9D0
JVALUE(2) = JPCRIT
RCONST(2) = 61.4D5
CALL AYPSTR( LNOC3, .FALSE., 2, JVALUE, RCONST, NERR, MXERR, IERR)
```

4.8.2 Setting numerical constants to undefined

When AYPSTR is called with one of the input property values set to undefined (-DLARGE), the input is ignored and the internal value is left unaltered. Sometimes there is a need to set an internal value to undefined. This can be done by calling AYPUND which sets all stored values specified in the input list JVALUE to undefined. The definition of AYPUND is:

```
      SUBROUTINE AYPUND( ICO, NVALUE, JVALUE, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) ::  NVALUE, JVALUE(*), MXERR
      INTEGER, INTENT(INOUT) :: ICO
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
C
C     Input arguments:
C     NVALUE   Number of properties to be set undefined
C     JVALUE   Property identification codes (from AYJPID.INC)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     ICO      Component load number (0=set up new component)
C
C     Output values:
C     ICO      Component load number
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              -13271  the JVALUE is not a recognised property
C              13271   error in component number (<1 or > maxncs)
C              13272   component storage exhausted
```

4.8.3 Entering character constants

The character constants are the quantities listed in section and Appendix , e.g. compound name. They are set by calling AYPSTK as follows:

```
      SUBROUTINE AYPSTK( ICO, INIT, NVALUE, JVALUE, KCONST, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NVALUE, JVALUE(*), MXERR
      INTEGER, INTENT(INOUT) :: ICO
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      CHARACTER*(*) KCONST(*)
      LOGICAL INIT

C
C     Input arguments:
C     INIT     .TRUE. if all properties to be initialised, .FALSE. if
C              only undefined properties to be intialised
C     NVALUE   Number of properties to be entered
C     JVALUE   Property identification codes (from ayjpid.inc)
```

```
 5
C      KCONST   Pure component constants. Blank means don't overwrite
C               existing value.
C      MXERR    maximum number of error flags to be returned in IERR
C
C      Input/output arguments:
C      Input values:
C      ICO      Component load number (0=set up new component)
C
C      Output values:
C      ICO      Component load number
C
C      Output arguments:
C      NERR     no. errors reported by this routine and lower level routines
C      IERR     Array of error codes. Values returned by this routine:
C               -13281  the JVALUE is not a recognised property
C               13281   error in component number (<1 or > maxncs)
C               13282   component storage exhausted
```

If the component load number ICO is zero a new component is set up and its load number is returned otherwise properties are stored for an existing component. The array of property codes JVALUE should be set using the parameter values listed in Appendix and defined in file ayjpid.inc. Setting a property value in KCONST to a blank means the existing value for that property is not to be overwritten.

For example, to set the name and formula for a component that is already loaded with load number LNOC3 to ethanol and C_2H_6O , the call of AYPSTK should be:

```
INTEGER MXERR
PARAMETER ( MXERR=10 )
INTEGER JVALUE(2), LNOC3, NERR, IERR(MXERR)
CHARACTER*7 KCONST(2)
LOGICAL INIT
INIT = .TRUE.
JVALUE(1) = JKNAME
JVALUE(2) = JKFORM
KCONST(1) = 'ETHANOL'
KCONST(2) = 'C2H6O'
CALL AYPSTK( LNOC3, INIT, 2, JVALUE, KCONST, NERR, MXERR, IERR )
```

4.8.4 Entering correlations for temperature-dependent properties

The temperature-dependent properties are listed in section and Appendix , e.g. vapour pressure. Temperature correlations are regarded as special cases of numerical constants and are also set by calling AYPSTR as specified above. The arrays JVALUE and RCONST are used to set up a data structure as follows:

element	JVALUE	RCONST
1	jval	equation type code
2	jval + 1	coefficient 1
3	jval + 2	coefficient 2
...	...	...

5

1 + ncof	jval + ncof	coefficient ncof
1 + ncof + 1	jval + ncof + 1	T_{min}
1 + ncof + 2	jval + ncof + 2	T_{max}

The property code jval should be set to the code number from file ayjpid.inc. The number of correlation coefficients ncof is equation-dependent. Details of the correlating equations used are given in Appendix .

For example, consider the extended Antoine equation which has the following form (see Appendix):

$$p = a_1 + \frac{a_2}{T + a_3} + a_4 T^{a_6} + a_5 T + a_7 / T^2$$

The property code is vapour pressure, JPSAT. The Antoine equation is vapour pressure equation type 2 which is specified by setting the coefficient corresponding to JPSAT to DBLE(2). Note: a REAL*8 number is required. The nine coefficients a_1 - a_7 , T_{min} and T_{max} are denoted by JPSAT+1 to JPSAT+9. To define an Antoine equation for a component that is already loaded with load number LNOC3 with $a_1 = 23.5$, $a_2 = 2537.2$, $a_3 = -11.3$, $a_4 = a_5 = 0$, $a_6 = 1.0$, $a_7 = 0$, $T_{min} = 275.0$ and $T_{max} = 420.0$, AYPSTR should be called as follows:

```
INTEGER MXERR
PARAMETER ( MXERR=10 )
INTEGER LNOC3, JVALUE(10), NERR, IERR(MXERR)
REAL*8 RCONST(10)
LOGICAL INIT
INIT = .TRUE.
JVALUE(1) = JPSAT
JVALUE(2) = JPSAT + 1
JVALUE(3) = JPSAT + 2
JVALUE(4) = JPSAT + 3
JVALUE(5) = JPSAT + 4
JVALUE(6) = JPSAT + 5
JVALUE(7) = JPSAT + 6
JVALUE(8) = JPSAT + 7
JVALUE(9) = JPSAT + 8
JVALUE(10) = JPSAT + 9
RCONST(1) = DBLE( 2 )
RCONST(2) = 23.5D0
RCONST(3) = 2537.2D0
RCONST(4) = -11.3D0
RCONST(5) = 0.0D0
RCONST(6) = 0.0D0
RCONST(7) = 1.0D0
RCONST(8) = 0.0D0
RCONST(9) = 275.0D0
RCONST(10) = 420.0D0
CALL AYPSTR( LNOC3, INIT, 10, JVALUE, RCONST, NERR, MXERR, IERR )
```

4.8.5 Entering Unifac subgroup structures

A component's subgroup structure is defined by property code JUNIF is then followed by up to 10 pairs of numbers giving the Multiflash code number for the subgroup in question followed by its abundance. The following example shows how to define the Unifac subgroup structure for hexane. Hexane consists of two CH3 subgroups and four CH2 subgroups. The Multiflash intger code numbers for the CH3 and CH2 subgroups are 1 and 2 respectively.

```
INTEGER MXERR
PARAMETER ( MXERR=10 )
INTEGER LNOC3, JVALUE(5), NERR, IERR(MXERR)
REAL*8 RCONST(5)
LOGICAL INIT
INIT = .TRUE.
JVALUE(1) = JUNIF
JVALUE(2) = JUNIF + 1
JVALUE(3) = JUNIF + 2
JVALUE(4) = JUNIF + 3
JVALUE(5) = JUNIF + 4
RCONST(1) = 0.0D0
RCONST(2) = DBLE( 1 )
RCONST(3) = DBLE( 2 )
RCONST(4) = DBLE( 2 )
RCONST(5) = DBLE( 4 )
CALL AYPSTR( LNOC3, INIT, 5, JVALUE, RCONST, NERR, MXERR, IERR )
```

All the integer code numbers and abundances have to be converted to double precision (REAL*8) before entering into AYPSTR. There is a useful utility AYPUFS which can return the Multiflash code number corresponding to the character-string name of the subgroup, or vice-versa. All current subgroups names are given in the Multiflash Command Reference. The call specification is:

```
      SUBROUTINE AYPUFS( MODE, SYMBOL, ISGNO )

      INTEGER, INTENT(INOUT) :: ISGNO
      CHARACTER*12, INTENT(INOUT) :: SYMBOL
      LOGICAL, INTENT(IN) :: MODE
C
C     Input arguments:
C     MODE     .TRUE. if SYMBOL is input and ISGNO is output,
C              .FALSE. if vice-versa
C
C     Input/output arguments:
C     SYMBOL   Symbol
C     ISGNO    Subgroup number
```

4.9 Petroleum fractions

Petroleum fractions or pseudo components are commonly used in modelling oil and gas processing operations. A petroleum fraction

may be used to represent the aggregate properties of complex mixtures that are characterised by standard tests such as ASTM distillation.

4.9.1 Correlations for petroleum fraction properties

Multiflash uses industry-standard correlations to estimate all the properties required to use equations of state and basic transport property models for mixtures that include petroleum fractions. These properties include: molecular weight, boiling point, critical temperature, critical pressure, critical volume, parachor, dipole moment, enthalpy of formation, standard entropy, liquid and solid solubility parameters, melting temperature, enthalpy change on melting, volume change on melting, heat capacity change on melting, triple point temperature and pressure, perfect gas Cp, saturated liquid density, latent heat, vapour pressure, liquid and vapour viscosity, liquid and vapour thermal conductivity and liquid heat capacity.

Before calling the routine AYPPF (see below) to define a petroleum fraction it is necessary to define the characterisation method (set of correlations) to use. This is done by calling AYPDBO (see) to obtain a load number for the characterisation method. At present only one characterisation method JCINFO is supported but other methods may be added in the future. This method uses the correlations of Riazi and Al-Sahhaf [1]. for molecular weight and specific gravity as a function of carbon number and the Lee-Kesler correlations [2] to obtain critical properties and the acentric factor.

The characterisation method load number returned by AYPDBO must be stored and passed without alteration to any routine (e.g. AYPPF) that subsequently uses the method.

4.9.2 Defining petroleum fractions

Petroleum fractions are defined by calling the routine AYPPF. This routine estimates any necessary properties and loads the component. It is necessary to supply some basic information which is used as the basis of estimation procedures. Up to seven property values may be entered for a petroleum fraction. Any user-supplied property values are used in preference to estimated values. As much data as possible should be entered but the following combinations are the minimum acceptable:

- Carbon number molecular weight and specific gravity (at 60 ºF);
- Carbon number and molecular weight or specific gravity boiling point and specific gravity
- boiling point and molecular weight critical temperature, critical pressure and acentric factor

1M. R. Riazi and T. A. Al-Sahhaf, Physical properties of heavy petroleum fractions and crude oils, Fluid Phase Equilibria, 117, 217 (1996)
2M.J. Kesler and B.I. Lee, Hydrocarbon Processing, 53, 153 (1976)

The physical properties estimated for petroleum fraction components may depend on the type of component which may be a petroleum fraction of unspecified type, normal-paraffin, non-n-paraffin, iso-paraffin, aromatic, resin or asphaltene. Currently the Infochem characterisation method does not distinguish between normal, iso- or aromatic components. The formula, Family Code and Chemical Abstracts Registry Number for a petroleum fraction are set to blank strings.

The calling specification of AYPPF is:

```
      SUBROUTINE AYPPF( LNOPF, ICO, INIT, NAME, ITYPE, CNN, AMW, SG,
     +                  TB, TC, PC, W, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOPF, ITYPE, MXERR
      INTEGER, INTENT(INOUT) ::ICO
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(INOUT) :: CNN, AMW, SG, TB, TC, PC, W
      CHARACTER*(*), INTENT(IN) :: NAME
      LOGICAL, INTENT(IN) :: INIT

C
C     Input arguments:
C     LNOPF    load number for characterisation method.
C              Obtained by calling aypdbo.
C     INIT     .TRUE. if all properties to be initialised, .FALSE. if
C              only undefined properties to be intialised
C     NAME     component name
C     ITYPE    type of petroleum fraction component
C              (parameters from ayjxid.inc)
C              The estimated physical properties will, in general,
C              depend on ITYPE but not all characterisation methods
C              distinguish between the type of component.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     ICO      Component load number (0=set up new component)
C     CNN      carbon number of petroleum fraction
C     AMW      molecular weight (g/mol) (relative molar mass)
C     SG       specific gravity at 60degF (relative density at 288.71K)
C     TB       boiling point (K)
C     TC       critical temperature (K)
C     PC       critical pressure (Pa)
C     W        acentric factor
C              If any of the arguments CNN to W is set to -DLARGE it is
estimated
C
C     Output values:
C     ICO      Component load number
C     CNN      carbon number of petroleum fraction
C     AMW      molecular weight (g/mol) (relative molar mass)
C     SG       specific gravity at 60degF (relative density at 288.71K)
C     TB       boiling point (K)
C     TC       critical temperature (K)
C     PC       critical pressure (Pa)
C     W        acentric factor
C
C     Output arguments:
```

```
 5
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes.
```

For example, the code below defines a new component called 'C7+' as a petroleum fraction of unspecified type with molecular weight of 224, specific gravity of 0.842 and boiling point of 552.59K. The critical temperature, critical pressure and acentric factor are unknown and will be estimated (along with other properties) using the Infochem characterisation method. The load number for this component is LNOC4

```
      INCLUDE 'ayjxid.inc'
      INTEGER MXERR
      PARAMETER(MXERR=10)
      REAL*8 DLARGE, CN, MW, SG, TB, TC, PC, W
      LOGICAL INIT
      INTEGER JBANK(1), LNOPF, LNOC4, NERR, IERR(MXERR)
      CHARACTER*1 NAMFIL(1)

      CALL AYNUMV( DLARGE )
      JBANK(1) = JCINFO
      CALL AYPDBO( 1, JBANK, 0, NAMFIL, LNOPF, NERR, MXERR, IERR )
      INIT = .TRUE.
      MW = 224.0D0
      SG = 0.842D0
      TB = 552.59D0
      TC = -DLARGE
      PC = -DLARGE
      W  = -DLARGE
      CN  = -DLARGE
      LNOC4 = 0
      CALL AYPPF( LNOPF, LNOC4, INIT, 'C7+', JTPET, CN, MW, SG, TB,
     +            TC, PC, W, NERR, MXERR, IERR )
```

4.10 Searching for compounds on a databank

The routines described in this section are provided to allow databanks previously opened by AYPDBO (see Section) to be searched in a number of ways. These routines can be used to support on-line searching facilities for users of databanks interfaced to Multiflash. Most of the routines return a databank record number corresponding to compound identified in the search. The record number can be used in calls to routines described in Section to look up data without loading a component from the databank.

4.10.1 AYPDBI: Search for a compound name:

The integer function subprogram AYPDBI searches the index associated with a databank for a unique compound name and returns the record number of the compound in the databank file if the name is identified. The calling specification is:

```
5
      INTEGER FUNCTION AYPDBI( LNODB, CNAME, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNODB, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      CHARACTER*(*), INTENT(IN) :: CNAME
C
C     Input arguments:
C     LNODB    Data bank load number
C     CNAME    component name
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Function return values:
C     0        cannot find compound or error (see IERR)
C     >0       record number on databank file for component
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10431   Unrecognised data bank load number
C              10432   Unrecognised data bank type
```

4.10.2 AYPDBS: Search for compound names containing specified string:

AYPDBS searches the index of compound names associated with a databank for names containing a specified string. One name is returned per call. The first call for a new name should be made with the logical flag FIRST set to .TRUE.. Subsequent calls with FIRST set to .FALSE. will return the next name matching the search criterion until the end of file is reached. The "*" character is used to specify a sub-string that may occur at the start, end or anywhere in the name. For example, to find all compounds starting with "METHANE", the string to search for should be set to "METHANE*". If the search string has the value ** it matches any name and the effect is to return the next name on the databank index. The calling specification is:

```
      SUBROUTINE AYPDBS( LNODB, KSTRNG, FIRST, IREC, KNAME, NERR, MXERR,
     +                   IERR )

      INTEGER, INTENT(IN) :: LNODB, MXERR
      INTEGER, INTENT(OUT) :: IREC, NERR, IERR(*)
      CHARACTER*(*), INTENT(IN) :: KSTRNG
      CHARACTER*(*), INTENT(OUT) :: KNAME
      LOGICAL, INTENT(IN) :: FIRST

C
C     Input arguments:
C     LNODB    Data bank load number
C     KSTRNG   string to search for:
C              STRING     exact match on STRING
C              *STRING    any name ending in STRING
C              STRING*    any name starting with STRING
C              *STRING*   any name containing STRING
C              **         the next name in the index
```

```
 5
C     FIRST    logical flag.
C              TRUE   first call for current string, index file is
C                     rewound prior to search
C              FALSE  for second and subsequent calls, starts search at
C                     current position of index file
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     IREC     record number in databank file of compound matching
C              search criterion.
C              Zero if no match found.
C              Negative if end of index file reached.
C     KNAME    component name matching search criterion. Blank if no
C              match found.
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10471   Unrecognised data bank load number
C              10472   Unrecognised data bank type
```

4.10.3 AYPDBY: Return synonyms for compound name:

AYPDBY searches the index of compound names associated with a databank for a specified name and returns synonyms, i.e. alternative names for the compound. One synonym is returned per call. The first call for a new name should be made with the logical flag FIRST set to .TRUE.. Subsequent calls with FIRST set to .FALSE. will return the next synonym until the final synonym is returned. The calling specification is:

```
      SUBROUTINE AYPDBY( LNODB, KNAME, FIRST, IREC, KSYNM, NERR, MXERR,
     +                   IERR )

      INTEGER, INTENT(IN) :: LNODB, MXERR
      INTEGER, INTENT(OUT) ::  IREC, NERR, IERR(*)
      CHARACTER*(*), INTENT(IN) :: KNAME
      CHARACTER*(*), INTENT(OUT):: KSYNM
      LOGICAL, INTENT(IN) :: FIRST
C
C     Input arguments:
C     LNODB    Data bank load number
C     KNAME    component name for which to find synonyms
C     FIRST    logical flag.
C              TRUE   first call for current string, index file is
C                     rewound prior to search
C              FALSE  for second and subsequent calls, starts search at
C                     current position of index file
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     IREC     record number in databank file if compound name found.
C              Zero if name not found.
C              Negative if last synonym has been returned.
C     KSYNM    synonym for component name (will include original
C              name specified)
C     NERR     no. errors reported by this routine and lower level routines
```

```
 5
C     IERR     array of error codes. Values returned by this routine:
C              10481   Unrecognised data bank load number
C              10482   Unrecognised data bank type
```

4.10.4 AYPDBF: Search for compounds with specified formula:

AYPDBF searches the databank for compounds with a specified chemical formula. One compound name is returned per call. The first call for a new formula should be made with the logical flag FIRST set to .TRUE.. Subsequent calls with FIRST set to .FALSE. will return the next name with the formula until the final name is returned. The formula must be entered in the correct upper- and lower-case characters. The symbol "*" can be used to specify any number (non-zero) of atoms of a particular element. The calling specification is:

```
      SUBROUTINE AYPDBF( LNODB, SFORM, FIRST, IREC, NAME, FORM, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: LNODB, MXERR
      INTEGER, INTENT(OUT) :: IREC, NERR, IERR(*)
      CHARACTER*(*), INTENT(IN) :: SFORM
      CHARACTER*(*), INTENT(OUT) :: NAME, FORM
      LOGICAL, INTENT(IN) :: FIRST
C
C     Input arguments:
C     LNODB    Data bank load number
C     SFORM    chemical formula to search for.
C              The formula should contain the standard chemical
C              symbols in upper/lower case. Abundances must be integer
C              values or * for any number of atoms. Brackets and other
C              punctuation are not allowed. An abundance of 1 need not
C              be specified.
C     FIRST    logical flag.
C              TRUE   first call for current formula, databank position
C                     is reset prior to search
C              FALSE  for second and subsequent calls, starts search at
C                     current position of databank
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     IREC     record number in databank file if a matching formula
C              found. Zero if formula not found.
C              Negative if end of file reached.
C     NAME     component name found (if irecdb is positive)
C     FORM     chemical formula found (if irecdb is positive)
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10491   Unrecognised data bank load number
C              10492   Unrecognised data bank type
```

4.11 Reading data directly from a databank

The routines described in this section are provided to return information from a databank without first loading the compound data into Multiflash as described in Section . The databanks referenced must have previously been opened by AYPDBO (see Section). These routines can be used to support on-line component selection on the basis of component properties.

4.11.1 AYPDBK: Pure component character data:

AYPDBK returns character constants from a specified record on a databank. Its function is similar to the routine AYPCNK which can only be used once a component has been loaded. See Appendix for a list of property codes. The calling specification is:

```
      SUBROUTINE AYPDBK( LNODB, IREC, JVALUE, KCONST, NERR, MXERR, IERR)

      INTEGER, INTENT(IN) :: LNODB, JVALUE, MXERR
      INTEGER, INTENT(INOUT) :: IREC
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      CHARACTER*(*), INTENT(OUT) ::  KCONST

C
C     Input arguments:
C     LNODB    Data bank load number
C     JVALUE   Property identification code (from ayjpid.inc)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     IREC     record number of desired data
C
C     Output values:
C     IREC     record number for next component
C
C     Output arguments:
C     KCONST   pure component character constant (blank if error)
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10451   Unrecognised data bank load number
C              10452   Unrecognised data bank type
```

4.11.2 AYPDBV: Pure component real-valued constant data:

AYPDBV returns real-valued constants from a specified record on a databank. Its function is similar to the routine AYPCNR which can only be used once a component has been loaded. See Appendix for a list of property codes. Note that only constant property values can be returned e.g. critical properties and not

coefficients in temperature-dependent correlations. The calling specification is:

```
      SUBROUTINE AYPDBV( LNODB, IREC, JVALUE, RCONST, NERR, MXERR, IERR)

      INTEGER, INTENT(IN) :: LNODB, JVALUE, MXERR
      INTEGER, INTENT(INOUT) :: IREC
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(OUT) :: RCONST

C
C     Input arguments:
C     LNODB    Data bank load number
C     JVALUE   Property identification code (from ayjpid.inc)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     IREC     record number of desired data
C
C     Output values:
C     IREC     record number for next component
C
C     Output arguments:
C     RCONST   pure component character constant
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10461   Unrecognised data bank load number
C              10462   Unrecognised data bank type
```

4.12 Other utilities

The following utility routines are provided:

4.12.1 AYPNC1: Number of components defined:

```
      SUBROUTINE AYPNC1( NC, IC )
      INTEGER, INTENT(OUT) :: NC, IC(*)
C
C     Output arguments:
C     NC       number of defined components
C     IC       load numbers of defined components
C              Ic has to be dimensioned at least MAXNCS to be safe.
```

4.12.2 AYPDEL: Delete data for a component:

```
      SUBROUTINE AYPDEL( ICO, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(INOUT) :: ICO
```

```
5
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
C
C     Input arguments:
C     MXERR    maximum number of error codes to return in IERROR
C
C     Input/output arguments:
C     Input values:
C     ICO      Component load number
C
C     Output values:
C     ICO      Component load number (0=deleted)
C
C     Output arguments:
C     NERR     number of error codes returned in IERROR
C     IERR     error codes. Values for this routine:
C              13291  Erroneous component number
C              13292  Maximum number of components has already been entered
```

AYPDEL frees the storage associated with the load number and makes it available for loading new components. However, the recomended routine to call when deleting components is AYBDEL as it also deletes BIP references for that component.

4.12.3 AYSALN: Equivalent salt component/ions concentration:

The purpose of this routine is to take a multicomponent ion analysis, salt analysis or total deposited solids in water and to find the concentration of sodium chloride, potassium chloride and calcium chloride which will have an equivalent effect on the activity of the water. This allows the effect of salt solutions on the hydrate formation point or the freezing point depression to be calculated.

Three electrolyte models are available. The first uses the concept of a salt pseudocomponent. The component called SALTCOMPONENT is included in the Infodata databank. The SALTCOMPONENT is intented to use with RKSA and PRA using van der Waals and Infochem mixing rules. To calculate the equivalent amount of the SALTCOMPONENT the MODEL argument should be set to 1. The usage of the SALTCOMPONENT option is not recomended if calculations are to be carried out at high pressures and or high temperatures.

The other two make the Infochem electrolyte model () handling sodium, potassium, calcium and chloride ions. Setting the MODEL argument to 2 will generate the equivalent concentrations of NaCl, KCl, $CaCl_2$ and NaBr salts by calculating the correct amounts of Na^+ , K^+ , Ca^{2+} , Cl^- and Br^- ions. Setting the model to 3 it will return the NaCl equivalent for the electrolyte model as amounts of Na^+ and Cl^- . If halide scales are not needed setting model to 3 is the recommended option.

The properties of the salt component or the ions have also been set to give reasonable estimates for the density of aqueous solutions. The enthalpy of solutions have not been fitted. The

procedure for calculating the equivalent concentration is based on that described in: A practical model for the effect of salinity on gas hydrate formation, B. Edmonds, R.A.S. Moorwood and R. Szczepanski, SPE 35569 (1996).

The calling specification is as follows:

```
      SUBROUTINE AYSALN( INOPT, MODEL, DEN, ANAL, ZSALT, NERR, MXERR,
     +                   IERR )

      INTEGER, INTENT(IN) ::  INOPT, MODEL, MXERR
      INTEGER, INTENT(OUT):: NERR, IERR(*)
      REAL*8, INTENT(IN) :: DEN, ANAL(*)
      REAL*8, INTENT(OUT) :: ZSALT(5)

C
C     Input arguments:
C     INOPT    Input option:
C                0  Ion concentrations in mg/litre
C                1  Total dissolved solid in mg/litre
C                2  Salt molalities
C                3  Salt mole fractions
C                4  Salt mass fractions
C     MODEL      1  NaCl equivalant ( salt component )
C                2  NaCl, KCl and CaCl2 equivalents
C                3  NaCl equivalent for electrolyte model
C     DEN      Salt density if known in kg/m3
C     ANAL     Concentrations:
C              INOPT=0  Ion analyses in mg/litre. Ions in order Na+, Ca++,
C                       Mg++, K+, Sr++, Ba++, Fe++, Cl-, SO4--, HCO3-, Br-
C              INOPT=1  Total dissolved solid in mg/litre in ANAL(1)
C              INOPT=2  Salt concentrations in molalities. Salts in order
C                       NaCl, CaCl2, MgCl2, KCl, SrCl2, BaCl2, FeCl2,
C                       Na2SO4, NaHCO3, NaBr
C              INOPT=3  Salt concentrations in mole fractions. Salts in
C                       order NaCl, CaCl2, MgCl2, KCl, SrCl2, BaCl2, FeCl2,
C                       Na2SO4, NaHCO3, NaBr
C              INOPT=4  Salt concentrations in mass fractions. Salts in
C                       order NaCl, CaCl2, MgCl2, KCl, SrCl2, BaCl2, FeCl2,
C                       Na2SO4, NaHCO3, NaBr
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     ZSALT    if model == 1, zsalt(1) will contain salt component amount in
moles per mol of water
C              if model == 2 or model ==3
C              NA+  = ZSALT (1)
C              K+   = ZSALT (2)
C              Ca++ = ZSALT (3)
C              Cl-  = ZSALT (4)
C              Br-  = ZSALT (5)
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Error codes. Values for this routine:
C              13611  total salt mole/mass fractions > 1
C              13612  ion analysis entered with no negative or no
C                     positive ions
```

5

5 Evaluating pure component properties

5.1 Summary of routines

Pure component properties may be evaluated by calling the following routines:

AYPCNK	character valued constants, e.g. name, formula etc.
AYPCNR	real valued constants, e.g. critical properties etc.
AYPCNR	returns the status of the set property. Status flags are defined in ayjuid.inc
AYPHC	tests whether components are hydrocarbons
AYPDIE	dielectric constant of liquid at 1 atm pressure
AYPDNL	density of liquid on the saturation line
AYPHLV	enthalpy change on evaporation (latent heat)
AYPHSI	perfect gas enthalpy, entropy and heat capacity
AYPLCP	heat capacity of liquid on the saturation line
AYPLH1	liquid enthalpy, entropy and heat capacity on the saturation line
AYPLTC	thermal conductivity of liquid on the saturation line
AYPLVS	viscosity of liquid on the saturation line
AYPSCP	heat capacity of solid
AYPSDN	density of solid
AYPST	surface tension of liquid on the saturation line
AYPSVP	saturated liquid vapour pressure
AYPSVR	second virial coefficient
AYPVTC	vapour thermal conductivity in zero density limit
AYPVVS	vapour viscosity in zero density limit

The models used are defined in Appendix . Complete information on how to call each routine is given below. A working example of a complete program is provided in Appendix .

5.2 Use of NCO and IC arguments

Properties may be returned for all the components currently loaded by setting NCO to zero. In this case the contents of the IC array are ignored. IC is the array of component load numbers.

To return properties for a subset of the complete set of loaded components NCO should be set to the number of components and the component load numbers should be set in the IC array. Values are returned in locations 1 to NCO of the output arrays.

If NCO is set to a negative value this is interpreted as a request for subset of $|NCO|$ components. The IC array must be set. The value for the Ith component is returned in the location IC(I) of the output arrays. Note that the component load numbers returned by the component loading routines are always in the range 1 to MAXNCS.

5.3 Constant properties

5.3.1 AYPCNK: character valued constants

Examples of the constants returned by AYPCNK include component name, formula, etc. See Appendix for a list of property codes. The calling specification is:

```
      SUBROUTINE AYPCNK( NCO, IC, JVALUE, KCONST, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), JVALUE, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      CHARACTER*(*), INTENT(OUT) :: KCONST(*)

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list
C     JVALUE    Property identification code (from ayjpid.inc)
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     KCONST    pure component character constants.
C               If NCO +ve prop. for component I placed in KCONST(I),
C               if NCO -ve prop. for component I placed in KCONST(IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               -13261  missing constants for one or more components
C               -13262  invalid component number
C               13261   property id (JVALUE) not recognised
```

5.3.2 AYPCNR: real valued constants

Examples of the constants returned by AYPCNR include critical properties, etc. See Appendix for a list of property codes. The calling specification is:

```
      SUBROUTINE AYPCNR( NCO, IC, JVALUE, RCONST, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), JVALUE, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(OUT) :: RCONST(*)

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list. Eg. setting NC to 2 , IC(1) to 1
C               and IC(2) to 3 would return properties for components 1
C               and 3. The simplest way to return properties for all
C               components is to set NC to 0 and pass IC as a dummy
C               array.
C     JVALUE    Property identification code.
C               The properties returned by AYPCNR are selected by
C               setting the appropriate code (J-number) from include
C               file ayjpid.inc.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     RCONST    Pure component constants. Set to -DLARGE if missing.
C               If NCO +ve prop. for component I placed in RCONST(I),
C               If NCO -ve prop. for component I placed in RCONST(IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               -13251  missing constants for one or more components
C               -13252  invalid component number
C               13251   property id (JVALUE) not recognised
```

5.3.3 AYPCNS: status of constants

This subroutine can be used to check if some property is defined from a databank or user defined. The calling specification is:

```
      SUBROUTINE AYPCNS( NCO, IC, JVALUE, STATUS, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), JVALUE, MXERR
      INTEGER, INTENT(OUT) :: STATUS(*), NERR, IERR(*)

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list
C     JVALUE    Property identification code (from AYJPID.INC)
C     MXERR     maximum number of error flags to be returned in IERR
C
```

```
 5
C      Output arguments:
C      STATUS   Pure component property status (see AYJUID.INC)
C               If NCO +ve prop. for component I placed in STATUS(I),
C               If NCO -ve prop. for component I placed in STATUS(IC(I))
C      NERR     no. errors reported by this routine and lower level routines
C      IERR     Array of error codes. Values returned by this routine:
C               -13821  invalid component number
C               13821   property id (JVALUE) not recognised
```

5.3.4 AYPHC: test for hydrocarbons

AYPHC returns a logical flag for each component to indicate whether it is a hydrocarbon. A .TRUE. value is returned if the component is a petroleum fraction or if its chemical formula contains just carbon and hydrogen.

```
      SUBROUTINE AYPHC( NCO, IC, HC, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      LOGICAL, INTENT(OUT) :: HC(*)

C
C      Input arguments:
C      NCO      Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C      IC       Subcomponent list
C      MXERR    maximum number of error flags to be returned in IERR
C
C      Output arguments:
C      HC       Elements .TRUE. if component a hydrocarbon, otherwise .FALSE.
C      NERR     no. errors reported by this routine and lower level routines
C      IERR     Array of error codes. Values returned by this routine:
C               -13351  missing constants for one or more components
C               -13352  invalid component number
```

5.4 Temperature-dependent properties

Temperature-dependent properties for pure components are usually stored on databanks for limiting conditions where the property value is fixed by the temperature alone. Thermal properties of gases are normally correlated in the perfect gas state, i.e. the low pressure limit of real gas behaviour. Gas phase transport properties are normally correlated in the zero-density limit where pressure effects are unimportant. Liquid phase properties are normally correlated on the saturation line, i.e. at a pressure corresponding to the saturated liquid vapour pressure at the given temperature. All the routines described in this section include extrapolation procedures to ensure that a value is returned even if the temperature is outside the limits for the correlation.

5.4.1 AYPDIE: dielectric constant of liquid

The property returned is the dimensionless relative permittivity of the liquid at low pressure (nominally 1 atm).

```
      SUBROUTINE AYPDIE( NCO, IC, TIN, IDT, DIEL, DDIEL, DDDIEL, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) ::  NCO, IC(*), IDT, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT):: DIEL(*), DDIEL(*), DDDIEL(*)

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list
C     TIN       Temperature (K)
C     IDT       derivative flag (0=none, 1=first derivatives,
C               2=second derivatives)
C     MXERR     Maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     DIEL      Dielectric constant
C               (If NCO +ve prop. for component I placed in element I,
C               if NCO -ve prop. for component I placed in element IC(I))
C     DDIEL     T derivative of dielectric constant
C               Only returned if IDT>0.
C     DDDIEL    Second T derivative of dielectric constant
C               Only returned if IDT=2.
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               -13921  Temperature below lower limit or above upper
C                       limit for one or more components. Extrapolated
C                       values returned
C               13921   Unrecognised equation type
```

5.4.2 AYPDNL: density of liquid on the saturation line

```
      SUBROUTINE AYPDNL( NCO, IC, TIN, IDT, DL, DDL, DDDL, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) ::  NCO, IC(*), IDT, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: DL(*), DDL(*), DDDL(*)

C
C     Input arguments:
```

```
5
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list.
C     TIN       temperature (K)
C     IDT       derivative flag (0=none, 1=first derivatives,
C               2=second derivatives)
C     MXERR     maximum number  of error flags to be returned in IERR
C
C     Output arguments:
C     DL        saturated liquid density for components selected (mol m-3)
C     DDL       T derivative of DL (mol m-3 K-1); if IDT=1 or 2
C     DDDL      second T derivative of DL (mol m-3 K-2); if IDT=2
C               (If NCO +ve prop. for component I placed in element I,
C               if NCO -ve prop. for component I placed in element IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               -13161  temperature below lower limit or above upper
C                       limit for one or more components. Extrapolated
C                       values returned
C               13161   unrecognised equation type
C               13162   erroneous/missing data for one or more components.
C                       -DLARGE returned.
```

5.4.3 AYPHLV: enthalpy change on evaporation (latent heat)

```
      SUBROUTINE AYPHLV( NCO, IC, TIN, LDERIV, HLV, DHLV, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: HLV(*), DHLV(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list.
C     TIN       temperature (K)
C     LDERIV    logical flag. If set to .TRUE. both property and
C               derivative are returned. If .FALSE. only the property
C               is returned.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     HLV       enthalpy change on evaporation for components selected
C               (J mol-1)
C     DHLV      T derivative of HLV (J mol-1 K-1); if LDERIV is .TRUE.
C               (If NCO +ve prop. for component I placed in element I,
C               if NCO -ve prop. for component I placed in element IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
```

```
 5
C              -13171  temperature below lower limit or above upper
C                      limit for one or more components. Extrapolated
C                      values returned
C              13171   unrecognised equation type
C              13172   erroneous/missing data for one or more components.
C                      -DLARGE returned.
```

5.4.4 AYPHSI: perfect gas enthalpy, entropy and heat capacity

```
      SUBROUTINE AYPHSI( NCO, IC, TIN, LDERIV, HIG, SIG, DHIG, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) ::  HIG(*), SIG(*), DHIG(*)
      LOGICAL, INTENT(OUT) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       Subcomponent list.
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     HIG      enthalpy of components in ideal gas state (J mol-1)
C     SIG      entropy of components in ideal gas state (J K-1 mol-1)
C     DHIG     T derivative of HIG (J mol-1 K-1); if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              -13241  temperature below lower limit or above upper
C                      limit for one or more components. Extrapolated
C                      values returned
C              13241   component type is not fluid
C              13242   unrecognised equation type
C              13243   T value unacceptable
```

5.4.5 AYPLCP: liquid heat capacity on the saturation line

```
      SUBROUTINE AYPLCP( NCO, IC, TIN, DPROP, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
```

```
5
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: DPROP(*)

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     DPROP    Cp (J/mol K)
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10151   Equation number not recognised.
C              10152   Critical temperature undefined for component
C             -10151   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

5.4.6 AYPLH1: liquid enthalpy, entropy and heat capacity on the saturation line

```
      SUBROUTINE AYPLH1( NCO, IC, TIN, LDERIV, HLIQ, CPLIQ, SLIQ, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: HLIQ(*), CPLIQ(*), SLIQ(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       Subcomponent list
C     TIN      Temperature (K)
C     LDERIV   Logical flag. If .TRUE. CPLIQ is also returned
C              If .FALSE. only HLIQ and SLIQ are returned.
C     MXERR    Maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     HLIQ     Saturated liquid enthalpy relative to 298.15 K (J/mol)
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     CPLIQ    Derivative of saturated liquid enthalpy (Cp) (J/mol K)
C     SLIQ     Saturated liquid entropy relative to 298.15 K (J/mol K)
C     NERR     No. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              13651   Equation number not recognised.
C              13652   Critical temperature undefined for component
```

```
5
C             -13651   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
C             -13652   Reference temperature below lower limit or above
C                      upper limit.
```

5.4.7 AYPLTC: thermal conductivity of liquid on the saturation line

```
      SUBROUTINE AYPLTC( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     saturated liquid thermal conductivity for components
C              selected (W/m K)
C     DPROP    T derivative of saturated liquid thermal conductivity
C              (W/m K2)
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10121   Equation number not recognised.
C              10122   Critical temperature undefined for component
C             -10121   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPLVS: viscosity of liquid on the saturation line

```
      SUBROUTINE AYPLVS( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
```

```
5
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     saturated liquid viscosity for components selected (Pas)
C     DPROP    T derivative of saturated liquid viscosity (Pas/K).
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10101   Equation number not recognised.
C             -10101   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPSCP: solid heat capacity

```
      SUBROUTINE AYPSCP( NCO, IC, TIN, DPROP, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: DPROP(*)

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     DPROP    Cp (J/mol K)
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10171   Equation number not recognised.
C             -10171   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPSDN: density of solid

```
      SUBROUTINE AYPSDN( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     density of solid for components selected (mol/m3)
C     DPROP    T derivative of solid density (mol/m3/K)
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10181   Equation number not recognised.
C             -10181   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPST: surface tension of liquid on the saturation line

```
      SUBROUTINE AYPST( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                  MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C
C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
```

```
 5
C               is returned.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP      saturated liquid surface tension for components selected (N/m)
C     DPROP     T derivative of saturated liquid surface tension (N/m K)
C               Only returned if LDERIV is .TRUE.
C               (If NCO +ve prop. for component I placed in element I,
C               if NCO -ve prop. for component I placed in element IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               10141   Equation number not recognised.
C               10142   Critical temperature undefined for component
C              -10141   Temperature below lower limit or above upper
C                       limit. Extrapolated values returned.
```

AYPSVP: saturated liquid vapour pressure

```
      SUBROUTINE AYPSVP( NCO, IC, TIN, LDERIV, VP, DVP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: VP(*), DVP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        Subcomponent list.
C     TIN       temperature (K)
C     LDERIV    logical flag. If set to .TRUE. both property and
C               derivative are returned. If .FALSE. only the property
C               is returned.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     VP        saturated vapour pressure for components selected (Pa)
C     DVP       T derivative of saturated vapour pressure (Pa/K).
C               Only returned if LDERIV is .TRUE.
C               (If NCO +ve prop. for component I placed in element I,
C               if NCO -ve prop. for component I placed in element IC(I))
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               -13151  temperature below lower limit or above upper
C                       limit for one or more components. Extrapolated
C                       values returned
C               13151   unrecognised equation type
C               13152   erroneous/missing data for one or more components.
C                       -DLARGE returned.
```

AYPSVR: second virial coefficient

```
      SUBROUTINE AYPSVR( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     second virial coefficient for components selected (m3/mol)
C     DPROP    T derivative of second virial coefficient (m3/mol/K)
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10191   Equation number not recognised.
C              10192   Critical temperature undefined for component
C             -10191   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPVTC: vapour thermal conductivity in the zero-density limit

```
      SUBROUTINE AYPVTC( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
```

```
 5
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     ideal gas thermal conductivity for components selected (W/m K)
C     DPROP    T derivative of ideal gas thermal conductivity (W/m K2).
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10131   Equation number not recognised.
C              10132   Critical temperature undefined for component
C              10133   Relative molar mass undefined for component
C             -10131   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

AYPVVS: vapour viscosity in the zero-density limit

```
      SUBROUTINE AYPVVS( NCO, IC, TIN, LDERIV, PROP, DPROP, NERR,
     +                   MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: TIN
      REAL*8, INTENT(OUT) :: PROP(*), DPROP(*)
      LOGICAL, INTENT(IN) :: LDERIV

C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero, constants returned for all components.
C              If negative, changes placing of output values.
C     IC       subcomponent list
C     TIN      temperature (K)
C     LDERIV   logical flag. If set to .TRUE. both property and
C              derivative are returned. If .FALSE. only the property
C              is returned.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     PROP     ideal gas viscosity for components selected (Pas)
C     DPROP    T derivative of ideal gas viscosity (Pas/K).
C              Only returned if LDERIV is .TRUE.
C              (If NCO +ve prop. for component I placed in element I,
C              if NCO -ve prop. for component I placed in element IC(I))
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              10111   Equation number not recognised.
C              10112   Critical temperature undefined for component
C              10113   Relative molar mass undefined for component
C             -10111   Temperature below lower limit or above upper
C                      limit. Extrapolated values returned.
```

5.5 Fetching coefficients in temperature-dependent property correlations

This section describes a routine (AYPCOR) that may be used to return values of correlation coefficients in the temperature-dependent property correlations for pure components. A complete set of routines (section) is provided for evaluating temperature-dependent pure component properties. These routines should be used wherever possible because they separate the property values from the details of the correlation equations and how the coefficients are stored. It is usually unnecessary and undesirable to access the coefficients directly.

The calling specification is:

```
      SUBROUTINE AYPCOR( NCO, IC, JPROP, NCRDIM, NEQN, NCOR, CORR, TMIN,
     +                   TMAX, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), JPROP, NCRDIM, MXERR
      INTEGER, INTENT(OUT) :: NEQN(*), NCOR(*), NERR, IERR(*)
      REAL*8, INTENT(OUT) :: CORR(NCRDIM,*), TMIN(*), TMAX(*)

C
C     Input arguments:
C     NCO       Number of components in subcomponent list.
C               If zero, constants returned for all components.
C               If negative, changes placing of output values.
C     IC        subcomponent list
C     JPROP     property identification code (must be one of the
C               temperature-dependent properties listed in ayjpid.inc)
C     NCRDIM    size of 1st dimension of CORR as declared in calling
C               routine. If NCRDIM is not large enough to contain all
C               the information to be returned in CORR, an error code
C               is returned.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     (If NCO is positive or zero , information  for component I is
C     placed in element I of the following arrays. If NCO is negative,
C     information is placed in element IC(I))
C     NEQN      equation type code for each component. If no correlation
C               is stored, a value of zero is returned and values are
C               not set in NCOR, CORR, TMIN and TMAX.
C     NCOR      number of coefficients in correlation (which may be
C               greater than NCRDIM)
C     CORR      correlation coefficients - a maximum of NCRDIM values
C               are returned. If NCOR(I) is greater than NCRDIM then
C               some coefficients are missing for component I.
C     TMIN      array of minimum temperature limits for correlation
C     TMAX      array of maximum temperature limits for correlation
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               10161   J-number not a recognised temperature dependent
C                       property
```

The models and equation numbers are defined in Appendix . The maximum number of coefficients for any property (the setting for

5

NCRDIM) can be found from the parameter values in include file ayjpid.inc.

