

# Embedding equation oriented models of process unit operations in a sequential modular flowsheet simulator

Study with a gas separation membrane model

Thesis  
Master Chemical Engineering  
Delft University of Technology

Wilma Hensen

October 2005



Process Systems Engineering

P.J.T. Verheijen  
C.S. Bildea



Engineering Solutions

J.W. Verwijs

## Abstract

Within the Dow Chemical Company the strategy is to centre process flowsheet design activities around the simulation tool Aspen Plus (from Aspen Tech Inc.). However, the model library of Aspen Plus is limited in type and number of models of process unit operations. Equation oriented modelling tools, such as gPROMS (from Process Systems Enterprise Ltd.) or Aspen Custom Modeler – ACM (from Aspen Tech Inc.), can be used to develop custom models of process unit operations which are not available in the Aspen Plus model library. For consistent flowsheet simulation and optimization it is required that these custom models can be exported to and used within Aspen Plus, just like any other model already available in the Aspen Plus model library.

Recently, interfaces – based on the CAPE-OPEN standards - have been implemented in the latest releases of the above mentioned process simulation tools. To test the status and performance of the software interoperability, as well as to examine the custom model performance in Aspen Plus, a model of a gas separation membrane unit is developed in gPROMS and in ACM and exported for use in Aspen Plus.

This thesis describes the achievements made in the development of the custom membrane model and its interfacing with Aspen Plus. Also, an improved method for model initialisation is presented. Model initialisation is one of the principal obstacles for the development of generic custom models, i.e. models which can run successfully irrespective of the set of components, physical property method, or range of operating conditions.

The main conclusions from this study are that several software features enhance the development of generic and robust custom models in equation oriented modelling tools, such as a hierarchical model structure, the usage of an external physical property package, as well as the incorporation of a model initialisation structure (e.g. like the one proposed in this thesis). Moreover, the current functionality and performance of the interfaces for custom models between gPROMS and ACM on one side – and Aspen Plus on the other side – are not sufficient to be used for industrial practice. They need to be improved significantly by the software vendors.

## Acknowledgements

During the last 8 months, from the beginning of February, I worked on my graduation thesis eventuating in this report. The project was done in close collaboration with the Process Systems Engineering section of Delft University of Technology and the Engineering Solutions department of Dow Benelux B.V. at Terneuzen.

In the first place I would like to thank Jan Willem Verwijs for giving me the opportunity to do my graduation project under his supervision. His experience, especially in process modelling helped me to gain a better insight in this field. Also his advice on practical issues for carrying out this project was very helpful. Furthermore I would like to acknowledge Peter Verheijen for his guidance on the scientific part of my work.

During this project it was a pleasure to work with Elodie Sureau. Together we were able to discover the exciting world of process modelling. I also would like to express my gratitude to the following people for their assistance and collaboration in this project: Philippe Hayot, Sorin Bildea, Deepa Rethinam and Emmanuel Lejeune.

Last but not least, my close colleagues at the Engineering Solutions department and my family are gratefully acknowledged for supporting me during this project.

Wilma Hensen  
Terneuzen, October 2005

# Table of contents

<i>Abstract</i> .....	2
<i>Acknowledgements</i> .....	3
<i>Table of contents</i> .....	4
<i>1 Introduction</i> .....	6
1.1 <i>Background and objectives</i> .....	6
1.2 <i>Scope of this thesis</i> .....	9
<i>2 Theory of gas separation membrane processes</i> .....	12
2.1 <i>Membrane processes</i> .....	12
2.2 <i>Membrane modules</i> .....	13
2.3 <i>Overview of membrane transport mechanisms</i> .....	15
2.4 <i>Literature review of existing membrane unit simulations</i> .....	17
<i>3 Custom model of a gas separation membrane unit</i> .....	20
3.1 <i>Model description</i> .....	20
3.1.1 <i>Modelling objective</i> .....	20
3.1.2 <i>Model structure and assumptions</i> .....	20
3.1.3 <i>Model description</i