6 Petroleum fluid characterisation

6.1 Introduction

The aim of the Infochem characterisation procedures is to make optimum use of available experimental data relating to petroleum fluid composition to construct an equation of state model of the fluid. The model must be sufficiently detailed, i.e. contain enough components, to describe accurately the phase behaviour of interest but be simple enough for practical engineering calculations. We also wish to apply the same characterisation procedure to model all types of phase behaviour including straightforward vapour-liquid and vapour-liquid-liquid equilibria, hydrate formation and wax and asphaltene formation. It is essential therefore that the same set of real components and pseudocomponents can be used simultaneously with models for all phases of interest (fluid and solid).

We take as a starting point a typical modern fluid analysis as provided by a PVT laboratory. The laboratory will depressurise the reservoir fluid to obtain separate liquid and gas samples. The gas-oil ratio (GOR) is recorded in one way or another. The samples are analysed by gas chromatography (GC) to obtain their compositions. For the gas, the components will be real discrete components of known molecular weight, so the composition can be expressed in mole or weight fractions with equal validity. The GC analysis of the liquid also identifies some discrete components, but mostly the individual hydrocarbons cannot be identified. Conventionally, the analysis is expressed as a series of single carbon number (SCN) cuts terminating with a plus fraction which gives the remainder. The liquid GC analysis measures the weight fractions not the mole fractions. In addition some properties of the liquid sample such as density and molecular weight can be measured.

Another option is to have the fluid defined as a series of boiling point cuts. In this method the smaple is distilled with a very high relux ratio to separate the several cuts. Multiflashcan handle two

methods: True Boiling Point curves (TBP) and D86 boiling point curves.

There are a number of steps in the characterisation of a petroleum fluid which are interrelated but can also be separated out. The procedure used within Infochem's characterisation algorithm is:

- The SCN weight fractions in the liquid, including the plus fraction, in the liquid are correlated against carbon number using a distribution function.
- By extrapolating the distribution function, the plus fraction can be split into a number of pseudocomponents of varying molecular weight as required. Some SCN cuts can also be grouped into pseudocomponents as required.
- Molecular weights and specific gravities are assigned to each component from a suitable correlation against carbon number.
- The molecular weights and specific gravities of the components are all adjusted to match the measured values for the whole liquid sample.
- The liquid weight fractions are converted to mole fractions using the molecular weights of the components.
- The gas and liquid mole fractions are combined in the correct proportions to give the mole fractions of the total recombined fluid.
- Using established correlations, the critical properties and acentric factors of the SCN cuts and pseudocomponents can be estimated from the values of molecular weight and specific gravity for that component. The correlations described in section are used to estimate these properties.
- For asphaltenes we use the information from a SARA characterisation to split off an asphaltene component and one or more resin components from the fluid.
- For waxes we use the information from a UOP wax content test or the normal-paraffin distribution to characterise the wax-forming fraction of the fluid.

6.2 Using the characterisation procedure

The Infochem characterisation procedure is implemented in the routine AYPVT1. The concept is to allow for the input of a gas steam analysis and a liquid stream analysis plus a separator GOR or, alternatively, a total fluid analysis without a GOR. The specific gravity (SG) and molecular weight of the liquid, total fluid or the plus fraction (the heavy fraction without a compositional analysis) may also be specified.

There is also a simplified version to be used with TBP and D86 boiling point curves: AYPVT1_TBP

The compositional analysis may contain a set of discrete components (real components such as methane etc.) plus a series

of SCN fractions. It is also possible to enter the liquid n-paraffin SCN analysis separately from then non-n-paraffin SCN analysis. The gas phase analysis may include an n-paraffin SCN distribution. Analytical information and may be entered in molar units or mass units. An analysis to C30+ is routinely available and it is often possible to obtain a much more detailed analysis. It is also possible to specify a water cut (as a volume ratio relative to the liquid stream at stock tank conditions).

The procedure will generate a set of petroleum fraction pseudocomponents based on the information provided. Although it is possible to represent each SCN with a separate pseudocomponent it is more usual to group SCNs to produce a smaller number of pseudocomponents if a detailed analysis is provided. Conversely, if the analysis only extends to, say, C10, it may be desirable to split the heavy end into several pseudocomponents. It is necessary to specify a starting SCN for the pseudocomponents and the number of pseudocomponents required.

If a SARA analysis is provided the procedure will automatically generate asphaltene and resin pseudocomponents with properties appropriate for the Infochem asphaltene model.

The n-paraffin distribution is used to support the Coutinho wax model (see section). If the distribution has not been measured it can be estimated from the UOP wax content (C20+ wax content).

The calling specification for AYPVT1 is:

```
      SUBROUTINE AYPVT1( LNODB, LNOPF, CSTART, NNMDIS, NMDIS, NDISL,
     +                   ADISL, NCNL, ACNL, CNCNL, MWCNL, SGCNL, NCNN,
     +                   ACNN, LUL, MW, LMW, SG, LSG, NPLUL, NPFRIN,
     +                   NDISG, ADISG, NCNG, ACNG, LUG, GOR, SARA,
     +                   ESARA, TWC, ETWC, CRSTRT, CSPLIT, NSPLIT,
     +                   CNSPLT, NDIS, ZDIS, MWDIS, SGDIS, NCUT, NMCUT,
     +                   ZCUT, ITCUT, CNLCUT, CNCUT, MWCUT, SGCUT,
     +                   TBCUT, TCCUT, PCCUT, WCUT, TMCUT, DHMCUT,
     +                   DSMCUT, DCPCUT, WATCUT, ZWAT, MWWAT, SGWAT,
     +                   TOTAL, LUT, NERR, MXERR, IERR )

      INTEGER LNODB, LNOPF, NNMDIS, NDISL, NDISG, CSTART, NCNL, NCNN,
     +        NPFRIN, NCNG, CRSTRT, CSPLIT(2), NSPLIT(2), NDIS, NCUT,
     +        ITCUT(*), LMW, LSG, LUL, NPLUL, NERR, MXERR, IERR(*)
      REAL*8 ADISL(*), ACNL(*), CNCNL(*), MWCNL(*), SGCNL(*), ACNN(*),
     +       MW, SG, ADISG(*), ACNG(*), GOR, SARA(4), TWC,
     +       CNSPLT(MAXNCS,2), ZDIS(*), MWDIS(*), SGDIS(*), ZCUT(*),
     +       CNLCUT(*), CNCUT(*), MWCUT(*), SGCUT(*), TBCUT(*),
     +       TCCUT(*), PCCUT(*), WCUT(*), TMCUT(*), DHMCUT(*),
     +       DSMCUT(*), DCPCUT(*), WATCUT, ZWAT, MWWAT, SGWAT, TOTAL
      CHARACTER*(*) NMDIS(*), NMCUT(*)
      LOGICAL LUG, ESARA, ETWC, LUT

C
C     Note: if a total fluid analysis is entered rather than separator
C     gas and liquid analyses the amounts should be entered in the
C     liquid phase arguments (ADIS, ACIN)
C
C               must be the last entry.
C     CNCNL     Lower boundary of liquid CN cut
```

```
5
C     MWCNL    Molecular weight of liquid CN cut
C     SGCNL    Specific gravity of liquid CN cut
C     NCNN     number of composition values for n-paraffins.
C              NCNN is the number of values entered in array ACNN.
C              If an n-paraffin distribution is not provided, set NCNN=0.
C     ACNN     compositional information for n-paraffins.
C              If NPRFIN=1 ACNN(I) is the percentage of n-paraffins in
C              the Ith CN fraction in the STO. The values must sum to
C              less than 100%. The n-paraffin distribution may extend to
C              higher carbon numbers than the analysis in ACNL in which
C              case NCNN will be greater than NCNL. Otherwise NCNN
C              may be less than or equal to NCNL.
C              If NPRFIN=2 ACNN(I) is the fraction of material in
C              ACNL(I) that is n-paraffin. Each fractions must be between
C              zero and one. In this case NCNN must be less than NCNL.
C              Any value entered for the plus fraction is ignored.
C              Percentages or fractions are either on a mass basis or a
C              molar basis depending on the setting of NPLUL
C     LUL      flag for liquid/total amounts: 0=molar units, 1=mass units,
C              2=volume units
C     MW       molecular weight (set to -DLARGE if not specified).
C     LMW      = 0  MW refers to the plus fraction
C              = 1  MW refers to the total fluid
C              = 2  MW refers to the STO
C     SG       specific gravity (set to -DLARGE if not specified)
C     LSG      = 0  SG refers to the plus fraction
C              = 2  SG refers to the STO
C     NPLUL    flag for n-paraffin amounts: 0=molar units, 1=mass units
C     NPFRIN   code for the percentage/fraction of n-paraffins.
C              1 for percentage of n-paraffins in STO
C              2 for fraction of n-paraffins in each CN cut.
C              The percentage/fraction is on a mass or molar basis
C              depending on the setting of NPLUL
C     NDISG    number of discrete components present in gas analysis
C              given in array ADISG. NDISG must be less than or equal to
C              NNMDIS. If NDISG<NNMDIS, this means that remaining components
C              defined in NMDIS are not present in this particular gas sample.

C     ADISG    amounts of discrete components in gas analysis in units
C              denoted by LUG (NDISG values).
C     NCNG     Number of CN cuts in gas, i.e. NCNG is the number of
C              values entered in array ACNG. If no gas distribution is to
C              be entered, set NCNG=0.
C     ACNG     amounts for CNs in gas starting at CSTART
C     LUG      .TRUE. if gas amounts (ADISG/ACNG) are on a mass basis,
C              .FALSE. if on a molar basis
C     GOR      recombination gas-oil ratio in sm3/m3
C     SARA     SARA analysis as mass fractions of saturates,
C              aromatics, resins and asphaltenes(nC7) in STO.
C              Only the values for resins and aspaltenes are actually used.
C              Undefined values should be set -DLARGE.
C     ESARA    .TRUE. if missing SARA data to be estimated, otherwise .FALSE.
C              This is only required if an asphaltene model is used.
C     TWC      total (UOP) wax content in STO as mass fraction.
C              An undefined value should be set -DLARGE.
C     ETWC     .TRUE. if n-paraffin distribution to be estimated.
C              This is only required if the Coutinho wax model is used.
C     CRSTRT   carbon number at which regression of CN analytical
C              data is to start
```

```
 5
C     CSPLIT   carbon number at which pseudocomponents to start
C              (1)  value for non n-paraffin distribution
C              (2)  value for n-paraffin distribution. Set to 0 for default.
C                   (Only required if wax content entered or estimated).
C     NSPLIT   number of pseudocomponents required
C              (1)  value for non n-paraffin distribution
C              (2)  value for n-paraffin distribution. Set to 0 for default.
C                   (Only required if wax content entered or estimated).
C     CNSPLT   Lower carbon number bounds defined for pseudocomponents
C     WATCUT   water cut as volume fraction of liquid hydrocarbon phase
C              at 60F and 1 atm. An undefined value should be set -DLARGE.
C     TOTAL    total amount of fluid. All compositions are scaled to give
C              this amount. An undefined value should be set -DLARGE.
C     LUT      .TRUE. if total amount (TOTAL) is in mass units, .FALSE.
C              if in molar units
C     MXERR    maximum number of error codes to be returned in IERR
C
C     Output arguments:
C     NDIS     number of discrete components with values set in ZDIS,
C              MWDIS and SGDIS. (excluding water)
C     ZDIS     mole numbers of discrete components
C     MWDIS    molecular weights of discrete components
C     SGDIS    specific gravities of discrete components
C     NCUT     number of pseudocomponents returned
C     NMCUT    assigned names of pseudocomponents
C     ZCUT     molar amounts of pseudocomponents
C     ITCUT    type of pseudocomponent (code from AYJXID.inc)
C     CNLCUT   lower carbon number boundary of pseudocomponents
C     CNCUT    average carbon number of pseudocomponents
C     MWCUT    average molecular weight of pseudocomponents
C     SGCUT    average specific gravity of pseudocomponents
C     TBCUT    average true boiling point of pseudocomponents
C     TCCUT    average critical temperature of pseudocomponents
C     PCCUT    average critical pressure of pseudocomponents
C     WCUT     average acentric factor of pseudocomponents
C     TMCUT    normal melting point of pseudocomponents
C     DHMCUT   enthalpy of fusion of pseudocomponents
C     DSMCUT   entropy of fusion of pseudocomponents
C     DCPCUT   change in Cp on fusion of pseudocomponents
C     ZWAT     number of moles of water present
C     MWWAT    molecular weight of water
C     SGWAT    specific gravity of water
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              13941 Discrete component not in databank
C              13942 Reading data bank has failed
C              13943 Discrete component not hydrocarbon, CO2, H2S or nitrogen
C              13944 Insufficient discrete components named
C              13946 Procedure to model CN distribution has failed
C             -13941 Fluid a gas and mol.wt. or sp.grav. specified
C             -13942 Fluid an oil and mol.wt. and sp.grav. not. specified
```

The simplified version to be used for TBP and D86 curves, AYPVT1_TBP is defined as:

```
      SUBROUTINE AYPVT1_TBP( LNOPF, D86, NCN, ACN, TB_CN, MWCN, SGCN,
     +   CUTUNITS, MW, LMW, SG, LSG, SARA, LSARA, TWC, LTWC, ISTART,
     +   SSPLT, NSPLT, CNSPLT, NCUT, NMCUT, ZCUT, ITCUT, CNLCUT, CNCUT,
     +   MWCUT, SGCUT, TBCUT, TCCUT, PCCUT, WCUT, TMCUT, DHMCUT, DSMCUT,
```

```
     +   DCPCUT, WVOL, ZWAT, MWWAT, SGWAT, TOTAL, LUT, NERR, MXERR,
     +   IERR)

      INTEGER, INTENT(IN) :: LNOPF, NCN, LMW,LSG,  NSPLT(2), SSPLT(2), MXERR
      INTEGER, INTENT(OUT) :: NCUT, ITCUT(*), NERR, IERR(*)
      INTEGER, INTENT(INOUT) :: ISTART
      LOGICAL, INTENT(IN) :: LUT, LSARA, LTWC, D86, CUTUNITS
      REAL*8, INTENT(IN) :: ACN(*), TB_CN(*), MWCN, SGCN, MW, SG,
     +   SARA(4), TWC, CNSPLT(MAXNCS,2), WVOL, TOTAL
      REAL*8, INTENT(OUT) :: ZCUT(*), CNLCUT(*), CNCUT(*), MWCUT(*),
     +   SGCUT(*), TBCUT(*), TCCUT(*), PCCUT(*), WCUT(*), TMCUT(*),
     +   DHMCUT(*), DSMCUT(*), DCPCUT(*), ZWAT, MWWAT, SGWAT
      CHARACTER*(*), INTENT(OUT) :: NMCUT(*)

C     Input arguments:
C     LNOPF    Load number of characterisation databank (correlations)
C     D86      Flag indicating if the analysis is D86 (.TRUE.) or TBP
(.FALSE.)
C     NCN      Number of boiling point cuts
C     ACN      cumulative volume of boiling point cuts (cm3)
C     TB_CN    Temperature in K or carbon number for boiling point cuts
C     MWCN     Molecular weight for boiling point cuts (set to -DLARGE if not
specified)
C     SGCN     Specific gravity for boiling point cuts (set to -DLARGE if not
specified)
C     CUTUNITS .TRUE. TB_CN units in K, .FALSE. TB_CN indicates carbon number
C     MW       molecular weight (set to -DLARGE if not specified).
C     LMW      = 0  MW refers to the plus fraction
C              = 1  MW refers to the total fluid
C              = 2  MW refers to the STO
C     SG       specific gravity (set to -DLARGE if not specified)
C     LSG      = 0  SG refers to the plus fraction
C              = 2  SG refers to the STO
C     SARA     values of the SARA analysis. Set not specified values to
-DLARGE.
C     LSARA    .TRUE. if missing SARA data to be estimated, otherwise .FALSE.
C               This is only required if an asphaltene model is used.
C     TWC      Total (UOP) wax content as mass fraction. (set to -DLARGE if
not specified)
C     LTWC     .TRUE. if n-paraffin distribution to be estimated.
C     SSPLT    carbon number at which pseudocomponents to start
C              (1)  value for non n-paraffin distribution
C              (2)  value for n-paraffin distribution. Set to 0 for default.
C                   (Only required if wax content entered or estimated).
C     NSPLT    number of pseudocomponents required
C              (1)  value for non n-paraffin distribution
C              (2)  value for n-paraffin distribution. Set to 0 for default.
C                   (Only required if wax content entered or estimated).
C     CNSPLT   User defined lower carbon number bounds defined for
pseudocomponents.
C              (set to -DLARGE if not specified)
C              (*,1)  value for non n-paraffin distribution
C              (*,2)  value for n-paraffin distribution. Set to 0 for default.

C     WVOL     water cut as volume fraction of liquid hydrocarbon phase. (set
to -DLARGE
C              if not specified)
C     TOTAL   Total amount of fluid. Compositions are scaled to give this
amount.
```

```
 5
C     LUT      .TRUE. if total amount (TOTAL) is in mass units (g), .FALSE.
C              if in molar units (mol).
C     MXERR    Dimension of IERR array.
C
C     Input/Output Arguments
C     ISTART   Carbon number at which regresion should start.
C              The output is the CN at which the regression actually starts.
C
C     Output arguments
C     NCUT     number of pseudocomponents returned
C     NMCUT    assigned names of pseudocomponents
C     ZCUT     molar amounts of pseudocomponents
C     ITCUT    type of pseudocomponent (code from AYJXID.inc)
C     CNLCUT   lower carbon number boundary of pseudocomponents
C     CNCUT    average carbon number of pseudocomponents
C     MWCUT    average molecular weight of pseudocomponents
C     SGCUT    average specific gravity of pseudocomponents
C     TBCUT    average true boiling point of pseudocomponents
C     TCCUT    average critical temperature of pseudocomponents
C     PCCUT    average critical pressure of pseudocomponents
C     WCUT     average acentric factor of pseudocomponents
C     TMCUT    normal melting point of pseudocomponents
C     DHMCUT   enthalpy of fusion of pseudocomponents
C     DSMCUT   entropy of fusion of pseudocomponents
C     DCPCUT   change in Cp on fusion of pseudocomponents
C     ZWAT     number of moles of water present
C     MWWAT    molecular weight of water
C     SGWAT    specific gravity of water
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              14221  Cut bounds must be defined for the D86 method - method
failed
C              14222  Amounts distilled must be defined by volume for the D86
method -
C                     method failed
```

The information provided by AYPVT1 or AYPVT1_TBP should be used to load the discrete components (using AYPDBC and pseudocomponents (using AYPPF) as described in section . The amounts of the components should be used to calculate the properties of the mixture using the thermodynamic and transport property models described in section .

6.3 Black Oil characterisation procedure

Black Oil correlations have traditionally been used to represent petroleum fluids as three pseudocomponents: oil, gas and water. Although such correlations can be effective when dealing with phases of essentially fixed composition they are not applicable when compositions change or if the fluid is a volatile oil or a condensate.

The Multiflash Black Oil characterisation procedure takes as input a minimal set of black oil parameters and generates a compositional fluid model. The minimum input required is liquid

gravity and GOR. A better model will be generated if other properties such as the gas gravity, Watson characterisation factor and the gas composition can be provided.

The calling specification for AYPVB1 is:

```
      SUBROUTINE AYPVB1( LNODB, LNOPF, GG, SG, LSG, GOR, WK, NADIS,
     +                   GASAMT, SARA, ESARA, TWC, ETWC, SSPLIT, NSPLIT,
     +                   CNSPLT, NDIS, ZDIS, NMDIS, NCUT, NMCUT, ZCUT,
     +                   ITCUT, CNLCUT, CNCUT, MWCUT, SGCUT, TBCUT,
     +                   TCCUT, PCCUT, WCUT, TMCUT, DHMCUT, DSMCUT,
     +                   DCPCUT, WATCUT, ZWAT, MWWAT, SGWAT, TOTAL, LUT,
     +                   NERR, MXERR, IERR )

      INTEGER LNODB, LNOPF, NSPLIT(*), NCUT, ITCUT(*), NERR, MXERR,
     +        IERR(*), NADIS, NDIS, SSPLIT(*), LSG
      REAL*8 GOR, GG, SG, WK, GASAMT(*),  SARA(*), TWC, ZDIS(*),
     +       ZCUT(*), CNLCUT(*), CNCUT(*), MWCUT(*), SGCUT(*), TBCUT(*),
     +       TCCUT(*), PCCUT(*), WCUT(*), TMCUT(*), DHMCUT(*),
     +       DSMCUT(*), DCPCUT(*), WATCUT, ZWAT, MWWAT, SGWAT, TOTAL
      CHARACTER*(*) NMDIS(*), NMCUT(*)
      LOGICAL ESARA, ETWC, LUT

C
C     Input arguments:
C     LNODB    load number for pure component databank used to obtain
C              properties of the discrete components
C              (obtained from AYPDBO call)
C     LNOPF    load number for characterisation method to use for
C              estimating petroleum fraction properties.
C              (obtained from AYPDBO call)
C     GOR      solution GOR (Rs) (m3/m3)
C     GG       gas gravity (relative to air) = MWgas/MWair = MWgas/28.966
C     SG       liquid specific gravity (relative to water at 1 atm, 60F)
C     LSG      code to identify liquid to which SG refers:
C              2  SG is for stock tank oil
C              1  SG is for the total fluid (oil plus dissolved gas)
C     WK       Watson K-factor for oil (optional).
C              An undefined value should be set -DLARGE.
C     NADIS    Number of gas discrete components
C     GASAMT   amounts (moles) of the gas discrete components in the
C              following order: N2, CO2, methane, ethane, propane,
C              isobutane, butane
C     TOTAL    total amount for fluid stream (after recombination) - in
C              units set bu LUL
C     LUT      units of TOTAL amount:
C              .TRUE.  for mass units (g)
C              .FALSE. for molar units (mole)
C     SARA     SARA analysis as weight fractions of saturates,
C              aromatics, resins and asphaltenes. Only the values for resins
C              and aspaltenes are actually used. Undefined values should
C              be set -DLARGE.
C     ESARA    .TRUE. if missing SARA data to be estimated, otherwise .FALSE.
C              This is only required if an asphaltene model is used.
C     TWC      Total (UOP) wax content in liquid as mass fraction
C              An undefined value should be set -DLARGE.
C     ETWC     .TRUE. if n-paraffin distribution to be estimated
C              This is only required if the Coutinho wax model is used.
C     SSPLIT   Carbon number at which pseudocomponents to start
C              (1)  value for non n-paraffin distribution
```

```
 5
C               (2)  value for n-paraffin distribution. Set to 0 for default.
C                    (Only required if wax content entered or estimated).
C     NSPLIT    Number of pseudocomponents required
C               (1)  value for non n-paraffin distribution
C               (2)  value for n-paraffin distribution. Set to 0 for default.
C                    (Only required if wax content entered or estimated).
C     CNSPLT    Lower carbon number bounds defined for pseudocomponents
C               An undefined value should be set -DLARGE.
C     WATCUT    Water cut as volume fraction of liquid at 60F and 1 atm
C     MXERR     maximum number of error codes to return in IERR
C
C     Output arguments:
C     NDIS      Number of discrete components required (excluding water)
C     ZDIS      Mole numbers of discrete components
C     NMDIS     Names of discrete components
C     NCUT      Number of pseudocomponents returned
C     NMCUT     Assigned names of pseudocomponents
C     ZCUT      Molar amounts of pseudocomponents
C     ITCUT     Type of pseudocomponent (from AYJXID.inc)
C     CNLCUT    Lower carbon number boundary of pseudocomponents
C     CNCUT     Average carbon number of pseudocomponents
C     MWCUT     Average molecular weight of pseudocomponents
C     SGCUT     Average specific gravity of pseudocomponents
C     TBCUT     Average true boiling point of pseudocomponents
C     TCCUT     Average critical temperature of pseudocomponents
C     PCCUT     Average critical pressure of pseudocomponents
C     WCUT      Average acentric factor of pseudocomponents
C     TMCUT     Normal melting point of pseudocomponents
C     DHMCUT    Enthalpy of fusion of pseudocomponents
C     DSMCUT    Entropy of fusion of pseudocomponents
C     DCPCUT    Change in Cp on fusion of pseudocomponents
C     ZWAT      Number of moles of water present
C     MWWAT     Molecular weight of water
C     SGWAT     Specific gravity of water
C     NERR      No. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               14061 Liquid specific gravity not defined
C               14062 SG specified is too small.
C               14063 The liquid specific gravity is too big.
C               14064 GG specified is either too small or too big.
C               14065 GOR not defined
C               14066 Discrete component not in databank
C               14067 Reading data bank has failed
C               14068 The Watson K-factor and the SG of the liquid are not
C                     consistent, so that MW of C5+ is too small from the
correlation
C               14069 Maximum iteration reached
C               14070 Blackoil analysis procedure to model CN distribution has
failed.
C              -14061 The gas analysis is not consistent with the gas gravity,
C                     default values used.
C              -14062 The Watson K-factor is too big, default value used.
C              -14063 The Watson K-factor is too small, default value used.
C              -14064 The amount or MW of the gas stream is too small based
C                     on the the gas analysis and the input data, so that
C                     the gas analysis is ignored.
```

6.4 Blending petroleum fluids

The blending method in Multiflash allows mixing, or blending, of already characterised petroleum fluids to provide a new fluid characterisation, for example when two pipes intersect. This routine AYBLND allows two fluids to be blended together in relative amounts specified to produce a new fluid described by its own set of pseudo-components. The properties and relative amounts of the blend's pseudo-components is automatically calculated by the Multiflash blending procedure. The method of averaging the properties of the blended pseudo-components is exactly the same as that used in the PVT analysis procedure to create the pseudo-components used to represent the properties of the original petroleum fluids. The method also handles waxy and asphaltenic crudes thereby predicting the likely wax or asphaltene formation from the fluid blend.

The calling specification for AYBLND is:

```
      SUBROUTINE AYBLND( MODEL, N, NAME, ALIAS, CARN, TYPE, HC,  Z, CN,
     +                   MW, SG, TB, TC, PC, W, TM, DHM, DSM, DCPM,
     +                   MCRKS, MCPR, VRKS, VPR, REFVIS, IDVIS, RA,
     +                   NERR, MXERR, IERR )

      INTEGER MODEL, N(3), TYPE(MAXNCS,3), NERR, MXERR, IERR(*),
     +        IDVIS(MAXNCS,3)
      REAL*8 Z(MAXNCS,3), CN(MAXNCS,3), MW(MAXNCS,3), SG(MAXNCS,3),
     +       TB(MAXNCS,3), TC(MAXNCS,3), PC(MAXNCS,3), W(MAXNCS,3),
     +       TM(MAXNCS,3), DHM(MAXNCS,3),DSM(MAXNCS,3),DCPM(MAXNCS,3),
     +       RA(2,3), MCRKS(MAXNCS,3,3), MCPR(MAXNCS,3,3),
     +       VRKS(MAXNCS,3,3), VPR(MAXNCS,3,3), REFVIS(MAXNCS,3)
      CHARACTER*72 NAME(MAXNCS,3), ALIAS(MAXNCS,3)
      CHARACTER*12 CARN(MAXNCS,3)
      LOGICAL HC(MAXNCS,3)

C     Input arguments:
C     MODEL    Model pointer, 1 for RKS model, 2 for PR model
C     N        Number of components, (1) for first stream; (2) for second
stream
C     NAME     Array of component names, (1) for first stream; (2) for second
stream
C     ALIAS    Array of component aliases, (1) for first stream; (2) for
second stream
```

```
C     CARN     Array of component CAR numbers, (1) for first stream; (2) for
second
C              stream
C     TYPE     Array of component types in first mixture, (1) for first
stream; (2) for
C              second stream
C     HC       Array of component hydrocarbon flags,(1) for first stream; (2)
for
C              second stream
C     Z        Array of component mole fractions, (1) for first stream; (2)
for second
C              stream
C     CN       Array of component carbon numbers, (1) for first stream; (2)
for second
C              stream
C     MW       Array of component molecular weights, (1) for first stream; (2)
for
C              second stream
C     SG       Array of component specific gravities, (1) for first stream;
(2) for
C              second stream
C     TB       Array of component normal boiling points, (1) for first stream;
(2) for
C              second stream
C     TC       Array of component critical temperatures, (1) for first stream;
(2) for
C              second stream
C     PC       Array of component critical pressures, (1) for first stream;
(2) for
C              second stream
C     W        Array of component acentric factors, (1) for first stream; (2)
for
C              second stream
C     TM       Array of component normal melting points, (1) for first stream;
(2) for
C              second stream
C     DHM      Array of component enthalpies of fusion, (1) for first stream;
(2) for
C              second stream
C     DSM      Array of component entropies of fusion, (1) for first stream;
(2) for
C              second stream
C     DCPM     Array of component changes in Cp on fusion, (1) for first
stream; (2)
C              for second stream
C     MCRKS    Array of component MC parameters for RKS model, (1) for first
stream;
C              (2) for second stream
C     MCPR     Array of component MC parameters for PR model, (1) for first
stream;
C              (2) for second stream
C     VRKS     Array of component volume shift for RKS model, (1) for first
stream;
C              (2) for second stream
C     VPR      Array of component volume shift for PR model, (1) for first
stream; (2)
C              for second stream
C     REFVIS   Array of component reference viscosity,, (1) for first stream;
(2) for
```

```
5
C               second stream
C     IDVIS     Viscosity model ID for reference viscosity, (1) for first
stream; (2)
C               for second stream
C     RA        Resin-asphaltene parameters, (1) for first, (2) for the second
mixture
C     MXERR     maximum number of error codes to return in IERR
C
C     Output arguments:
C     N         Number of components, (3) for blended mixture
C     NAME      Array of component names, (3) for blended mixture
C     ALIAS     Array of component aliases , (3) for blended mixture
C     CARN      Array of component CAR numbers , (3) for blended mixture
C     TYPE      Array of component types , (3) for blended mixture
C     Z         Array of component mole fractions , (3) for blended mixture
C     CN        Array of component carbon numbers , (3) for blended mixture
C     MW        Array of component molecular weights , (3) for blended mixture
C     SG        Array of component specific gravities , (3) for blended mixture

C     TB        Array of component normal boiling points , (3) for blended
mixture
C     TC        Array of component critical temperatures , (3) for blended
mixture
C     PC        Array of component critical pressures , (3) for blended mixture

C     W         Array of component acentric factors , (3) for blended mixture
C     TM        Array of component normal melting points , (3) for blended
mixture
C     DHM       Array of component enthalpies of fusion , (3) for blended
mixture
C     DSM       Array of component entropies of fusion , (3) for blended
mixture
C     DCPM      Array of component changes in Cp on fusion , (3) for blended
mixture
C     MCRKS     Array of component MC parameters for RKS model , (3) for
blended mixture
C     MCPR      Array of component MC parameters for PR model , (3) for blended
mixture
C     VRKS      Array of component volume shift parameters , (3) for blended
mixture
C               for RKSA
C     VPR       Array of component volume shift parameters , (3) for blended
mixture
C               for PRA
C     REFVIS    Array of component reference viscosity parameters , (3) for
blended
C               mixture
C     IDVIS     Viscosity model ID for reference viscosity, (3) for blended
mixture
C     RA        Resin-asphaltene parameters, (3) for the blended mixture.
C     NERR      no. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               14151 Mol. wt. and/or carbon number missing for pseudocomponent

C              -14151 Viscosity models are not consistent among the blended
fluids.
```

7 Binary interaction parameters

Binary interaction parameters (BIPs) are required by most of the mixture property models in Multiflash. Some models, such as cubic equations of state, can produce useful property predictions for many mixtures without specially fitted BIPs. Other models e.g. , most activity models, require specific non-default BIPs to make realistic predictions. For more information on the models available and their BIPs see Appendices - .

Before setting up a mixture model that requires non-default BIPs it is necessary to define the BIP values. The procedures for doing this are described in this section.

7.1 Defining BIPs

Before defining BIPs the components in the mixture must be loaded as described in section . To define BIPs the following steps are involved:

- Initialise the BIP software.
- Open BIP databanks (if required).
- Load BIPs
  - from databanks.
  - and/or entering directly

Each step is described in detail below.

7.2 Initialising the BIP software

The following call initialises the BIP data system:

```
CALL AYBINI()
```

7.3 Using BIP databanks

7.3.1 Opening BIP databanks

If binary interaction parameters (BIPs) are to be loaded from a databank or other data source it must first be opened or selected by calling routine AYPDBO. This is the same routine that is used for handling pure component data sources (section). The current options are the Infochem INFOBIPS BIP databank for VLE, the Infochem INFOLLBIPS BIP databank for LLE and the 'oilandgas' estimation methods for equations of state (see section). The required option is selected by setting the argument JBANK to one of the values in ayjdbk.inc. Note that the pure component databank structures are not valid for BIP databanks and are not described in this section. Once the BIP bank is opened successfully the BIP data source is subsequently referred to by the load number LNODBB, which is returned. The calling specification is descibed in :

For example, the following call opens the INFOBIPS BIP bank:

```
INCLUDE 'ayjdbk.inc'
INTEGER MXERR
PARAMETER ( MXERR = 10 )
INTEGER JBANK(1), LNODBB, NERR, IERR(MXERR)
CHARACTER*1 CDUM(1)
JBANK(1) = JBINFO
CALL AYPDBO( 1, JBANK, 0, CDUM, LNODBB, NERR, MXERR, IERR )
```

This call returns BIP data source load number in LNODBB which must be passed without alteration to subroutine AYBDBL which loads the BIP values. Note: all BIP databanks must be in the required Infochem format.

AYPDBO can also open a user's own data file with the same structure as the Infochem BIP bank. To do this the name of the files containing the BIP data and the index file (BIPFIL) must be passed to AYPDBO rather than a blank. For example:

```
INCLUDE 'ayjdbk.inc'
INTEGER MXERR
PARAMETER ( MXERR = 10 )
INTEGER JBANK(1), LNODBB, NERR, IERR(MXERR)
CHARACTER*10 BIPFIL(2)
BIPFIL(1) = 'myfile.bin'
BIPFIL(2) = 'myfile.idx'
JBANK(1) = JBOINF
CALL AYPDBO( 1, JBANK, 2, BIPFIL, LNODBB, NERR, MXERR, IERR )
```

7.3.2 Loading BIPs from a databank

The routine AYBDBL can be called repeatedly to load as many BIP arrays into Multiflash as are needed. Each array can then be referred to by a unique number, the BIP load number returned by

AYBDBL and then passed to the model loading routines. The calling specification is:

```
      SUBROUTINE AYBDBL( LNODBB, NC, IC, INIT, NMDL, LMDL, NBIP,
     +                   LNO, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNODB, NC, IC(*), NMDL, LMDL(NMDL), NBIP, MXERR
      INTEGER, INTENT(INOUT) :: LNO
      INTEGER, INTENT(OUT) ::  NERR, IERR(*)
      LOGICAL, INTENT(IN) :: INIT

C
C     Input arguments:
C     LNODBB   BIP databank load number
C     NC       Number of components in subcomponent list IC
C              0=all components selected from data bank
C     IC       Subcomponent list: array of component load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     INIT     .TRUE.  any BIPs for which information is available on the
C                      databank should be initialised,
C              .FALSE. only BIPs that are currently unset to be given
C                      values if found on the BIP databank
C     NMDL     Dimension of array LMDL
C     LMDL     Integer array specifying the model
C     NBIP     BIP serial number (1=Aji or kij, 2=Aij, 3=alpha)
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     LNO      BIP load number. Set to 0 if a new set of BIPs is
C              to be created.
C
C     Output values:
C     LNO      BIP load number.
C              If input value was 0, LNO will be set to a new value
C              unless no BIPs or only 'ideal' default BIPs found.
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     array of error codes. Values returned by this routine:
C              10441   Unrecognised data bank load number
C              10442   Unrecognised data bank type
```

For example, the following code sets up a BIP load number (LNOB1 for the RKS model by loading BIPs from a databank:

```
      INTEGER MAXMDL
      PARAMETER ( MAXMDL = 5 )
      INTEGER MXERR
      PARAMETER ( MXERR = 10 )
      INCLUDE 'aymncs.inc'
      INCLUDE 'ayjmid.inc'
      INTEGER IC(1), NMDL, LMDL(MAXMDL), LNOB1, LNOB2, NBIP,
     +        LNODBB, NERR, IERR(MXERR)
      LOGICAL INIT
      LMDL(1) = JMRKS
      NMDL = 1
      INIT = .TRUE.
      NBIP = 1
```

```
5
      LNOB1 = 0
      CALL AYBDBL( LNODBB, 0, IC, INIT, NMDL, LMDL, NBIP, LNOB1, NERR,
     +             MXERR, IERR )
```

The thermodynamic model for which the BIPs are required is referred to by array LMDL with LMDL(1) set to the model identifier (JM-number from file ayjmid.inc) and LMDL(2), LMDL(3), etc. set to the (optional) model variant numbers (other J numbers from file ayjmid.inc). The BIPs are stored in a linear form as half-arrays: for most equations of state one half-array is needed as the BIP array is symmetric (NBIP=1); for the Wilson and UNIQUAC equations two half-arrays are needed as the BIP array is asymmetric (NBIP=1,2); for the NRTL equation and the Infochem-NRTL mixing rule for cubic equations of state three half-arrays are needed to store the asymmetric BIP array (NBIP=1,2) and the symmetric alpha array (NBIP=3).

7.3.3 Unloading databanks

As with pure-component databanks, binary databanks can be closed to free storage. As with pure-component databanks, this is done by calling AYPDBU as follows:

```
      SUBROUTINE AYPDBU( LNODBB )

      INTEGER, INTENT(INOUT) :: LNODBB

C
C     Input/output arguments:
C     LNODBB   Data bank load number (set to zero on return)
```

If AYPDBU is called with an invalid load number no action is taken.

7.4 Entering BIPs directly

If required, additional BIPs may be entered or the existing values overwritten by calling AYBIP_USR. The old routine, AYBUSR is stil provided for backwards compatibility. This routine can also be used to create a new BIP load number if a BIP databank call (AYBDBL) was not used to do this. The calling specification for AYBIP_USR is:

```
      SUBROUTINE AYBIP_USR( NC, IC, INIT, LNO, IBFUN, NBCO, DEFBIP,
     +             ORD_DIM, BCOIN, NERR, MXERR, IERR )

      INTEGER, INTENT(IN):: NC, IC(*), IBFUN, NBCO,MXERR, ORD_DIM
      INTEGER, INTENT(INOUT):: LNO
      INTEGER, INTENT(OUT) :: NERR,  IERR(*)
      REAL*8, INTENT(IN):: DEFBIP, BCOIN(NBCO,*)
      LOGICAL, INTENT(IN) :: INIT

C
C     NC       number of components for which BIPs are entered.
```

```
 5
C               Zero means "all components currently defined".
C     IC        subcomponent list; if NC is not set to zero the array
C               IC must be filled with the NC component load numbers
C               for which BIPs are supplied.
C     INIT      logical flag.
C               If .TRUE. all BIP values entered in the BCOn arrays will
C               be stored internally.
C               If .FALSE. the BIPs entered will only be stored for those
C               cases where a BIP has not already been set. This option
C               is provided to allow the routine to be called several
C               times with subsets of BIPs estimated by different methods.
C     IBFUN     BIP function number (id codes defined in aybfid.inc)
C               JBFUND     undefined
C               JBFT1      a + bT + cT^2 (dimensionless, T in K, for eos)
C               JBFT2      a + bT + c/T (dimensionless, T in K, for eos)
C               JBACT      a + bT + cT^2 (J/mol, T in K, for activity models)
C               JBFASS     for association models
C               JBGERT     for gerg mixing rule in csma model, TC
C               JBGERV     for gerg mixing rule in csma model, VC
C     NBCO      number of coefficients entered for the T-function
C               specified by IBFUN:
C                 NBCO     values must be entered in
C                   1        BCOIN(1,*)
C                   2        BCOIN(1,*) to  BCOIN(2,*)
C                   3        BCOIN(1,*) to  BCOIN(3,*)
C                   4        BCOIN(1,*) to  BCOIN(4,*)
C     DEFBIP    default value of BIPs for BCO(1,*). This value is stored
C               in any unspecifed BIPs when the BIPload number is set up.
C               Eg. for a cubic eos DEFBIP would usuallly be set to 0.
C     ORD_DIM   Leading dimension of the BCOIN array.
C     BCOIN(1,*) first order BIP coefficients
C     BCOIN(2,*) second order BIP coefficients
C     BCOIN(3,*) third order BIP coefficients
C     BCOIN(4,*) fourth order BIP coefficients
C               Note: BIPs in BCOIN1,2,3,4 are entered as linear arrays.
C               Values in these arrays must be ordered correctly to
C               correspond with the pairs of components in listed in IC.
C               The BIP between the Ith and Jth components , where J < I,
C               must be placed in BCO(K), where K = (I-1)(I-2)/2 + J
C               the size of the second dimension must be at least 1+NC/2(NC-1)
C     MXERR     maximum number of error codes to return in IERR
C
C     Input/output argument:
C     Input values:
C     LNO       BIP load number. If set to 0 a new BIP load number is
C               returned and all BIPs are set to default values before
C               the specified values are entered.
C
C     Output values:
C     LNO       BIP load number as entered, or new load number if 0 entered
C
C     Output arguments:
C     NERR      number of error codes returned in IERR
C     IERR      error codes. Values for this routine:
C               13121   Run out of BIP load numbers
C               13122   Run out of space in BIP load array
C               13123   Incompatible function type
C              -13121    BIP temperature function incompatible with some/all
C                        stored BIPs
```

The function AYBUSR is also supplied and it is applicable for the majority of the BIPs necessary.

```
      SUBROUTINE AYBUSR( NC, IC, INIT, LNO, IBFUN, NBCO, DEFBIP, BCO1,
     +          BCO2, BCO3, NERR, MXERR, IERR )

      INTEGER, INTENT(IN):: NC, IC(*),IBFUN, NBCO,MXERR
      INTEGER, INTENT(INOUT):: LNO
      INTEGER, INTENT(OUT) :: NERR,  IERR(*)
      REAL*8, INTENT(IN):: DEFBIP, BCO1(*), BCO2(*), BCO3(*)
      LOGICAL, INTENT(IN) :: INIT

C     Input arguments:
C     NC       number of components for which BIPs are entered.
C              Zero means "all components currently defined".
C     IC       subcomponent list; if NC is not set to zero the array
C              IC must be filled with the NC component load numbers
C              for which BIPs are supplied.
C     INIT     logical flag.
C              If .TRUE. all BIP values entered in the BCOn arrays will
C              be stored internally.
C              If .FALSE. the BIPs entered will only be stored for those
C              cases where a BIP has not already been set. This option
C              is provided to allow the routine to be called several
C              times with subsets of BIPs estimated by different methods.
C     IBFUN    BIP function number (id codes defined in aybfid.inc)
C              JBFUND     undefined
C              JBFT1      a + bT + cT^2 (dimensionless, T in K, for eos)
C              JBFT2      a + bT + c/T (dimensionless, T in K, for eos)
C              JBACT      a + bT + cT^2 (J/mol, T in K, for activity models)
C              JBFASS     for association models
C              JBGERT     for gerg mixing rule in csma model, TC
C              JBGERV     for gerg mixing rule in csma model, VC
C     NBCO     number of coefficients entered for the T-function
C              specified by IBFUN:
C                NBCO     values must be entered in
C                  1        BCO1
C                  2        BCO1, BCO2
C                  3        BCO1, BCO2, BCO3
C     DEFBIP   default value of BIPs for BCO1. This value is stored
C              in any unspecifed BIPs when the BIPload number is set up.
C              Eg. for a cubic eos DEFBIP would usuallly be set to 0.
C     BCO1     first order BIP coefficients
C     BCO2     second order BIP coefficients
C     BCO3     third order BIP coefficients
C              Note: BIPs in BCO1,2,3 are entered as linear arrays.
C              Values in these arrays must be ordered correctly to
C              correspond with the pairs of components in listed in IC.
C              The BIP between the Ith and Jth components , where J < I,
C              must be placed in BCO(K), where K = (I-1)(I-2)/2 + J
C     MXERR    maximum number of error codes to return in IERR
C
C     Input/output argument:
C     Input values:
C     LNO      BIP load number. If set to 0 a new BIP load number is
C              returned and all BIPs are set to default values before
C              the specified values are entered.
C
C     Output values:
```

```
 5
C      LNO      BIP load number as entered, or new load number if 0 entered
C
C      Output arguments:
C      NERR     number of error codes returned in IERR
C      IERR     error codes. Values for this routine:
C               -13122    BIP temperature function needs more bip values.
C                         Bip index 4 is set to zero.
```

AYBUSR may be called as often as is required to enter BIPs. If the load number LNO is entered as zero, new BIP half-arrays are created and the load number is returned in LNO. If an existing load number is entered, the existing values of the BIPs are overwritten. Setting a BCO value to -DLARGE (from routine AYNUMV) means that the existing value is not to be overwritten. If INIT is .TRUE., all other existing values are overwritten, if INIT is .FALSE., only default existing values are overwritten.

The ordering of values in the arrays BCO1, BCO2 and BCO3 or BCOIN(1:4,:) is according to the following rules. The BIP between the the i th. and j th. components in the subcomponent list IC must be placed in BCO (k) where $k=(i-1)(i-2)/2+j$ and $j<i$. The diagonal elements of the BIP array are not stored as they are not significant. For models that require more than one BIP it is necessary to define a separate BIP load number and associated BIP values for each set of BIPs. The number of BIP load numbers required be each model is listed in section . Note that it is only necessary to define BIP load numbers when non-default BIP values are required.

The function used to correlate BIPs is denoted by the argument IBFUN. The following options are available:

- JBFUND: the correlation is undefined which means that a set of values is stored but no restriction is placed on their use. When the BIP set is connected to a model the model is responsible for determining how the values are used.
- JBFT1: the correlation has the form: $B=a+bT+cT^2$ where the temperature T is in K. The BIP is dimensionless and is intended for use by an equation of state model.
- JBFT2: the correlation has the form: $B=a+bT+c/T$ where the temperature T is in K. The BIP is dimensionless and is intended for use by an equation of state model.
- JBFACT: the correlation has the form: $B=a+bT+cT^2$ where the temperature T is in K. The BIP has units of J/mol and is intended for use by an activity model.
- JBFASS: Up to 3 parameters may be specified for use by the association model (CPA).
- JBEOSV: Up to 3 parameters may be specified for use by the CSMA BIP for Vc.
- JBGERGT: Up to 3 parameters may be specified for use by the GERG model.
- JBGERGV: Up to 3 parameters may be specified for use by the GERG model.

- JBGERGF: 1 parameter that specified the weighting factor used by the GERG model.
- JBGELDH: up to 3 parameters for electrolyte model Deby-Huckel BIP.
- JBGELV2: up to 4 parameters for electrolyte model second virial BIP.
- JBGELV3: up to 3 parameters for electrolyte model third virial BIP.

For example, the following code sets up new load numbers and enters two sets of constant BIPs for an activity model for a three component mixture, returning the BIP load numbers in LNOB1 and LNOB2:

```
 INCLUDE 'aymncs.inc'
 INCLUDE 'aybfid.inc'
 INTEGER MXERR
 PARAMETER (MXERR=10)
 INTEGER IC(1), LNOB1, LNOB2, NERR, IERR(MXERR)
 REAL*8 BCO(4,MAXNCS*(MAXNCS-1)/2)
 LOGICAL INIT
 INIT = .TRUE.
 LNOB1 = 0
 BCO1(1,1) = -459.02D0
 BCO1(1,2) = 1916.20D0
 BCO1(1,3) = 7846.20D0
 CALL AYBIP_USR( 0, IC, INIT, LNOB1, JBFACT, 1, 0.0D0, 4, BCO,
+             NERR, MXERR, IERR )
 LNOB2 = 0
 BCO1(1,1) = 5578.10D0
 BCO1(1,2) = 606.30D0
 BCO1(1,3) = -2229.10D0
 CALL AYBIP_USR( 0, IC, INIT, LNOB2, JBFACT, 1, 0.0D0, 4, BCO,
+             NERR, MXERR, IERR )
```

7.5 BIPs for oil and gas mixtures

BIPs for components that are typically found in oil and gas mixtures have been corelated for the principal cubic equations of state: RKS, PR and their variants. The components included in the correlations are: methane, other defined hydrocarbons, pseudocomponents, CO_2 , nitrogen, H_2S , water, methanethiol, ethanethiol, propanethiol, butanethiol and COS. Hydrate inhibitors may include methanol, MEG, DEG, TEG, SALTCOMPONENT, ethanol, isopropanol, propylene glycol and glycerol. Note that parameters for SALTCOMPONENT are only available for the RKSA (or PRA) equation of state with the Infochem-NRTL mixing rules. For the CPA model BIPs are included for hydrocarbons, water, methanol, ethanol, MEG, CO_2 and H_2S . For the LKP model correlations are provided for hydrocarbons, CO_2 , nitrogen and H_2S .

BIP values may be set by using the 'oilandgas' estimation method by passing parameter JBOG4 from ayjdbk.inc to subroutine AYPDBO which selects data sources. JBOG4 selects the latest

version of the Infochem BIP correlations and is the recommended option. Access to previous correlations is provided for compatibility with earlier versions of Multiflash. JBOG3 was the default in Multiflash versions 2.7 and 2.8. JBOG2 was the default in versions 2.3 to 2.5 and JBOG1 was the correlation used prior to version 2.3.

For example, for an equation of state with a single BIP per binary:

```
      INCLUDE 'ayjdbk.inc'
      INTEGER MAXMDL
      PARAMETER ( MAXMDL = 5 )
      INTEGER MXERR
      PARAMETER ( MXERR = 10 )
      INCLUDE 'aymncs.inc'
      INTEGER IC(MAXNCS), NMDL, LMDL(MAXMDL), LNOB1, N, LNODBB, NERR,
     +        IERR(MXERR), JBANK(1)
      CHARACTER*1 CDUM(1)
      LOGICAL INIT
      JBANK(1) = JBOG4
      CALL AYPDBO( 1, JBANK, 0, CDUM, LNODBB, NERR, MXERR, IERR )
      INIT = .TRUE.
      N = 1
      LNOB1 = 0
      CALL AYBDBL( LNODBB, 0, IC, INIT, NMDL, LMDL, N, LNOB1, NERR, MXERR,
IERR )
```

7.6 Utilities

7.6.1 Getting values of stored BIP coefficients

The routine AYBIP_VAL returns the values of stored BIP coefficients for a particular BIP load number. The calling specification is:

```
      SUBROUTINE AYBIP_VAL( NC, IC, LNO, IBFUN, NBCO, ORD_DIM, BCO_OUT,
     +          NERR, MXERR, IERR )

      INTEGER, INTENT(IN)  :: NC, IC(*), LNO, MXERR, ORD_DIM
      INTEGER, INTENT(OUT) :: IBFUN, NBCO, NERR, IERR(*)
      REAL*8, INTENT(OUT) :: BCO_OUT(ORD_DIM,*)

C     Input arguments:
C     NC       number of components for which BIP values are required.
C              Zero means "all components currently defined".
C     IC       subcomponent list; if NC is not set to zero the array
C              IC must be filled with the NC component load numbers
C              for which BIPs are required.
C     LNO      BIP load number
C     ORD_DIM  Leading dimension of BCO_OUT
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     IBFUN    BIP function number (id codes defined in aybfid.inc)
```

```
 5
C               JBFUND     undefined
C               JBFT1      a + bT + cT^2 (dimensionless, T in K, for eos)
C               JBFT2      a + bT + c/T (dimensionless, T in K, for eos)
C               JBACT      a + bT + cT^2 (J/mol, T in K, for activity models)
C               JBFASS     for association models
C     NBCO      number of coefficients entered for the T-function
C               specified by IBFUN:
C                 NBCO     values must be entered in
C                   1        BCO_OUT(1,*)
C                   2        BCO_OUT(1,*) to  BCO_OUT(2,*)
C                   3        BCO_OUT(1,*) to  BCO_OUT(3,*)
C                   4        BCO_OUT(1,*) to  BCO_OUT(4,*)
C     BCO_OUT   BIP coefficients
C               Note: BIPs in BCO_OUT 1,2,3 are returned as linear arrays.
C               The BIP between the Ith and Jth components , where J < I,
C               is returned in BCO(K), where K = (I-1)(I-2)/2 + J
C     NERR      number of error codes returned in IERR
C     IERR      error codes. Values for this routine:
C               13131   Unrecognised BIP load number
```

The previous utility to fetch the values of BIPs is also provided for backward compatibility.

```
      SUBROUTINE AYBVAL( NC, IC, LNO, IBFUN, NBCO, BCO1, BCO2, BCO3,
     +                   NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NC, IC(*), LNO, MXERR
      INTEGER, INTENT(OUT) :: IBFUN, NBCO, NERR, IERR(*)
      REAL*8, INTENT(OUT) :: BCO1(*), BCO2(*), BCO3(*)

C
C     Input arguments:
C     NC        number of components for which BIP values are required.
C               Zero means "all components currently defined".
C     IC        subcomponent list; if NC is not set to zero the array
C               IC must be filled with the NC component load numbers
C               for which BIPs are required.
C     LNO       BIP load number
C     MXERR     maximum number of error codes to return in IERR
C
C     Output arguments:
C     IBFUN     BIP function number (id codes defined in aybfid.inc)
C               JBFUND     undefined
C               JBFT1      a + bT + cT^2 (dimensionless, T in K, for eos)
C               JBFT2      a + bT + c/T (dimensionless, T in K, for eos)
C               JBACT      a + bT + cT^2 (J/mol, T in K, for activity models)
C               JBFASS     for association models
C     NBCO      number of coefficients stored for the T-function
C               specified by IBFUN:
C                 NBCO     values returned in
C                   1        BCO1
C                   2        BCO1, BCO2
C                   3        BCO1, BCO2, BCO3
C     BCO1      first order BIP coefficients
C     BCO2      second order BIP coefficients
C     BCO3      third order BIP coefficients
C               Note: BIPs in BCO1,2,3 are returned as linear arrays.
C               The BIP between the Ith and Jth components , where J < I,
C               is returnrd in BCO(K), where K = (I-1)(I-2)/2 + J
```

```
5
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              13131   Unrecognised BIP load number
```

Besides being possible to get the BIP values, it is also possible to know where do the values come from. As for the pure component data, the BIP values also stored its status. Databank defined, user defined or undefined. The function used to get the BIP values status is AYBVLS

```
      SUBROUTINE AYBVLS( NC, IC, LNO, IBFUN, NBCO, ORD_DIM, IBCO_OUT,
     +   NERR, MXERR, IERR )

      INTEGER, INTENT(IN)  :: NC, IC(*), LNO, MXERR, ORD_DIM
      INTEGER, INTENT(OUT) :: IBFUN, NBCO, IBCO_OUT(ORD_DIM,*),
     +         NERR, IERR(*)

C     Input arguments:
C     NC       number of components for which BIP values are required.
C              Zero means "all components currently defined".
C     IC       subcomponent list; if NC is not set to zero the array
C              IC must be filled with the NC component serial numbers
C              for which BIPs are required.
C     LNO      BIP load number
C     ORD_DIM  Leading dimenstion for the IBCO_OUT array
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     IBFUN    BIP function number (JBFT1=a+bT+cT^2(eos), JBFT2=a+bT+c/T(eos),

C              JBACT=a+bT+cT^2(activity eqn), JBFASS=association,
C              JBGERT=GERG mixing rule in CSMA model)
C              JBGERV=GERG mixing rule in CSMA model)
C     NBCO     number of coefficients entered for the T-function
C              specified by IBFUN:
C                NBCO     values must be entered in
C                  1        IBCO_OUT(1,*)
C                  2        IBCO_OUT(1,*) to  IBCO_OUT(2,*)
C                  3        IBCO_OUT(1,*) to  IBCO_OUT(3,*)
C                  4        IBCO_OUT(1,*) to  IBCO_OUT(4,*)
C     IBCO_OUT status of  BIP coefficients
C              Note: BIPs in IBCO_OUT are returned as linear arrays.
C              The status between the Ith and Jth components , where J < I,
C              is returned in IBCO(K), where K = (I-1)(I-2)/2 + J
C              status values are found in ayjuid.inc
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              13141   Unrecognised BIP load number
```

7.6.2 Freeing BIP load numbers

The routine AYBNUL is provided to free BIP load numbers:

```
      CALL AYBNUL( LNOB )

      INTEGER, INTENT(INOUT) :: LNOB
C
```

```
 5
C      Input arguments:
C      LNOB     BIP load number to be released
```

The load number LNOB and associated storage is released for subsequent reuse. This routine is provided, principally, for use in interactive programs as there has to be a finite limit to the total number of BIP load numbers that the software can handle. The limit can be altered by changing the appropriate dimensioning parameters in files aymbco.inc, aymbin.inc, and recompiling the Models software. See also section on freeing model load numbers.

7.6.3 Deleting BIPs for a component

The routine AYBDEL deletes all BIPs associated with a particular component and then deletes it:

```
      SUBROUTINE AYBDEL( ICO, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(INOUT) :: ICO
      INTEGER, INTENT(OUT) :: NERR, IERR(*)

C
C      Input arguments:
C      MXERR    maximum number of error codes to return in IERR
C
C      Input/output arguments:
C      Input values:
C      ICO      Component load number
C
C      Output values:
C      ICO      Component load number (0=deleted)
C
C      Output arguments:
C      NERR     number of error codes returned in IERROR
C      IERR     error codes. None specific to this routine.
```

Deleting the BIPs means that default values will be used by the models.

8 Setting up mixture property models

The following steps are required:

- Initialise the models software.
- Load BIPs if required by the models (as described in section).
- Load models.

Those steps not previously defined are described in detail below.

8.1 Initialising the Models software

The following call initialises the models and phase descriptor software:

```
SUBROUTINE AYMINI( NERR, MXERR, IERR )

INTEGER, INTENT(IN) :: MXERR
INTEGER, INTENT(OUT) :: NERR, IERR(*)
```

8.2 Loading models

When a model is loaded the information that defines the model is checked and as much initialisation as possible is carried out. This typically involves fetching any pure component properties required and precalculating any temperature-independent model parameters or coefficients in temperature-dependent correlations. The loading procedure returns a model load number which is subsequently used to refer to the model. Most models can be set up with a number of options. Each instance of the model plus options will correspond to a different load number.

The required thermodynamic or transport property models may be loaded by calling AYLOAD and specifying all options and which BIPs (if any) they are to use. The calling specification is:

```
5
      SUBROUTINE AYLOAD( NMDL, LMDL, FULL, LNO, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NMDL, LMDL(*), MXERR
      INTEGER, INTENT(OUT) :: LNO, NERR, IERR(*)
      LOGICAL, INTENT(IN) :: FULL

C
C     Input arguments:
C     NMDL     number of integers passed in array LMDL that fully
C              define the model and its options
C     LMDL     integer array specifying the model. The information
C              required in LMDL is model-dependent.
C     FULL     type of load operation required
C              .TRUE. if full load required, ie. normal model load.
C              .FALSE. if only a test load to check model definition
C              (this option is provided to support interactive
C              interfaces and is not intended for general use)
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     LNO      Model load number (unique identifier used subsequently to
C              refer to the the model).
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              10021  model identification code is not valid
C              10022  model cannot be initialised
C              10023  run out of model load numbers
C              10024  Run out of space in model load information array
```

The value of NMDL and the information required in LMDL is model-dependent and is defined in the tables below. If the full options list is not passed to AYLOAD, the missing options are set to the defaults; the default values are always the first ones in each list. Any option values beyond those expected are ignored. The model option identifiers are all held in file ayjmid.inc. The models available are defined in Appendices - .

The flag FULL, as stated in the description of the arguments is to be used to test model definitions. For example, if the hydrate model is to be loaded but no water is present and FULL == .TRUE., the model loading will fail as it needs water to perform calculations.

One model may be defined in terms of another model. Whenever a model depends on another, the model on which it is dependent must be loaded first and its load number is passed to the dependent model in the LMDL array. AYLOAD may be called as many times as is necessary to load the models that will be required. The models software permits several models to be loaded at any time. The number of models that may be loaded is limited by parameters in files aymlno.inc and aymlda.inc.

8.3 Thermodynamic models

8.3.1 Cubic equations of state

The cubic EOS are described in Appendices to . A more in depth description can be read in the Models Manual that is distributed with Multiflash. The model specification requires a primary model identifier plus a number of variant identifiers to specify the treatment of the *a* parameter, whether a Peneloux volume correction is used, the type of mixing rule and the correct number of BIP load numbers to match the mixing rule. The number of possible models resulting from all combinations of these variants is large. Some of the more common or useful combinations are shown in the table below.

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)	(6)	(7)
RKS type EOS								
GUI Models:								
RKS	5	JMRKSA	JVRKS	JVRKS	JVVDW	LBIP1		
RKS API	5	JMRKSA	JVAPI	JVRKS	JVVDW	LBIP1		
RKS (Advanced)	5	JMRKSA	JVSVP	JVDEN	JVVDW	LBIP1		
RKSA (Infochem)	7	JMRKSA	JVSVP	JVDEN	JVNRTL	LBIP1	LBIP2	LBIP3
RKS-HVP	7	JMRKSA	JVRKS	JVDEN	JVHVPS	LBIP1	LBIP2	LBIP3
PSRK	5	JMRKSA	JVMC	JVDEN	JVPSRK	LNOGEX		
PSRK-NRTL model	5	JMRKSA	JVMC	JVDEN	JVPSRK	LNOGEX		
Other variants:								
RK EOS	5	JMRKSA	JVRK	JVRKS	JVVDW	LBIP1		
RKS with volume shift	5	JMRKSA	JVRKS	JVDEN	JVVDW	LBIP1		
PR type EOS								
GUI Models:								
PR	5	JMPRA	JVPR	JVPR	JVVDW	LBIP1		
PR (Advanced)	5	JMPRA	JVSVP	JVDEN	JVVDW	LBIP1		
PR78	5	JMPRA	JVPR78	JVPR	JVVDW	LBIP1		
PR78 (Advanced)	5	JMPRA	JVSV78	JVDEN	JVVDW	LBIP1		
PR-HVP	7	JMPRA	JVPR	JVDEN	JVHVPS	LBIP1	LBIP2	LBIP3
Other variants:								
PR with volume shift	5	JMPRA	JVPR	JVDEN	JVVDW	LBIP1		

5								
PR78 with volume shift	5	JMPRA	JVPR78	JVDEN	JVVDW	LBIP1		
PRA with NRTL-type mixing rule	7	JMPRA	JVSVP	JVDEN	JVNRTL	LBIP1	LBIP2	LBIP3
ZJ EoS	5	JMZJ	JVZJ	LBIP				

The model variants are as follows:

JVAPI API version of RKS (Daubert and Graboski)
JVDEN T-dependent volume shift for EOS to liquid density (Peneloux)
JVDEN1 constant volume shift for EOS to liquid density (Peneloux)
JVMC Mathias-Copeman treatment for a parameter
JVNRTL Infochem NRTL-type mixing rule
JVHVPS Huron-Vidal-Pedersen (HVP) mixing rule
JVPSRK Predictive SRK parameters/mixing rule
JVPR original PR equation
JVPR78 PR78 equation
JVRK original RK equation
JVRKS original Soave version
JVSVP a parameter fitted to vapour pressure
JVSV78 a parameter for PR78 model fitted to vapour pressure
JVVDW van der Waals mixing rule
JVZJ The original ZJ Equation of State method
LBIP1 Load number for kij, Aij, or Δ gij/R binary interaction parameters
LBIP2 Load number for Aji or Δ gji/R binary interaction parameters
LBIP3 Load number for NRTL or HVP 'alpha' binary interaction parameters
LNOGEX Load number of activity model to use for G^E mixing rule

8.3.2 Non-cubic equations of state

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)
Ideal gas EoS	1	JMIDG				
HOC (2nd virial)	1	JMHOC				
LKP method	3	JMLKP	JVLKP	LBIP1		
LK method	3	JMLKP	JVLK	LBIP1		
BWR(S) EoS						
BWRS	3	JMBWRS	JVBWRS	LBIP1		
BWRS (Starling-Han)	3	JMBWRS	JVSH	LBIP1		
BWR	3	JMBWRS	JVBWR	LBIP1		
Corresponding states						
CSMA with vdw mixing rule	4	JMCSMA	JVVDW	LBIP1		
CSMA with GERG	5	JMCSMA	JVCGER	LBIP1	LBIP2	LBIP3

5						
mixing rule GERG-2008 model	5	JMGERG	JVCGER	LBIP1	LBIP2	LBIP3
IAPWS-95 water model	1	JMSTEA				
CO2 high-accuracy model	1	JMCDIO				
Water-Ammonia high-accuracy eos	5	JMWAMM	JVIORG	LBIP1	LBIP2	LBIP3
CPA models						
CPA (Infochem)	4	JMASSO	JVSVP	LBIP1	LBIP2	LBIP3
Asphaltene model						
Default model	1	JMASP				
Full specification	2+ NDPAR*2	JMASP	NDPAR	JPi	Pi	repeated
PC-SAFT model						
Standard model	5	JMSAFT	JVPC	0	LBIP1	LBIP2
Simplified Michelsen version	5	JMSAFT	JVLYNG	0	LBIP1	LBIP2

Model variants:
JPi model data parameter identifier (model-dependent, see ayjmid.inc)
JVBWR Original BWR eos
JVBWRS Starling modified BWR eos using stored parameters (where available) and SH correlation otherwise
JVLKP Use standard LKP mixing rules
JVLK Use original LK mixing rules
JVLYNG Simplified PC-SAFT model
JVPC Original PC-SAFT model
JVSH Starling modified BWR eos using Staring-Han correlation for parameters
JVSVP a parameter fitted to vapour pressure
LBIP1 Load number for kij or Aij binary interaction parameters
LBIP2 Load number for Aji binary interaction parameters
LBIP3 Load number for Fij weighting factor of binary departure eos contribution (GERG), or CPA dimerisation BIP
LNOFL Load number of fluid (gas/liquid) phase model
NDPAR number of data parameters for model
Pi model data parameter value (model-dependent)

8.3.3 Free energy addition model

The free energy model allows you to combine models. This is for example necessary if you want to use the asphaltene model or the electrolyte model.

5

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)
Free energy addition model	5	JMADD	JVANAL	LNOM	LNOM	LNOM

Model variants:
JVANAL Original variant of the addition model
LNOM Load number of previously loaded model

Asphaltene model. To use the asphaltene model it is necessary to combine it with a fluid model. In principle the asphaltene model can be combined with any fluid model, but it has been optimized for the RKSA model, and it is not recommended to combine it with any other model.

model	NMDL	LMDL(1)	(2)	(3)	(4)
Free energy addition model	4	JMADD	JVANAL	LNOFL	LNOASP

Model variants:
LNOFL Load number of fluid (gas/liquid) phase model
LNOASP Load number for asphaltene model

Electrolyte model. The electrolyte model also needs to be combined with a fluid model using the free energy addition model. See also section .

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)
Free energy addition model	5	JMADD	JVANAL	LNOLRI	LNOFL	LNOLRE

Model variants:
LNOLRI Load number for electrolyte phase check model
LNOLRE Load number for long-range electrolyte model
LNOFL Load number of fluid (gas/liquid) phase model

8.3.4 Liquid activity equations

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)	(6)	(7)
Ideal solution	3	JMIDL	LNOG	Jpc				
UNIFAC method								
VLE variant	4	JMUNF	JVVLE	LNOG	LNOG	Jpc		
LLE	4	JMUNF	JVLLE	LNOG	LNOG	Jpc		

variant								
PSRK variant	4	JMUNF	JVPSRK	LNOG	LNOG	Jpc		
UNIFAC A method								
VLE variant	6	JMUNFA	JVORIG	JVVLE	0	LNOG	Jpc	
LLE variant	6	JMUNFA	JVORIG	JVLLE	0	LNOG	Jpc	
PSRK variant	6	JMUNFA	JVORIG	JVPSRK	0	LNOG	Jpc	
PSRK no hydrocarbon correction	6	JMUNFA	JVORIG	JVPNHC	0	LNOG	Jpc	
Dortmund modified UNIFAC	6	JMUNFA	JVDORT	JVDORT	0	LNOG	Jpc	
UNIQUAC equation	5	JMUNQ	LNOG	LBIP1	LBIP2	Jpc		
NRTL equation								
VLE variant	7	JMNRTL	JVVLE	LNOG	LBIP1	LBIP2	LBIP3	Jpc
LLE variant	7	JMNRTL	JVLLE	LNOG	LBIP1	LBIP2	LBIP3	Jpc
Wilson equation								
A variant	6	JMWIL	JVWILA	LNOG	LBIP1	LBIP2	Jpc	
E variant	6	JMWIL	JVSL	LNOG	LBIP1	LBIP2	Jpc	
Regular Solution								
Standard regular solution	5	JMREG	JVREG	LNOG	LBIP1	Jpc		
Flory-Huggins	5	JMREG	JVFH	LNOG	LBIP1	Jpc		
Activity coefficient model for gas phase method	4	JMACG	JVACG	LNOG	LNOL			

Model variants:
JVFH Flory-Huggins model
JVLLE Use LLE variant
JVPSRK Predictive SRK parameters/mixing rule
JVPNHC Predictive SRK parameters/mixing rule with no hydrocarbon correction
JVORIG Original UNIFAC parameters/mixing rule
JVDORT Dortmund modified UNIFAC parameters/mixing rule
JVREG Regular solutions model
JVSL Use E (or small lambda) variant
JVVLE Use VLE variant
JVWILA Use A (or large lambda) variant
JVACG Activity coefficient model for gas phase

5

LBIP1 Load number for kij or Aij binary interaction parameters
LBIP2 Load number for Aji binary interaction parameters
LBIP3 Load number for NRTL 'alpha' binary interaction parameters
LNOG Load number of gas phase model
LNOL Load number of liquid activity model
Jpc Switch for Poynting correction: JVPOYN (default) or JVNOPO

8.3.5 User-modified Unifac groups and subgroups

For each definition of a Unifac model the user can amend the subgroup and group definitions by extending the UNIFACA definition beyond LMDL(5). LMDL(6) is set equal to the total number of subgroup and group definitions. Each subgroup definition requires a sequence of 5 contiguous elements of LMDL as shown in the following table.

LMDL(n)	LMDL(n+1)	LMDL(n+2)	LMDL(n+3)	LMDL(n+4)
JPSUB	IDSUB	IEQR	IEQQ	IDGRP

Data entries:
n A subscript number of array LMDL
JPSUB Integer parameter specifying a subgroup
IDSUB Subgroup identifier - value between -1 and -10 denotes a user-defined subgroup
IEQR r-value, single precision real equivalenced to an integer
IEQQ q-value, single precision real equivalenced to an integer
IDGRP Identifier of group to which subgroup belongs - value between -1 and -10 denotes a user-defined group

Each group definition requires a sequence of 6 contiguous elements LMDL as shown in the following table.

LMDL(n)	LMDL(n+1)	LMDL(n+2)	LMDL(n+3)	LMDL(n+4)	LMDL(n+5)
JPGRP	IDGRP1	IDGRP2	IEQIP0	IEQIP1	IEQIP2

Data entries:
n A subscript number of array LMDL
JPGRP Integer parameter specifying a group interaction parameter
IDGRP1 First group identifier - value between -1 and -10 denotes a user-defined group
IDGRP2 Second group identifier - value between -1 and -10 denotes a user-defined group
IEQIP0 constant term in (non-symmetric) group interaction parameter - single precision real equivalenced to an integer
IEQIP1 linear temperature term in (non-symmetric) group interaction parameter - single precision real equivalenced to an integer
IEQIP2 quadratic temperature term in (non-symmetric) group interaction parameter - single precision real equivalenced to an integer

8.3.6 Electrolyte model

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)	(6)	(7)
Long-range electrolyte model								
Debye-Huckel	7	JMLRE	JVDH			IBIP1	IBIP2	IBIP3
Debye-Huckel+Born	7	JMLRE	JVDH	JVBORN		IBIP1	IBIP2	IBIP3
Debye-Hckel+virial	7	JMLRE	JVDH	0	JVVIR	IBIP1	IBIP2	IBIP3
Debye-Huckel+Born+virial	7	JMLRE	JVDH	JVBORN	JVVIR	IBIP1	IBIP2	IBIP3
Electrolyte phase check model	1	JMLRI						

Model variants:
JVBORN Born charging term
JVDH Debye-Huckel long-range model
JVVIR Virial term for electrolytes
IBIP1 Bip load number for Deby-Huckel term
IBIP2 Bip load number for second virial term
IBIP3 Bip load number for third virial term

8.3.7 Solids

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)	(6)	
Gas hydrate model								
Hydrate I	3	JMHYD	JVTYP1	LNOL				
Hydrate II	3	JMHYD	JVTYP2	LNOL				
Hydrate H	3	JMHYD	JVTYPH	LNOL				
Hydrate/ice Nucleation model	2	JMBPN	LNOS					
Freeze-out model	3	JMFRZ	ICO	LNOL				
Advanced freeze-out model								
Relative to liquid	3	JMFRZA	JVLIQ	LNOL	NDPAR	JPi	Pi	repeated

5

Absolute	3	JMFRZA	JVACT	LNOL	NDPAR	JPi	Pi	repeated
Wax Coutinho model								
Wilson variant (recommended)	3	JMCTO	JVWIL	LNOL				
UNIQUAC variant	3	JMCTO	JVUNQ	LNOL				

Model variants:
ICO Component number (for freeze-out model)
JVTYP1 Type I hydrate
JVTYP2 Type II hydrate
JVTYPH Type H hydrate
JVLIQ Relative to liquid phase
JVACT Absolute
JVUNQ Use UNIQUAC mixing rule in Coutinho wax model
JVWIL Use Wilson mixing rule in Coutinho wax model
LNOL Load number of liquid phase model
LNOS Load number of solid phase model
NDPAR Number of data parameters for model
JPi Model parameter identifier (see ayjmid.inc)
Pi Model parameter value

Note that if model parameter identifier JPCOMP is used it must be followed by two values, the component load number followed by the component abundance.

8.3.8 Other thermodynamic models

model	NMDL	LMDL(1)	(2)	
Henry's Law water	2	JMHWT	LNOL	
COSTALD model	1	JMCOS	LBIP1	

Model variants:
LBIP1 Load number for kij or Aij binary interaction parameters
LNOL Load number of liquid phase model

8.4 Transport property models

model	NMDL	LMDL(1)	(2)	(3)	(4)	(5)	(6)	
Viscosity (gas and liquid)								

Lohrenz-Bray-Clark (standard)	3	JMLBC	JVLBC	LNOVOL				
Full specification of LBC	3+2*	JMLBC	JVLBC	LNOVOL	NDPAR	JPi	Pi	repeat
LBC (fit liquid viscosity)	NDPAR 3	JMLBC	JVLFIT	LNOVOL				
Pedersen (standard)	2	JMPDV	JVPDV					
Pedersen (heavy fluids)	2	JMPDV	JVHEAV					
Pedersen (fit liquid viscosity)	2	JMPDV	JVLFIT					
superTRAPP (standard)	2	JMSPTV	JVSPTV					
superTRAPP (fit liquid viscosity)	2	JMSPTV	JVLFIT					
Viscosity (gas)								
Vapour mixing rule 1	1	JMVVS1						
Viscosity (liquid)								
Liquid mixing rule 1	1	JMLVS1						
Twu (standard)	2	JMTWV	JVTWV					
Twu (fit liquid viscosity)	2	JMTWV	JVLFIT					
Thermal conductivity (gas and liquid)								
Chung-Lee-Starling	2	JMCLS	LNOVOL					
superTRAPP	2	JMSPTH	JVSPTH					
Thermal conductivity (gas)								
Vapour mixing rule 1	1	JMVTC1						

5

Thermal conductivity (liquid)									
Liquid mixing rule 1	1	JMLTC1							
Surface tension									
Linear Gradient Theory	4	JMLGST	0	LNOVOL	LBIP				
MacLeod-Sugden 1-phase	3	JMMCS	JV1PHA	LNOVOL					
MacLeod-Sugden 2-phase	3	JMMCS	JV2PHA	LNOVOL					
Sutton Model	3	JMSUT	JVSUT	LNOVOL					
Mixing rule 1	1	JMST1							
Diffusivity (gas)									
Chapman-Enskog (LJ)	3	JMCED	JVLJ	LBIP					
Chapman-Enskog (Brockaw)	3	JMCED	JVBROK	LBIP					
Chapman-Enskog (Wilke-Lee)	3	JMCED	JVWL	LBIP					
Chapman-Enskog (Fuller)	3	JMCED	JVFULL	LBIP					
Diffusivity (liquid)									
Hayduk-Minhas	5	JMLQD	JVHM	LNOLIQ	LNOVIS	LBIP			
Tyn-Calus	5	JMLQD	JVTC	LNOLIQ	LNOVIS	LBIP			

Model variants:
JV1PHA 1-phase variant
JV2PHA 2-phase variant
JVLBC standard LBC method
JVSPPV standard superTRAPP viscosity method
JVLFIT fit model parameters to match pure component liquid viscosity
JVPDV standard Pedersen viscosity method
JVSPTH standard superTRAPP thermal conductivity method
JVTWV standard Twu method
LBIP BIP load number
LNOVOL Load number of model to use for calculating volumetric properties (may be gas or liquid)

5

LNOVIS Load number of viscosity model
LNOLIQ Load number of thermodynamic model for liquid phase
NDPAR Number of data parameters for model
JPi Model parameter identifier (see ayjmid.inc)
Pi Model parameter value

The JVLFIT option changes the way that pure component model parameters are set. If a value is set for the reference viscosity (pure component constant property JLVSRF) the internal model parameters are adjusted to reproduce this value. If the property JLVSRF is not set (i.e. equalt to -DLARGE) the default method for the model is used to set the required parameters. To use the viscosity data matching routine AXMT04 the JVLFIT variant should be specified.

8.5 Information about a loaded model

The definition of a model can be requested from the model load number using AYGTLO. The LMDL array returns the integers used to specify the model when it was defined using AYLOAD.

```
      SUBROUTINE AYGTLO( LNO, N, NMDL, LMDL )

      INTEGER, INTENT(IN) :: LNO, N
      INTEGER, INTENT(OUT :: NMDL, LMDL(N)

C
C     Input arguments:
C     LNO      Model load number
C     N        Dimension of array LMDL
C
C     Output arguments:
C     NMDL     Number of integers stored to specify model,
C              0=load number not found
C     LMDL     Integer array specifying the model
```

8.6 Setting model parameter data values

The model parameter data values (Pi in the above table) are real numbers that must be entered in the LMDL array using the function AYMPDV defined below.

```
      INTEGER*4 FUNCTION AYMPDV( PV )

      REAL*8, INTENT(IN) :: PV
```

For example, the following code fragment defines the asphaltene model and sets the parameter JPAAA to 1.06.

```
      NMDL = 4
      LMDL(1) = JMASP
      LMDL(2) = 1
      LMDL(3) = JPAAA
```

```
5
      LMDL(4) = AYMPDV( 1.06D0 )
      CALL AYLOAD( 4, LMDL, .TRUE., LNO, NERR, MXERR, IERR )
```

The reverse operation is also possible. Convert the integer that comes from LMDL into a real number. This is acomplished using AYMPDV_R

```
      REAL*8 FUNCTION AYMPDV_R( PARVAL )

      INTEGER, INTENT(IN) :: PARVAL
```

8.7 Freeing model load numbers

The routine AYUNLO is provided to free model load numbers:

```
      SUBROUTINE AYUNLO( LNO )

      INTEGER, INTENT(INOUT) :: LNO
```

which releases the model load number LNO for subsequent reuse. This routine is provided, principally, for use in interactive programs as there has to be a finite limit to the total number of model load numbers that the software can handle. The limit can be checked in appropriate dimensioning parameters in files aymlno.inc and aymlda.inc. See also section on freeing BIP load numbers.

8.8 Reinitialising models

Routine AYRELA is provided to force all currently defined models to be reinitialised using the currently defined pure-component data. The procedure is useful if any of the pure-component data have been modified by calling subroutines such as AYPSTR, etc. The calling procedure is:

```
      CALL AYRELA( NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(OUT) :: NERR, MXERR, IERR(*)

C
C     Input arguments:
C     MXERR    Maximum number of error codes to return in IERR
C
C     Output arguments:
C     NERR     number of error codes returned in IERR
C     IERR     error codes. None for this routine
```

8.9 Example

The following code fragment shows how to define the Redlich-Kwong equation of state with default (zero) interaction parameters. The RK eos is treated as a variant of the RKS eos.

```
C
C     Codes for models:
C
      INCLUDE 'ayjmid.inc'
C
      INTEGER MXERR
      PARAMETER (MXERR=10)
      INTEGER NERR, IERR(MXERR), NMDL, LMDL(5), LNO
C
C     Initialise models module:
C
      CALL AYMINI( NERR, MXERR, IERR )
      IF ( NERR.NE.0 ) THEN
        PRINT*,'Error initialising models module'
        GO TO 990
      ENDIF
C
C     Load RK model:
C
      NMDL = 5
      LMDL(1) = JMRKSA
      LMDL(2) = JVRK
      LMDL(3) = JVRKS
      LMDL(4) = JVVDW
      LMDL(5) = 0
      CALL AYLOAD( NMDL, LMDL, .TRUE., LNO, NERR, MXERR, IERR )
      IF ( NERR.NE.0 ) THEN
        PRINT*,'Error loading RK model'
        GO TO 990
      ENDIF
```

Further examples are provided in Appendix .

9 Phase descriptors

The idea of a phase descriptor (PD) is central to the operation of all the modules of Multiflash that deal with mixtures. These include single phase mixture properties (section) where the PD is used directly in the subroutine calls and the phase and chemical equilibrium calculations (sections and) where the PDs are encapsuslated in the stream type identifier. The PD contains all the information required to identify a phase and to retrieve its thermophysical properties. A PD must be specified for any phase for which mixture properties are required and for each possible phase in a phase/chemical equilibrium calculation. Usually, only a subset of the list of possible phases will actually be present at equilibrium.

9.1 Setting up phase descriptors

The following steps are required to set up PDs.

9.1.1 Initialise PD system

The PD system is initialised at the same time as the models system by calling AYMINI. See section .

9.1.2 Define PDs

Each phase descriptor is set up by calling AY_DEFINE_PD. This routine returns a PD load number which is subsequently used in calls to all routines that deal with mixtures. It also allows you to specify key components for each PD, as a means to identify it from other phase PDs. The calling specification is as follows:

```
      SUBROUTINE AY_DEFINE_PD( Phid, Nkey, Key_Array, Lnomod, Lnopd, Nerr,
Mxerr, Ierr )

      INCLUDE 'aympd.inc'

      INTEGER, INTENT(IN) :: Phid, Nkey, Lnomod(MXMDPD), Mxerr,
```

```
 5
Key_array(Nkey)
      INTEGER, INTENT(INOUT) :: Lnopd
      INTEGER, INTENT(OUT) :: Nerr, Ierr(Mxerr)
C
C
C     Input arguments:
C     Phid     phase type identifier for the phase descriptor.
C              The phase codes used are defined in AYPTID.INC
C     Nkey     Number of key components for this phase descriptor.
C              Must not be larger than MAXNCS as specified in 'axpara.inc'
C              Nkey should be set to 0 if no key component is required for a
C              particular phase.
C     Key_array Array of key component identifiers for the phase descriptor.
C              The key components are used to label PDs which
C              would otherwise be indistinguishable and/or to control
C              the setting up of new phases in phase stability tests.
C              ABS(Key_array(i)) is the component load number for the i'th key

C              component.
C              A positive value of Key_array(i) means that the component
should
C              be present in the phase with a relatively high concentration
and a
C              negative value of Key_array(i) means that the component should
C              be present in a relatively low concentration.
C              Alternatively, one of the parameters that summarize key
components (in
C              AYJXID.INC) may be given.
C     Lnomod   model load numbers in phase descriptor definitions.
C              The load number for at least one of the properties listed
C              below must be defined. The load numbers for other
C              properties may be set to zero if not required.
C        (1)   fugacity coefficient model load number
C        (2)   volumetric properties model load number
C        (3)   thermal properties model load number
C        (4)   viscosity model load number
C        (5)   thermal conductivity model load number
C        (6)   surface tension model load number
C        (7)   nucleation model load number
C        (8)   diffusivity model load number
C     MXERR    maximum number of error codes to return in IERR
C
C     Input/output arguments:
C     Input values:
C     Lnopd    phase descriptor identifier (0=define new one)
C     Output values:
C     Lnopd    identifier assigned to the defined phase descriptor
C
C     Output arguments:
C     Nerr     number of error codes returned in IERR
C     Ierr     error codes. Values for this routine:
C              10291   key component pointer out of range
C              10292   no free space for any PDs
```

In addition, there also exists the routine AYDFPD which has a similar functionality to AY_DEFINE_PD but only allows for the specification of a single key component per PD. From version 4.4 of Multiflash we recommend the use of AY_DEFINE_PD, however,

5

AYDFPD is still kept in the API for backward compatibility. This routine should be called as follows.

```
      SUBROUTINE AYDFPD( PHID, KEY, LNOMOD, LNOPD, NERR, MXERR, IERR )

      INCLUDE 'aympd.inc'
      INTEGER, INTENT(IN) :: Phid, Key, Lnomod(MXMDPD), Mxerr
      INTEGER, INTENT(INOUT) :: Lnopd
      INTEGER, INTENT(OUT) :: Nerr, Ierr(Mxerr)

C     Input arguments:
C     PHID     phase type identifier for the phase descriptor.
C              The phase codes used are defined in AYPTID.INC
C     KEY      key component identifier for the phase descriptor.
C              The key component is used to label PDs which
C              would otherwise be indistinguishable and/or to control
C              the setting up of new phases in phase stability tests.
C              KEY may be set to zero if no key component is required.
C              Otherwise, ABS(KEY) is the component load number for the
C              key component.
C              If KEY is equal to WATER it gets replaced by the constant
JCM_AQUEOUS
C              defined in 'ayjxid.inc' to extend the key components to any
aqueous
C              components. If key is negative, it gets replaced by the
constant
C              JCM_DEFAULT defined also in 'ayjxid.inc' to indicate that the
phase
C              should not be rich in any 'positive' key components.
C
C     LNOMOD   model load numbers in phase descriptor definitions.
C              The load number for at least one of the properties listed
C              below must be defined. The load numbers for other
C              properties may be set to zero if not required.
C        (1)   fugacity coefficient model load number
C        (2)   volumetric properties model load number
C        (3)   thermal properties model load number
C        (4)   viscosity model load number
C        (5)   thermal conductivity model load number
C        (6)   surface tension model load number
C        (7)   nucleation model load number
C        (8)   diffusivity model load number
C     MXERR    maximum number of error codes to return in IERR
C
C     Input/output arguments:
C     Input values:
C     LNOPD    phase descriptor identifier (0=define new one)
C     Output values:
C     LNOPD    identifier assigned to the defined phase descriptor
C
C     Output arguments:
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
```

For examples of defining and using PDs see the examples in section .

9.2 Getting PD information

9.2.1 PDs defined

The routine AYNPD returns the number of PDs currently defined and their identifiers:

```
      SUBROUTINE AYNPD( NPD, IPD )

      INCLUDE 'aympd.inc'
      INTEGER, INTENT(OUT) :: NPD, IPD(MAXPD)

C
C     Output arguments:
C     NPD      number of phase descriptors defined
C     LNOPD    array of phase descriptor load numbers
```

9.2.2 Phase, key components and models associated with a PD

The routine AY_GET_PD_INFO returns information about a PD:

```
      SUBROUTINE AY_Get_PD_Info(Lnopd, Max_nkey, Phid, Nkey, Key_array,
     & Lnomod, Nerr, Mxerr, Ierr )

      INCLUDE 'aympd.inc'
      INTEGER, INTENT(IN) :: Lnopd, Mxerr, Max_nkey
      INTEGER, INTENT(OUT) :: Phid, Nkey, Lnomod(MXMDPD), Nerr, Ierr(Mxerr)
      INTEGER, INTENT(OUT) :: Key_array(Max_nkey)

C     Input arguments:
C     Lnopd    phase descriptor load (id) number
C     Max_nkey Maximum number of key components expected by caller routine
C     Mxerr    maximum number of error codes to return in IERR
C
C     Output arguments:
C     Phid     phase type identifier for the phase descriptor.
C              The phase codes used are defined in AYPTID.INC
C     Nkey     Number of key components for this phase descriptor.
C              If NKey is zero, no key component has been specified
C     Key_Array Array of KEY components identifiers for the phase descriptor.
C              ABS(Key_Array(i)) indicates a component load number.
C              If Key_Array(i) is positive it means that the component should
be
C              present in the phase in a relatively high concentration. If
C              Key_Array(i) is negative it means that the component should be
C              present in zero or relatively low concentration.
C              Alternatively, one of the parameters that summarize key
components in
C              AYJXID.INC may be returned.
C     Lnomod   model load numbers in phase descriptor definitions.
C              The load numbers for which models are not defined are
C              set to zero.
```

```
 5
C         (1)   fugacity coefficient model load number
C         (2)   volumetric properties model load number
C         (3)   thermal properties model load number
C         (4)   viscosity model load number
C         (5)   thermal conductivity model load number
C         (6)   surface tension model load number
C         (7)   nucleation model load number
C         (8)   diffusivity model load number
C      Nerr     number of error codes returned in IERR
C      Ierr     error codes. Values for this routine:
C               10311   LNOPD not a valid PD identifier
```

Additionally, although not recommended from MF 4.4, the routine AYGTPD can be used to return information about a PD. This routine has only been kept for backward compatibility, as it can only return one key component for each phase descriptor.

```
      SUBROUTINE AYGTPD( LNOPD, PHID, KEY, LNOMOD, NERR, MXERR, IERR )

      INCLUDE 'aympd.inc'
      INTEGER, INTENT(IN) :: LNOPD, MXERR
      INTEGER, INTENT(OUT) :: PHID, KEY, LNOMOD(MXMDPD), NERR, IERR(MXERR)

C
C     Input arguments:
C     LNOPD    phase descriptor load number
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     PHID     phase type identifier for the phase descriptor.
C              The phase codes used are defined in AYPTID.INC
C     KEY      key component identifier for the phase descriptor.
C              KEY is zero if no key component has been specified
C              otherwise ABS(KEY) is a component load number.
C              If KEY is positive it means that the component should be
C              present in the phase in a high concentration. If KEY is
C              negative it means that the component should be present in
C              zero or low concentration.
C              Alternatively, one of the parameters JCMHVY and JCMLIT from
C              AYJXID.INC may be returned.
C              If a key compnent for a given PD was equal to JCM_AQUEOUS
C              (defined in 'ayjxid.inc')
C              and water is present, KEY returns the load number of water.
C              If more than one key component has been defined for a phase
C              (by calling AY_Define_PD) the first key component is returned.
C     LNOMOD   model load numbers in phase descriptor definitions.
C              The load numbers for which models are not defined are
C              set to zero.
C         (1)  fugacity coefficient model load number
C         (2)  volumetric properties model load number
C         (3)  thermal properties model load number
C         (4)  viscosity model load number
C         (5)  thermal conductivity model load number
C         (6)  surface tension model load number
C         (7)  nucleation model load number
C         (8)  diffusivity model load number
C     NERR     number of error codes returned in IERR
```

```
 5
C      IERR     error codes. Values for this routine:
```

9.3 Removing a PD

The maximum number of PDs that can be defined is set by the parameter MAXPD in the include file aympd.inc. If a previously defined PD is no longer needed it may be removed by calling AYRMPD. This call frees storage and allows new PDs to be defined if the maximum had been reached. The calling specification is:

```
      SUBROUTINE AYRMPD( LNOPD, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(INOUT) :: LNOPD
      INTEGER, INTENT(OUT) :: NERR, IERR(*)

C
C     Input arguments:
C     LNOPD    load number of the PD to be removed
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              10301   LNOPD not a valid PD identifier
```

After this call, the PD referred to by LNOPD is no longer defined.

10 Evaluating mixture properties

The routines described in this section evaluate the thermodynamic or transport properties of a mixture in a single phase (vapour, liquid, hydrate,... etc.). Before calling any of the routines it is necessary to define a phase descriptor for each phase of interest. The procedure for doing this is described in section .

10.1 Thermodynamic properties: AYMIX

Calculations of the thermodynamic properties of mixtures using any successfully loaded phase descriptor may be performed by calling the standard interface AYMIX. AYMIX can calculate the fugacity coefficients, volume, enthalpy and entropy together with the derivatives of these quantities with respect to pressure, temperature and mole numbers for a mixture of specified composition and conditions. Any combination of properties and derivatives can be requested with any combination of models previously defined in the phase descriptor. All the derivatives are calculated using analytical expressions. A working example of a complete program is provided in Appendix . The calling specification for AYMIX is as follows:

```
      SUBROUTINE AYMIX( LNOPD, IPHREQ, IPHID, AMBIG, ZFUG, T, P, NC, IC,
     +                  Z, DV, V, VT, VP, VN, DF, FUG, FUGT, FUGP, FUGN,
     +                  IHTYPE, DH, H, HT, HP, HN, S, ST, SP, SN,
     +                  NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOPD, IPHREQ, IPHID, NC, IC(*), IHTYPE, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: T, P
      REAL*8, INTENT(OUT) :: Z(*), V, VT, VP, VN(*), FUG(*), FUGT(*), FUGP(*)
      REAL*8, INTENT(OUT) :: FUGN(*), H, HT, HP, HN(*), S, ST, SP, SN(*), ZFUG

      LOGICAL, INTENT(IN) :: DV(0:3), DF(0:3), DH(0:3)
      LOGICAL, INTENT(OUT) :: AMBIG

C
C     Input arguments:
C     LNOPD    Phase descriptor load number
```

```
 5
C     IPHREQ   Requested phase
C              (parameter values from file ayjxid.inc):
C              JPHDEF  default, i.e as in phase descriptor
C              JPHMGE  phase of minimum Gibbs energy
C              JPHASP  effective phase (Aspen method) for equations of state
C              JPHPSU  as JPHASP but default phase is returned if the model
C                      is able to solve for that phase
C     T        Temperature (K)
C     P        Pressure (Pa)
C     NC       Number of components in subcomponent list IC.
C              0=all components selected from data bank
C     IC       Subcomponent list: array of load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     Z        Mole numbers of components
C     DV       Array of logical flags to control calculation of volume
C              and derivatives. Set elements TRUE to return required
C              quantities, otherwise set FALSE. The volume itself must
C              always be requested if any derivatives are required.
C              DV(0) = volume
C              DV(1) = T derivative
C              DV(2) = P derivative
C              DV(3) = mole number derivatives
C     DF       Array of logical flags to control calculation of fugacity
C              coefficients and derivatives. Set elements TRUE to return
C              required quantities, otherwise set FALSE. The fugacity
C              coefficients themselves must always be requested if any
C              derivatives are required.
C              DF(0) = log fugacity coefficients
C              DF(1) = T derivatives
C              DF(2) = P derivatives
C              DF(3) = mole number derivatives
C     IHTYPE   Total/residual properties flag for enthalpy and entropy
C              (flag values from file AYJXID.INC):
C              JHTTOT = total quantities
C              JHTRES = residual quantities
C              JGEX = excess quantities
C     DH       Array of logical flags to control calculation of enthalpy/
C              entropy and derivatives. Set elements TRUE to return
C              required quantities, otherwise set FALSE. The enthalpy/
C              entropy must always be requested if any derivatives
C              are required.
C              DH(0) = enthalpy and entropy
C              DH(1) = T derivatives
C              DH(2) = P derivatives
C              DH(3) = mole number derivatives
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     IPHID    Phase type found (flag values from file ayptid.inc):
C              UNPHAS = no stable phase found
C              VPHAS = vapour phase
C              LPHAS = liquid phase
C              HPHAS = hydrate phase
C              CPHAS = pure condensed (solid) phase
C              MPHAS = multisolid phase
C              SPHAS = mixed solid phase
C     AMBIG    .TRUE. if fluid supercritical, otherwise .FALSE.
C     ZFUG     Compressibility factor as calculated by the fugacity
```

```
 5
C               coefficient model
C      V        Total volume (m3)
C      VT       Derivative of V wrt temperature
C      VP       Derivative of V wrt pressure
C      VN       Derivative of V wrt mole numbers
C      FUG      Logarithm of the fugacity coefficients (see note below)
C      FUGT     Derivative of FUG wrt temperature
C      FUGP     Derivative of FUG wrt pressure
C      FUGN     Derivative of FUG wrt mole numbers
C      H        Total enthalpy (J)
C      HT       Derivative of H wrt temperature
C      HP       Derivative of H wrt pressure
C      HN       Derivative of H wrt mole numbers
C      S        Total entropy (J K-1)
C      ST       Derivative of S wrt temperature
C      SP       Derivative of S wrt pressure
C      SN       Derivative of S wrt mole numbers
C      NERR     number of error codes returned in IERR
C      IERR     error codes. Values for this routine:
C               10251 cannot match fugacity coefficient phase type in doing
C                     volumetric calculations
C               10252 cannot match fugacity coefficient phase type in doing
C                     enthalpy/entropy calculations
C               10253 requested phase unstable
C               10254 model not valid for fugacity calculation
C               10255 model not valid for volume calculation
C               10256 model not valid for enthalpy, entropy calculation
C               10257 no calculations requested
C               10258 model not valid for activity/GE calculation
C
C      Note on derivatives of fugacity coefficient with respect to the
C      mole numbers:
C      These derivatives form a NCxNC symmetric matrix which is returned
C      by AYMIX in a linear array FUGN. The I,Jth element of the matrix
C      is placed in FUGN(I*(I-1)/2+J) where J is less than or equal to I.
C      Cases where J is greater than I do not have to be returned as the
C      matrix is symmetrical.
C      The dimension of FUGN must be at least FUGN(NC*(NC+1)/2).
```

10.2 Enthalpy and entropy for activity coefficient models

Pure component enthalpy and entropy for activity coefficient models are by default calculated from the perfect gas heat capacity, residual vapour enthalpy/entropy and heat of vaporisation. To calculate enthalpy and entropy for the liquid phase from liquid heat capacity instead, specify the pure component property, CPIDLIQ = 1. This is done by calling AYPSTR with:

```
JVALUE = JCPLIQ
RCONST = 1.D0
```

It is also possible to calculate gas phase enthalpy and entropy from the liquid heat capacity. To do this set CPIDLIQ = 2 by calling AYPSTR with:

```
JVALUE = JCPLIQ
RCONST = 2.D0
```

Load the ACG model (Activity coefficient model for gas phase) as descriped in section , and define this model for the gas phase. The ACG model uses an EoS to calculate fugacity and volumetric properties for the gas phase, and saturated liquid heat capacity and heat of vaporisation to calculate perfect gas thermal properties.

10.3 Average molecular weight: AYAVMW

To calculate the average molecular weight of a mixture call AYAVMW:

```
      SUBROUTINE AYAVMW( NCO, IC, Z, D, AVMW, AVMWN, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: Z(*)
      REAL*8, INTENT(OUT) :: AVMW, AVMWN(*)
      LOGICAL, INTENT(IN) :: D

C
C     Input arguments:
C     NCO       Number of components in subcomponent list IC.
C               0=all components selected from data bank
C     IC        Subcomponent list: array of load numbers of the
C               components selected from the data bank.
C               For NC=0, values of IC ignored.
C     Z         Mole numbers of components
C     D         .TRUE.=component molecular weights returned in AVMWN,
C                  otherwise AVMWN values are undefined
C     MXERR     Maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     AVMW      Average molecular weight
C     AVMWN     Component molecular weights
C     NERR      No. errors reported by this routine and lower level routines
C     IERR      Array of error codes. Values returned by this routine:
C               10091   Undefined molecular weight
C               10092   Negative mole number
C               10093   zero total number of moles
```

10.4 Viscosity: AYVISC

To obtain the viscosity of a phase and its analytical derivatives call AYVISC:

```
5
      SUBROUTINE AYVISC( LNOPD, T, P, NC, IC, Z, DER, VIS, VIST,
     +                   VISP, VISN, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOPD, NC, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: T, P, Z(*)
      REAL*8, INTENT(OUT) :: VIS, VIST, VISP, VISN(*)
      LOGICAL, INTENT(IN) :: DER(3)

C
C     Input arguments:
C     LNOPD    Phase descriptor load number
C     T        temperature (K)
C     P        pressure (Pa)
C     NC       number of components in subcomponent list IC.
C              0=all components selected from data bank
C     IC       subcomponent list: array of load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     Z        mole numbers of components
C     DER      derivative flags. Set elements TRUE to
C              return required derivatives, otherwise set FALSE:
C              DER(1) = T derivatives
C              DER(2) = P derivatives
C              DER(3) = mole number derivatives
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     VIS      viscosity (Pa s)
C     VIST     derivative of VIS wrt temperature
C     VISP     derivative of VIS wrt pressure
C     VISN     derivative of VIS wrt mole numbers
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C             -10201  phase descriptor has no defined viscosity model
C              10201  load number does not refer to a valid
C                     viscosity model
C              10202  failure in viscosity model
C              10203  requested phase type does not exist (unstable)
```

10.5 Thermal conductivity: AYTCND

To obtain the thermal conductivity of a phase and its analytical derivatives call AYTCND:

```
      SUBROUTINE AYTCND( LNOPD, T, P, NC, IC, Z, DER, TCN, TCNT,
     +                   TCNP, TCNN, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOPD, NC, IC(*), MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: T, P, Z(*)
      REAL*8, INTENT(OUT) :: TCN, TCNT, TCNP, TCNN(*)
      LOGICAL, INTENT(IN) :: DER(3)

C
C     Input arguments:
```

```
 5
C     LNOPD    Phase descriptor load number
C     T        temperature (K)
C     P        pressure (Pa)
C     NC       number of components in subcomponent list IC.
C              0=all components selected from data bank
C     IC       subcomponent list: array of load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     Z        mole numbers of components
C     DER      derivative flags. Set elements TRUE to
C              return required derivatives, otherwise set FALSE:
C              DER(1) = T derivatives
C              DER(2) = P derivatives
C              DER(3) = mole number derivatives
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     TCN      thermal conductivity (W/m K)
C     TCNT     derivative of TCN wrt temperature
C     TCNP     derivative of TCN wrt pressure
C     TCNN     derivative of TCN wrt mole numbers
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C             -10241  phase descriptor has no defined thermal conductivity
model
C              10241  load number does not refer to a valid
C                     thermal conductivity model
C              10242  failure in thermal conductivity model
C              10243  requested phase type does not exist (unstable)
```

10.6 Surface tension: AYSURF

To obtain the surface tension of a phase and its analytical derivatives call AYSURF. The surface tension routine differs from all other property routines in that it accepts two phase descriptor load numbers. This is because the surface tension depends on the properties of two phases. The load numbers LNOPD1 and LNOPD2 must refer to PDs with the same surface tension model. However, the surface tension models may refer to different subsidiary models such as models for volumetric properties. In practise, most engineering models are one phase models that ignore the gas phase - examples include the ST1 mixing rule and the one-phase variant of the MCS model. In such cases LNOPD2 should be set to zero.

The Linear Gradient Theory, which always needs the definition of two phases, will fail if only one phase load number is passed.

```
      SUBROUTINE AYSURF( T, P, NC, IC, LNOPD1, Z1, LNOPD2,
     +                   Z2, DER, ST, STT, STP, STN1, STN2,
     +                   NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NC, IC(*), LNOPD1, LNOPD2, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)
      REAL*8, INTENT(IN) :: T, P, Z1(*), Z2(*)
      REAL*8, INTENT(OUT):: ST, STT, STP, STN1(*), STN2(*)
```

```
5
      LOGICAL, INTENT(OUT) :: DER(4)

C
C     Input arguments:
C     T        temperature (K)
C     P        pressure (Pa)
C     NC       number of components in subcomponent list IC.
C              0=all components selected from data bank
C     IC       subcomponent list: array of load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     LNOPD1   Phase descriptor load number for phase 1 (usually the
C              the liquid phase)
C     Z1       mole numbers of components in phase 1
C     LNOPD2   Phase descriptor load number for phase 2 (usually the
C              gas phase for a 2-phase model, should be set to zero if
C              a one-phase model)
C     Z2       mole numbers of components in phase 2 (unused if
C              LNOPD2=0)
C     DER      derivative flags. Set elements TRUE to
C              return required derivatives, otherwise set FALSE:
C              DER(1) = T derivatives
C              DER(2) = P derivatives
C              DER(3) = mole number derivatives for phase 1
C              DER(4) = mole number derivatives for phase 2
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     ST       surface tension (N m-1)
C     STT      derivative of ST wrt temperature
C     STP      derivative of ST wrt pressure
C     STN1     derivative of ST wrt mole numbers in phase 1
C     STN2     derivative of ST wrt mole numbers in phase 2
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C             -10221  phase descriptor has no defined surface tension model
C              10221  load number does not refer to a valid
C                     surface tension model
C              10222  failure in ST model
C              10223  load numbers for ST models are different for the
C                     2 phases
C              10224  invalid option for ST model
```

10.7 Binary diffusion coefficient: AYDIFF

To obtain the binary diffusion coeffients of components in a fluid phase and the analytical derivatives call AYDIFF:

```
      SUBROUTINE AYDIFF( LNOPD, T, P, NCDIM, NC, IC, Z, DER, DC, DCT,
     +                   DCP, DCN, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOPD, NC, IC(*), NCDIM,  MXERR
      INTEGER, INTENT(OUT) :: NERR,  IERR(*)
      REAL*8, INTENT(IN) :: T, P, Z(*)
      REAL*8, INTENT(OUT) :: DC(NCDIM,*), DCT(NCDIM,*), DCP(NCDIM,*),
     +       DCN(NCDIM,NCDIM,*)
```

```
5
      LOGICAL, INTENT(IN) :: DER(3)

C
C     Input arguments:
C     LNOPD    Phase descriptor load number
C     T        Temperature (K)
C     P        Pressure (Pa)
C     NCDIM    Dimension of diffusion coefficient array DC declared in
C              calling routine. NCDIM must be greater than or equal to the
C              the number of components for the stream type.
C     NC       Number of components in subcomponent list IC.
C              0=all components selected from data bank
C     IC       Subcomponent list: array of load numbers of the
C              components selected from the data bank.
C              For NC=0, values of IC ignored.
C     Z        Mole numbers of components (ignored by models that do not
C              require the fluid composition as input)
C     DER      Derivative flags. Set elements TRUE to
C              return required derivatives, otherwise set FALSE:
C              DER(1) = T derivatives
C              DER(2) = P derivatives
C              DER(3) = mole number derivatives
C     MXERR    Maximum number of error codes to return in IERR
C
C     Output arguments:
C     DC       Diffusion coefficients
C     DCT      Derivative of DC wrt temperature
C     DCP      Derivative of DC wrt pressure
C     DCN      Derivative of DC wrt mole numbers
C     NERR     Number of error codes returned in IERR
C     IERR     Error codes. Values for this routine:
C             -10381  phase descriptor has no defined diffusivity model
C              10381  load number does not refer to a valid diffusivity model
C              10382  failure in diffusivity model
C              10383  requested phase type does not exist (unstable)
```

10.8 Specification of datum: AYDATM

Set the enthalpy and entropy datum for the specified components by calling: AYDATM:

```
      SUBROUTINE AYDATM( NCO, IC, HDATM, SDATM,
     +                   NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NCO, IC(*), HDATM, SDATM, MXERR
      INTEGER, INTENT(OUT) :: NERR, IERR(*)

C     Input arguments:
C     NCO      Number of components in subcomponent list.
C              If zero the datum is set for all components.
C     IC       subcomponent list
C     HDATM    enthalpy datum (codes in ayjxid.inc)
C              JHSCOM  zero for compounds at 298.15 K for pg
C              JHSELE  enthalpy of formation in PG phase 298.15 K, 1 bar
C     SDATM    entropy datum (codes in ayjxid.inc)
C              JHSCOM  zero for compounds at 298.15 K for pg
```

```
 5
C              JHSELE  entropy of formation in PG phase 298.15 K, 1 bar
C              JHSSTD  standard entropy at 298.15 K, 1 bar
C
C     Output arguments:
C     NERR     no. errors reported by this routine and lower level routines
C     IERR     Array of error codes. Values returned by this routine:
C              16921   code for enthalpy datum is not recognised
C              16922   code for entropy datum is not recognised
```

11 Streams

A stream is defined as a collection of components and phase descriptors. Multiflash uses the concept of a stream when carrying out phase and chemical equilibrium calculations. The components in a stream may be a subset of all the components defined. The PDs in a stream define a list of the possible phases that may be present at equilibrium. It is not necessary to define a stream explicitly in order to use the phase/chemical equilibrium routines. The default stream identifier of 0 includes all the currently defined PDs and components.

11.1 Setting up streams

The following steps are required to set up streams.

11.1.1 Initialise stream type system

The stream type system is initialised at the same time as the phase equilibrium module by making a call to AXINIT. See section .

11.1.2 Define stream types

Each stream type is set up by calling AXDFST. This routine returns a stream load number which is subsequently used in calls to all routines that deal with equilibrium calculations. The calling specification is as follows:

```
      SUBROUTINE AXDFST( NC, IC, NPD, IPD, IST, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: NC, IC(*), NPD, IPD(*), MXERR
      INTEGER, INTENT(OUT) :: IST, NERR, IERR(*)

C
C     Input arguments:
C     NC       number of components in stream type
C              (-1=use all defined components)
```

```
5
C     IC        subcomponent list for stream type (array of component
C               load numbers). Not used if NC = -1
C     NPD       number of phase descriptors in stream type
C               (-1=use all defined PDs)
C     IPD       List of phase descriptors in stream type
C     MXERR     Maximum number of error codes to return in IERR
C
C     Output arguments:
C     IST       Identifier assigned to the defined stream type
C     NERR      Number of error codes returned in IERR
C     IERR      Error codes. Values for this routine:
C               20421   Too many components defined in subcomponent list
C               20422   Invalid subcomponent number
C               20423   Repeated subcomponent number
C               20424   Too many phase descriptors defined
C               20425   Invalid phase descriptor load number
C               20426   Repeated phase descriptor load number
C               20427   No free space for any stream descriptor
```

11.2 Getting stream type information

11.2.1 Number of stream types defined

The routine AXNST returns the number of stream types currently defined and their identifiers:

```
      SUBROUTINE AXNST( NST, IST )

      INCLUDE 'axpara.inc'
      INTEGER, INTENT(OUT) :: NST, IST(MAXST)

C
C     Output arguments:
C     NST       Number of stream types defined
C     IST       Stream type identifiers
```

11.2.2 Phase descriptors and components associated with a stream type

The routine AXGTST returns information about a stream type:

```
      SUBROUTINE AXGTST( IST, NC, IC, NPD, IPD, NERR, MXERR, IERR )

      INCLUDE 'aympd.inc'
      INCLUDE 'axpara.inc'
      INTEGER, INTENT(IN) :: IST, MXERR
      INTEGER, INTENT(OUT) :: NC, IC(MAXNCS), NPD, IPD(MAXPD), NERR, IERR(*)

C
C     Input arguments:
C     IST       Stream type identifier
```

```
 5
C     MXERR    Maximum number of error codes to return in IERR
C
C     Output arguments:
C     NC       number of components in stream type
C     IC       subcomponent list (array of component load numbers)
C              for stream type
C     NPD      number of phase descriptors in stream type
C     IPD      List of phase descriptors in stream type
C     NERR     Number of error codes returned in IERR
C     IERR     Error codes. Values for this routine:
C              20441   IST not a valid stream type identifier
```

11.3 Removing a stream type

The maximum number of stream types that can be defined is set by the parameter MAXST in the include file axpara.inc. If a previously defined stream type is no longer needed it may be removed by calling AXRMST. This call frees storage and allows new stream types to be defined if the maximum had been reached. The calling specification is:

```
      SUBROUTINE AXRMST( IST, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: MXERR
      INTEGER, INTENT(INOUT) :: IST
      INTEGER, INTENT(OUT) :: NERR, IERR(*)

C
C     Input arguments:
C     IST      Identifier of the stream type to be removed. It is set to zero
on output.
C     MXERR    Maximum number of error codes to return in IERR
C
C     Output arguments:
C     NERR     Number of error codes returned in IERR
C     IERR     Error codes. Values for this routine:
C              20431   IST not a valid stream type identifier
```

After this call, the stream type referred to by IST is no longer defined.

12 Phase equilibrium calculations

12.1 Introduction

The Multiflash phase equilibrium module can handle any number of vapour, liquid, and solid phases. It performs the full range engineering flash calculations for multiphase situations including:

- flashes at fixed pressure and temperature;
- isenthalpic flashes at fixed pressure or temperature;
- isentropic flashes at fixed pressure or temperature;
- isochoric flashes at fixed pressure or temperature;
- flashes at fixed internal energy and pressure or temperature;
- fixed (molar) phase fraction flashes (e.g. dew or bubble points) at defined pressure or temperature.
- fixed mass phase fraction flashes at defined pressure or temperature.
- fixed volume phase fraction flashes at defined pressure or temperature.

In addition, Multiflash offers ‘double’ flashes where two variables are adjusted to match two constraints and tolerance calculations where a second fluid is added to the feed to match a constraint.

The solution procedures used in Multiflash are extensions of methods described by Michelsen:

M. L. Michelsen, The isothermal flash problem. Part I. Stability, Fluid Phase Equilibria, 9, 1-19 (1982).

M. L. Michelsen, The isothermal flash problem. Part II. Phase-split calculations, Fluid Phase Equilibria, 9, 21-40 (1982).

12.2 Multiflash operation

Multiflash is designed to allow use of different thermodynamic models for different phases and for different properties of a given phase. The solution methods used do not make any assumptions about thermodynamic consistency between fugacity coefficients, volumetric properties, and thermal properties. It is the implementer's responsibility to ensure that the same fugacity reference state is used for all phases and that a sensible combination of models for different phases is specified.

A typical sequence of operations for using Multiflash is:

1. Initialise physical properties data system, define databanks and load the components (section).
2. Define or load sets of BIPs (section).
3. Specify and initialise the set of models that will be available (section).
4. Initialise phase descriptor system and set up all the phase descriptors (PDs) which Multiflash may use (section).
5. Initialise the stream type system and the phase equilibrium module by calling AXINIT.
6. Set up all the stream types required (section).
7. Algorithm parameters for Multiflash eg. (convergence tolerances or iteration limits) may be set to values different from the internal defaults by calling any of the following entry functions:
   a) AXSE01 parameters for the 1st order phase split calculation
   b) AXSE02 parameters for the 2nd order phase split calculation
   c) AXSE03 parameters for the stability test
   d) AXSE04 parameters for the T/P search algorithms
   See section for guidelines on setting algorithm parameters to enhance convergence in difficult regions or to reduce computation time.
8. Call one of the top level routines to do a calculation:
   a) AXFL01 isothermal flash
   b) AXFL02 fixed phase fraction (FPF), isenthalpic, isentropic, isochoric flash or
   c) flash at constant internal energy
   d) AXFL04 double flash (HV, UV, etc.
   e) AXST01 stability test on a mixture (all flashes include stability tests, this routine is
9. intended for doing a stability test without a flash)

Multiflash is designed to work with a subset of the previously defined components and PDs. These subsets may be specified in the calls to AXFL01 etc. through the stream type load number.

12.3 Thermodynamic property interface

The maximum number of components that Multiflash can handle is set by the parameter MAXNCS. The maximum number of phases that may be present at equilibrium is set by the parameter MAXPHA. Both parameters are defined in the include file axpara.inc. Note that the maximum number of phases that may be considered (as possible equilibrium phases) by Multiflash is set by the parameter MAXPD in the file aympd.inc.

The algorithms used in Multiflash do not assume that the same thermodynamic model is used to represent all phases, or that different properties such as the enthalpy and the fugacity are mutually consistent. The characteristics required of the property package are:

- A model may return one or more of the following three classes of properties: fugacity coefficients; volume; enthalpy and entropy. An enthalpy model must also handle the entropy.
- A model may evaluate the temperature, pressure and mole number derivatives of the property for which it is defined. Finite difference derivatives are not adequate unless they are indistinguishable from analytic derivatives. Multiflash will operate with or without property derivatives.

  If mole number derivatives of the fugacity coefficients are available, the second order minimisation method can be used in the isothermal flash. Variable I2ORD, set by calling AXSE02 controls whether the second order method is used. Most problems can be solved reliably without use of the second order method. The default behaviour is recomended as the second order method is only called if needed.
- A model must be able to return properties for a specified phase type or, if the model can describe more than one phase, for the phase of minimum Gibbs energy. This is only meaningful for equation-of-state type models. Models which can only represent one phase type, eg. activity models, should always regard that phase as the one of minimum Gibbs energy. If a specific phase is requested and the model cannot represent that phase, it should return an error flag.
- For models based on equations of state, the phase type returned by the model may be ambiguous. The model should be able to detect whether this is the case, i.e. whether a van der Waals loop exists or not, and return a status flag. For non-eos type models the phase type should always be identified as unambiguous.
- Every model must be related to a well-defined reference state for the fugacity coefficients and thermal properties. This will usually be the perfect gas state, with the fugacity coefficients having their conventional interpretation. The

model description must contain enough information to relate the model to the reference state. For example, a liquid-phase activity model must have an accompanying equation of state model that relates the liquid properties to the perfect gas reference state.

- Multiflash is designed to work with subsets of the total list of components defined to the property package. To use this feature the models must be able to return properties for a subset of components. If it is not possible to work on subsets then Multiflash should always be called with a full component list.

Details of the models routines provided by Infochem are given in section .

12.4 Initialising the phase equilibrium module

Once the databank(s), components, models and phase descriptors have been defined, Multiflash and the stream type system should be initialised by calling AXINIT. The calling specification is as follows:

```
 SUBROUTINE AXINIT( NERR, MXERR, IERR )

 INTEGER, INTENT(IN) :: MXERR,
 INTEGER, INTENT(OUT) :: NERR, IERR(*)
```

It is not necessary to set up stream types explicitly if only one stream is to be used. The default steam type identifier of zero may be used. It includes all components and all PDs defined. If it is required to use a subcomponent list or a subset of the defined PDs it is necessary to define suitable suitable stream types explicitly by calling the routines described in section .

12.5 Flash at fixed pressure and temperature: AXFL01

AXFL01 calculates a flash at fixed pressure and temperature. This combination of variables leads to a unique solution of the phase equilibrium problem. The calling specification is:

```
 SUBROUTINE AXFL01( LNOST, T, P, NCDIM, NMOLIN, NPHMAX, FEST1,
+                   ISTABI, NP, LNOPD, X, NMOLPH, ZCOMP, IPHID,
+                   ISTABO, NPRINT, NERR, MXERR, IERR )

 INTEGER, INTENT(IN) :: LNOST, NCDIM, NPHMAX, ISTABI, NPRINT, MXERR
 INTEGER, INTENT(INOUT) :: NP, LNOPD(*)
 INTEGER, INTENT(OUT) :: IPHID(*),
+        ISTABO, NERR, IERR(*)
 REAL*8, INTENT(IN) :: T, P, NMOLIN(*)
 REAL*8, INTENT(INOUT) :: X(NCDIM,*), NMOLPH(*)
```

```
      REAL*8, INTENT(OUT) :: ZCOMP(*)
      LOGICAL, INTENT(IN) :: FEST1

C
C     Input arguments:
C     LNOST    stream type load number for flash (obtained by calling
C              AXDFST). This defines the phase descriptors (PDs) and
C              components to be considered in the flash. If set to zero
C              this means that all PDs and components will be included.
C     T        temperature (K)
C     P        pressure (Pa)
C     NCDIM    dimension of mole fraction array X declared in calling
C              routine. NCDIM must be greater than or equal to the
C              the number of components for the stream type.
C     NMOLIN   number of moles of each component in the feed
C     NPHMAX   The maximum number of coexisting phases that the flash
C              calculation is to search for. The maximum number of
C              phases that may actually be returned by AXFL01 is the
C              minimum of NPHMAX, the number of phases in the stream
C              type (NPD) and MAXPHA (the maximum number of coexisting
C              phases for which AXFL01 is set up). The parameter MAXPHA
C              is defined in the file axpara.inc. If NPHMAX is outside
C              the range 1 to MAXPHA it is equivalent to setting it to
C              MAXPHA.
C              The arrays LNOPD, X, NMOLPH, ZCOMP and IPHID must be
C              dimensioned to at least MIN( NPHMAX, NPD, MAXPHA ) in
C              the calling routine.
C     FEST1    logical flag indicating whether an estimate of the
C              solution is supplied.
C              .TRUE.  values of NP, LNOPD, X and NMOLPH must be supplied.
C              .FALSE. the initial phase distribution is obtained from a
C                      stability analysis using the phase descriptors
C                      specified in the stream type.
C     ISTABI   controls whether a stability test is done on a final
C              solution that contains all PDs in the stream type.
C              Omitting the stability test saves time but an
C              erroneous solution may be returned.
C              (codes are defined in axjnum.inc)
C              JXNOSC   no final stability check
C              JXDOSC   do final stability check
C     NPRINT   diagnostic output flag.
C              -1   no output
C              0    minimum output in case of serious errors
C              1    errors and serious warnings
C              2    warnings plus above
C              3    informative messages plus above
C     MXERR    maximum number of error codes to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     NP       If FEST1 is .TRUE. NP must be set to the number of
C              phases present at the estimate of the solution supplied
C              in LNOPD, X and NMOLPH.
C     LNOPD    If FEST1 is .TRUE. LNOPD must be set on entry.
C              Array of NP PD load numbers corresponding to input
C              values in XX and NMOLPH.
C     X        If FEST1 is .TRUE. X must be set on entry.
C              See below for full description.
C     NMOLPH   If FEST1 is .TRUE. NMOLPH must be set on entry.
```

```
 5
C               See below for full description.
C
C     Output values:
C     NP        number of phases at equilibrium or at best estimate of
C               solution. When an estimate of the solution is supplied
C               some of the input phases may be eliminated.
C     LNOPD     Array of NP PD load numbers corresponding to output
C               values in XX, NMOLPH, ZCOMP and IPHID.
C     X         X(I,J) holds the mole fraction of component I in phase
C               J for J = 1,NP.
C     NMOLPH    NMOLPH(J) holds the number of moles in phase J for
C               J = 1,NP.
C
C     Output arguments:
C     ZCOMP     compressibility factor of each output phase (as
C               calculated by the fugacity model, not the volumetric
C               model).
C     IPHID     identity (V,L,..etc) of each output phase. If the
C               calculation is successful, the phase id. is set to that
C               specified in the phase descriptor definition. Note that
C               for supercritical phases described by equations of state,
C               the phase label is ambiguous.
C               The phase codes used are defined in AYPTID.INC
C     ISTABO    stability indicator for solution.
C               (codes are defined in axjnum.inc)
C               JXSTUN  definitely unstable. Another phase would be
C                       formed or an existing phase removed if problem
C                       specification allowed.
C               JXSTMU  marginally unstable (see above).
C               JXSTIN  indeterminate.
C                       This value is returned if no final stability
C                       test was requested (ISTABI=JXNOSC) and no
C                       other error condition was detected.
C                       If ISTABI=JXDOSC it means that the solution is
C                       on the stability limit.
C               JXSTMS  marginally stable.
C               JXSTST  definitely stable.
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               -20001 unstable solution returned.
C                      Phase split converged but solution is not stable.
C                      All specified phase descriptors have been used up
C                      or the maximum number of phases has been reached
C                      so no further phase split can be done.
C               -20002 more phases found than requested. Best solution
C                      with NPD phases returned (unstable).
C               -20003 unstable solution returned.
C                      Cannot allocate PDs to the phase distribution.
C                      i.e. the actual phase split cannot be described by
C                      the specified phase types.
C                      Values returned correspond to the last phase split
C                      with an acceptable set of phases but this phase
C                      distribution is unstable.
C               -20004 phase split converged to low accuracy
C               -20005 cannot allocate PDs to the phase distribution
C                      i.e. the actual phase split cannot be described by
C                      the specified phase types.
C               20001  illegal input specification negative or zero
C                      P or T
```

```
5
C              20002  illegal input specification reported by startup
C                     routine. Negative or zero:
C                     number of moles of feed; feed mole
C                     fractions; number of phase descriptors;
C                     phase descriptors wrongly specified in input or
C                     wrongly set up prior to calling AXFL01.
C              20003  fatal error finding most stable single phase.
C              20004  fatal error in stability analysis
C              20005  PD identifier has unacceptable value
C              20006  fatal error in phase split calculation.
C              20007  reached maximum number of cycles of stability
C                     analysis/phase split
```

12.6 Temperature or pressure searches and tolerance calculations: AXFL02

AXFL02 performs flashes where one variable is adjusted to satisfy a constraint. The variables may be either the temperature, the pressure or, for tolerance calculations, the amount of a second fluid mixture (expressed as a molar fraction) that should be added to the feed. The possible constraints are:

- Total enthalpy H
- Total entropy S
- Total volume V
- Total internal energy U
- Phase fraction (molar, mass or volume basis)
- Nucleation rate

The phase fraction can be fixed for any phase for which a phase descriptor has been defined in the stream type passed to AXFL02 and includes dew and bubble point calculations as special cases. On convergence, the pressure, temperature or fluid fraction is returned together with the amounts and compositions of the phases present at equilibrium.

For tolerance calculations, a combination ratio, t, is returned in the argument VAR, which is defined such that:

$$n_i = n_{i^{feed}}(1-t) + n_{i^{add}} t \quad (1)$$

where $n_{i^{feed}}$ is the number of moles of component i in the feed, and $n_{i^{add}}$ is the number of moles of component i in the fluid mixture that is to be added to the feed. For a tolerance calculation, the array NMOLIN passed to AXFL02 must be doubly dimensioned to carry both compositions: NMOLIN(I,1) is assigned the value of $n_{i^{feed}}$ and NMOLIN(I,2) is assigned the value of $n_{i^{add}}$. For non-tolerance calculations, NMOLIN may be either singly or doubly dimensioned and only carries the feed compositions either as NMOLIN(I) or NMOLIN(I,1).

If the same total number of moles is defined in each of the two fluids, t is equal to the molar fraction of fluid 2 relative to the

mixed fluid, and the molar amount of the second fluid to be added to obtain the specification is then given by:

$$N_2 = N^{feed}\frac{t}{1-t} \quad (2)$$

where N^{feed} is the sum of $n_{i^{feed}}$. If mole fractions are passed in, N_2 in equation is the amount of the second fluid to add per mole feed.

Only certain combinations of specifications handled by AXFL02 lead to a phase equilibrium problem that has a unique solution. These combinations are: p,H ; p,S and V,T . The remaining specifications may have more than one solution or no solution.

One of the most common applications for AXFL02 is to calculate the dew point or bubble point of a mixture. For multi-phase systems the best way to specify a dew point calculation is to set phase fraction of the vapour phase PD to 1.0 . This specification identifies the primary dew point. To find the dew point of a specific liquid phase, e.g. the water dew point, the target phase fraction of the PD corresponding to an aqueous phase should be set to 0.0 . For a bubble point calculation the target fraction of the gas phase PD should be set to 0.0. Note that there may not be any solution or multiple solutions corresponding to these specifications.

In addition, AXFL02 can be called at fixed temperature and pressure to return the deviation index. The deviation index is a smoothly varying function that shows how far the constrained equilibrium is from satisfying the constraint at the defined pressure or temperature. If the contraint is satisfied, the deviation index is zero. (The search procedures in AXFL02 operate by adjusting either temperature, pressure or amount of second fluid until the deviation index is zero.)

The calling specification for AXFL02 is:

```
      SUBROUTINE AXFL02( LNOST, IVAR, T, P, VAR, NCDIM, NMOLIN,
     +                   NPHMAX, IPROB, TARGET, FEST1, FEST2, ISTABI,
     +                   NP, LNOPD, XX, NMOLPH, ZCOMP, IPHID, ISTABO,
     +                   NPRINT, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, IVAR, NCDIM, NPHMAX, IPROB(3), ISTABI,
     +                       NPRINT, MXERR
      INTEGER, INTENT(INOUT) :: NP, LNOPD(*)
      INTEGER, INTENT(OUT) :: IPHID(*), ISTABO, NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(NCDIM,*), TARGET
      REAL*8, INTENT(INOUT):: T, P, VAR, XX(NCDIM,*), NMOLPH(*)
      REAL*8, INTENT(OUT) :: ZCOMP(*)
      LOGICAL, INTENT(IN) :: FEST1, FEST2

C
C     Input arguments:
C     LNOST    stream type load number for flash (obtained by calling
C              AXDFST). This defines the phase descriptors (PDs) and
C              components to be considered in the flash. If set to zero
C              this means that all PDs and components will be included.
C     IVAR     code denoting variable to search for:
C              (codes are defined in axjnum.inc)
C              JXTEMP  temperature
C              JXPRES  pressure
```

```
5
C               JXTOL   tolerance variable
C               JXDVIX  deviation index
C     NCDIM     dimension of mole fraction arrays NMOLIN and XX declared
C               in calling routine. NCDIM must be greater than or equal
C               to the the number of components for the stream type.
C     NMOLIN    number of moles of each component in the feed and in the
C               fluid mixture to be added in tolerance calculations.
C     NPHMAX    The maximum number of coexisting phases that the flash
C               calculation is to search for. The maximum number of
C               phases that may actually be returned by AXFL02 is the
C               minimum of NPHMAX, the number of phases in the stream
C               type (NPD) and MAXPHA (the maximum number of coexisting
C               phases for which AXFL02 is set up). The parameter MAXPHA
C               is defined in the file axpara.inc. If NPHMAX is outside
C               the range 1 to MAXPHA it is equivalent to setting it to
C               MAXPHA.
C               The arrays LNOPD, X, NMOLPH, ZCOMP and IPHID must be
C               dimensioned to at least MIN( NPHMAX, NPD, MAXPHA ) in
C               the calling routine.
C     IPROB     identifer for type of calculation required
C               IPROB(1) = problem type
C               (codes are defined in axjnum.inc)
C                 JXENTH   total enthalpy (J)
C                 JXENTR   total entropy (J K-1)
C                 JXVOL    total volume (m3)
C                 JXINTE   total internal energy (J)
C                 JXPHML   phase molar fraction
C                 JXPHMS   phase mass fraction
C                 JXPHVL   phase volume fraction
C                 JXNUCL   phase nucleation
C               IPROB(2) = fixed phase load number if phase fraction problem
C               IPROB(3) = upper/lower phase fraction search
C               (codes are defined in axjnum.inc)
C                 JSNL     lower normal solution
C                 JSRU     upper retrograde solution
C                 JSNU     upper normal solution
C                 JSRL     lower retrograde solution
C                 JSANY    any solution
C                 JSDEF    default normal solution
C
C               The normal settings for dew/bubble points are:
C                                          Search for    setting
C               Normal dew point               T         upper
C                                              P         lower
C
C               Retrograde dew point           T         lower
C                                              P         upper
C
C               Normal bubble point            T         lower
C                                              P         upper
C
C               Retrograde bubble point        T         upper
C                                              P         lower
C
C               For phase fractions between 0 and 1 the best setting
C               depends on the shape of the phase diagram.
C
C               It is important to specify the correct search code
C               because the value of IPROB(3) is used as a guide, both to
```

```
C               generate initial estimates of T or P, and also to decide
C               on a search direction towards the required solution.
C               Particularly for difficult calculations, the user needs
C               to have an idea of the layout of the phase diagram.
C               Note that the JSANY specification is less reliable than
C               any of the others because the routine has no guidance
C               about where to look for a solution.
C     TARGET    function values to be matched
C     FEST1     first estimates of the phase fractions and compositions
C               at the solution must be supplied if FEST1 is .TRUE.
C               If FEST1 is .FALSE. the initial phase distribution
C               is obtained from a stability analysis using the
C               phase descriptors specified in the stream type.
C     FEST2     first estimates of the variable (temperature, pressure or
C               tolerance ratio ) at the solution must be supplied if FEST2
C               is .TRUE. If FEST2 is .FALSE. the initial value of T or P
C               is obtained using an internal procedure.
C     ISTABI    controls whether a stability test is done on a final
C               solution that contains all PDs in the stream type.
C               Omitting the stability test saves time but an
C               erroneous solution may be returned.
C               (codes are defined in axjnum.inc)
C               JXNOSC  no final stability check
C               JXDOSC  do final stability check
C     NPRINT    diagnostic output flag.
C               -1   no output
C               0    minimum output in case of serious errors
C               1    errors and serious warnings
C               2    warnings plus above
C               3    informative messages plus above
C     MXERR     maximum number of error codes to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     T         if IVAR=JXPRES or JXTOL, temperature (K),
C               if IVAR=JXTEMP and FEST2=TRUE, estimated temperature (K),
C               else not defined
C     P         if IVAR=JXTEMP or JXTOL, pressure (Pa),
C               if IVAR=JXPRES and FEST2=TRUE, estimated pressure (Pa),
C               else not defined
C     VAR       If IVAR=JXTOL and FEST2=TRUE, estimated tolerance
C               variable else not defined
C     NP        If FEST1 is .TRUE. NP must be set to the number of
C               phases present at the estimate of the solution supplied
C               in LNOPD, XX and NMOLPH.
C     LNOPD     If FEST1 is .TRUE. LNOPD must be set on entry.
C               Array of NP PD load numbers corresponding to input
C               values in XX and NMOLPH.
C     XX        If FEST1 is .TRUE. XX must be set on entry.
C               See below for full description.
C     NMOLPH    If FEST1 is .TRUE. NMOLPH must be set on entry.
C               See below for full description.
C
C     Output values:
C     T         Temperature (K)
C     P         Pressure (Pa)
C     VAR       Tolerance variable, if IVAR=JXTOL,
C               deviation index, if IVAR=JXDVIX.
C     NP        number of phases at equilibrium or at best estimate of
```

```
C               solution. When an estimate of the solution is supplied
C               some of the input phases may be eliminated.
C     LNOPD     Array of NP PD load numbers corresponding to output
C               values in XX, NMOLPH, ZCOMP and IPHID.
C     XX        XX(I,J) holds the mole fraction of component I in phase
C               J for J = 1,NP.
C     NMOLPH    NMOLPH(J) holds the number of moles in phase J for
C               J = 1,NP. A zero value will be returned for phases at a
C               saturation point (eg. a dew point or bubble point).
C
C     Output arguments:
C     ZCOMP     compressibility factor of each output phase (as
C               calculated by the fugacity model, not the volumetric
C               model).
C     IPHID     identity (V,L,..etc) of each output phase. If the
C               calculation is successful, the phase id. is set to that
C               specified in the phase descriptor definition. Note that
C               for supercritical phases described by equations of state,
C               the phase label is ambiguous.
C               The phase codes used are defined in AYPTID.INC
C     ISTABO    stability indicator for solution.
C               (codes are defined in axjnum.inc)
C               JXSTUN  definitely unstable. Another phase would be
C                       formed or an existing phase removed if problem
C                       specification allowed.
C               JXSTMU  marginally unstable (see above).
C               JXSTIN  indeterminate.
C                       This value is returned if no final stability
C                       test was requested (ISTABI=JXNOSC) and no
C                       other error condition was detected.
C                       If ISTABI=JXDOSC it means that the solution is
C                       on the stability limit.
C               JXSTMS  marginally stable.
C               JXSTST  definitely stable.
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               -20021 unstable solution returned.
C                      Phase split converged but solution is not stable.
C                      All specified phase descriptors have been used up
C                      or the maximum number of phases has been reached
C                      so no further phase split can be done.
C               -20022 phase split converged to low accuracy
C               -20023 other warning
C               20021  condition to match IPROB(1) is not correctly specified
C               20022  illegal input specification reported by startup
C                      routine. Negative or zero:
C                      number of moles of feed; feed mole
C                      fractions; number of phase descriptors;
C                      phase descriptors wrongly specified in input or
C                      wrongly set up prior to calling AXFL02.
C               20023  inconsistent user-supplied estimates.
C                      Number of input phases NP does not agree
C                      with flags for phases present in LNOPD
C                      or initial number of phases NP is greater than
C                      number of available phase descriptors NPD.
C               20024  cannot find starting point for calculation - this
C                      may mean there is no solution.
C               20025  PD identifier has unacceptable value
C               20026  cannot converge temperature or pressure search
```

```
 5
C              20027  not enough phase descriptors to solve problem.
C                     Calculation abandoned.
C              20028  other fatal error
C              20029  input pressure not in acceptable range
C              20030  input temperature not in acceptable range
C              20031  variable to search for (IVAR) is not correctly specified

C              20032  input volume not in acceptable range
C              20033  inconsistent user-supplied estimates.
C                     An input phase has a value in LNOPD which does
C                     not correspond to any of the current PDs.
```

12.7 Search for hydrate temperature or pressure: AXHYD

A special version of AXFL02 has been developed that is useful for hydrates.

In hydrates, when solving for the hydrate phase is not alwyas possible to predict if the hydrate that is going to form is Hydrate I or II or H.

This function mimics what is done in the GUI when the user presses the hydrate P or T buttons to get the most stable hydrate at the current Pressure and Temperature , respectevily.

The particular difference between AXHYD and AXFL02 are:

- the constraint for phase amount is set to 0.0
- can only search T or P
- the most stable hydrate phase load number is returned

The calling specification for AXHYD is:

```
SUBROUTINE AXHYD ( Lnost, Ivar, T, P, Ncdim, Nmolin, Nphmax, Fest1,      &
                   Fest2, Istabi, Np, Lnopd, Xx, Nmolph, Zcomp, Iphid,    &
                   Istabo, Lnbest, Nprint,  Nerr, Mxerr, Ierr)

  LOGICAL, INTENT(IN) :: Fest1, Fest2
  INTEGER, INTENT(IN) :: Lnost, Ivar, Ncdim, Nphmax, Istabi, Nprint, Mxerr
  INTEGER, INTENT(OUT) :: Istabo, Nerr, Ierr(*), Iphid(*), Np, Lnbest
  INTEGER, INTENT(INOUT) :: Lnopd(*)
  REAL*8, INTENT(IN) :: Nmolin(Ncdim, *)
  REAL*8, INTENT(INOUT) ::  T, P, Xx(Ncdim, *), Nmolph(*)
  REAL*8, INTENT(OUT) :: Zcomp(*)

!
!     Input arguments:
!     Lnost    stream type load number for flash (obtained by calling
!              AXDFST). This defines the phase descriptors (PDs) and
!              components to be considered in the flash. If set to zero
!              this means that all PDs and components will be included.
!     Ivar     code denoting variable to search for:
!              (codes are defined in axjnum.inc)
!              JXTEMP  temperature
!              JXPRES  pressure
!     Ncdim    dimension of mole fraction arrays NMOLIN and XX declared
!              in calling routine. NCDIM must be greater than or equal
```

```
5
!               to the the number of components for the stream type.
!     Nmolin    number of moles of each component in the feed and in the
!               fluid mixture to be added in tolerance calculations.
!     Nphmax    The maximum number of coexisting phases that the flash
!               calculation is to search for. The maximum number of
!               phases that may actually be returned by AXFL02 is the
!               minimum of NPHMAX, the number of phases in the stream
!               type (NPD) and MAXPHA (the maximum number of coexisting
!               phases for which AXFL02 is set up). The parameter MAXPHA
!               is defined in the file AXPARA.INC. If NPHMAX is outside
!               the range 1 to MAXPHA it is equivalent to setting it to
!               MAXPHA.
!               The arrays LNOPD, X, NMOLPH, ZCOMP and IPHID must be
!               dimensioned to at least MIN( NPHMAX, NPD, MAXPHA ) in
!               the calling routine.
!     Fest1     first estimates of the phase fractions and compositions
!               at the solution must be supplied if FEST1 is .TRUE.
!               If FEST1 is .FALSE. the initial phase distribution
!               is obtained from a stability analysis using the
!               phase descriptors specified in the stream type.
!     Fest2     first estimates of the variable (temperature, pressure or
!               tolerance ratio ) at the solution must be supplied if FEST2
!               is .TRUE. If FEST2 is .FALSE. the initial value of T or P
!               is obtained using an internal procedure.
!     Istabi    controls whether a stability test is done on a final
!               solution that contains all PDs in the stream type.
!               Omitting the stability test saves time but an
!               erroneous solution may be returned.
!               (codes are defined in axjnum.inc)
!               JXNOSC  no final stability check
!               JXDOSC  do final stability check
!     Nprint    diagnostic output flag.
!               -1   no output
!               0    minimum output in case of serious errors
!               1    errors and serious warnings
!               2    warnings plus above
!               3    informative messages plus above
!     Mxerr     maximum number of error codes to be returned in IERR
!
!     Input/output arguments:
!       Input values:
!       T         if IVAR=JXPRES, temperature (K),
!                 if IVAR=JXTEMP and FEST2=TRUE, estimated temperature (K),
!                 else not defined
!       P         if IVAR=JXTEMP, pressure (Pa),
!                 if IVAR=JXPRES and FEST2=TRUE, estimated pressure (Pa),
!                 else not defined
!       Np        If FEST1 is .TRUE. NP must be set to the number of
!                 phases present at the estimate of the solution supplied
!                 in LNOPD, XX and NMOLPH.
!       Lnopd     If FEST1 is .TRUE. LNOPD must be set on entry.
!                 Array of NP PD load numbers corresponding to input
!                 values in XX and NMOLPH.
!       Xx        If FEST1 is .TRUE. XX must be set on entry.
!                 See below for full description.
!       Nmolph    If FEST1 is .TRUE. NMOLPH must be set on entry.
!                 See below for full description.
!
!       Output values:
```

```
5
!       T        Temperature (K)
!       P        Pressure (Pa)
!       Np       number of phases at equilibrium or at best estimate of
!                solution. When an estimate of the solution is supplied
!                some of the input phases may be eliminated.
!       Lnopd    Array of NP PD load numbers corresponding to output
!                values in XX, NMOLPH, ZCOMP and IPHID.
!       Xx       XX(I,J) holds the mole fraction of component I in phase
!                J for J = 1,NP.
!       Nmolph   NMOLPH(J) holds the number of moles in phase J for
!                J = 1,NP. A zero value will be returned for phases at a
!                saturation point (eg. a dew point or bubble point).
!
!     Output arguments:
!     Zcomp    compressibility factor of each output phase (as
!              calculated by the fugacity model, not the volumetric
!              model).
!     Iphid    identity (V,L,..etc) of each output phase. If the
!              calculation is successful, the phase id. is set to that
!              specified in the phase descriptor definition. Note that
!              for supercritical phases described by equations of state,
!              the phase label is ambiguous.
!              The phase codes used are defined in AYPTID.INC
!     Lnbest   The load number of the most satble hydrate phase
!     Istabo   stability indicator for solution.
!              (codes are defined in axjnum.inc)
!              JXSTUN  definitely unstable. Another phase would be
!                      formed or an existing phase removed if problem
!                      specification allowed.
!              JXSTMU  marginally unstable (see above).
!              JXSTIN  indeterminate.
!                      This value is returned if no final stability
!                      test was requested (ISTABI=JXNOSC) and no
!                      other error condition was detected.
!                      If ISTABI=JXDOSC it means that the solution is
!                      on the stability limit.
!              JXSTMS  marginally stable.
!              JXSTST  definitely stable.
!     Nerr     no. errors reported by this routine and sub-routines.
!     Ierr     error codes returned.
!              21071: Number of phase descriptors in unsufficient to continue
!              21072: No hydrate phase descriptors found.
!              21073: The calculation of the more stable hydrate phase has
failed.
!              21074: A stable hydrate phase was not found.
!              21075: Invalid temperature input.
!              21076: Invalid pressure input.
!              21077: Water not present or amount not defined.
!              21078: Invalid specification for the flash routine.
```

12.8 Double flashes: AXFL04

AXFL04 calculates flashes where two variables are adjusted to match two constraints. The variables and constraints are as for

AXFL02, i.e. temperature, pressure or fluid fraction (tolerance) and:

- Total enthalpy H
- Total entropy S
- Total volume V
- Total internal energy U
- Phase fraction (molar, mass or volume basis)
- Phase nucleation rate
- Cricondentherm/cricondenbar

The fluid fractions for tolerance calculations are defined as previously. It is possible for both constraints to be phase fractions but in this case the phase fractions must be fixed for two different phases for which phase descriptors are included in the stream type passed to AXFL04. On convergence, the pressure, temperature and fluid fraction are returned together with the amounts and compositions of the phases present at equilibrium.

Only certain combinations of specifications handled by AXFL04 lead to a phase equilibrium problem that has a unique solution. These combinations are: U,V ; S,V ; U,S and H,S . The remaining specifications may have more than one solution or no solution. The calling specification for AXFL04 is:

```
      SUBROUTINE AXFL04( LNOST, IVAR1, IVAR2, T, P, TOLIN, NCDIM,
     +                   NMOLIN, NPHMAX, IPROB1, IPROB2, TARG1, TARG2,
     +                   FEST1, FEST2, ISTABI, NP, LNOPD, XX, NMOLPH,
     +                   ZCOMP, IPHID, ISTABO, NPRINT, NERR, MXERR,
     +                   IERR )

      INTEGER, INTENT(IN) :: LNOST, IVAR1, IVAR2, NCDIM, NPHMAX, IPROB1(3),
IPROB2(3),
     +        ISTABI, NPRINT, MXERR
      INTEGER, INTENT(INOUT) :: NP, LNOPD(*)
      INTEGER, INTENT(OUT) :: IPHID(*), ISTABO,  NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(NCDIM,*), TARG1, TARG2, TOLIN
      REAL*8, INTENT(INOUT) :: T, P, XX(NCDIM,*), NMOLPH(*)
      REAL*8, INTENT(OUT) :: ZCOMP(*)
      LOGICAL, INTENT(IN) :: FEST1, FEST2

C
C     Input arguments:
C     LNOST    stream type load number for flash (obtained by calling
C              AXDFST). This defines the phase descriptors (PDs) and
C              components to be considered in the flash. If set to zero
C              this means that all PDs and components will be included.
C     IVAR1    code denoting variable to search for in outer loop:
C              (codes are defined in axjnum.inc)
C              JXTEMP  temperature
C              JXPRES  pressure
C              JXTOL   tolerance variable
C     IVAR2    code denoting variable to search for in inner loop
C              (codes as for IVAR1)
C     NCDIM    dimension of mole fraction array XX declared in calling
C              routine. NCDIM must be greater than or equal to the
```

```
 5
C               the number of components for the stream type.
C      NMOLIN   number of moles of each component in the feed
C      NPHMAX   The maximum number of coexisting phases that the flash
C               calculation is to search for. The maximum number of
C               phases that may actually be returned by AXFL04 is the
C               minimum of NPHMAX, the number of phases in the stream
C               type (NPD) and MAXPHA (the maximum number of coexisting
C               phases for which AXFL04 is set up). The parameter MAXPHA
C               is defined in the file axpara.inc. If NPHMAX is outside
C               the range 1 to MAXPHA it is equivalent to setting it to
C               MAXPHA.
C               The arrays LNOPD, X, NMOLPH, ZCOMP and IPHID must be
C               dimensioned to at least MIN( NPHMAX, NPD, MAXPHA ) in
C               the calling routine.
C      IPROB1   identifer for type of calculation required for outer loop
C               IPROB1(1) = problem type
C               (codes are defined in axjnum.inc)
C                 JXENTH   total enthalpy (J)
C                 JXENTR   total entropy (J K-1)
C                 JXVOL    total volume (m3)
C                 JXINTE   total internal energy (J)
C                 JXPHML   phase molar fraction
C                 JXPHMS   phase mass fraction
C                 JXPHVL   phase volume fraction
C                 JXNUCL   phase nucleation
C                 JXCCTH   cricondentherm
C                 JXCCBA   cricondenbar
C               IPROB1(2) = fixed phase load number if phase fraction problem
C               IPROB1(3) = upper/lower phase fraction search
C               (codes are defined in axjnum.inc)
C                 JSNL     lower normal solution
C                 JSRU     upper retrograde solution
C                 JSNU     upper normal solution
C                 JSRL     lower retrograde solution
C                 JSANY    any solution
C                 JSDEF    default normal solution
C
C               The normal settings for dew/bubble points are:
C                                       Search for    setting
C               Normal dew point            T         upper
C                                           P         lower
C
C               Retrograde dew point        T         lower
C                                           P         upper
C
C               Normal bubble point         T         lower
C                                           P         upper
C
C               Retrograde bubble point     T         upper
C                                           P         lower
C
C               For phase fractions between 0 and 1 the best setting
C               depends on the shape of the phase diagram.
C
C               It is important to specify the correct search code
C               because the value of IPROB1(3) is used as a guide, both to
C               generate initial estimates of T or P, and also to decide
C               on a search direction towards the required solution.
C               Particularly for difficult calculations, the user needs
```

```
5
C               to have an idea of the layout of the phase diagram.
C               Note that the JSANY specification is less reliable than
C               any of the others because the routine has no guidance
C               about where to look for a solution.
C     IPROB2    identifer for type of calculation required for inner loop
C               (codes as for IPROB1)
C     TARG1     function values to be matched (depends on IPROB1) in outer loop

C     TARG2     function values to be matched (depends on IPROB2) in inner loop

C     FEST1     first estimates of the phase fractions and compositions
C               at the solution must be supplied if FEST1 is .TRUE.
C               If FEST1 is .FALSE. the initial phase distribution
C               is obtained from a stability analysis using the
C               phase descriptors specified in the stream type.
C     FEST2     first estimates of the temperature and pressure
C               at the solution must be supplied if FEST2 is .TRUE.
C               If FEST2 is .FALSE. the initial value of T and P
C               are obtained using an internal procedure.
C     ISTABI    controls whether a stability test is done on a final
C               solution that contains all PDs in the stream type.
C               Omitting the stability test saves time but an
C               erroneous solution may be returned.
C               (codes are defined in axjnum.inc)
C               JXNOSC  no final stability check
C               JXDOSC  do final stability check
C     NPRINT    diagnostic output flag.
C               -1   no output
C               0    minimum output in case of serious errors
C               1    errors and serious warnings
C               2    warnings plus above
C               3    informative messages plus above
C     MXERR     maximum number of error codes to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     T         If FEST2=TRUE, estimated temperature (K), else not
C               defined
C     P         If FEST2=TRUE, estimated pressure (Pa), else not defined
C     TOLIN     If IVAR1=JXTOL or IVAR2=JXTOL and FEST2=TRUE, estimated
C               tolerance variable, else not defined
C     NP        If FEST1 is .TRUE. NP must be set to the number of
C               phases present at the estimate of the solution supplied
C               in LNOPD, XX and NMOLPH.
C     LNOPD     If FEST1 is .TRUE. LNOPD must be set on entry.
C               Array of NP PD load numbers corresponding to input
C               values in XX and NMOLPH.
C     XX        If FEST1 is .TRUE. XX must be set on entry.
C               See below for full description.
C     NMOLPH    If FEST1 is .TRUE. NMOLPH must be set on entry.
C               See below for full description.
C
C     Output values:
C     T         Temperature (K)
C     P         Pressure (Pa)
C     TOLIN     Tolerance variable, if IVAR1=JXTOL or IVAR2=JXTOL
C     NP        number of phases at equilibrium or at best estimate of
C               solution. When an estimate of the solution is supplied
C               some of the input phases may be eliminated.
```

```
 5
C     LNOPD    Array of NP PD load numbers corresponding to output
C              values in XX, NMOLPH, ZCOMP and IPHID.
C     XX       XX(I,J) holds the mole fraction of component I in phase
C              J for J = 1,NP.
C     NMOLPH   NMOLPH(J) holds the number of moles in phase J for
C              J = 1,NP. A zero value will be returned for phases at a
C              saturation point (eg. a dew point or bubble point).
C
C     Output arguments:
C     ZCOMP    compressibility factor of each output phase (as
C              calculated by the fugacity model, not the volumetric
C              model).
C     IPHID    identity (V,L,..etc) of each output phase. If the
C              calculation is successful, the phase id. is set to that
C              specified in the phase descriptor definition. Note that
C              for supercritical phases described by equations of state,
C              the phase label is ambiguous.
C              The phase codes used are defined in AYPTID.INC
C     ISTABO   stability indicator for solution.
C              (codes are defined in axjnum.inc)
C              JXSTUN  definitely unstable. Another phase would be
C                      formed or an existing phase removed if problem
C                      specification allowed.
C              JXSTMU  marginally unstable (see above).
C              JXSTIN  indeterminate.
C                      This value is returned if no final stability
C                      test was requested (ISTABI=JXNOSC) and no
C                      other error condition was detected.
C                      If ISTABI=JXDOSC it means that the solution is
C                      on the stability limit.
C              JXSTMS  marginally stable.
C              JXSTST  definitely stable.
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              -20041 unstable solution returned.
C                     Phase split converged but solution is not stable.
C                     All specified phase descriptors have been used up
C                     or the maximum number of phases has been reached
C                     so no further phase split can be done.
C              -20043 other warning
C              20041  conditions to match (IPROB1/2) not correctly specified
C              20042  illegal input specification reported by startup
C                     routine. Negative or zero:
C                     number of moles of feed; feed mole
C                     fractions; number of phase descriptors;
C                     phase descriptors wrongly specified in input or
C                     wrongly set up prior to calling AXFL02.
C              20043  inconsistent user-supplied estimates.
C                     Number of input phases NP does not agree
C                     with flags for phases present in LNOPD
C                     or initial number of phases NP is greater than
C                     number of available phase descriptors NPD.
C              20045  PD identifier has unacceptable value
C              20046  cannot converge temperature or pressure search
C              20047  Same variable specified twice
C              20048  Same constraint specified twice
C              20049  other serious error
C              20050  variables to search for (IVAR1/2) not correctly
specified
```

```
 5
C              20051  input pressure not in acceptable range
C              20052  input temperature not in acceptable range
C              20053  input volume not in acceptable range
C              20054  inconsistent user-supplied estimates.
C                     An input phase has a value in LNOPD which does
C                     not correspond to any of the current PDs.
```

12.9 Stability test: AXST01

AXST01 can perform an independent phase stability analysis on a mixture to detect if any further phases may form. The routine will determine which is the most stable single phase for the input composition (out of the list of available phases in the stream LNOST) and then finds composition of incipient coexisting phases. The incipient phases may either reduce the Gibbs energy, in which case the original mixture is unstable, or increase the Gibbs energy which indicates that the mixture is stable. The Gibbs energy stability indicator for each incipient phase is returned in array GSTB. The calling specification is:

```
      SUBROUTINE AXST01( LNOST, T, P, NMOLIN, MAXIP, FEST, IPD0, IPHID0,
     +                   ZCOMP0, NIP, IPDIP, GSTB, XIP, ZCOMP, IPHID,
     +                   ALLTP, ISTABO, NPRINT, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, MAXIP, NPRINT, MXERR
      INTEGER, INTENT(INOUT):: IPD0,  NIP, IPDIP(MAXIP)
      INTEGER, INTET(OUT) :: IPHID(MAXIP), ISTABO, NERR, IERR(*), IPHID0,
      REAL*8, INTENT(IN) ::  T, P, NMOLIN(MAXNCS)
      REAL*8, INTENT(INOUT) :: XIP(MAXNCS,MAXIP)
      REAL*8, INTENT(OUT) :: ZCOMP0, GSTB(MAXIP),ZCOMP(MAXIP)
      LOGICAL, INTENT(IB) :: FEST
      LOGICAL, INTENT(UNOUT) :: ALLTP

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     T        temperature (K)
C     P        pressure (Pa)
C     NMOLIN   number of moles of each component in the mixture to be
C              tested for stability
C     MAXIP    maximum number of incipient phases for which compositions
C              etc. may be returned
C     FEST     Logical flag: determines whether information on most
C              stable phase and estimates of incipient phase
C              compositions are supplied or internally generated
C              = T    information must be supplied in IPD0, NIP, XIP and
C                     IPDIP (see below)
C              = F    most stable phase and incipient phases are
C                     determined automatically.
C     NPRINT   diagnostic output flag.
C              = -1   no output
C              = 0    minimum output in case of serious errors
C              = 1    errors and serious warnings
C              = 2    warnings plus above
C              = 3    informative messages plus above
C     MXERR    maximum number of error flags to be returned in IERR
C
```

```
 5
C     Input/output arguments:
C     Input values:
C     ALLTP    Logical flag, controls number of trial phases generated
C              for the stability test.
C              = T    TPs for all PDs (including that used for the most
C                     stable phase) this is the most exhaustive search
C              = F    TPs for PDs corresponding to the phases other
C                     than the most stable phase.
C     If FEST is .TRUE. the following must be set:
C     IPD0     phase descriptor for test mixture in reference phase,
C              i.e. the phase for which stability is being tested.
C     NIP      number of incipient phases set in IPDIP and XIP
C     IPDIP    phase descriptor for each incipient phase.
C     XIP      initial estimates of incipient phase compositions.
C              XIP(I,K) holds the composition (mole fraction) of
C              component I in the Kth incipient phase
C
C     Output values:
C     ALLTP    if the input value was F, ALLTP will be reset to T if
C              NP=NPD or the PD specification is such that that all
C              possible TPs were used in the stability test.
C              (See AXSTPC).
C     XIP      XIP(I,K) holds the composition (mole fraction) of
C              component I in the Kth incipient phase
C
C     Output arguments:
C     IPD0     phase descriptor for test mixture in most stable phase
C              (set if FEST is FALSE)
C     IPHID0   identity (V,L,..etc) of most stable phase
C     ZCOMP0   compressibility factor for test mixture in most stable
C              phase
C     NIP      number of incipient phases found. NIP <= MAXIP.
C              (set if FEST is FALSE)
C     IPDIP    phase descriptor for each incipient phase.
C              (set if FEST is FALSE)
C     GSTB     stability indicator for each incipient phase.
C              Modified tangent plane distance (g* in eq. 14) for
C              each phase returned by stability test arranged in
C              order of increasing g*.
C              A negative g* shows the phase can decrease the GE,
C              i.e. system is unstable. A zero g* shows the phase is
C              in equilibrium with existing phases. A positive g*
C              shows the system is stable but the phase composition
C              is an indication of the composition of an incipient
C              phase which may form at a different T or P.
C     ZCOMP    compressibility factor of each incipient phase.
C     IPHID    identity (V,L,..etc) of each incipient phase.
C              The phase id. is set to that specified in the phase
C              descriptor definition. Note that for supercritical
C              phases described by equations of state, the phase label
C              is ambiguous.
C              The phase codes used are defined in AYPTID.INC
C     ISTABO   stability indicator for solution.
C              (codes are defined in axjnum.inc)
C              JXSTUN  definitely unstable. Another phase would be
C                      formed or an existing phase removed if problem
C                      specification allowed.
C              JXSTMU  marginally unstable (see above).
C              JXSTIN  indeterminate.
```

```
 5
C                    This value is returned if no final stability
C                    test was requested (ISTABI=JXNOSC) and no
C                    other error condition was detected.
C                    If ISTABI=JXDOSC it means that the solution is
C                    on the stability limit.
C               JXSTMS  marginally stable.
C               JXSTST  definitely stable.
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               20061  bad input specification: negative or zero
C                      P or T
C               20062  bad input specification reported by startup
C                      routine. Negative or zero:
C                      number of moles of feed; feed mole
C                      fractions; number of phase descriptors;
C                      phase descriptors wrongly specified in input or
C                      wrongly set up prior to calling AXST01.
C               20063  fatal error getting properties of reference phase
C               20064  fatal error in stability analysis
C               20065  bad input specification of phase descriptor for
C                      reference phase (negative, exceeds MAXPD or not
C                      in input list of PDs)
```

12.10 Setting low-level calculation options

All the phase equilibrium algorithms use the following elements: first order phase split; second order phase split; stability test. See the references to Michelsen's work for a mathematical description. In some circumstances it may be desirable to change some of the numerical parameters that affect the performance of these algorithms although this is not normally recommended. The default settings for a number of algorithm parameters can be changed by calls to the following entry points. A negative value for any of the numerical parameters in the call means 'set the value to deafult'.

12.10.1 First-order phase split

To set the parameters for the first-order phase-split method call AXSE01:

```
      SUBROUTINE AXSE01( ITMX0, EEPS0, ITMX1, EEPS1 )

      INTEGER, INTENT(IN) :: ITMX0, ITMX1
      REAL*8, INTENT(IN) :: EEPS0, EEPS1

C
C     Input arguments:
C     ITMX0    maximum number of iterations for 1st order phase split
C     EEPS0    convergence tolerance on gradient norm for 1st order
C              phase split
C     ITMX1    maximum number of iterations for material balance
C     EEPS1    convergence tolerance for material balance
```

5

Notes:

ITMX0

Default is 60, should not normally be changed unless second-order method is disabled.

EEPS0

Default is 10^{-16} . A larger value e.g. (10^{-12}) may reduce calculation time but it is possible that flashes in the critical region may not be adequately converged. If EEPS0 is set to a value greater than the default, the convergence of the temperature/pressure search flashes (AXFL02 and AXFL04) could be impaired because the low-level calls to the T, P flash may return noisy values. The convergence parameters for the temperature/pressure searches should therefore be set to the lower-accuracy values recommended for AXSE04.

ITMX1

Default is 60, should not normally be changed.

EEPS1

Default is 10^{-20} , should not normally be changed.

12.10.2 Second-order phase split

To set the parameters for the second-order phase-split method call AXSE02:

```
      SUBROUTINE AXSE02( ITMX0, EEPS0, XXTRA, I2OR0 )

      INTEGER, INTENT(IN) :: ITMX0
      REAL*8, INTENT(IN) :: EEPS0, XXTRA
      LOGICAL, INTENT(IN) :: I2OR0

C
C     Input arguments:
C     ITMX0    maximum number of iterations for 2nd order phase split
C     EEPS0    convergence tolerance on gradient norm for 2nd order
C              phase split
C     XXTRA    limit for mole fraction of trace components
C     I2OR0    second order minimization flag. If I2OR0=.TRUE.
C              2nd order methods may be used to solve the flash.
C              Otherwise only methods that do not require composition
C              derivatives of the fugacity coefficients are used.
```

Notes:

ITMX0

Default is 50, should not normally be changed.

EEPS0

Default is 10^{-16} . This should not normally be changed.

XXTRA

Default is 10^{-6} , should not normally be changed.

I2ORD

Default is .TRUE..

12.10.3 Stability test

To set the parameters for the stability test call AXSE03:

```
      SUBROUTINE AXSE03( ITMX0, EEPS0, ITP )

      INTEGER, INTENT(IN) :: ITMX0, ITP
      REAL*8, INTENT(IN) :: EEPS0

C
C     Input arguments:
C     ITMX0    maximum number of iterations for stability test
C     EEPS0    convergence tolerance on gradient norm for stability
C              test
C     ITP      flag which controls the type and number of trial
C              phases (TP) generated.
C              = 0  generate maximum number of TP based on pure
C                   components (PC), approximate K-values (KV),
C                   ideal mixtures (IM) and several combinations
C                   of existing phases
C              = 1  generate TP based on PC, KV, IM and arithmetic
C                   mean of existing phases
C              = 2  generate TP based on PC, KV and  IM
C              = 3  generate TP based on PC and  KV
C              = 4  generate TP based on PC only
C              = 5  generate TP based on KV only
C              = 6  generate TP based on PC for phases with key
C                   components and KV only
```

Notes:

ITMX0

Default is 100, may be reduced to 20 in most cases but extra phases in the near-critical region may be missed.

EEPS0

Default is 10^{-9} , should not normally be changed.

ITP

The default value is ITP=0, i.e. the most exhaustive search. For oil/gas/water systems ITP=6 is much faster and quite reliable.

12.10.4 Temperature/pressure searches

To set the parameters for the temperature/pressure search routines call AXSE04:

```
      SUBROUTINE AXSE04( TMAX, TMIN, PMAX, PMIN, EPST, EPSP, EPSPH,
     +                   EPSH, EPSS, EPSU, EPSV )

      REAL*8, INTENT(IN) :: TMAX, TMIN, PMAX, PMIN, EPST, EPSP,
```

```
5
     +       EPSPH, EPSH, EPSS, EPSU, EPSV

C
C     Input arguments:
C     TMAX     upper temperature limit at which search should stop (K)
C     TMIN     lower temperature limit at which search should stop (K)
C     PMAX     upper pressure limit at which search should stop (Pa)
C     PMIN     lower pressure limit at which search should stop (Pa)
C     EPST     convergence tolerance for temperature (K)
C     EPSP     convergence tolerance for pressure (Pa)
C     EPSPH    convergence tolerance for phase fraction
C     EPSH     convergence tolerance for enthalpy (J)
C     EPSS     convergence tolerance for entropy (JK-1)
C     EPSU     convergence tolerance for internal energy (J)
C     EPSV     convergence tolerance for volume (m3)
```

Notes:

TMAX

Default is 2000K.

TMIN

Default is 1K.

PMAX

Default is 10^9 Pa.

PMIN

Default is 10^{-4} Pa.

EPST

Default is 10^{-4} K, may often be increased safely to 10^{-3} K.

EPSP

Default is 1Pa, may often be increased safely to 10Pa.

EPSPH

Default is 10^{-5} , may often be increased safely to 10^{-4} .

EPSH

Default is 10^{-3} J, may often be increased safely to 0.1 J.

EPSS

Default is $10^{-5} J K^{-1}$, may often be increased safely to $10^{-3} J K^{-1}$.

EPSU

Default is 10^{-3} J, may often be increased safely to 0.1 J.

EPSV

Default is 10^{-4} , may often be increased safely to 10^{-3} .

It is recommended that the lower-accuracy values are used if the parameter EEPS0 in AXSE01 or AXSE02 is set to a value greater than the default.

13 Flash sensitivity calculations

The routines described in this section return the (analytical) derivatives of output quantities in flash calculations with respect to the specified conditions (such as temperature, pressure and feed composition) whilst keeping the phases at equilibrium. Note that to operate correctly, the equilibrium values returned by the flash routines must be passed unaltered into the sensitivity routines.

13.1 Flashes at fixed pressure and temperature: AXSN01

AXSN01 differentiates flashes performed by AXFL01 at specified pressure and temperature. The calling specification is:

```
      INTEGER LNOST, NCDIM, NP, LNOPD(*), NERR, MXERR, IERR(*)
      REAL*8 T, P, X(NCDIM,*), XT(NCDIM,*), XP(NCDIM,*),
     +       XN(NCDIM,NCDIM,*), PHF(*), PHFT(*), PHFP(*), PHFN(NCDIM,*),
     +       NMOLIN(*), V(0:*), VT(0:*), VP(0:*), VN(NCDIM,0:*), H(0:*),
     +       HT(0:*), HP(0:*), HN(NCDIM,0:*), S(0:*), ST(0:*), SP(0:*),
     +       SN(NCDIM,0:*)
      LOGICAL D(3), RETV, RETH

      SUBROUTINE AXSN01( LNOST, D, T, P, NCDIM, NMOLIN, NP, LNOPD, X,
     +                   PHF, XT, XP, XN, PHFT, PHFP, PHFN, RETV, V, VT,
     +                   VP, VN, RETH, H, HT, HP, HN, S, ST, SP, SN,
     +                   NERR, MXERR, IERR )
C
C     Input arguments:
C     LNOST    stream type load number for flash
C     D        Derivative flags:
C              D(1)  .TRUE. return T derivatives, otherwise not
C              D(2)  .TRUE. return P derivatives, otherwise not
C              D(3)  .TRUE. return mole number derivatives, otherwise not
C     T        Temperature (K)
C     P        Pressure (Pa)
C     NCDIM    dimensioning parameter for arrays X, XH, XP, XN and PHFN as
C              declared in calling routine. NCDIM must be greater than or
```

```
5
C               equal to the number of components for the stream type.
C     NMOLIN    Number of moles of each component in the feed
C     X         X(I,J) holds the composition (mole fraction) of
C               component I in phase J.
C     NP        Number of phases in equilibrium
C     PHF       Number of moles of each phase
C     LNOPD     Array of NP PD load numbers corresponding to values
C               in X and PHF
C     RETV      .TRUE. if volumes and their derivatives are to be returned,
C               otherwise .FALSE.
C     RETH      .TRUE. if enthalpies and entropies and their derivatives are
C               to be returned, otherwise .FALSE.
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     XT        Partial derivative of mole fraction X(I,J) w.r.t. T
C     XP        Partial derivative of mole fraction X(I,J) w.r.t. P
C     XN        Partial derivative of mole fraction X(I,J) w.r.t.
C               feed composition
C     PHFT      Partial derivative of phase amount PHF(J) w.r.t. T
C     PHFP      Partial derivative of phase amount PHF(J) w.r.t. P
C     PHFN      Partial derivative of phase amount PHF(J) w.r.t.
C               feed composition
C     V         Total and phase volumes (index=0 denotes total)
C     VT        Partial derivative of volume V w.r.t. T
C     VP        Partial derivative of volume V w.r.t. P
C     VN        Partial derivative of volume V w.r.t. feed composition
C     H         Total and phase enthalpies (index=0 denotes total)
C     HT        Partial derivative of enthalpy H w.r.t. T
C     HP        Partial derivative of enthalpy H w.r.t. P
C     HN        Partial derivative of enthalpy H w.r.t. feed composition
C     S         Total and phase entropies (index=0 denotes total)
C     ST        Partial derivative of entropy S w.r.t. T
C     SP        Partial derivative of entropy S w.r.t. P
C     SN        Partial derivative of entropy S w.r.t. feed composition
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               20151  error reported by startup routine AXSTRT
C               20153  cannot evaluate fugacity and derivatives
C               20154  cannot solve for derivatives
C               20155  cannot solve for mole number derivatives
```

13.2 Flashes with a temperature or pressure search: AXSN02

AXSN02 differentiates flashes performed by AXFL02 where one variable is adjusted to satisfy a constraint. (Tolerance calculations are excluded.) The variables may be either the temperature or the pressure. The possible constraints are:

- Total enthalpy H
- Total entropy S
- Total volume V

- Total internal energy U
- Phase fraction (molar, mass or volume basis)

The phase fraction can be fixed for any phase for which a phase descriptor has been defined in the stream type passed to AXSN02 and includes dew and bubble point calculations as special cases. The calling specification for AXSN02 is:

```
      SUBROUTINE AXSN02( LNOST, IVAR, IPROB, D, T, T1, T2, TN, P, P1,
     +                   P2, PN, NCDIM, NMOLIN, NP, LNOPD, X, X1, X2,
     +                   XN, PHF, PHF1, PHF2, PHFN, RETV, V, V1, V2, VN,
     +                   RETH, H, H1, H2, HN, S, S1, S2, SN, NERR,
     +                   MXERR, IERR )

      INTEGER LNOST, IVAR, IPROB(2), NCDIM, NP, LNOPD(*), NERR, MXERR,
     +        IERR(*)
      REAL*8 T, T1, T2, TN(MAXNCS), P, P1, P2, PN(MAXNCS), X(NCDIM,*),
     +       X1(NCDIM,*), X2(NCDIM,*), XN(NCDIM,NCDIM,*), PHF(*),
     +       PHF1(*), PHF2(*), PHFN(NCDIM,*), NMOLIN(*), V(0:*),
     +       V1(0:*), V2(0:*), VN(NCDIM,0:*), H(0:*), H1(0:*), H2(0:*),
     +       HN(NCDIM,0:*), S(0:*), S1(0:*), S2(0:*), SN(NCDIM,0:*)
      LOGICAL D(3), RETV, RETH

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     IVAR     code denoting variable to search for:
C              (codes are defined in axjnum.inc)
C              JXTEMP  temperature (constraint 2 = pressure)
C              JXPRES  pressure (constraint 2 = temperature)
C     IPROB    identifer for type of calculation required
C              IPROB(1) problem type (= constraint 1)
C                       JXENTH   total enthalpy (J)
C                       JXENTR   total entropy (J K-1)
C                       JXVOL    total volume (m3)
C                       JXINTE   total internal energy (J)
C                       JXPHML   phase molar fraction
C                       JXPHMS   phase mass fraction
C                       JXPHVL   phase volume fraction
C                                  (codes are defined in axjnum.inc)
C              IPROB(2) fixed phase load number if phase fraction problem
C     D        Derivative flags:
C              D(1)  .TRUE. return derivatives w.r.t. constraint 1,
C                    otherwise not
C              D(2)  .TRUE. return P derivatives w.r.t. constraint 2,
C                    otherwise not
C              D(3)  .TRUE. return mole number derivatives, otherwise not
C     T        Temperature (K)
C     P        Pressure (Pa)
C     NCDIM    dimensioning parameter for arrays X, X1, X2, XN and PHFN as
C              declared in calling routine. NCDIM must be greater than or
C              equal to the number of components for the stream type.
C     NMOLIN   Number of moles of each component in the feed
C     X        X(I,J) holds the composition (mole fraction) of
C              component I in phase J.
C     NP       Number of phases in equilibrium
C     PHF      Number of moles of each phase
C     LNOPD    Array of NP PD load numbers corresponding to values
C              in X and PHF
```

```
5
C     RETV     .TRUE. if volumes and their derivatives are to be returned,
C              otherwise .FALSE.
C     RETH     .TRUE. if enthalpies and entropies and their derivatives are
C              to be returned, otherwise .FALSE.
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     T1       Partial derivative of temperature T w.r.t. constraint 1
C     T2       Partial derivative of temperature T w.r.t. constraint 2
C     TN       Partial derivative of temperature T w.r.t. feed composition
C     P1       Partial derivative of pressure P w.r.t. constraint 1
C     P2       Partial derivative of pressure P w.r.t. constraint 2
C     PN       Partial derivative of pressure P w.r.t. feed composition
C     X1       Partial derivative of mole fraction X(I,J) w.r.t. constraint 1
C     X2       Partial derivative of mole fraction X(I,J) w.r.t. constraint 2
C     XN       Partial derivative of mole fraction X(I,J) w.r.t.
C              feed composition
C     PHF1     Partial derivative of phase amount PHF(J) w.r.t. constraint 1
C     PHF2     Partial derivative of phase amount PHF(J) w.r.t. constraint 2
C     PHFN     Partial derivative of phase amount PHF(J) w.r.t.
C              feed composition
C     V        Total and phase volumes (index=0 denotes total)
C     V1       Partial derivative of volume V w.r.t. constraint 1
C     V2       Partial derivative of volume V w.r.t. constraint 2
C     VN       Partial derivative of volume V w.r.t. feed composition
C     H        Total and phase enthalpies (index=0 denotes total)
C     H1       Partial derivative of enthalpy H w.r.t. constraint 1
C     H2       Partial derivative of enthalpy H w.r.t. constraint 2
C     HN       Partial derivative of enthalpy H w.r.t. feed composition
C     S        Total and phase entropies (index=0 denotes total)
C     S1       Partial derivative of entropy S w.r.t. constraint 1
C     S2       Partial derivative of entropy S w.r.t. constraint 2
C     SN       Partial derivative of entropy S w.r.t. feed composition
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              20161  error reported by startup routine AXSTRT
C              20162  unacceptable variable type
C              20163  cannot evaluate fugacity and derivatives
C              20164  cannot solve for derivatives
C              20165  cannot solve for T and mole number derivatives
C              20166  unacceptable problem specification
C              20167  fixed phase fraction flash cannot be single phase
```

13.3 Double flashes: AXSN04

AXSN04 differentiates flashes performed by AXFL04 where two variables are adjusted to match two constraints. The variables are temperature and pressure and two contraints are required from the following:

- Total enthalpy H
- Total entropy S
- Total volume V
- Total internal energy U

- Phase fraction (molar, mass or volume basis)

It is possible for both constraints to be phase fractions but in this case the phase fractions must be fixed for two different phases for which phase descriptors are included in the stream type passed to AXSN04. The calling specification for AXSN04 is:

```
      SUBROUTINE AXSN04( LNOST, IPROB1, IPROB2, D, T, T1, T2, TN, P, P1,
     +                   P2, PN, NCDIM, NMOLIN, NP, LNOPD, X, X1, X2,
     +                   XN, PHF, PHF1, PHF2, PHFN, RETV, V, V1, V2, VN,
     +                   RETH, H, H1, H2, HN, S, S1, S2, SN, NERR,
     +                   MXERR, IERR )

      INTEGER LNOST, IPROB1(2), IPROB2(2), NCDIM, NP, LNOPD(*), NERR,
     +        MXERR, IERR(*)
      REAL*8 T, T1, T2, TN(MAXNCS), P, P1, P2, PN(MAXNCS), X(NCDIM,*),
     +       X1(NCDIM,*), X2(NCDIM,*), XN(NCDIM,NCDIM,*), PHF(*),
     +       PHF1(*), PHF2(*), PHFN(NCDIM,*), NMOLIN(*), V(0:*),
     +       V1(0:*), V2(0:*), VN(NCDIM,0:*), H(0:*), H1(0:*), H2(0:*),
     +       HN(NCDIM,0:*), S(0:*), S1(0:*), S2(0:*), SN(NCDIM,0:*)
      LOGICAL D(3), RETV, RETH

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     IPROB1   identifer for type of calculation required (constraint 1)
C              IPROB1(1)  problem type
C                         JXENTH   total enthalpy (J)
C                         JXENTR   total entropy (J K-1)
C                         JXVOL    total volume (m3)
C                         JXINTE   total internal energy (J)
C                         JXPHML   phase molar fraction
C                         JXPHMS   phase mass fraction
C                         JXPHVL   phase volume fraction
C                                  (codes are defined in axjnum.inc)
C              IPROB1(2)  fixed phase load number if phase fraction problem
C     IPROB2   identifer for type of calculation required (constraint 2)
C              (Codes as for IPROB1)
C     D        Derivative flags:
C              D(1)  .TRUE. return derivatives w.r.t. constraint 1,
C                    otherwise not
C              D(2)  .TRUE. return P derivatives w.r.t. constraint 2,
C                    otherwise not
C              D(3)  .TRUE. return mole number derivatives, otherwise not
C     T        Temperature (K)
C     P        Pressure (Pa)
C     NCDIM    dimensioning parameter for arrays X, X1, X2, XN and PHFN as
C              declared in calling routine. NCDIM must be greater than or
C              equal to the number of components for the stream type.
C     NMOLIN   Number of moles of each component in the feed
C     X        X(I,J) holds the composition (mole fraction) of
C              component I in phase J.
C     NP       Number of phases in equilibrium
C     PHF      Number of moles of each phase
C     LNOPD    Array of NP PD load numbers corresponding to values
C              in X and PHF
C     RETV     .TRUE. if volumes and their derivatives are to be returned,
C              otherwise .FALSE.
C     RETH     .TRUE. if enthalpies and entropies and their derivatives are
C              to be returned, otherwise .FALSE.
```

```
 5
C      MXERR    maximum number of error flags to be returned in IERR
C
C      Output arguments:
C      T1       Partial derivative of temperature T w.r.t. constraint 1
C      T2       Partial derivative of temperature T w.r.t. constraint 2
C      TN       Partial derivative of temperature T w.r.t. feed composition
C      P1       Partial derivative of pressure P w.r.t. constraint 1
C      P2       Partial derivative of pressure P w.r.t. constraint 2
C      PN       Partial derivative of pressure P w.r.t. feed composition
C      X1       Partial derivative of mole fraction X(I,J) w.r.t. constraint 1
C      X2       Partial derivative of mole fraction X(I,J) w.r.t. constraint 2
C      XN       Partial derivative of mole fraction X(I,J) w.r.t.
C               feed composition
C      PHF1     Partial derivative of phase amount PHF(J) w.r.t. constraint 1
C      PHF2     Partial derivative of phase amount PHF(J) w.r.t. constraint 2
C      PHFN     Partial derivative of phase amount PHF(J) w.r.t.
C               feed composition
C      V        Total and phase volumes (index=0 denotes total)
C      V1       Partial derivative of volume V w.r.t. constraint 1
C      V2       Partial derivative of volume V w.r.t. constraint 2
C      VN       Partial derivative of volume V w.r.t. feed composition
C      H        Total and phase enthalpies (index=0 denotes total)
C      H1       Partial derivative of enthalpy H w.r.t. constraint 1
C      H2       Partial derivative of enthalpy H w.r.t. constraint 2
C      HN       Partial derivative of enthalpy H w.r.t. feed composition
C      S        Total and phase entropies (index=0 denotes total)
C      S1       Partial derivative of entropy S w.r.t. constraint 1
C      S2       Partial derivative of entropy S w.r.t. constraint 2
C      SN       Partial derivative of entropy S w.r.t. feed composition
C      NERR     no. errors reported by this routine and sub-routines
C      IERR     error codes returned. Values for this routine:
C               20511  error reported by startup routine AXSTRT
C               20513  cannot evaluate fugacity and derivatives
C               20514  cannot solve for derivatives
C               20515  cannot solve for T and mole number derivatives
C               20516  unacceptable problem specification
C               20517  fixed phase fraction flash cannot be single phase
```

14 Phase envelope calculations

14.1 Introduction

The Multiflash phase envelope routines are designed for tracing lines of constant phase fraction, enthalpy, entropy, internal energy or volume for multiphase systems in the pressure-temperature plane.

There are two subroutines, AXPT01 and AXPT02 which are called as follows. To trace a particular iso-property line an initial flash calculation must be done by calling AXPT01 at a defined pressure or temperature. The line is then traced by repeated calls of AXPT02. On each call, the pressure and temperature returned by AXPT02 is automatically incremented by a small amount. Calling AXPT02 should continue until either it returns in an error state indicating that it is impossible to trace the iso-property line further, or else if the pressure or temperature has reached some high or low value beyond which one does not wish to proceed. Once a line has been traced, the results can be plotted with a suitable graphics facility, if required. A new iso-property line can then be traced by calling AXPT01 again with a different specification followed by repeated calls of AXPT02. Likewise, a new fluid can be introduced by again calling AXPT01 followed by repeated calls of AXPT02.

Before any of the phase envelope routines are called, Multiflash must be initialised as described in section . The phase envelope routines may be called interchangeably with the Multiflash flash routines AXFL01, AXFL02 and AXFL04.

14.2 Calling the phase envelope routines

AXPT01 defines the stream (components in mixture and phase descriptors) to be used for the phase envelope tracing routines

and performs the initial flash calculation for each new line that is to be traced. Its calling specification is as follows:

```
      SUBROUTINE AXPT01( LNOST, IVAR, UP, T, P, VV, HH, SS, NCDIM,
     +                   NMOLIN, NPHMAX, IPRBIN, TARGIN, FEST1, FEST2,
     +                   ISTABI, NPOUT, LNOPD, XX, NMOLPH, ZCOUT,
     +                   IPHOUT, ISTABO, NPRINT, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, IVAR, IPRBIN(3), ISTABI, NPHMAX, NCDIM,
NPRINT, MXERR
      INTEGER, INTENT(INOUT) :: NPOUT, LNOPD(*)
      INTEGER, INTENT(OUT) :: ISTABO, IPHOUT(*), NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), TARGIN
      REAL*8, INTENT(INOUT) :: T, P, XX(NCDIM,*), NMOLPH(*)
      REAL*8, INTENT(OUT) :: VV, HH, SS, ZCOUT(*)
      LOGICAL, INTENT(IN) :: UP, FEST1, FEST2

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     IVAR     independent variable to be incremented first along
C              phase boundary (codes are defined in axjnum.inc):
C              JXTEMP = temperature
C              JXPRES = pressure
C     UP       .TRUE. if independent variable is to be increased to start
C              plot, .FALSE. if it is to be decreased.
C     NCDIM    Dimension of mole fraction array XX declared
C              in calling routine. NCDIM must be greater than or equal
C              to the the number of components for the stream type.
C     NMOLIN   mole numbers of components in mixture
C     NPHMAX   The maximum number of coexisting phases that the flash
C              calculation is to search for. The maximum number of phases
C              that may actually be returned by AXPT01 is the minimum
C              of NPHMAX, NPD (the number of phases in the stream type
C              and MAXPHA (the maximum number of coexisting phases for
C              the phase equilibrium module). The parameter MAXPHA is
C              defined in the file AXPARA.INC. If NPHMAX is outside the
C              range 1 to MAXPHA it is equivalent to setting it to MAXPHA.
C              The arrays LNOPD, X, NMOLPH, ZCOMP and IPHID must be
C              dimensioned to at least MIN( NPHMAX, NPD, MAXPHA ) in
C              the calling routine.
C     IPRBIN   identifer for type of calculation required
C              (codes are defined in axjnum.inc)
C              IPRBIN(1) = problem type
C                JXENTH   total enthalpy (J)
C                JXENTR   total entropy (J K-1)
C                JXVOL    total volume (m3)
C                JXINTE   total internal energy (J)
C                JXNUCL   nucleation current
C                JXPHML   phase molar fraction
C                JXPHMS   phase mass fraction
C                JXPHVL   phase volume fraction
C              IPRBIN(2) = fixed phase load number if phase fraction problem
C              IPRBIN(3) = upper/lower phase fraction search
C                JSNL     lower normal solution
C                JSRU     upper retrograde solution
C                JSNU     upper normal solution
C                JSRL     lower retrograde solution
C                JSANY    any solution
```

```
 5
C                JSDEF    default normal solution
C
C              The normal settings for dew/bubble points are:
C                                    Search for    setting
C              Normal dew point          T         upper
C                                        P         lower
C
C              Retrograde dew point      T         lower
C                                        P         upper
C
C              Normal bubble point       T         lower
C                                        P         upper
C
C              Retrograde bubble point   T         upper
C                                        P         lower
C
C              For phase fractions between 0 and 1 the best setting
C              depends on the shape of the phase diagram.
C
C              It is important to specify the correct search code
C              because the value of IPROB(3) is used as a guide, both to
C              generate initial estimates of T or P, and also to decide
C              on a search direction towards the required solution.
C              Particularly for difficult calculations, the user needs
C              to have an idea of the layout of the phase diagram.
C              Note that the JSANY specification is less reliable than
C              any of the others because the routine has no guidance
C              about where to look for a solution.
C     TARGIN   Function values to be matched
C     FEST1    First estimates of the phase fractions and compositions
C              at the solution must be supplied if FEST1 is TRUE.
C              If FEST1 is FALSE the initial phase distribution
C              is obtained from a stability analysis using the
C              phase descriptors specified in the stream type.
C     FEST2    First estimates of the variable (temperature, pressure or
C              tolerance ratio ) at the solution must be supplied if FEST2
C              is TRUE. If FEST2 is FALSE the initial value of T or P
C              is obtained using an internal procedure.
C     ISTABI   Controls whether a stability test is done on the final
C              solution which matches the specifications in NPD and
C              IPD. Omitting the stability test saves time but an
C              erroneous solution may be returned.
C              JXNOSC = no final stability check
C              JXDOSC = do final stability check
C     NPRINT   Parameter for diagnostics output:
C              =  -1 none
C              =  0 none unless serious error
C              >  0 diagnostics, increasing as NPRINT increased
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     T        If IVAR=JXTEMP, temperature (K).
C              If IVAR=JXPRES and FEST2=TRUE, estimated temperature (K),
C              else not defined
C     P        If IVAR=JXPRES, pressure (Pa).
C              If IVAR=JXTEMP and FEST2=TRUE, estimated pressure (Pa),
C              else not defined
C     NPOUT    If FEST1 is .TRUE. NPOUT must be set to the number of
```

```
5
C               phases present at the estimate of the solution supplied
C               in LNOPD, XX and NMOLPH.
C     LNOPD     If FEST1 is .TRUE. LNOPD must be set on entry.
C               Array of NP PD load numbers corresponding to input
C               values in XX and NMOLPH.
C     XX        If FEST1 is .TRUE. XX must be set on entry.
C               See below for full description.
C     NMOLPH    If FEST1 is .TRUE. NMOLPH must be set on entry.
C               See below for full description.
C
C     Output values:
C     T         Temperature (K)
C     P         Pressure (Pa)
C     NPOUT     number of phases at equilibrium or at best estimate of
C               solution. When an estimate of the solution is supplied
C               some of the input phases may be eliminated.
C     LNOPD     Array of NP PD load numbers corresponding to output
C               values in XX, NMOLPH, ZCOMP and IPHID.
C     XX        XX(I,J) holds the mole fraction of component I in phase
C               J for J = 1,NPOUT.
C     NMOLPH    NMOLPH(J) holds the number of moles in phase J for
C               J = 1,NPOUT. A zero value will be returned for phases at a
C               saturation point (eg. a dew point or bubble point).
C
C     Output arguments:
C     VV        Total volume (m3)
C     HH        Total enthalpy (J)
C     SS        Total entropy (J/K)
C     ZCOUT     Compressibility factor of each output phase (as
C               calculated by the fugacity model, not the volumetric
C               model).
C     IPHOUT    identity (V,L,..etc) of each output phase. If the
C               calculation is successful, the phase id. is set to that
C               specified in the phase descriptor definition. Note that
C               for supercritical phases described by equations of state,
C               the phase label is ambiguous.
C               The phase codes used are defined in AYPTID.INC
C     ISTABO    Stability indicator for solution.
C               = -2  definitely unstable. Another phase would be formed
C                     or an existing phase removed if problem
C                     specification allowed.
C               = -1  marginally unstable (see above).
C               =  0  indeterminate. This value is returned if no final
C                     stability test was requested (ISTABI=0) and no
C                     other error condition was detected. If ISTABI=1 it
C                     means that the solution is on the stability limit.
C               =  1  marginally stable.
C               =  2  definitely stable.
C     NERR      No. errors reported by this routine and sub-routines
C     IERR      Error codes returned. Values for this routine:
C               -21011 Unstable solution returned.
C                      Phase split converged but solution is not stable.
C                      All specified phase descriptors have been used up
C                      or the maximum number of phases has been reached
C                      so no further phase split can be done.
C               -21012 Phase split converged to low accuracy
C               -21013 Other warning
C               21011  Illegal input specification. IVAR, IPROB
C                      or TARGET set wrongly. Negative or zero:
```

```
5
C                     P; T; volume.
C               21012  Inconsistent user-supplied estimates.
C                      Number of input phases NPOUT does not agree
C                      with flags for phases present in LNOPD
C                      or initial number of phases NPOUT is greater than
C                      number of available phase descriptors NPD.
C               21013  Cannot find starting point for calculation - this
C                      may mean there is no solution.
C               21014  PD identifier has unacceptable value
C               21015  Cannot converge temperature or pressure search
C               21016  Not enough phase descriptors to solve problem.
C                      Calculation abandoned.
C               21017  Other fatal error
C               21018  Phase envelope not enabled
```

AXPT02 is called repeatedly to calculate subsequent points on the line being traced. Critical points and gradient discontinuities are fixed by interpolation and codes to denote such points are returned in argument KI. The calling specification is:

```
      SUBROUTINE AXPT02( T, P, VV, HH, SS, ISTABI, NCDIM, NPOUT, LNOPD,
     +                   XX, NMOLPH, ZCOUT, IPHOUT, ISTABO, NI, NIDIM,
     +                   KI, TI, PI, STEPI, NPRINT, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: ISTABI, NCDIM, NI, NIDIM, NPRINT, MXERR
      INTEGER, INTENT(OUT) :: ISTABO, NPOUT, LNOPD(*), IPHOUT(*), KI(*), NERR,
IERR(*)
      REAL*8, INTENT(OUT) :: T, P, VV, HH, SS, XX(NCDIM,*), NMOLPH(*),
ZCOUT(*), TI(*),
     +       PI(*), STEPI(*)

C
C     Input arguments:
C     ISTABI   controls whether a stability test is done on the final
C              solution which matches the specifications in NPDIN and
C              IPDIN. Omitting the stability test saves time but an
C              erroneous solution may be returned.
C              JXNOSC = no final stability check
C              JXDOSC = do final stability check
C     NCDIM    dimension of mole fraction arrays NMOLIN and XX declared
C              in calling routine. NCDIM must be greater than or equal
C              to the the number of components for the flash.
C     NIDIM    dimension of arrays KI, TI, PI
C     NPRINT   parameter for diagnostics output:
C              =  -1 none
C              =  0 none unless serious error
C              >  0 diagnostics, increasing as NPRINT increased
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     T        Temperature (K)
C     P        Pressure (Pa)
C     VV       Total volume (m3)
C     HH       Total enthalpy (J)
C     SS       Total entropy (J/K)
C     NPOUT    number of phases at equilibrium or at best estimate of
C              solution.
C     LNOPD    Array of NP PD load numbers corresponding to output
C              values in XX, NMOLPH, ZCOMP and IPHOUT.
```

```
5
C     XX        XX(I,J) holds the mole fraction of component I in phase
C               J for J = 1,NPOUT.
C     NMOLPH    NMOLPH(J) holds the number of moles in phase J for
C               J = 1,NPOUT. A zero value will be returned for phases at a
C               saturation point (eg. a dew point or bubble point).
C     ZCOUT     Compressibility factor of each output phase (as
C               calculated by the fugacity model, not the volumetric
C               model).
C     IPHOUT    Identity (V,L,..etc) of each output phase. If the
C               calculation is successful, the phase id. is set to that
C               specified in the phase descriptor definition. Note that
C               for supercritical phases described by equations of state,
C               the phase label is ambiguous.
C               The phase codes used are defined in AYPTID.INC
C     ISTABO    Stability indicator for solution.
C               = -2  definitely unstable. Another phase would be formed
C                     or an existing phase removed if problem
C                     specification allowed.
C               = -1  marginally unstable (see above).
C               =  0  indeterminate. This value is returned if no final
C                     stability test was requested (ISTABI=0) and no
C                     other error condition was detected. If ISTABI=1 it
C                     means that the solution is on the stability limit.
C               =  1  marginally stable.
C               =  2  definitely stable.
C     NI        Number of special interpolated points returned
C     KI        Codes of special interpolated points
C               = 1 critical point
C               = 2 discontuity on crossing phase boundary
C     TI        Interpolated temperature
C     PI        Interpolated pressure
C     STEPI     Step to discontinuity/critical point as fraction of normal step

C     NERR      No. errors reported by this routine and sub-routines
C     IERR      Error codes returned. Values for this routine:
C               -21021 Unstable solution returned.
C                      Phase split converged but solution is not stable.
C                      All specified phase descriptors have been used up
C                      or the maximum number of phases has been reached
C                      so no further phase split can be done.
C               -21022 Phase split converged to low accuracy
C               21021  Cannot converge temperature or pressure search
C               21022  Other fatal error
C               21023  PD identifier has unacceptable value
C               21024  Outside allowed range for p/T search
C               21025  No initial points stored, subroutine aborted
```

14.3 Phase envelope for flow assurance

Apart from the aboce functions, there is a convinience subroutine that can be called that automatically calculate the phase boundaries for phases relevant to flow assurance applications: - Vapour-liquid phase envelope - Hydrate I (hydrate model needed) - Hydrate II (hydrate model needed) - Water dew point line - Ice - Wax (Wax model needed) - Asphaltene (Asphaltene model needed)

5

The subroutine that does this calculation automatically is AXPHES.

```
      SUBROUTINE AXPHES( LNOST, NMOL, INP, TPGRID, MTEMP, MPRES,
     +                   MISTAT, NP, NERR, MXERR, IERR )

      REAL*8, INTENT (IN) :: NMOL(*), TPGRID(4)
      INTEGER, INTENT (IN) :: LNOST, INP, MXERR
      REAL*8, INTENT (OUT) :: MTEMP(INP,MAXPHA), MPRES(INP,MAXPHA)
      INTEGER, INTENT (OUT) :: MISTAT(INP,MAXPHA), NP(MAXPHA),
     +                         NERR, IERR(MXERR)

!
!     Input arguments:
!     LNOST   Stream type load number
!     NMOL    Feed molar composition
!     INP     Number of points to be calculated (Max number of points for
output of each line)
!     TPGRID  Vector of temperature and pressure minimum and maximum values
for
!             phase ennvelope calculations. If = 0, default values are used.
!             (1) Tmin (minimum temperature)
!             (2) Tmax (maximum temperature)
!             (3) Pmin (minimum pressure)
!             (4) Pmax (maximum pressure)
!     MXERR   Maximum number of status codes to return
!
!     Output arguments:
!     MTEMP   Temperature points on phase boundaries (matrix of
!             (INP,7) values). Current temperature units.
!             Phase boundaries:
!             (:,1) = VLE
!             (:,2) = Hydrate I
!             (:,3) = Hydrate II
!             (:,4) = Water dew point
!             (:,5) = Ice
!             (:,6) = Wax
!             (:,7) = Asphaltene
!     MPRES   Pressure points on phase boundaries (matrix of (INP,7)
!             values). Current pressure units.
!             Phase boundaries: Same as for MTEMP.
!     MISTAT  Status codes for points on phase boundaries (matrix of
!             (INP,7) values).
!             0  normal point
!             1  discontinuity (where phase boundaries cross)
!             2  critical point
!             Phase boundaries: Same as for MTEMP.
!     NP      Actual number of points for each phase boundary
!     NERR    Number of status codes returned in IERR
!     IERR    Multiflash warning/error codes
!
!     Arguments
!
```

15 Property matching routines

15.1 Introduction

Starting with Multiflash version 6, with the introduction of the PVT experiments, the model tuning methods have changed to take into account the additional availability of experimental data. Also more component properties can be changed: Critical Temperature and Pressure, acentric factor, volume shifts and viscosity parameters.

The Multiflash API matching routines are provided for backward compatability. It is recomended to use the methods available in the Graphical User interface as they provide more flexibility and an interactive way of checking the performance of the tned model.

The Multiflash matching routines are provided to allow the user to match measured phase equilibrium data (e.g. bubble points, dew points or wax appearance points), volumetric data or viscosity values. They operate by adjusting the pure-component properties of the petroleum fractions present in the mixture. Note that if no petroleum fractions are present, none of the matching routines can function.

15.2 Phase equilibrium data

AXMT03 may be called to match a set of phase equilibrium data. Typical applications might be to match one or more dew points for a condensate or bubble point points for an oil. In addition this routine can be used to match solid deposition points such as the wax appearance temperature (WAT) provided a suitable model is available. The properties adjusted depend on the model and the number of data points. The number of (pure component) parameters fitted and the identifiers for the fitted parameters are returned by the routine in the arguments NFITP and IFITP. To find

the values of the matched parameters use the routine AYPCNR as described in section .

The routine will operate with the following fluid phase models: RKS/RKSA (and variants), PR/PRA (and variants), CPA. For matching wax phase fractions the routine will operate with the Coutinho and GFP models.

The calling specification is as follows:

```
      SUBROUTINE AXMT03( LNOST, NMOLIN, IPROB, SR, NPTS, TARGET, T, P,
     +                   WT, NFITP, IFITP, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, IPROB(*), NPTS, MXERR
      INTEGER, INTENT(OUT) :: NFITP, IFITP(*), NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), SR, TARGET(*), T(*), P(*), WT(*)

C
C     Input arguments:
C     LNOST    stream type load number for flash calculations
C     NMOLIN   number of moles of each component in the mixture
C     IPROB    array of codes for type of calculation required
C              (codes are defined in axjnum.inc)
C              IPROB(1)  quantity to match
C                        JXPHML   phase molar fraction
C                        JXPHMS   phase mass fraction
C                        JXPHVL   phase volume fraction
C              IPROB(2)  fixed phase load number
C              IPROB(3)  upper/lower phase fraction search
C                        JSNL     lower normal solution
C                        JSRU     upper retrograde solution
C                        JSNU     upper normal solution
C                        JSRL     lower retrograde solution
C                        JSANY    any solution
C                        JSDEF    default normal solution
C
C              The normal settings for dew/bubble points are:
C                                       Search for    setting
C              Normal dew point            T         upper
C                                          P         lower
C
C              Retrograde dew point        T         lower
C                                          P         upper
C
C              Normal bubble point         T         lower
C                                          P         upper
C
C              Retrograde bubble point     T         upper
C                                          P         lower
C
C              For phase fractions between 0 and 1 the best setting
C              depends on the shape of the phase diagram.
C
C              It is important to specify the correct search code
C              because the value of IPROB(3) is used as a guide, both to
C              generate initial estimates of T or P, and also to decide
C              on a search direction towards the required solution.
C              Particularly for difficult calculations, the user needs
C              to have an idea of the layout of the phase diagram.
C              Note that the JSANY specification is less reliable than
```

```
 5
C               any of the others because the routine has no guidance
C               about where to look for a solution.
C     SR        super/sub saturation ratio. The value of SR
C               should normally be set to 1.0. Values greater than 1
C               mean that the phase is super-saturated and less than 1
C               mean that the phase is undersaturated.
C     NPTS      number of data points to be matched (in T, P, and TARGET
C               arrays)
C     TARGET    array of property values (phase fractions) to be matched
C               (corresponding to the setting of IPROB(1))
C     T         arrays of temperature (K) and pressure (Pa) points
C     P         corresponding to values in TARGET
C     WT        array of weighting factors for data points
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output arguments:
C     NFITP     number of parameters fitted
C     IFITP     pure component property ID codes (array) for the
C               parameters fitted (codes are in ayjpid.inc).
C               IFITP should be dimensioned to receive the maximum number
C               of parameters that can be fiited (currently this is 2).
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               -20551 could not fit to all points - there were some
C                      failures
C               20551  there are no data points to match
C               20552  unacceptable problem specification. This includes
C                      IPROB(1) not being set to one of the codes listed
C                      above, a zero or negative T/P, or a negative phase
C                      fraction target
C               20553  failure initialising Multiflash
C               20554  failure getting model info. for matching
C               20555  failure during single-point match
C               20556  zero total weighting factor - cannot evaluate
C                      optimum parameter
```

The saturation ratio allows for departure from equilibrium in phase formation. A value of 1.0 means equilibrium, values less than 1.0 is sub-saturated relative to equilibrium and valued greater than 1.0 mean super-saturation. The saturation ratio is used to scale the fugacities of the phase that is present in fixed amount.

15.3 Gas/Liquid viscosity

AXMT04 matches a set of gas or liquid viscosity points for a mixture by adjusting the reference viscosities (property code JLVSRF) for all the petroleum fractions in the mixture. The routine works with the generalised Lohrenz-Bray-Clark, Pedersen, Twu and SuperTRAPP models provided in the Multiflash models library. To use the matching procedure with these models the JVLFIT variant must be specified when the model is loaded, see section and Appendix .

The calling specification is as follows:

```
5
      SUBROUTINE AXMT04( LNOST, NMOLIN, IPDIN, NPTS, VISC, T, P, WT,
     +                   IFLASH, NFITP, IFITP, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, IPDIN, NPTS, MXERR
      INTEGER, INTENT(INOUT) :: IFLASH(*)
      INTEGER, INTENT(OUT) :: NFITP, IFITP(*), NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), VISC(*), T(*), P(*), WT(*)
C
C     Input arguments:
C     LNOST    stream type load number: used to find the number of
C              components and subcomponent list
C     NMOLIN   number of moles of each component in the mixture
C     IPDIN    phase descriptor (PD) for the phase to be used to match
C              viscosity
C     NPTS     number of data points to be matched (in T, P, and VISC
C              arrays)
C     VISC     array of viscosity (Pas) points to be matched
C     T        arrays of temperature (K) and pressure (Pa) points
C     P        corresponding to viscosities in VISC
C     WT       array of weighting factors for viscosity data points
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input values:
C     IFLASH   indicates whether the overall composition should be
C              flashed at the given P,T conditions to obtain the
C              phase compositions.
C              0  do not flash (P > bubble point pressure)
C              1  do flash (P < bubble point pressure)
C              Note that if a flash is requested it will be performed at
C              each iteration in the attempt to match viscosities and
C              will take much more time than the 'no flash' option.
C              This option is also less reliable than the 'no flash'
C              option because the specified phase may not be present
C              or may be mis-identified.
C
C     Output values:
C     IFLASH   if IFLASH(i) is set to 1 the mixture is flashed and the
C              value is reset to 0 if only one phase (with PD IPDIN)
C              is found. This assumes that matching the viscosity model
C              has no effect on the phase behaviour of the mixture.
C
C     Output arguments:
C     NFITP    number of parameters fitted
C     IFITP    pure component property ID codes (array) for the
C              parameters fitted (codes are in ayjpid.inc).
C              IFITP should be dimensioned to receive the maximum number
C              of parameters that can be fiited (currently this is 2).
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              20531  PD specified for matching is not part of stream
C              20532  there are no data points to match
C              20533  temp, pressure or viscosity is zero or negative
C              20534  cannot evaluate objective function at initial point
C              20535  cannot increment solution to get a 2nd point
C              20536  function value independent of variable, model
C                     is probably not set up for fitting
C              20537  cannot improve solution to get a third point
```

```
 5
C              20538  quadratic extrapolation failed to improve solution
C              20539  run out of iterations
```

15.4 Volumetric data

AXMT06 matches a set of molar volume or specific volume data points for a mixture by adjusting the properties of the petroleum fractions in the mixture. The volume matched may be the overall volume or the volume of a a single phase in a multiphase mixture.

The routine works with the RKSA and PRA models with the JVDEN or JVDEN1 variants and with the CPA model with the JVSVP variant (see section and Appendix). The number of (pure component) parameters fitted and the identifiers for the fitted parameters are returned by the routine in the arguments NFITP and IFITP. To find the values of the matched parameters use the routine AYPCNR as described in section .

The calling specification is as follows:

```
      SUBROUTINE AXMT06( LNOST, NMOLIN, IPDIN, NPTS, VOL, T, P, WT,
     +                   IUNIT, NFITP, IFITP, NERR, MXERR, IERR  )

      INTEGER, INTENT(IN) :: LNOST, IPDIN, NPTS, IUNIT, MXERR
      INTEGER, INTENT(OUT) :: NFITP, IFITP(*), NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), VOL(*), T(*), P(*), WT(*)

C
C     Input arguments:
C     LNOST    stream type load number: used to find the number of
C              components and subcomponent list
C     NMOLIN   number of moles of each component in the mixture
C     IPDIN    phase descriptor load number corresponding to the phase
C              for which volume is to be matched. If set to zero
C              this means match the total mixture volume.
C     NPTS     number of data points to be matched (in T, P, and VOL
C              arrays)
C     VOL      array of volume points to be matched. The units must be
C              either m3/mol or m3/kg as set by bthe IUNIT argument.
C              If the volume of a single phase is to be matched and if
C              more than one phase is present the molecuar weight used
C              to convert mass units is obtained by flashing the total
C              fluid composition.
C     T        arrays of temperature (K) and pressure (Pa) points
C     P        corresponding to volumes in VOL
C     WT       array of weighting factors for viscosity data points
C     IUNIT    code for units of data in VOL array.
C              0   m3/mol
C              1   m3/kg
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Output values:
C     NFITP    number of parameters fitted
C     IFITP    pure component property ID codes (array) for the
C              parameters fitted (codes are in ayjpid.inc).
C              IFITP should be dimensioned to receive the maximum number
C              of parameters that can be fiited (currently this is 2).
```

```
 5
C
C     Output arguments:
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              20651 The model for the volume calculation and the fluid
C                    model should be same.
C              20652 Matching the volume data only works with RKSA and PRA.
C              20653 Error in problem specification
C              20654 There are no data points to match
C              20655 Too many failures in matching the volume data,
calculation was terminated.
C              20656 Maximum iteration reached, calculation was terminated.
C              20657 specified phase for matching is not in the stream
```

15.5 GOR and Bubble Point Matching

Similar to the Match Bubble Point utility in the GUI, there is an API call to match the GOR alongside the Bubble points. It can be used to match GOR for single stage separators, even if no Bubble point information is available. The new function AXMT07_A allows to match the following properties independently:

- Bubble points
- Single separator GOR
- Water cut

Only one of the above properties will have to be supplied. Additionally, when matching GOR, the hydrocarbon liquid density can also be matched. Note that when matching GOR and/or water cut the compositions (NMOL) will have changed.

The calling specification for AXMT07_A is:

```
      SUBROUTINE AXMT07_A( LNOST, NMOLIN, IPROB, SR, IPDV, IPDL,
     +                     NPT, T, P, PHFRAC, WT, IWAT, WVOL,
     +                     GOR, TGOR, PGOR, LDGOR,
     +                     NMOUT, NFITPH, IFITPH, NFITLD, IFITLD,
     +                     NERR, MXERR, IERR  )

      INTEGER, INTENT(IN) ::  LNOST, IPROB(3), IPDV, IPDL, NPT, MXERR, IWAT
      INTEGER, INTENT(OUT) :: NFITPH, IFITPH(*), NFITLD, IFITLD(*), NERR,
IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), SR, PHFRAC(*), T(*), P(*), WT(*), WVOL
     +  GOR, TGOR, PGOR, LDGOR
      REAL*8, INTENT(INOUT) :: NMOUT(*)

C     Input arguments:
C     LNOST    stream type load number: used to find the number of
C              components and subcomponent list
C     NMOLIN   number of moles of each component in the mixture
C     IPROB    array of codes for type of calculation required
C              (codes are defined in axjnum.inc)
C              IPROB(1)  quantity to match
C                JXPHML    phase molar fraction
C                JXPHMS    phase mass fraction
```

```
C               JXPHVL    phase volume fraction
C             IPROB(2)  fixed phase load number
C             IPROB(3)  upper/lower phase fraction search
C               JSNL      lower normal solution
C               JSRU      upper retrograde solution
C               JSNU      upper normal solution
C               JSRL      lower retrograde solution
C               JSANY     any solution
C               JSDEF     default normal solution
C
C             The normal settings for dew/bubble points are:
C                                     Search for    setting
C             Normal dew point            T         upper
C                                         P         lower
C
C             Retrograde dew point        T         lower
C                                         P         upper
C
C             Normal bubble point         T         lower
C                                         P         upper
C
C             Retrograde bubble point     T         upper
C                                         P         lower
C
C             For phase fractions between 0 and 1 the best setting
C             depends on the shape of the phase diagram.
C
C             It is important to specify the correct search code
C             because the value of IPROB(3) is used as a guide, both to
C             generate initial estimates of T or P, and also to decide
C             on a search direction towards the required solution.
C             Particularly for difficult calculations, the user needs
C             to have an idea of the layout of the phase diagram.
C             Note that the JSANY specification is less reliable than
C             any of the others because the routine has no guidance
C             about where to look for a solution.
C     SR      super/sub saturation ratio. The value of SR
C             should normally be set to 1.0. Values greater than 1
C             mean that the phase is super-saturated and less than 1
C             mean that ther phase is undersaturated.
C     IWAT    Position of water component in component sublist for the
C             stream type refered by LNOST
C     IPDV    PD load number for gas phase for GOR calculation
C     IPDL    PD load number for liquid phase for GOR and density
C             calculation
C     NPT     number of point data ppoints to be matched (in T
C             and P)
C     T       arrays of temperature (K) and pressure (Pa) points
C     P       corresponding to phase fractions in PHFRAC
C     PHFRAC  values of phase fractions to be matched (for phase
C             with PD given by IPROB(2))
C     WT      array of weighting factors for each data point
C     WVOL    Fraction of water present in liquid phase (%)
C     GOR     value of gas/oil ratio in sm3/m3, i.e. m3 of gas at
C             standard conditions (15C, 1bar) per m3 of liquid at
C             TGOR and PGOR
C     TGOR    temperature corresponding to GOR (K)
C     PGOR    pressure corresponding to GOR (Pa)
C     LDGOR   liquid density at GOR conditions (kg/m3). If no value
```

```
 5
C               is available LDGOR should be set to -DLARGE
C     MXERR     maximum number of error flags to be returned in IERR
C
C     Output values:
C     NMOUT     component mole numbers adjusted to match the GOR
C     NFITPH    number of parameters fitted for the fixed phase fraction data
C     IFITPH    pure component property ID codes (array) for the
C               parameters fitted (codes are in ayjpid.inc).
C               IFITPH should be dimensioned to receive the maximum number
C               of parameters that can be fiited (currently this is 2).
C     NFITLD    number of parameters fitted for the liquid density data
C     IFITLD    pure component property ID codes (array) for the
C               parameters fitted (codes are in ayjpid.inc).
C               IFITPH should be dimensioned to receive the maximum number
C               of parameters that can be fitted (currently this is 2).
C
C     Output arguments:
C     NERR      no. errors reported by this routine and sub-routines
C     IERR      error codes returned. Values for this routine:
C               21051 failure matching phase fraction point(s)
C               21052 failure flashing stream at GOR conditions
C               21053 vapour phase not present at GOR conditions
C               21054 liquid phase not present at GOR conditions
C               21055 failure matching liquid volume
C               21056 failure calculating liquid MW
C               21057 failure calculating liquid volume at GOR conditions
C               21058 failure calculating vapour volume at GOR conditions
C               21059 Failed to find the minimum amount of water to add
C               21101 Water phase descriptor not present. Water phase fraction
matching not possible.
C               21102 Failure calculating water phase volume at GOR conditions.

C              -21051 specified GOR is too big, not consistent with the fluid.
C              -21052 specified GOR is too small, not consistent with the
fluid.
```

16 Chemical equilibrium calculations

16.1 Introduction

The Multiflash chemical equilibrium module (Chemreact) is a suite of software for performing simultaneous phase and chemical equilibrium calculations. It can handle mixtures in vapour and/or liquid phases, plus any number of pure solid phases. It incorporates a phase stability analysis procedure whereby it can establish which of the possible phases are present at equilibrium automatically. It has been improved in version 4.2 to handle more liquid phases and make it more robust for generic applications. There is the option of using reaction squemes but this feature is not supported yet.

Chemreact can determine phase and chemical equilibrium for the following specifications:

- fixed pressure and temperature;
- dew point at fixed pressure or temperature;
- bubble point at fixed pressure or temperature;
- fixed pressure and enthalpy;

Additional facilities include the ability to impose a number of empirical corrections to the computed equilibria to allow for situations in real reactors when equilibrium has not been attained.

16.2 Chemreact operation

Chemreact is designed to allow use of different thermodynamic models for different phases and for different properties of a given phase. The solution methods used do not make any assumptions about thermodynamic consistency between fugacity coefficients, volumetric properties, and thermal properties. It is the implementer's responsibility to ensure that the same fugacity

reference state is used for all phases and that a sensible combination of models for different phases is specified.

A typical sequence of operations for using Chemreact is:

1. Initialise physical properties data system.
2. Define list of all components that may be present and load the data for these components.
3. Define or load sets of binary interaction parameters (BIPs).
4. Specify and initialise the set of models that will be available. Define which variants of the models are required and which BIPs the models will use. It is assumed that models are defined in advance of use. A subset of the models defined may be used in any given calculation.
5. Set up all the phase descriptors (PDs) which Multiflash may use: CALL AY_DEFINE_PD. See section for a description of PDs.
6. Initialise stream type system and set up all the stream types required (section).
7. Initialise Chemreact: CALL CRINIT
8. Call one of the top level routines to do a calculation:

Chemreact is designed to work with a subset of the previously defined components and PDs. These subsets are specified in the calls to CREA01 etc.

16.3 Initialising Chemreact

Once the databank(s), components, models, phase descriptors and streams, have been defined, Chemreact should be initialised by calling CRINIT. The calling specification is as follows:

```
 SUBROUTINE CRINIT( NERR, MXERR, IERR )

 INTEGER, INTENT(IN) :: MXERR
 INTEGER, INTENT(OUT) :: NERR, IERR(*)
```

16.4 Chemical equilibrium at fixed pressure and temperature: CREA01

CREA01 can be called to perform chemical equilibrium calculations as follows:

```
 SUBROUTINE CREA01( LNOST, NR, CRNAME, CRSTCO, CRCPNO, T, P, NCDIM,
+                   NMOLIN, NPHMAX, KREAIN, ZETOT, ZEQUIL, NP,
+                   LNOPD, ZCOMP , VAP, XX, IPHID, NPRINT, NERR,
+                   MXERR, IERR )

 INTEGER, INTENT(IN) :: LNOST, NCDIM, NPHMAX, KREAIN(*), NPRINT, MXERR,
NR
```

```
5
      INTEGER, INTENT(OUT) :: NP, LNOPD(*), IPHID(*), NERR, IERR(*)
      REAL*8, INTENT(IN) :: T, P, NMOLIN(*), CRSTCO(MAXRXN, *)
      REAL*8, INTENT(INOUT) :: ZEQUIL(*), XX(NCDIM,*), VAP(*)
      REAL*8, INTENT(OUT) :: ZETOT, ZCOMP(*)
      CHARACTER*72, INTENT(IN) :: CRNAME(MAXRXN, *)

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     NR       Number of reactions
C     CRNAME   Component names present in reactions
C     CRSTCO   Stoquiometric coefficients for each component in each reaction.

C     T        temperature (K)
C     P        pressure (Pa)
C     NCDIM    dimension of mole fraction array XX declared in calling
C              routine. NCDIM must be greater than or equal to the
C              the number of mixture components for the calculation.
C     NMOLIN   mole numbers in the feed.
C     NPHMAX   The maximum number of coexisting phases that the equilibrium
C              calculation is to search for. Used to set value of NPHASE.
C              If NPHMAX is outside the range 1 to MXPHCR it
C              is equivalent to setting it to MXPHCR.
C     KREAIN   code numbers marking reacting fluid components
C              (1=reacting, 0=non-reacting), condensed
C              components (1=solid only, 2=solid or liquid)
C     NPRINT   diagnostic output flag.
C              = -1   no output
C              = 0    minimum output in case of serious errors
C              = 1    errors and serious warnings
C              = 2    warnings plus above
C              = 3    informative messages plus above
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input value:
C     ZEQUIL   starting values for the equilibrium mole fractions
C     VAP      starting value for phase distribution
C     XX       starting values for fluid phase mole fractions
C
C     Output value:
C     ZEQUIL   mole fractions at equilibrium
C     VAP      equilibrium phase distribution
C     XX       equilibrium phase mole fractions
C
C     Output arguments:
C     ZETOT    total number of moles in fluid phases at
C              equilibrium
C     NP       number of phases at equilibrium or at best estimate of
C              solution. When an estimate of the solution is supplied
C              some of the input phases may be eliminated.
C     LNOPD    Array of NP PD load numbers corresponding to output
C              values in XX, NMOLPH, ZCOMP and IPHID.
C     ZCOMP    compressibility factor of each output phase (as
C              calculated by the fugacity model, not the volumetric
C              model).
C     IPHID    identity (V,L,..etc) of each output phase. If the
C              calculation is successful, the phase id. is set to that
C              specified in the phase descriptor definition. Note that
```

```
 5
C              for supercritical phases described by equations of state,
C              the phase label is ambiguous.
C              The phase codes used are defined in AYPTID.INC
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              -30001 WARNING: calculation successful but warning from
C                     thermodynamic model(s) getting properties of
C                     reactants or products
C              30001  illegal input specification negative or zero
C                     P or T
C              30002  illegal input specification reported by startup
C                     routine. Negative or zero:
C                     number of moles of feed; feed mole
C                     fractions; number of phase descriptors;
C                     phase descriptors wrongly specified in input or
C                     wrongly set up prior to calling CREA01.
C              30003  cannot get equilibrium constants
C              30004  convergence failure in CREQ01
```

16.5 Dew and bubble point searches at chemical equilibrium: CREA02

CREA02 searches for either the temperature or pressure at which a dew or bubble point occurs for a system that is simultaneously at chemical equilibrium and at vapour-liquid phase equilibrium:

```
      SUBROUTINE CREA02( LNOST, NR, CRNAME, CRSTCO, CRCPNO, IVAR, TIO,
     +                   PIO, NCDIM, NMOLIN, NPHMAX, KREAIN, NTYPE,
     +                   FEST,ZETOT, ZEQUIL, NP, LNOPD, ZCOMP, VAP, XX,
     +                   IPHID, NPRINT, NERR, MXERR, IERR )

      INTEGER, INTENT(IN) :: LNOST, IVAR, NCDIM, NPHMAX, KREAIN(*), NTYPE,
     +    MXERR, NR, CRCPNO(*)
      INTEGER, INTENT(OUT) :: NP, LNOPD(*), IPHID(*), NPRINT, NERR, IERR(*)
      REAL*8, INTENT(IN) :: NMOLIN(*), CRSTCO(MAXRXN, *)
      REAL*8, INTENT(INOUT) :: TIO, PIO, VAP(*), XX(NCDIM,*)
      REAL*8, INTENT(OUT) :: ZETOT, ZEQUIL(*), ZCOMP(*),
      LOGICAL, INTENT(IN) :: FEST
      CHARACTER*72, INTENT(IN) :: CRNAME(MAXRXN, *)

C
C     Input arguments:
C     LNOST    stream type load number for flash
C     NR       Number of reactions
C     CRNAME   Names of components present in reaction.
C     CRSTCO   Stoquimetric coefficients for components in reactions.
C     CRCPNO   Number of components in each reaction.
C     IVAR     code denoting variable
C              = 1     search for temperature at given pressure
C              = 2     search for pressure at given temperature
C     NCDIM    dimension of mole fraction array XX declared in calling
C              routine. NCDIM must be greater than or equal to the
C              the number of mixture components for the calculation.
C     NMOLIN   mole numbers in the feed (NCOMIX fluid components
C              followed by NCOCND condensed phase components)
```

```
 5
C     NPHMAX   The maximum number of coexisting phases that the equilibrium
C              calculation is to search for. Used to set value of NPHASE.
C              If NPHMAX is outside the range 1 to MXPHCR it
C              is equivalent to setting it to MXPHCR.
C     KREAIN   code numbers marking reacting fluid components
C              (1=reacting, 0=non-reacting), condensed
C              components (1=solid only, 2=solid or liquid)
C     FEST     If .TRUE. an estimate of the equilibrium temperature
C              must be supplied in TIO
C     NTYPE    calculation type (1=dew point, 2=bubble point)
C     NPRINT   diagnostic output flag.
C              = -1   no output
C              = 0    minimum output in case of serious errors
C              = 1    errors and serious warnings
C              = 2    warnings plus above
C              = 3    informative messages plus above
C     MXERR    maximum number of error flags to be returned in IERR
C
C     Input/output arguments:
C     Input value:
C     TIO      If IVAR=2, temperature (K) at which to solve for
C              incipient phase.
C              If IVAR=1 and FEST=TRUE, estimated temperature (K),
C              else not defined
C     PIO      If IVAR=1, pressure (Pa) at which to solve for
C              incipient phase.
C              If IVAR=2 and FEST=TRUE, estimated pressure (Pa),
C              else not defined
C     VAP      starting value for phase distribution
C     XX       starting values for fluid phase mole fractions
C
C     Output value:
C     TIO      temperature (K)
C     PIO      pressure (Pa)
C     VAP      equilibrium phase distribution
C     XX       equilibrium phase mole fractions
C
C     Output arguments:
C     ZETOT    total number of moles in fluid phases at
C              equilibrium
C     ZEQUIL   mole fractions at equilibrium
C     NP       number of phases at equilibrium or at best estimate of
C              solution. When an estimate of the solution is supplied
C              some of the input phases may be eliminated.
C     LNOPD    Array of NP PD load numbers corresponding to output
C              values in XX, NMOLPH, ZCOMP and IPHID.
C     ZCOMP    compressibility factor of each output phase (as
C              calculated by the fugacity model, not the volumetric
C              model).
C     IPHID    identity (V,L,..etc) of each output phase. If the
C              calculation is successful, the phase id. is set to that
C              specified in the phase descriptor definition. Note that
C              for supercritical phases described by equations of state,
C              the phase label is ambiguous.
C              The phase codes used are defined in AYPTID.INC
C     NERR     no. errors reported by this routine and sub-routines
C     IERR     error codes returned. Values for this routine:
C              30021  illegal input specification negative or zero
C                     P or T, Bad IVAR or NTYPE
```

```
      5
C               30022  illegal input specification reported by startup
C                      routine. Negative or zero:
C                      number of moles of feed; feed mole
C                      fractions; number of phase descriptors;
C                      phase descriptors wrongly specified in input or
C                      wrongly set up prior to calling CREA02.
C               30023  cannot find starting point (P or T) for calculation
C               30024  convergence failure in CREQ01
C               30024  unconverged - too many successive failures in
C                      equilibrium routine
C               30025  unconverged - out of iterations
```

17 Configuring Multiflash with a command file

17.1 Introduction

The previous sections of this manual have described how to initialise Multiflash by making low-level subroutine calls. This is the most efficient and direct method of setting up precisely those Multiflash features required by a higher level program such as a process simulator. An alternative and simpler approach is to perform the initialisation by reading a Multiflash command file. The Multiflash command language allows all aspects of a problem setup to be defined. This would typically include databank(s), components, models and phase descriptors. The command language is described in the Multiflash Command Reference.

17.2 Reading a command file

The subroutine MFRDFL_NMOL will read a Multiflash command file and process the commands in the file. To initialise Multiflash the file should contain information on the components, models and phase descriptors. The intention is that MFRDFL_NMOL will be called once to perform the initialisation followed by multiple calls to property/phase equilibrium routines as described in section . The previous version of this routine MFRDFL is also provided for backwards comptibility. The calling specification is as follows:

```
 SUBROUTINE MFRDFL_NMOL( CMDFIL, APPNAM, ERRFIL, NC, IC, NMOL,
+          CNAME,NPD,IPD, PDNAME, IPHID, NST, IST, STNAME, NERR,MXERR,
+          IERR )

 INTEGER, INTENT(OUT) :: NC, IC(*), NPD, IPD(*), IPHID(*),
+                 NST, IST(*), NERR, IERR(*)
 INTEGER, INTENT(IN) :: MXERR
 CHARACTER*(*), INTENT(IN) :: APPNAM, CMDFIL, ERRFIL
 CHARACTER*(*), CNAME(*), PDNAME(*), STNAME(*)
 REAL*8, INTENT(OUT) :: NMOL(*)
```

```
C     Input arguments:
C     CMDFIL   name of Multiflash command file to be processed
C     APPNAM   directory path used to locate Multiflash files
C              (databanks etc.)
C     ERRFIL   name of file for error information generated as the
C              command processor parses the command file. Note that
C              error codes are always returned in the IERR array
C              whatever the setting of ERRFIL.
C              Three operating modes are possible:
C              1. If set to a blank character error codes are only
C                 returned in the IERR array.
C              2. If set to a '*' character, the contents of the command
C                 file are echoed to the default output unit (eg. the
C                 screen) and any errors or warnings generated by the
C                 commands in the file are also displayed.
C                 The output is paused after each screenfull and
C                 the <Enter> key must be pressed to resume.
C              3. If set to a valid file name, the contents of the command
C                 file are echoed to the error file together with any
C                 errors or warnings
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     NC       highest seial number for defined components.
C              This is usually the same as the number of components
C              defined but it possible that the component list
C              (in CNAME) contains gaps.
C     IC       component load numbers
C     NMOL     Component amounts in mol stored in MFL file.
C     CNAME    component names (aliases). The alias is a name by which a
C              user wishes to refer to a component. It need not be one
C              of the names by which the component is recognises on
C              any databank. If an aliases is not specified explicitly
C              the name is the same as the databank name.
C              The list may contain gaps which are denoted by blanks.
c              The highest non-blank entry is CNAME(NC)
C     NPD      number of phase descriptors defined
C     IPD      array of PD load numbers
C     PDNAME   PD names defined in the command file
C     IPHID    phase type of each PD (codes from ayptid.inc)
C     NST      number of stream types defined
C     IST      array of stream type load numbers
C     STNAME   stream names defined in the command file.
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              -230 processing the command file produced warnings
C              576  cannot open MF command file CMDFIL
C              577  cannot open the error file ERRFIL
C              578  MF command file contains serious errors
```

And the old subroutine:

```
      SUBROUTINE MFRDFL( CMDFIL, APPNAM, ERRFIL, NC, IC, CNAME, NPD,
     +                   IPD, PDNAME, IPHID, NST, IST, STNAME, NERR,
     +                   MXERR, IERR )

      INTEGER NC, IC(*), NPD, IPD(*), IPHID(*), NST, IST(*), NERR,
```

```
5
     +        MXERR, IERR(*)
      CHARACTER*(*) APPNAM, CMDFIL, ERRFIL, CNAME(*), PDNAME(*),
     +              STNAME(*)

C
C     Input arguments:
C     CMDFIL   name of Multiflash command file to be processed
C     APPNAM   directory path used to locate Multiflash files
C              (databanks etc.)
C     ERRFIL   name of file for error information generated as the
C              command processor parses the command file.
C              Three operating modes are possible:
C              1. If set to a blank character, all errors are returned
C                 in the IERR array.
C              2. If set to a '*' character, the contents of the command
C                 file are echoed to the default output unit (eg. the
C                 screen) and any errors or warnings generated by the
C                 commands in the file are also displayed.
C                 The output is paused after each screenfull and
C                 the <Enter> key must be pressed to resume.
C              3. If set to a valid file name, the contents of the command
C                 file are echoed to the error file together with any
C                 errors or warnings
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     NC       highest seial number for defined components.
C              This is usually the same as the number of components
C              defined but it possible that the component list
C              (in CNAME) contains gaps.
C     IC       component load numbers
C     CNAME    component names (aliases). The alias is a name by which a
C              user wishes to refer to a component. It need not be one
C              of the names by which the component is recognises on
C              any databank. If an aliases is not specified explicitly
C              the name is the same as the databank name.
C              The list may contain gaps which are denoted by blanks.
c              The highest non-blank entry is CNAME(NC)
C     NPD      number of phase descriptors defined
C     IPD      array of PD load numbers
C     PDNAME   PD names defined in the command file
C     IPHID    phase type of each PD (codes from ayptid.inc)
C     NST      number of stream types defined
C     IST      array of stream type load numbers
C     STNAME   stream names defined in the command file.
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              -230 processing the command file produced warnings
C              576  cannot open MF command file CMDFIL
C              577  cannot open the error file ERRFIL
C              578  MF command file contains serious errors
```

Notes:

1. Before calling MFRDFL_NMOL the security system must be initialised by calling MFSINI as described in section .

2. The Multiflash command file must be a sequential formatted ASCII file.

3. The arguments CMDFIL and, optionally, ERRFIL should contain complete file access strings (pathnames).

4. For a description of the databank files used by Multiflash see section .

5. Component load numbers and names are only returned if the components are defined in the command file.

6. Phase descriptor information is only returned if the models and PDs are defined in the command file.

7. All components and PDs defined in all stream types are returned. To get information on the component list and PD list for a particular stream it is necessary to call the routines described in section .

8. Only stream types defined explicitly are returned. The default stream type (0) can be used to refer to all components and PDs.

17.3 Reading a command file without loading it

For convenience a subroutine MFRMFL is provided to read a command file and return the components defined in that file, but without loading it. That means the currently defined model in Multiflash is unaffected. The calling specification is:

```
      SUBROUTINE MFRMFL( FILNAM, NCOMP, NAME, ALIAS, TYPE, NERR, MXERR,
     +                   IERR )

      INTEGER, INTENT(OUT) :: NCOMP, TYPE(*), NERR, IERR(*)
      INTEGER, INTENT(IN):: MXERR
      CHARACTER*(*), INTENT(OUT) :: NAME(*), ALIAS(*)
      CHARACTER*(*), INTENT(IN)  :: FILNAM

C
C     COPYRIGHT (C) INFOCHEM COMPUTER SERVICES LTD., 2009.
C     ALL RIGHTS RESERVED.
C
C     Reads component information from mfl files.
C
C     Version 1.0,  Tony Moorwood,  17th. February 2009.
C
C     Input arguments:
C     FILNAM   Name of file
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     NCOMP    Number of components
C     NAME     Array of component names
C     ALIAS    Array of component aliases
C     TYPE     Array of component types
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              384  Cannot get a unit number with which to open the file
C              385  Cannot open the file
```

17.4 Carrying out Calculations

Provided that all the necessary information has been supplied in the command file, the call to MFRDFL_NMOL will initialise Multiflash. The program may now make direct calls to any of the pure component data routines (described in section), flash routines, chemical equilibrium routines, or else to AYMIX, AYMIXH, AYVISC, AYTCND, AYSURF or AYDIFF to obtain mixture physical properties. Note that it is not necessary to call any of the initialisation routines such as AXINIT etc. since these will be called as part of reading the command file.

For example, if the file ex1a.mfl contains the following Multiflash commands

```
puredata infodata;
components WATER;
```

Provided that the security system has been successfully initialised the following code fragment will return the vapour pressure of water at 373.15 K using the correlation on the INFODATA databank (which is located in directory c:\mfl).

```
C
C     Error reporting system:
C
      INTEGER MXERR
      PARAMETER (MXERR=10)
      INTEGER NERR, IERR(MXERR)
C
      INTEGER NC, IC(MAXNCS), NPD, IPD(MAXPD), IPHID(MAXPD), IC(1),
     +        NST, IST(MAXST)
      REAL*8 VP(1), ADUM(1), NMOL(MAXNCS)
      CHARACTER*20 CNAME(MAXNCS), PDNAME(MAXPD), STNAME(MAXST)
C
C     Read and process Multiflash command file:
C
      CALL MFRDFL_NMOL( 'ex1a.mfl', 'c:\mfl', ' ', NC, IC, NMOL, CNAME,
     +             NPD, IPD, PDNAME, IPHID, NST, IST, STNAME,
     +             NERR, MXERR, IERR )
      IF ( NERR.NE.0 ) THEN
        PRINT*,'Error processing MF command file. Errors:'
        PRINT*,(IERR(I),I=1,NERR)
        GO TO 990
      ENDIF
C
C     Get vapour pressure:
C
      CALL AYPSVP( 1, IC, 373.15D0, .FALSE., VP, ADUM,
     +             NERR, MXERR, IERR )
      IF ( NERR.NE.0 ) THEN
        PRINT*,'Error evaluating VP'
      ELSE
        PRINT*,'VP of ',CNAME(1),' at 373.15K is ',VP(1),' Pa'
      ENDIF
```

It would be straightforward to calculate properties for a different substance simply by changing the command file. No program changes are necessary. Further examples are provided in sections ,

and which contain complete working programs. These can be compared with the equivalent low-level calls in sections , and which require more code to acomplish the same result.

17.5 Writing a command file

The current state of the Multiflash models as stored in the command processor can be written out at any time by calling MFWRPF. The user enters the name of the file on which the commands are to be written and Multiflash writes out a description of the model. The calling specification is:

```
      CALL MFWRPF( FILNAM, NERR, MXERR, IERR )

      INTEGER NERR, MXERR, IERR(*)
      CHARACTER*(*) FILNAM

C
C     Input arguments:
C     FILNAM   Name of file
C     MXERR    maximum number of error codes to return in IERR
C
C     Output arguments:
C     NERR     number of error codes returned in IERR
C     IERR     error codes. Values for this routine:
C              384  Cannot get a unit number with which to open the file
C              385  Cannot open the file
```

17.6 Ending the Multiflash session

When MFRDFL_NMOL is used to initialise Multiflash a number of files will be opened and the security system (see section) will be initialised. To close all files and release any licences a call should be made to MFEND. This INTEGER function has no arguments. For example:

```
      INTEGER STATUS

      STATUS = MFEND()
```

MFEND should only be called if no further calls will be made to the Multiflash dll.

17.7 Converting a PVTSim CHC/CTM file into an MFL file

Starting with version 4.2 of Multiflash it is possible to read PVTSim CHC files. It is not possible to get an exact model replication for the transport properties models (viscosity, thermal

conductivity, and surface tension) since the two software packages have different models or different implementations of common models. However, in the case of the Lohrenz-Bray-Clark (LBC) model for viscosity, the converter can retrieve all the necessary parameters from the CHC file, if this model is selected for the calculation of the viscosity.

Starting with version 4.3 the converter allows you to define the number of phases that you wish to use (e.g., Vapour, Liquid 1, Liquid 2, and/or Water). In addtion from version 4.3.11, the user has the option of calculating the density of the vapor phase using the GERG 2008 model if so desired, while estimating the rest of the thermodynamic properties with the model defined in the original CHC file.

The import tool will also import CTM files, which is the format used to export multiple feed from PVTSim to be used in OLGA compositional tracking.

The converter can retrieve from the CHC/CTM file the following information:

- Pure component parameters. (This includes checking differences in library components);
- van der Waals mixing rule binary parameters. Constant and temperature dependent;
- Huron-Vidal-Pedersen (HVP) mixing rule binary parameters. Constant and temperature dependent;
- Which equation of state was used (RKS or PR);
- Which Peneloux volume correction function was used: none, constant, linear with temperature;
- Amounts of the defined components.
- Parameters for the LBC and Pedersen viscosity models.
- composition of streams defined in CTM files

In addtion, the converter gives the user the following extra options:

- Allows adding water to the list of define components (if water was not defined in the CHC file).
- Allows defining phases (e.g., vapour, liquid 1, water, etc.).
- Gives the choice of calculating vapour phase density using GERG 2008 model (from version 4.3.11).

The resulting MFL can then be read using MFRDFL_NMOL.

The interface to the import tool is:

```
      SUBROUTINE MF_PVTSF_Convert( PVTSFile, MFLFile, Opts, Nerr, Mxerr, Ierr
)

     CHARACTER(*), INTENT(IN) :: PVTSFile
     CHARACTER(*), INTENT(IN) :: MFLFile
     INTEGER, INTENT(IN) :: Mxerr
```

```
5
      INTEGER, INTENT(IN) :: opts(*)
      INTEGER, INTENT(OUT) :: Nerr
      INTEGER, DIMENSION(*), INTENT(OUT) :: Ierr

!
!     Input arguments:
!     PVTSFile       full pathname of PVTsim .chc file
!     MFLSFile       full pathname of Multiflash .mfl file to be written
!     opts           User input options: (For transport properties models, 0,
means no model.)
!                    (1) Viscosity model (for Pedersen Heavy variant use
JMPDV+1000)
!                    (2) Thermal conductivity model
!                    (3) ST model
!                    (4) Add water? (0 for no, any other number for yes)
!                    (5) Define Vapour phase ( 0 = No, 1 = Yes, with model
from CHC, 2 = Vapour phase with GERG )
!                    (6) Define Liquid 1 phase ( 0 = No, 1 = Yes, with model
from CHC )
!                    (7) Define Liquid 2 phase (  0 = No, 1 = Yes, with model
from CHC )
!                    (8) Define Water phase (  0 = No, 1 = Yes, with model
from CHC )
!                    Model codes come from AYJMID.INC
!     Mxerr          maximum number of error codes to return in Ierr
!
!     Output arguments:
!     Nerr           no. errors reported by this routine and lower level
routines
!     Ierr           Error codes. Values for this routine:
!                    1129 Unable to open PVTSim CHC file
!                    1130 Unable to open MFL file to write.
!                    1131 Error reading the CHC file contents.
!                    1132 The CHC file is empty
!                    1133 The CHC file does not define any components or
models.
!                    1134 Error reading component information from CHC file.
!                    1135 Error writing components into MFL file.
!                    1136 Unable to allocate components or BIP arrays.
!                    1137 Error reading BIP information from CHC files.
!                    1138 Error writting BIP information to MFL file.
!                    1139 Error writting models information to MFL file.
!                    1140 Error writting amounts information to MFL file.
!                    1141 Error writting amounts information to MFL file.
```

18 Unit conversion utilities

A set of of basic unit conversion routines is provided. Before any of the following routines is called it is necessary to initialise the machine-dependent numerical constants stored by Multiflash. This is done by calling any of the initialisation routines described in other sections or by a direct call to AYNUMI (the routine has no arguments). The following conversion routines are available:

property	conversion routine
amount/composition/flowrate	UNF
density/volume (molar/specific)	UND
diffusion coefficient	UNI
energy (molar/specific)	UNH
entropy/heat capacity (molar/specific)	UNS
pressure	UNP
surface tension	UNR
temperature	UNT
thermal conductivity	UNU
viscosity	UNV
volume	UNVOL

18.1 UNF: Amount/composition/flowrate

This routine converts amounts, compositions or flowrates. To convert between mole fractions and mass fractions specify mole and kg units as input and output units. The calling specification is:

```
      SUBROUTINE UNF( I, O, NC, MW, F, FTOT, AVMW, IERROR )

      INTEGER I, O, NC, IERROR
      REAL*8 MW(*), F(*), FTOT, AVMW

C
C     Input arguments:
C     IU        code for input units:
C               1     mol
C               2     kmol
C               3     kg
C               4     lb
```

```
5
C               5     lbmol
C               6     g
C     OU        code for output units - as for input units
C     NC        number of component amounts in vector F
C     MW        array of relative molar masses for each component
C               (average molecular weight in g/mol). Must be set
C               correctly if conversion between molar and mass units
C               is required but can be set to arbitrary positive
C               non-zero values (say 1) if conversion is between two
C               sets of mass or molar units. If MW(I) is set to -DLARGE
C               (aycnum.inc) this is taken to mean missing data and the
C               value in F(I) is not changed.
C
C     Input/output arguments:
C     Input values:
C     F         component amounts (array of NC values) in units
C               corresponding to IU
C     FTOT      total amount
C
C     Output values:
C     F         component mole or mass fractions amounts for components
C               with defined MW in units corresponding to OU
C     FTOT      total amount for components with defined MW
C
C     Output arguments:
C     AVMW      relative molar mass for mixture. The value returned is
C               only meaningful if MW has been correctly set on input.
C     IERROR    Error code  (defined in unjerr.inc)
C               JUNERR_OK         OK
C               JUNERR_BADVALUE   numerical value is below physical limit
C               JUNERR_BADCODE    unit id. code is not recognised
C               JUNERR_NOMW       the molecular weight is not set
C               JUNERR_NOMATERIAL total amount of material is zero
```

18.2 UND: Density/volume

This routine converts density, specific volume or molar volume quantities. For total volume use UNVOL. The calling specification is:

```
      SUBROUTINE UND( I, O, AVMW, D, IERROR )

      INTEGER I, O, IERROR
      REAL*8 AVMW, D

C
C     Input arguments:
C     I         Input density/volume units
C               1   mol m-3
C               2   kg m-3
C               3   lb ft-3
C               4   lbmol ft-3
C               5   mol cm-3
C               6   g cm-3
C               7   kmol m-3
C               8   m3 mol-1
```

```
5
C               9   m3 kg-1
C               10  ft3 lb-1
C               11  ft3 lbmol-1
C               12  cm3 mol-1
C               13  cm3 g-1
C               14  m3 kmol-1
C     O         Output density/volume units - codes as above
C     AVMW      relative molar mass for mixture (average molecular
C               weight in g/mol). Must be set correctly if conversion
C               between molar and mass units is required but can be set
C               to -DLARGE if conversion is between two sets of mass or
C               molar units.
C
C     Input/output arguments:
C     D         Density/volume
C
C     Output argument:
C     IERROR    Error code  (defined in unjerr.inc)
C               JUNERR_OK         OK
C               JUNERR_BADVALUE   numerical value is below physical limit
C               JUNERR_BADCODE    unit id. code is not recognised
C               JUNERR_NOMW       the molecular weight is not set
```

18.3 UNI: Diffusion coefficient

This routine converts diffusion coefficient units. The calling specification is:

```
      SUBROUTINE UNI( I, O, DC, IERROR )

      INTEGER I, O, IERROR
      REAL*8 DI

C
C     Input arguments:
C     I         Input diffusion coefficient units
C               1     m2/s
C               2     cm2/s
C     O         Output units - codes as above
C               (if O and I are the same no conversion is attempted)
C
C     Input/output arguments:
C     DC     diffusion coefficient
C
C     Output argument:
C     IERROR    Error code  (defined in unjerr.inc)
C               JUNERR_OK         OK
C               JUNERR_BADVALUE   numerical value is below physical limit
C               JUNERR_BADCODE    unit id. code is not recognised
```

18.4 UNH: Energy (molar/specific)

This routine converts molar or specific energy units - this includes enthalpy, internal energy and Gibbs energy. The calling specification is:

```
      SUBROUTINE UNH( I, O, AVMW, H, IERROR )

      INTEGER I, O, IERROR
      REAL*8 AVMW, H

C
C     Input arguments:
C     I         Input specific/molar energy units
C               1   J mol-1     (kJ kmol-1)
C               2   kJ mol-1    (MJ kmol-1)
C               3   MJ mol-1    (GJ kmol-1)
C               4   cal mol-1
C               5   kcal lbmol-1
C               6   BTU lbmol-1
C               7   kJ kg-1     (J g-1)
C               8   BTU lb-1
C               9   kcal lb-1
C               10  kcal mol-1
C               11  cal g-1     (kcal kg-1)
C               12  J/kmol      (mJ/mol)
C               13  J/kg
C     O         Output specific energy units - codes as above
C     AVMW      relative molar mass for mixture (average molecular
C               weight in g/mol). Must be set correctly if conversion
C               between molar and mass units is required but can be set
C               to -DLARGE if conversion is between two sets of mass or
C               molar units.
C
C     Input/output arguments:
C     H         Specific/molar energy
C
C     Output argument:
C     IERROR    Error code  (defined in unjerr.inc)
C               JUNERR_OK         OK
C               JUNERR_BADVALUE   numerical value is below physical limit
C               JUNERR_BADCODE    unit id. code is not recognised
C               JUNERR_NOMW       the molecular weight is not set
```

18.5 UNS: Entropy/heat capacity (molar/specific)

This routine converts molar or specific units of energy/temperature interval - this includes entropy and heat capacity. The calling specification is:

```
      SUBROUTINE UNS( I, O, AVMW, S, IERROR )

      INTEGER I, O, IERROR
      REAL*8 AVMW, S
```

```
C
C     Input arguments:
C     I         Input specific entropy/heat capacity unit code
C     O         Output specific entropy/heat capacity unit code
C               1   J/mol/K
C               2   cal/mol/K
C               3   BTU/lbmol/F
C               4   kcal/lbmol/F
C               5   J/kg/K
C               6   kJ/kg/K
C               7   BTU/lb/F
C               8   kcal/lb/F
C               9   GJ/kmol/K
C               10  MJ/mol/K
C               11  kJ/kmol/K
C               12  J/kmol/K
C     AVMW      relative molar mass for mixture (average molecular
C               weight in g/mol). Must be set correctly if conversion
C               between molar and mass units is required but can be set
C               to -DLARGE if conversion is between two sets of mass or
C               molar units.
C
C     Input/output arguments:
C     S         Specific entropy/heat capacity
C
C     Output argument:
C     IERROR    Error code  (defined in unjerr.inc)
C               JUNERR_OK        OK
C               JUNERR_BADVALUE  numerical value is below physical limit
C               JUNERR_BADCODE   unit id. code is not recognised
C               JUNERR_NOMW      the molecular weight is not set
```

18.6 UNP: Pressure

This routine converts pressure units. The calling specification is:

```
SUBROUTINE UNP( I, O, P, IERROR )

INTEGER I, O, IERROR
REAL*8 P
```

```
C
C     Input arguments:
C     I         Input pressure units
C               1      pascal
C               2      kilo pascal
C               3      mega pascal
C               4      bar
C               5      bar gauge
C               6      atmosphere
C               7      atmosphere gauge
C               8      psi
C               9      psi gauge
C               10     mm mercury
C               11     inches water gauge
```

```
5
C               12     at (technical atmosphere kgf/cm2)
C               13     at gauge
C      O        Output pressure units - codes as above
C
C      Input/output arguments:
C      P        Pressure
C
C      Output argument:
C      IERROR   Error code  (defined in unjerr.inc)
C               JUNERR_OK        OK
C               JUNERR_BADVALUE  numerical value is below physical limit
C               JUNERR_BADCODE   unit id. code is not recognised
```

18.7 UNR: Surface tension

This routine converts surface tension units. The calling specification is:

```
      SUBROUTINE UNR( I, O, ST, IERROR )

      INTEGER I, O, IERROR
      REAL*8 ST

C
C     Input arguments:
C     I        Input surface tension units
C              1     N/m
C              2     mN/m
C              3     dyne/cm
C     O        Output units - codes as above
C              (if O and I are the same no conversion is attempted)
C
C     Input/output arguments:
C     ST     surface tension
C
C     Output argument:
C     IERROR   Error code  (defined in unjerr.inc)
C              JUNERR_OK        OK
C              JUNERR_BADVALUE  numerical value is below physical limit
C              JUNERR_BADCODE   unit id. code is not recognised
```

18.8 UNT: Temperature

This routine converts temperature units. The calling specification is:

```
      SUBROUTINE UNT( I, O, T, IERROR )

      INTEGER I, O, IERROR
      REAL*8 T

C
C     Input arguments:
```

```
 5
C      I         Input temperature units
C                1     kelvin
C                2     degrees Celcius
C                3     degrees Fahrenheit
C                4     degrees Rankine
C      O         Output temperature units - codes as above
C
C      Input/output arguments:
C      T         Temperature
C
C      IERROR    Error code  (defined in unjerr.inc)
C                JUNERR_OK         OK
C                JUNERR_BADVALUE   numerical value is below physical limit
C                JUNERR_BADCODE    unit id. code is not recognised
```

18.9 UNU: Thermal conductivity

This routine converts thermal conductivity units. The calling specification is:

```
      SUBROUTINE UNU( I, O, U, IERROR )

      INTEGER I, O, IERROR
      REAL*8 U

C
C      I         Input thermal conductivity units
C                1     W m-1 K-1
C                2     kW m-1 K-1
C                3     kJ hr-1 m-1 K-1
C                4     BTU hr-1 ft-1 F-1
C                5     kcal hr-1 ft-1 F-1
C                6     cal cm-1 s-1 K-1
C                7     kcal cm-1 hr-1 K-1
C                8     mW cm-1 K-1
C                9     W cm-1 K-1
C                10    mW m-1 K-1
C      O         Output units - codes as above
C
C      Input/output arguments:
C      U         Thermal conductivity
C
C      Output argument:
C      IERROR    Error code  (defined in unjerr.inc)
C                JUNERR_OK         OK
C                JUNERR_BADVALUE   numerical value is below physical limit
C                JUNERR_BADCODE    unit id. code is not recognised
```

18.10 UNV: Viscosity

This routine converts viscosity units. The calling specification is:

```
5
      SUBROUTINE UNV( I, O, V, IERROR )

      INTEGER I, O, IERROR
      REAL*8 V

C
C     Input arguments:
C     I        Input viscosity units
C              1     Pa s
C              2     cP (centipoise = mPas)
C              3     mPas
C              4     microPas
C              5     dynes/cm2 (dyne.s/cm2)
C              6     microP (micropoise)
C     O        Output units - codes as above
C
C     Input/output arguments:
C     V        Viscosity
C
C     Output argument:
C     IERROR   Error code  (defined in unjerr.inc)
C              JUNERR_OK        OK
C              JUNERR_BADVALUE  numerical value is below physical limit
C              JUNERR_BADCODE   unit id. code is not recognised
```

18.11 UNVOL: Volume

This routine converts units of total volume. For specific or molar volume or density use UND. The calling specification is:

```
      SUBROUTINE UNVOL( I, O, VOL, IERROR )

      INTEGER I, O, IERROR
      REAL*8 VOL

C
C     Input arguments:
C     I        Input volume units
C              1   m3
C              2   dm3 (l)
C              3   cm3
C              4   mm3
C              5   yd3
C              6   ft3
C              7   in3
C              8   pint (UK)
C              9   gal (UK)
C              10  pint (US)
C              11  gal (US)
C              12  barrel (US)
C     O        Output volume units - codes as above
C
C     Input/output arguments:
C     VOL      volume
C
C     Output argument:
```

```
 5
C     IERROR   Error code  (defined in unjerr.inc)
C              JUNERR_OK        OK
C              JUNERR_BADVALUE  numerical value is below physical limit
C              JUNERR_BADCODE   unit id. code is not recognised
```

5

Appendices

Infochem data banks

This section describes the contents and structures of the databanks supplied by and supported by Infochem. These include databanks for pure component data and for binary interaction parameters. This information should allow the programmer to extend or modify the supplied databanks or to create new databanks in a compatible format.

Infodata

Infodata is the Infochem pure component data bank for normal components. These are components that may be present in mixed fluid or solid phases. Some thermodynamic models may only be valid for pure solid or fluid phases but any component from an Infodata format databank could, in principle, be part of a mixed phase. The other class of components recognised by Multiflash are pure condensed components (see Section). This format is also used for the Infochem version of the DIPPR databank.

Data bank structure

The Infodata data bank is supplied in its standard form as two files INFODATA.BIN and INFODATA.IDX which contain the physical property data and names index respectively. The .BIN file is a direct access unformatted file with the data for each compound. The .IDX file is a sequential formatted file that contains a list of all compound names with pointers to the data record in the .BIN file. The DIPPR data bank is also available in the same format and is supplied as the files DIPPR.BIN and DIPPR.IDX. To create data files in the Infodata structure, an ASCII file needs to be created which contains the databank header and a set of keywords for the properties for each component.

Table 1: Header structure

Keywords Descriptions	Fortran input format	
DATABANK	Name of databank	Up to 12 characters
REVISIONDATE	The date that the databank was changed	Format: dd.mm.yy
VERSION	Version number of the databank	Numeric value

Table 2: The component name and its synonyms in Infodata ASCII file record

data entry	Description	
COMPONENT	keyword for a component name	
Name1	The component name	Up to 72 characters
DATA SYNONYMS	keyword for starting synonyms	
Name2	1st synonym	Up to 72 characters
Name3	2nd synonym	Up to 72 characters
etc.		

Table 3: Constant Properties in Infodata ASCII file record

keywords	Description
ACENTRICFACTOR	acentric factor
ASSAC	CPA a_c parameter ($J\,m^3\,mol^{-2}$)
ASSBC	CPA b parameter ($m^3\,mol^{-1}$)
ASSBETA	CPA β parameter
ASSDELTA	CPA δ_{AB} parameter (K^{-1})
ASSEPSILON	CPA ϵ_{AB} parameter ($J\,mol^{-1}$)
ASSFF	Number of association sites (CPA)
ASSGAMMA	CPA γ_{AB} parameter
ASSKAPPA	CPA κ parameter
CARNUMBER	CAR number (up to 12 character)
COMPREFNO	Component reference number
CNUMBER	Carbon number for petroleum fractions
CPMELT	Difference in C_p on fusion at the normal melting point (in $J\,mol^{-1}$)
DIPOLEMOMENT	Dipole moment (Debye units)
EOSC	Corresponding states reference equation code
EXPANSIVITY	thermal expansivity of liquid at 1 atm and 60 F (K^{-1})
FAMILYCODE	Chemical family code (up to 20 character)
FLAMLOWER	Lower flammability limit (volume % in air)
FLAMUPPER	Upper flammability limit (volume % in air)
FORMULA	Chemical formula (up to 20 character)
GFORMATION	Gibbs energy change on formation in the perfect gas state at 298.15K and 10^5 Pa ($J\,mol^{-1}$)
HCOMBUSTION	Standard net enthalpy change on combustion at 298.15K and 10^5 Pa ($J\,mol^{-1}$)
HFORMATION	Enthalpy change on formation in the perfect gas state at 298.15K ($J\,mol^{-1}$)
HMELT	Enthalpy change on fusion at the normal melting point ($J\,mol^{-1}$)
HOCASS	Hayden-O'Connell self-association parameter
HYD1	Hydrate parameter 1
HYD2	Hydrate parameter 2
HYD3	Hydrate parameter 3
HYDOC	Hydrate cavity occupation codes
LJBVISC	Lennard-Jones σ parameter (m)

LJEVISC	Lennard-Jones ϵ / k parameter for viscosity (K)
MCPR1	Mathias-Copeman parameter 1 for PRA EOS
MCPR2	Mathias-Copeman parameter 2 for PRA EOS
MCPR3	Mathias-Copeman parameter 3 for PRA EOS
MCRKS1	Mathias-Copeman parameter 1 for RKSA EOS
MCRKS2	Mathias-Copeman parameter 2 for RKSA EOS
MCRKS3	Mathias-Copeman parameter 3 for RKSA EOS
MOLECULARWEIGHT	Molecular weight (relative molar mass)
OMASCALE	scaling factor to give conventional value of Ω_a for cubic EOS
OMBSCALE	scaling factor to give conventional value of Ω_b for cubic EOS
PARACHOR	Parachor ($\frac{(dynecm^{-1})^1}{4} cm^3/mol$)
PCRIT	Critical pressure (Pa)
PTRIPLE	Triple point pressure (Pa)
QUNIQUAC	Uniquac q parameter
RADGYR	Radius of gyration (m)
REFRACTINDEX	Refractive index at 298.15K
REFVISCOSITY	liquid viscosity at reference temperature and 1 atm (Pa s)
RUNIQUAC	Uniquac r parameter
SAFTBOND	PC-SAFT: bond segment id. and abundance
SAFTEPSILON	PC-SAFT: ϵ_{AB} parameter (K)
SAFTEK	PC-SAFT: ϵ / k parameter (K)
SAFTFF	PC-SAFT: number of association sites
SAFTGAMMA	PC-SAFT: γ_{AB} parameter
SAFTKAPPA	PC-SAFT: κ parameter
SAFTLAMBDA	PC-SAFT: λ parameter
SAFTM	PC-SAFT: m parameter
SAFTSIGMA	PC-SAFT: σ parameter (m)
SMELT	Entropy change on fusion at the normal melting point ($Jmol^{-1}K^{-1}$)
SOLIDSOLUPAR	Solubility parameter for solid at 298.15K ($(Jm^{-3})^{1/2}$)
SOLUPAR	Solubility parameter for liquid at 298.15K ($(Jm^{-3})^{1/2}$)
SPGRAVITY	Specific gravity of liquid at 60F, 1atm relative to water
SSTANDARD	Standard entropy in the perfect gas state at 298.15K and 10^5 Pa ($Jmol^{-1}K^{-1}$)
TAUTO	Autoignition temperature (K)
TBOIL	Normal boiling point (K)
TCRIT	Critical temperature (K)
TFLASH	Flash point temperature (K)
THLWATER	Characteristic temperature for Henry's Law correlation for solubility of a component in water (K)
TMELT	Normal melting point (K)
TTRIPLE	Triple point temperature (K)
TYPE	Type of components
UNIFAC	UNIFAC subgroups and abundance
V25	Liquid molar volume at 298.15K (m^3mol^{-1})
VCRIT	Critical molar volume (m^3mol^{-1})
VHLWATER	Characteristic volume for Henry's Law correlation for

	solubility of a component in water ($m^3 mol^{-1}$)
VMELT	Difference in molar volume on fusion at the normal melting point ($m^3 mol^{-1}$)
VSRKS1	Peneloux volume shift parameter 1 for RKSA EOS
VSRKS2	Peneloux volume shift parameter 2 for RKSA EOS
VSRKS3	Peneloux volume shift parameter 3 for RKSA EOS
VSPR1	Peneloux volume shift parameter 1 for PRA EOS
VSPR2	Peneloux volume shift parameter 2 for PRA EOS
VSPR3	Peneloux volume shift parameter 3 for PRA EOS
ZCRIT	Critical compressibility factor

Table 4: Temperature-dependent properties in Infodata ASCII file record

keywords	Description
CPIDEAL	Perfect gas C_p equation number Coefficients for perfect gas C_p
CPLIQUID	Saturated liquid C_p equation number Coefficients for saturated liquid C_p
CPSOLID	Solid C_p equation number Coefficients for solid C_p
DIELECTRIC	Dielectric constant equation number Coefficients for dilecrtic constsnt correlation
HVAP	Enthalpy of vaporisation equation number Coefficients for enthalpy of vaporisation
LDENS	Saturated liquid density equation number Coefficients for saturated liquid density
LVISC	Saturated liquid viscosity equation number Coefficients for saturated liquid viscosity
LTHCOND	Saturated liquid thermal conductivity equation number Coefficients for saturated liquid thermal conductivity
PSAT	Saturated vapour pressure equation number Coefficients for saturated vapour pressure
STENSION	Surface tension equation number Coefficients for surface tension
SDENS	Solid density equation number Coefficients for solid density
VIRIALCOEFF	Second virial coefficient equation number Coefficients for second virial correlation
VTHCOND	Zero density gas thermal conductivity equation number Coefficients for zero density thermal conductivity
VVISC	Zero density gas viscosity equation number Coefficients for zero density gas viscosity

Notes:

- The Chemical Abstracts Registry (CAR) number should be a 12 character string consisting of 6 digits (left padded with zeros if necessary) followed by a hyphen (-) followed by 2 digits followed by another hyphen followed by a single digit followed by a blank. Multiflash uses the CAR numbers to recognise special components such as water etc. Any

character string may be entered for the CAR number but the component may not then be recognised for what it is.

- The compound names are any set of 72 characters. The program suppresses blanks and (most) punctuation marks in the names. Each name in every file must be unique in its converted form.
- The chemical formula is a string of up to 20 characters giving the empirical chemical formula. Each element is indicated by its chemical symbol followed by its (integer) abundance in the molecule. The letters in the chemical symbols must be in the correct case: first or only letter upper case, second letter lower case. An abundance of 1 can be omitted. e.g. CO or C1O1 represents carbon monoxide, Co cobalt. Brackets must not be used.
- The chemical family code is a string of up to 20 characters. The codes stored in the Infodata and DIPPR databanks are the DIPPR chemical family codes which use 3 characters.
- The UNIFAC subgroups are indicated by their identifiers enclosed in apostrophes. The Multiflash UNIFAC subgroup identifiers valid for the cuurent version of Multiflash can be found in the Multiflash Command Reference.
- The component reference number can be assigned any value the user may wish. The number is stored by Multiflash but not used in any other way. The number entered must be an integer, but it is subsequently stored as a real by Multiflash.
- All numerical data are stored as REAL*4 values. The precision and range are machine-dependent but on most computer systems 7 significant (decimal) digits are stored and the (decimal) exponent must be in the range -38 to $+38$.
- The databank header must be at the top of the Infodata ASCII file with a correct order and keywords which are given in Table .
- The data record for each component is terminated by a semi-colon and separated by a line starting with # which is followed by hyphen to separate the data record of each component. Within a data block for a component or the data bank header, each line starts with a keyword followed by either a character string or numeric value, then terminated by a semi-colon except for the keywords COMPONENT and DATA SYNONYMS.
- For each component, the data record starts with the keyword COMPONENT, which is then followed by the component's main name on a new line. Then the keyword DATA SYNONYMS must be present in the next new line for the components' synonyms. Each component can have several synonyms which includes its CARN number. Each of the synonyms occupies a line after the keyword DATA SYNONYMS. Finally the keyword COMPONENT is terminated by a semi-colon. The correct order for a component name and its synonyms is given in Table .

- The other constant properties in the data record are defined in Tables and . In general, each entry of the constant physical properties starts with a Multiflash keyword on a new line, followed by either character string or numeric value and terminated by a semi-colon. For the UNIFAC structure, it starts with the keyword UNIFAC, but followed by a set of subgroups. Each set of the subgroup includes character string and its abundance separated by a space, and then terminated by a semi-colon. The constant properties can be in any kind of orders as long as the keywords for physical properties are recognised by Multiflash. If the value of any is unknown, the entry of the property can be omitted. However, if it is not omitted, it should be set to the data unknown value of -SLARGE (SLARGE is set up by subroutine AYNUMI and is currently 1.0E35).
- For each of the temperature dependent properties, the data starts with a Multiflash keyword followed by the Infochem equation number, the appropriate number of coefficients and the minimum and maximum temperature limits for the correlation. The record is terminated by a semi-colon. Details of all the equations are given in Appendix . The correct number of coefficients for each equation can also obtained from Appendix by adding 2 to the number of coefficients listed (for T_{min} and T_{max} . Note that for the perfect gas C_p the coefficients start at coefficient a_3 ; coefficients a_1 and a_2 are calculated by the program.
- If the data for any temperature dependent property are unknown, the data entry for the temperature dependent property can be omitted or the equation number should be set to zero and no coefficients should be entered for that property. There is no restriction about the order of the temperature properties.

The data structure described above contains a complete specification of the data for one compound. To build a up a data file, data is entered in sequence for as many compounds as required.

Creating the data bank

Once the sequential data file has been written, it is converted to a direct access data file and a names index by the program MKIDBK3. The program works interactively with the user naming the file he wants to read and the files to be created. An appropriate version of MKIDBK3 is required for each target operating system and compiler.

The direct access file and names index for the Infodata bank are normally given the names infodata.bin and infodata.idx respectively. The usual names for the DIPPR databank files are dippr.bin and dippr.idx. The Infochem models software can recognise these files if they are present in the databank directory specified in the call to AYFLFL as described in section .

Infocond

Infocond is the Infochem condensed component data bank. It is only used by the chemical equilibrium module when considering pure condensed phases. It is not used for phase equilibrium calculations.

Data bank structure

The data bank is supplied in its standard form as two files INFOCOND.BIN and INFOCOND.IDX which contain the physical property data and names index respectively. The file organisation and formats are the same as for Infodata. To create data files in the Infocond structure, an ASCII file needs to be created first which contains sequential data for a number of compounds.

The ASCII data file must include the following elements:

- File header.
- Component name information.
- Constant properties. MOLECULARWEIGHT, HFORMATION, SSTANDARD, TBOIL, TMELT.
- T-dependent properties. CPSOLID.

Other properties may be present but are not used by Multiflash.

Binary interaction parameter databanks

BIP databanks in the INFOBIPS binary format can be created with the MKBIPBANK utility program. This program processes an ASCII input file that contains the BIP information and produces a binary format file that can be opened by AYPDBO.

Input specification

The input file is an ASCII text file.
Each line may be up to 256 characters long.
Any text on a line following a comment character is ignored.
Comment characters are # or !.
Blank lines are allowed.
A data line must contain a complete entry - it cannot be split over more than one line.

BIP data line

The format for a BIP data line is:
NAME1 NAME2 model_id model_variant T_function BIPcoeffs

NAME1 name of first component (string, up to 72 characters)

5

NAME2 name of second component (string, up to 72 characters)
model_id identifier for model (string) - see below
model_variant identifier for model variant (string) - see below
T_function Code for temperature function used to evaluate the BIP values from the stored coefficients (integer) - see below
BIPcoeffs coefficients in T-dependent correlation, 3 values per BIP (real).
The number of BIPs is model-dependent - see below.

Component Names

Names must be unique and may not contain embedded blanks. Names are translated to CARNs before the BIPs are stored. It must be possible to determine the CARN for each name. The NAME is looked for in the Infodata and/or DIPPR databanks and a CARN is returned. The location of the databanks is given to the program by the user when running the program. Note that names on Infochem supplied databanks are stored without embedded blanks so a name like 'CARBONDIOXIDE' is equivalent to 'CARBON DIOXIDE'.

Model/variant identifiers

The following character strings corresponding to each model identifier in ayjmid.inc must be used:

ayjmid.inc character number
model id string variants of BIPs units
JMPR PR * 1 -
JMRKS RKS * 1 -
JMLKP LKP LKP (default) 1 -
JMLKP LKP LK 1 -
JMBWRS BWRS * 1 -
JMSAFT SAFT * 1 -
JMASSO ASSOC * 2 (See below)
JMREG REGULAR REGULAR (default) 1 -
FH 1 J/mol
JMWIL WILSON A (default) 2 J/mol
E 2 -, -
JMNRTL NRTL * 3 J/mol, J/mol, -
JMUNQ UNIQUAC * 2 J/mol, J/mol
JMOWN OWN *
JMPRA PRA VDW (default) 1 -
NRTL 3 -, -, -
JMRKS RKSA VDW (default) 1 -
NRTL 3 -, -, -
JMCOS COSTALD * 1 -
JMCSM CSM * 1 -
JMCSMA CSMA VDW 2 -, -
JMCSMA CSMA GERG 3 -, -, -
JMGERG GERG GERG 3 -, -, -
JMPDV PDVISC * 1 -
JMASP ASPHALTENE * 1 -
JMLGST LGST * 1 -

Note: the * model variant is used for those models that have no variants (as far as BIPs are concerned). If a * is entered for a model with a variant, the default is taken, i.e. WILSON * is equivalent to WILSON A, etc.

The CPA model ASSOC has two sets of BIPs. The fist is dimensionless, but the second has a dimensionless first temperature coefficient, a second coefficient in J/mol and the third coefficient is again dimensionless.

BIP T-dependent correlation functions

Currently the only function available has the id. code of 1 corresponding to the function

$$BIP_i = a_{i1} + T\,a_{i2} + T^2\,a_{i3} \qquad (3)$$

where T is the temperature in K. The coefficients a_{i1} , a_{i2} , a_{i3} are the values entered in the file for the ith BIP.

Entering BIP correlation coefficients

Up to 9 values may be entered corresponding to the correlation coefficients for 3 BIPs. If 9 values are provided they are stored as entered and are interpreted as: a_{11} , a_{12} , a_{13} a_{21} , a_{22} , a_{23} , a_{31} , a_{32} , a_{33} . If fewer than 9 values are entered the program attempts to assign them as follows.

Models with 1 BIP.
1 - 3 values: a_{11} , a_{12} , a_{13} coefficients

Models with 2 BIPs.
2 values: a_{11} , a_{21} (ie. assumed constant BIPs)
4 values: a_{11} , a_{12} , a_{21} , a_{22} (ie. assumed linearly T-dependent)
6 values: a_{11} , a_{12} , a_{13} , a_{21} , a_{22} , a_{23} (ie. quadratic T-dependence)

Models with 3 BIPs.
3 values: a_{11} , a_{21} , a_{31} (ie. assumed constant BIPs)
6 values: a_{11} , a_{12} , a_{21} , a_{22} , a_{31} , a_{32} (ie. assumed linearly T-dependent)

The units of BIPs are model-dependent (see above). For cubic eos the BIP are usually dimensionless. For the activity models the Infochem implementations use BIPs in energy units (J/mol).

Examples

1. Comment line
value fitted to bubble point data 100-120K

2. BIP data lines

```
METHANE ETHANE PRA VDW 1 .011  ! trailing comment
ACETONE WATER NRTL * 1 1514 4342 .2908 # values from Dechema I/1 p248
```

Identification codes for pure component properties

The following i.d. codes are defined as PARAMETER values in the file ayjpid.inc

C.1 Numerical values

For use in calls to AYPCNR and AYPSTR:

JACNTR acentric factor

JCPLIQ liquid enthalpy flag for activity models

JCPMLT solid/liquid difference in C_p at melting point (J/mol K)

JCNUMB carbon number (for petroleum fractions)

JDIPMT dipole moment (debye)

JPOLARZ Polarizability

JQUAD Quadrupole moment

JGFORM Gibbs energy change on formation in perfect gas state at 298.15K and 1 atm (J/mol)

JHCOMB standard enthalpy change on combustion (net) (J/mol)

JHFORM standard enthalpy change on formation in perfect gas state at 298.15K (J/mol)

JHMELT enthalpy change on fusion at melting point (J/mol)

JHOCAS Hayden-O'Connell self-association parameter

JLBCVC critical volume for LBC model (m^3)

JLEX60 thermal expansivity of liquid at 1 atm and 60 F (K^{-1})

JLFLAM lower flammability limit (vol. % in air)

JLMV25 liquid molar volume at 25 C (m^3/mol)

JLSG60 specific gravity of liquid (relative to water) at 60 F, 1 atm

JLBCVC Critical volume paramter used in LBC model m^3/mol

JLVSRF liquid viscosity at reference temperature and 1 atm (Pa s)

JLVSRT liquid viscosity reference temperature (K)

JLVSST The reference viscosity for STRAPP model

JLVSPD The reference viscosity for Pedersen model

JLVSTW The reference viscosity for TWU model

JLVSLB The reference viscosity for LBC model

JMCPR1 Mathias-Copeman vapour pressure parameters for PR EOS

JMCPR2

JMCPR3

5

JMCRK1 Mathias-Copeman vapour pressure parameters for RKS EOS

JMCRK2

JMCRK3

JMOLWT molecular weight (g/mol)

JOMARKS Ω_a scaling factor for specific for RKS EOS

JOMBRKS Ω_b scaling factor for specific for RKS EOS

JOMAPR Ω_a scaling factor for specific for PR EOS

JOMBPR Ω_b scaling factor for specific for PR EOS

JPARAC Parachor ($\left(dynecm^{-1}\right)^{1/4} cm^3/mol$)

JPCRIT critical pressure (Pa)

JPHTYP component type identifier (from ayjxid.inc)

JPTRPL triple point pressure (Pa)

JREFNO reference number for component (databank specific)

JRFINX refractive index at 25 C

JRGYRT radius of gyration (m)

JSFORM Entropy change on formation in perfect gas state at 298.15K and 10^5 Pa (J/mol K)

JSMELT entropy change on fusion at melting point (J/Kmol)

JSOLPR solubility parameter (for liquid) at 25 C ($\left(J/m^3\right)^{1/2}$)

JSSLPR solubility parameter (for solid) at 25 C ($\left(J/m^3\right)^{1/2}$)

JSSTND standard entropy in perfect gas state at 298.15K and 10^5 Pa (J/mol K)

JTAUIG autoignition temperature (K)

JTBOIL normal boiling point temperature (K)

JTCRIT critical temperature (K)

JTFLSH flash point temperature (K)

JTHLWT Henry's law characteristic T for solubility in water (K)

JTMELT normal melting point temperature (K)

JTTRPL triple point temperature (K)

JUFLAM upper flammability limit (vol. % in air)

JUNIF UNIFAC group structure

JUNIQQ UNIQUAC q parameter

JUNIQR UNIQUAC r parameter

JVCRIT critical volume (m^3/mol)

JVHLWT Henry's law characteristic v for solubility in water (m^3/mol)

JVMELT solid/liquid difference in volume at melting point (m^3/mol)

JVSPR1 Peneloux volume shift parameters for PR EOS (m^3/mol)

JVSPR2 ($m^3/molK$)

JVSPR3 (m^3K/mol)

JVSRK1 Peneloux volume shift parameters for RKS EOS (m^3/mol)

JVSRK2 ($m^3/molK$)

JVSRK3 (m^3K/mol)

JZCRIT critical compressibility factor

JCPLIQ Set to use saturated liquid heat capacity as enthalpy reference.

Character values

For use in calls to AYPCNK and AYPSTK:

JKCARN Chemical Abstracts Registry Number in the form xxxxxx-yy-z

JKFORM chemical formula

JKNAME compound name

JKFCOD DIPPR chemical family code

Temperature-dependent property correlations

For use in calls to AYPSTR and AYPCOR:

JDIEL relative permittivity/dielectric constant

JVCHS perfect gas heat capacity

JDENS saturated liquid density

JLATH saturated liquid enthalpy change on evaporation (latent heat)

JLCHS saturated liquid heat capacity

JSURT saturated liquid surface tension

JLTHC saturated liquid thermal conductivity

JPSAT saturated liquid vapour pressure

JLVIS saturated liquid viscosity

JSVR second virial coefficient

JSDN solid density

JSCP solid heat capacity

JVTHC zero-density gas thermal conductivity

JVVIS zero-density gas viscosity

The J-values listed above are used to access the property correlation coefficients not the properties themselves. To evaluate a property at a given temperature it is necessary to call one of the routines listed in section .

Pure component properties

Constant properties

The following constant properties and information can be stored in the Multiflash data structure for each compound during execution of the program. Many of these quantities are not stored on any databank but may be calculated when a component is loaded or as a result of matching experimental data. Note that some of the property values may be missing for some compounds depending on the databank(s) in use.

Chemical Abstracts Registry number (CAR number)

Name of compound

Chemical formula

DIPPR chemical family code

Component type identifier (from ayjxid.inc)

Compound data bank reference number

Molecular weight

Critical temperature (K)

Critical pressure (Pa)

Critical molar volume ($m^3 mol^{-1}$)

Critical compressibility factor

Acentric factor

Normal boiling point (K)

Triple point temperature (K)

Triple point pressure (Pa)

Enthalpy change on formation in the perfect gas state at 298.15K ($J mol^{-1}$)

Entropy change on formation in the perfect gas state at 298.15K and 10^5 Pa ($J mol^{-1} K^{-1}$)

Gibbs energy change on formation in the perfect gas state at 298.15K and 10^5 Pa ($J mol^{-1}$)

Standard entropy in the perfect gas state at 298.15K and 10^5 Pa ($J mol^{-1} K^{-1}$)

Normal melting point temperature (K)

Enthalpy change on fusion at the normal melting point ($J mol^{-1}$)

Entropy change on fusion at the normal melting point ($J mol^{-1} K l^{-1}$)

Difference in C_p on fusion at the normal melting point (in $J mol^{-1}$)

Difference in molar volume on fusion at the normal melting point ($m^3 mol^{-1}$)

Uniquac r parameter

5

Uniquac q parameter

Characteristic temperature for Henry's Law correlation for solubility of a component in water (K)

Characteristic volume for Henry's Law correlation for solubility of a component in water ($m^3 mol^{-1}$)

Dipole moment (Debye units)

Parachor ($\left(dyne cm^{-1}\right)^{1/4} cm^3/mol$)

Radius of gyration (m)

Hayden-O'Connell self-association parameter

Standard net enthalpy change on combustion at 298.15K and 10^5 Pa ($J mol^{-1}$)

Liquid molar volume at 298.15K ($m^3 mol^{-1}$)

Solubility parameter at 298.15K ($\left(J m^{-3}\right)^{1/2}$)

Refractive index at 298.15K

Flash point temperature (K)

Autoignition temperature (K)

Lower flammability limit (volume % in air)

Upper flammability limit (volume % in air)

Solubility parameter (for solid) at 298.15K ($\left(J m^{-3}\right)^{1/2}$)

Specific gravity of liquid at 60 F and 1 atm relative to water

Thermal expansivity of liquid at 60 F and 1 atm (K^{-1})

Ω_a and Ω_b scaling parameters for cubic EOS

carbon number (for petroleum fractions)

Viscosity of liquid at reference temperature and 1 atm (Pas)

Reference temperature for viscosity of liquid (K)

Unifac group structure Type of reference to use for enthalpy in excess energy models

Temperature dependent properties

Multiflash can store coefficients for the following temperature dependent properties. Note that property correlations may be missing for some compounds depending on the databank(s) in use.

Perfect gas heat capacity C_p ($J mol^{-1} K^{-1}$)

Zero-density gas viscosity (Pa s)

Zero-density gas thermal conductivity ($W m^{-1} K^{-1}$)

Second virial coefficient ($m^3 mol^{-1}$)

Saturated liquid vapour pressure (Pa)

Enthalpy change on evaporation for saturated liquid (latent heat) ($J mol^{-1}$)

Saturated liquid density ($mol m^{-3}$)

Saturated liquid viscosity (Pa s)

Saturated liquid thermal conductivity ($W m^{-1} K^{-1}$)

5

Surface tension of saturated liquid (Nm^{-1})

Saturated liquid heat capacity C_p ($Jmol^{-1}K^{-1}$)

Relative permittivity (dielectric constant)

Solid density ($molm^{-3}$)

Solid heat capacity C_p ($Jmol^{-1}K^{-1}$)

Several alternative correlations can be used for each property. The actual equation used depends on the databank and component. Different equations may be used for the same property for different components in a given databank. If the user wishes to override databank values or to define a component without using a databank, any of the equation forms may be used. Details of the correlating equations currently implemented are set out in the Models Manuals that is distributed with below. T denotes the temperature in K. For each temperature correlation a minimum and maximum temperature T_{min} and T_{max} is stored. In most cases, extrapolating equations are given for the cases $T<T_{min}$ and $T>T_{max}$. These extrapolations have two coefficients b_1 and b_2 which are found by fitting to the value of the main correlation and its temperature derivative at either T_{min} or T_{max} . The b coefficients are calculated internally by Multiflash and are not stored in the databanks.

Perfect gas molar heat capacity/ enthalpy/ entropy

Coefficients for correlating the pg heat capacity C_p are stored. H and S are calculated by integrating C_p appropriately with the constants of integration stored as a_1 and a_2 respectively. On loading the coefficients for any correlation from a data bank, Multiflash automatically calculates and stores values for a_1 and a_2 for a datum of 298.15K and 10^5 Pa, i.e. the enthalpy and entropy returned by the correlation are zero at these conditions.

Infochem equation 1: Harmens' equation Harmens [3] proposed the following equation

$$C_p/R=a_3+(a_4-a_3)y^2(1+(y-1)(a_6+a_7y+a_8y^2+a_9y^3)) \quad (4)$$

where R is the gas constant and:

$$y=\frac{T}{T+a_5} \quad (5)$$

Harmens' equation extrapolates to the limits a_3 and a_4 at zero and infinite temperatures and therefore if these coefficients are chosen correctly the equation has an effectively unlimited range. For a linear molecule a_3 should be set to 3.5 and a_4 should be

3A.Harmens, Proceedings of the NPL Conference: Chemical Thermodynamic Data on Fluids and Fluid Mixtures, NPL 11-12 September 1978, IPC Science and Technology Press, 1979.

set to $3N-1.5$ where N is the number of atoms in the molecule. For a non-linear molecule a_3 should be set to 4 and a_4 should be set to $3N-2$. The values of T_{min} and T_{max} are only used to return warnings, no extrapolation is applied.

Infochem equation 2: DIPPR equation 107

$$C_p = a_3 + a_4\left(\frac{a_5/T}{s(a_5/T)}\right)^2 + a_6\left(\frac{a_7/T}{c(a_7/T)}\right)^2 \quad (6)$$

$$T > T_{max}: C_p = C_p(T_{max}) \quad (7)$$

$$T < T_{min}: C_p = C_p(T_{min}) \quad 8)$$

The properties of this equation are similar to Harmens' equation, it can be made to match the physical limits at zero and infinite temperatures.

Infochem equation 3: IUPAC Nitrogen eos form

$$C_p = a_3/T^3 + a_4/T^2 + a_5/T + a_6 + a_7T + a_8T^2 + a_9/T^3 + a_{10}\frac{U^2 e(U)}{(e(U)-1)^2} \quad (9)$$

$$U = a_{11}/T \quad (10)$$

$$T > T_{max}: C_p = C_p(T_{max}) \quad (11)$$

$$T < T_{min}: C_p = C_p(T_{min}) \quad (12)$$

The properties of this equation are similar to Harmens' equation, it can be made to match the physical limits at zero and infinite temperatures.

Infochem equation 4: Wagner Oxygen eos form

$$C_p = a_3/T^{1.5} + a_4 + a_5T^2 + a_6\frac{U^2 e(U)}{(e(U)-1)^2} + a_7\frac{(2/3)\eta^2 e(-\eta)}{(1+(2/3)e(-\eta))^2} \quad (13)$$

$$U = a_8/T \quad (14)$$

$$\eta = a_9/T \quad (15)$$

$$T > T_{max}: C_p = C_p(T_{max}) \quad (16)$$

$$T < T_{min}: C_p = C_p(T_{min}) \quad (17)$$

The properties of this equation are similar to Harmens' equation, it can be made to match the physical limits at zero and infinite temperatures.

Infochem equation 5: DIPPR equation 100

$$C_p = a_3 + a_4 T + a_5 T^2 + a_6 T^3 + a_7 T^4 \quad (18)$$

$$T > T_{max} : C_p = C_p(T_{max}) \quad (19)$$

$$T < T_{min} : C_p = C_p(T_{min}) \quad (20)$$

This equation does not extrapolate to physical limits at zero and infinite temperature.

Gas viscosity at zero density

Infochem equation 1: DIPPR equation 102

$$\eta = \frac{a_1 T^{a_2}}{1 + a_3 / T + a_4 / T^2} \quad (21)$$

$$T > T_{max} : \eta = b_1 / \sqrt{T} + b_2 \sqrt{T} \quad (22)$$

$$T < T_{min} : \eta = \frac{b_1 \sqrt{T}}{b_2 + T} \quad (23)$$

Infochem equation 2: Reichenberg equation:

$$\eta = \frac{a_1 T_r}{\left(1 + a_2 T_r^{a_3} (T_r - 1)\right)^{1/6}} \quad (24)$$

where:

$$T_r = T / T_c \quad (25)$$

$$T > 0.99 T_c : \eta = b_1 / \sqrt{T} + b_2 \sqrt{T} \quad (26)$$

$$T < T_{min} : \eta = \frac{b_1 \sqrt{T}}{b_2 + T} \quad (27)$$

Infochem equation 3: Chapman-Enskog equation

$$\eta = \frac{26.69 \times 10^{-7} (MT)^{1/2}}{a_{1^2} \Omega^{2,2}(T^*, a_3)} \quad (28)$$

where M is the molecular weight and $\Omega^{2,2}$ is a dimensionless collision cross-section calculated from the correlation of Neufeld, Janzen and Aziz [4]:

$$\Omega^{2,2} = A T^{*-B} + C e(-D T^*) + E e(-F T^*) + 0.2 a_{3^2} / T^* \quad (29)$$

4P.D.Neufeld, A.R.Janzen and R.A.Aziz, J. Chem. Phys., 57, 1100-1102, 1972.

and $T^{*}=T/a_2$, $A=1.16145$, $B=0.14874$, $C=0.52487$, $D=0.77320$, $E=2.16178$ and $F=2.43787$. See Reid, Prausnitz and Poling [5] , p.393.

$$T>T_{max}: \eta=b_1/\sqrt{T}+b_2\sqrt{T} \tag{30}$$

$$T<T_{min}: \eta=\frac{b_1\sqrt{T}}{b_2+T} \tag{31}$$

Infochem equation 5: DIPPR equation 100

$$\eta=a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \tag{32}$$

$$T>T_{max}: \eta=b_1/\sqrt{T}+b_2\sqrt{T} \tag{33}$$

$$T<T_{min}: \eta=\frac{b_1\sqrt{T}}{b_2+T} \tag{34}$$

Gas thermal conductivity at zero density

Infochem equation 1: DIPPR equation 102

$$\lambda=\frac{a_1T^{a_2}}{1+a_3/T+a_4/T^2} \tag{35}$$

$$T>T_{max}: \lambda=b_1/\sqrt{T}+b_2\sqrt{T} \tag{36}$$

$$T<T_{min}: \lambda=\frac{b_1\sqrt{T}}{b_2+T} \tag{37}$$

Infochem equation 2: reduced correlation

$$\lambda=\frac{\sqrt{T_r}}{a_1+a_2T_r+a_3T_{r^2}+a_4T_{r^3}} \tag{38}$$

where:

$$T_r=T/T_c \tag{39}$$

$$T_H=min(0.99T_c, T_{max}) \tag{40}$$

$$T>T_H: \lambda=b_1/\sqrt{T}+b_2\sqrt{T} \tag{41}$$

$$T<T_{min}: \lambda=\frac{b_1\sqrt{T}}{b_2+T} \tag{42}$$

5R. C. Reid, J. M. Prausnitz and B. E. Poling, 'The Properties of Gases and Liquids', 4th. ed., McGraw-Hill, New York, (1987)

Infochem equation 3: Monatomic ideal gas

$$\lambda = \eta_0 C_v F(T) / M \tag{43}$$

where η_0 is the zero density viscosity, F is the Eucken factor and M is the molecular weight. For monatomic gases $C_v = 1.5R$ and $F = 2500$, hence:

$$\lambda = 3750 \eta_0 R / M \tag{44}$$

The values of T_{min} and T_{max} are only used to return warnings, no extrapolation is applied.

Infochem equation 5: DIPPR equation 100

$$\lambda = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{45}$$

$$T > T_{max}: \lambda = b_1 / \sqrt{T} + b_2 \sqrt{T} \tag{46}$$

$$T < T_{min}: \lambda = \frac{b_1 \sqrt{T}}{b_2 + T} \tag{47}$$

Second virial coefficient

Infochem equation 1: reduced form of DIPPR equation 104

$$B = a_1 + a_2 \tau + a_3 \tau^3 + a_4 \tau^8 + a_5 \tau^9 \tag{48}$$

where:

$$\tau = T_c / T \tag{49}$$

$$T > T_{max}: B = B(T) \tag{50}$$

$$T < T_{min}: B = b_1 + (T - T_{min}) b_2 \tag{51}$$

Infochem equation 5: DIPPR equation 100

$$B = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{52}$$

$$T > T_{max}: B = b_1 \tag{53}$$

$$T < T_{min}: B = b_1 + b_2 (T - T_{min}) \tag{54}$$

Saturated liquid vapour pressure

Infochem equation 1: Simplified Wagner equation:

$$p = p_c + (a_1 \tau + a_2 \tau^2 + a_3 \tau^3) / T_r \tag{55}$$

where:

$$T_r = T / T_c \tag{56}$$

$$\tau = 1 - T_r \tag{57}$$

$$T > T_c : p = p_c + (a_1 \tau) / T_r \tag{58}$$

For a description of the original Wagner equation see Ambrose [6]

Infochem equation 2: Extended Antoine equation (DIPPR equation 101/115)

$$p = a_1 + \frac{a_2}{T + a_3} + a_4 T^{a_6} + a_5 T + a_7 / T^2 \tag{59}$$

$$T > T_{max} : p = b_1 + b_2 / T \tag{60}$$

$$T < T_{min} : p = b_1 + b_2 / T \tag{61}$$

Infochem equation 3: Wagner equation

$$p = p_c + \left(a_1 \tau + a_2 \frac{\tau^3}{2} + a_3 \tau^3 + a_4 \tau^6 \right) / T_r \tag{62}$$

where:

$$T_r = T / T_c \tag{63}$$

$$\tau = 1 - T_r \tag{64}$$

$$T > Tc : p = p_c + (a_1 \tau) / T_r \tag{65}$$

$$T < T_{min} : p = p_c + (b_1 + b_2 / T_r) \tag{66}$$

Infochem equation 4: Wagner equation (alternative form)

$$p = p_c + \left(a_1 \tau + a_2 \frac{\tau^3}{2} + a_3 \frac{\tau^5}{2} + a_4 \tau^5 \right) / T_r \tag{67}$$

where:

$$T_r = T / T_c \tag{68}$$

$$\tau = 1 - T_r \tag{69}$$

$$T > T_c : p = p_c + (a_1 \tau) / T_r \tag{70}$$

$$T < T_{min} : p = p_c + (b_1 + b_2 / T_r) \tag{71}$$

Infochem equation 5: DIPPR equation 100

$$p = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{72}$$

$$T > T_{max} : p = b_1 + b_2 / T \tag{73}$$

6Ambrose, D. and Ghiassee, N. B., J. Chem. Thermodynamics, 19, 903 (1987)

$$T<T_{min}: p=b_1+b_2/T \tag{74}$$

Infochem equation 7: IAPWS equation This equation form is used by IAPWS to correlate the vapour pressure of water.

$$p=p_c+\left(a_1\tau+a_2\frac{\tau^3}{2}+a_3\tau^3+a_4\frac{\tau^7}{2}+a_5\tau^4+a_6\frac{\tau^{15}}{2}\right)/T_r \tag{75}$$

where:

$$T_r=T/T_c \tag{76}$$

$$\tau=1-T_r \tag{77}$$

$$T>T_c: p=p_c+\left(a_1\tau\right)/T_r \tag{78}$$

$$T<T_{min}: p=p_c+\left(b_1+b_2/T_r\right) \tag{79}$$

Enthalpy change on evaporation for saturated liquid (latent heat)

Infochem equation 1: DIPPR equation 106/Watson equation

$$H_{vap}=a_1\tau^Y \tag{80}$$

where:

$$Y=a_2+a_3T_r+a_4T_{r^2}+a_5T_{r^3} \tag{81}$$

$$T_r=T/T_c \tag{82}$$

and:

$$\tau=1-T_r \tag{83}$$

$$T>0.999T_c: H_{vap}=\frac{b_1}{b_2+T} \tag{84}$$

Infochem equation 2: Wagner-type equation

$$\frac{H_{vap}}{RT_c}=a_1\frac{\tau^1}{3}+a_2\frac{\tau^2}{3}+a_3\tau+a_4\frac{\tau^4}{3}+a_5\frac{\tau^5}{3}+a_6\tau^2+a_7 \tag{85}$$

where:

$$\tau=1-T/T_c \tag{86}$$

$$T_H=min\left(0.999T_c,T_{max}\right) \tag{87}$$

$$T>T_H: H_{vap}=\frac{b_1}{b_2+T} \tag{88}$$

$$T<T_{min}: H_{vap}=b_1+b_2\tau^{0.3775} \tag{89}$$

Infochem equation 5: DIPPR equation 100

$$H_{vap}=a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \quad (90)$$

$$T>T_{max}: H_{vap}=\frac{b_1}{b_2+T} \quad (91)$$

$$T<T_{min}: H_{vap}=b_1+b_2\tau^{0.3775} \quad (92)$$

Saturated liquid density

Infochem equation 1

$$\rho=a_1+a_2\tau^{a_3} \quad (93)$$

where:

$$\tau=1-T/T_c \quad (94)$$

$$T_H=min(0.999T_c, T_{max}) \quad (95)$$

$$T>T_H: \rho=0.9a_1\frac{b_1+T}{b_2+T} \quad (96)$$

Infochem equation 2: Modified Hankinson and Thompson equation (DIPPR equation 116)

$$\rho'=1/V_{c'}+a_1\tau^{0.35}+a_2\frac{\tau^2}{3}+a_3\tau+a_4\frac{\tau^4}{3} \quad (97)$$

where: ρ' is the mass density, $V_{c'}$ is the critical volume per unit mass and

$$\tau=1-T/T_c \quad (98)$$

$$T_H=min(0.99T_c, T_{max}) \quad (99)$$

$$T>T_H: \rho=\frac{b_1+T}{V_c(b_2+T)} \quad (100)$$

Infochem equation 3: Extended Rackett equation

$$1/\rho=a_1(a_2+a_3\tau)^Y \quad (101)$$

where:

$$Y=1+\frac{\tau^2}{7} \quad (102)$$

and:

$$\tau=1-T/T_c \quad (103)$$

$$T_H=min(0.99T_c, T_{max}) \quad (104)$$

$$T>T_H: \frac{1}{\rho}=a_1\frac{b_1+T}{b_2+T} \tag{105}$$

Infochem equation 4: DIPPR equation 105

$$\frac{1}{\rho}=\frac{a_{2^Y}}{a_1} \tag{106}$$

where:

$$Y=1+(1-T/a_3)^{a_4} \tag{107}$$

$$T_H=min(0.99a_3,T_{max}) \tag{108}$$

$$T>T_H: \frac{1}{\rho}=\frac{b_1+T}{a_1(b_2+T)} \tag{109}$$

Infochem equation 5: DIPPR equation 100

$$\rho=a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \tag{110}$$

$$T_H=min(0.99T_c,T_{max}) \tag{111}$$

$$T>T_H: \frac{1}{\rho}=\frac{b_1+T}{\rho_c(b_2+T)} \tag{112}$$

where:

$$\rho_c=\rho(T_c) \tag{113}$$

$$T<T_{min}: \rho=b_1+b_2\frac{\tau^2}{7} \tag{114}$$

where:

$$\tau=1-T/T_c \tag{115}$$

Infochem equation 7: IAPWS equation This equation form is used by IAPWS to correlate the density of saturated liquid water.

$$\frac{\rho}{\rho_c}=1+a_1\frac{\tau^1}{3}+a_2\frac{\tau^2}{3}+a_3\frac{\tau^5}{3}+a_4\frac{\tau^{16}}{3}+a_5\frac{\tau^{43}}{3}+a_6\frac{\tau^{110}}{3} \tag{116}$$

where: ρ_c is the critical density and

$$\tau=1-T/T_c \tag{117}$$

$$T>T_H: \text{extrapolation} \tag{118}$$

$$T<T_{min}: \text{extrapolation} \tag{119}$$

Saturated liquid viscosity

Infochem equation 1: Reid, Prausnitz and Poling

This is the Reid, Prausnitz and Poling(pp.441-455) equation 1:

$$\eta = a_1 T^{a_2} \tag{120}$$

The values of T_{min} and T_{max} are only used to return warnings, no extrapolation is applied.

Infochem equation 2: DIPPR equation 101

$$\eta = a_1 + a_2 / T + a_3 T + a_4 T^{a_5} \tag{121}$$

$$T > T_{max} : \eta = b_1 + b_2 T \tag{122}$$

$$T < T_{min} : \eta = b_1 + b_2 T \tag{123}$$

Infochem equation 3: reduced correlation

$$\frac{\eta}{a_5} = a_1 X^{1/3} + a_2 X^{4/3} \tag{124}$$

where:

$$X = \frac{a_3 - a_4}{T - a_4} - 1 \tag{125}$$

$$T_H = min(0.99 a_3, T_{max}) \tag{126}$$

$$T > T_H : \eta = \frac{b_1}{b_2 + T} \tag{127}$$

$$T_L = max(1.01 a_4, T_{min}) \tag{128}$$

$$T < T_L : \eta = b_1 + b_2 l(T) \tag{129}$$

Infochem equation 4: Reid, Prausnitz and Poling

This is a combination of Reid, Prausnitz and Polingequations 2/3:

$$\eta = a_1 + a_2 / T + a_3 T + a_4 T^2 \tag{130}$$

$$T > T_{max} : \eta = b_1 + b_2 T \tag{131}$$

$$T < T_{min} : \eta = b_1 + b_2 T \tag{132}$$

Infochem equation 5: DIPPR equation 100

$$\eta = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{133}$$

$$T > T_{max} : \eta = b_1 + b_2 T \tag{134}$$

$$T < T_{min} : \eta = b_1 + b_2 T \tag{135}$$

Saturated liquid thermal conductivity

Infochem equation 1: Jamieson equation

$$\lambda = a_1\left(1 + a_2\frac{\tau^1}{3} + a_3\frac{\tau^2}{3} + a_4\tau\right) \quad (136)$$

where:

$$\tau = 1 - T / T_c \quad (137)$$

$$T_H = min\left(0.99T_c, T_{max}\right) \quad (138)$$

$$T > T_H : \lambda = a_1\frac{b_1 + T}{b_2 + T} \quad (139)$$

$$T < T_{min} : \text{extrapolation} \quad (140)$$

Infochem equation 2: DIPPR equation 101

$$\lambda = a_1 + a_2 / T + a_3 T + a_4 T^{a_5} \quad (141)$$

$$T > T_{max} : \lambda = b_1 + b_2 T \quad (142)$$

$$T < T_{min} : \lambda = b_1 + b_2 T \quad (143)$$

Infochem equation 5: Reid, Prausnitz and Poling

This is the equation defined by Reid et al.(pp.546-548):

$$\lambda = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \quad (144)$$

$$T > T_{max} : \text{extrapolation} \quad (145)$$

$$T < T_{min} : \text{extrapolation} \quad (146)$$

Surface tension of saturated liquid

Infochem equation 1: DIPPR equation 106

$$\sigma = a_1\left(1 - T_r\right)^Y \quad (147)$$

where:

$$Y = a_2 + a_3 T_r + a_4 T_{r^2} + a_5 T_{r^3} \quad (148)$$

and:

$$T_r = T / T_c \quad (149)$$

$$T_H = min\left(0.99T_c, T_{max}\right) \quad (150)$$

$$T > T_H : \sigma = \sigma\left(T_H\right)\frac{1}{1 + b_1 \Delta T} \quad (151)$$

where

$$b_1=\sigma'(T_H)\quad \Delta T=T_H-T \tag{152}$$

$$T<T_{min}:\text{extrapolation} \tag{153}$$

Infochem equation 2: Extended Sprow and Prausnitz equation

$$\sigma=a_1\tau^{a_2}(1+a_3\tau) \tag{154}$$

where:

$$\tau=1-T/T_c \tag{155}$$

and:

$$T_r=T/T_c \tag{156}$$

$$T_H=min(0.99T_c,T_{max}) \tag{157}$$

$$T>T_H:\sigma=\frac{b_1}{b_2+T} \tag{158}$$

$$T<T_{min}:\text{extrapolation} \tag{159}$$

Infochem equation 5: DIPPR equation 100

$$\sigma=a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \tag{160}$$

$$T>T_{max}:\sigma=\frac{b_1}{b_2+T} \tag{161}$$

$$T<T_{min}:\text{extrapolation} \tag{162}$$

D.13 Saturated liquid Cp

Infochem equation 1: DIPPR equation 114

$$C_p=a_1/\tau+a_2+a_3\tau+a_4\tau^2+a_5\tau^3+a_6\tau^4+a_7\tau^5 \tag{163}$$

where:

$$\tau=1-T/T_c \tag{164}$$

$$T_H=min(0.95T_c,T_{max}) \tag{165}$$

$$T>T_H:C_p=C_p(T_{max})\frac{1+2(b_1+b_2T)}{1+(b_1+b_2T)} \tag{166}$$

$$T<T_{min}:C_p=C_p(T_{min})(b_1+b_2T) \tag{167}$$

Infochem equation 5: DIPPR equation 100

$$C_p=a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \tag{168}$$

5

$$T>T_{max}: C_p = C_p(T_{max})\frac{1+2(b_1+b_2T)}{1+(b_1+b_2T)} \quad (169)$$

$$T<T_{min}: C_p = C_p(T_{min})(b_1+b_2T) \quad (170)$$

Dielectric constant of liquid

Infochem equation 1

$$\epsilon = a_1+a_2T+a_3T^2+a_4T^3 \quad (171)$$

$$T>T_{max}: \epsilon = b_1+b_2T \quad (172)$$

$$T<T_{min}: \epsilon = b_1+b_2T \quad (173)$$

Infochem equation 2 Matyott and Smith.

$$\epsilon = exp(a_1+a_2T) \quad (174)$$

The above equation is also used for extrapolation.

Solid density

Infochem equation 5: DIPPR equation 100

$$\rho = a_1+a_2T+a_3T^2+a_4T^3+a_5T^4 \quad (175)$$

$$T>T_{max}: \rho = b_1 \quad (176)$$

$$T<T_{min}: \rho = b_1 \quad (177)$$

Solid heat capacity Cp

Infochem equation 3: Multiple T-ranges The thermal properties are stored as a series of correlations for C_p over a number of temperature ranges. The equation used for C_p in the k th. range is:

$$C_p = a_{1k}+a_{2k}T+a_{3k}/T^2+a_{4k}T^3 \quad (178)$$

In total, 31 coefficients a_0 , a_{11} , a_{21} , a_{31} , a_{41} , a_{51} , a_{61} , a_{12} , a_{22} , a_{32} , a_{42} , a_{52} , a_{62} , a_{13} , a_{23} , a_{33} , a_{43} , a_{53} , a_{63} , a_{14} , a_{24} , a_{34} , a_{44} , a_{54} , a_{64} , a_{15} , a_{25} , a_{35} , a_{45} , a_{55} , a_{65} are used. a_0 is set to the number of temperature ranges used; the maximum permitted value is 5. Coefficient a_{5k} represents the transition temperature at which the k th. temperature range ends and which the $(k+1)$ th. temperature range, if it exists, begins. Coefficient a_{6k} represents the enthalpy of transition between the k th. and, if it exists, the $(k+1)$ th. temperature range.

H is calculated by integrating C_p appropriately. At each transition temperature, the enthalpy of transition is added to the enthalpy at that point and then the next expression for C_{rmp} is integrated, and so on until the desired temperature is reached. S is also found by the appropriate integration. The transition entropies are calculated by dividing the enthalpies of transition by the transition temperatures, since the Gibbs energies of transition must be zero (condition for equilibrium).

C_p is in $Jmol^{-1}K^{-1}$, a_{5k} is in K and a_{6k} is in $Jmol^{-1}$.

Infochem equation 5: DIPPR equation 100

$$C_p = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{179}$$

$$T > T_{max} : C_p = b_1 \tag{180}$$

$$T < T_{min} : C_p = b_1 \tag{181}$$

Nomenclature

a Equation of state parameter
A_{ij} Activity coefficient equation interaction parameter
b Equation of state parameter
C_p Molar heat capacity at constant pressure
C_v Molar heat capacity at constant volume
G^E Excess Gibbs energy
H Enthalpy
H_{vap} Enthalpy of vaporisation
k_{ij} Equation of state interaction parameter
M Molecular weight
n Number of moles
N Total number of moles

p Pressure
q UNIQUAC parameter
r UNIQUAC parameter
R Gas constant
S Entropy
T Temperature
V Volume
α_{ij} NRTL equation α parameter
γ Activity coefficient
η Viscosity
λ Thermal conductivity
Π Poynting correction
ρ Molar density
σ Surface tension
ϕ Fugacity coefficient
ω Acentric factor
$\Omega^{2,2}$ Collision integral for zero density viscosity

Subscripts:

0 Zero-density value
c Critical value
i, j Component numbers
r Reduced value

Examples of use

Pure component properties

Vapour pressure and saturated liquid density for water as a function of temperature. Two versions of the example are provided. The first uses low-level calls to initialise Multiflash and the second uses a Multiflash command file and the command processor interface described in section .

Low-level initialisation

PGExamples/ex1.for

Command file initialisation

The file ex1a.mfl contains commands to set up the same databank as used in the previous example but with a different component. The contents are as follows:

```
puredata infodata;
components nitrogen;
```

The following program processes the command file and evaluates the requested properties.

PGExamples/ex1a.for

Thermodynamic and transport properties of a mixture

Thermodynamic properties and viscosity of a methane-heptane mixture are calculated from the RKS equation of state and the LBC viscosity model. Two versions of the example are provided. The first uses low-level calls to initialise Multiflash and the second uses a Multiflash command file and the command processor interface described in section .

Low-level initialisation

PGExamples/ex2.for

Command file initialisation

The file ex2a.mfl contains commands to set up the same databank, components, models and phase descriptors as used in the previous example. The contents are as follows:

```
puredata infodata;
bipdata oilandgas;
model Meos rks;
model Mvvisc vvs1;
model Mlvisc lvs1;
model Mst mcs Meos;
pd VAPOUR vapour Meos * * Mvvisc;
pd LIQUID liquid Meos * * Mlvisc * Mst;
components METHANE HEPTANE;
```

The following program processes the command file and evaluates mixture properties.

PGExamples/ex2a.for

Simple flash with an equation of state

The API RKS equation of state is used for both phases.

PGExamples/ex3.for

Simple flash with an activity model

The NRTL equation with user-entered BIPs is used for for the liquid phase with the RK equation as gas phase correction. Pure component properties are taken from the Infodata data bank:

PGExamples/ex4.for

Flash with hydrates

The gas hydrate model is loaded for both type I and type II hydrates using the PR equation as the model for pure liquid water. Pure component properties are taken from the Infochem data bank with BIPs all zero:

PGExamples/ex5.for

Flash and phase properties from LKP model

This example shows how some thermodynamic properties not directly returned by Multiflash may be evaluated. Two versions of the example are provided. The first uses low-level calls to initialise Multiflash and the second uses a Multiflash command file and the command processor interface described in section .

Low-level initialisation

PGExamples/ex6.for

Command file initialisation

The file ex6a.mfl contains commands to set up the same databank, components, models and phase descriptors as used in section . The contents are as follows:

```
puredata infodata;
model Meos lkp;
model Mvvisc vvs1;
model Mlvisc lvs1;
model Mst st1;
pd VAPOUR vapour Meos * * Mvvisc;
pd LIQUID liquid Meos * * Mlvisc * Mst;
components BUTANE, CO2;
```

5

The following program processes the command file and carries out a flash calculation and dew point calculation followed by property evaluations.

PGExamples/ex6a.for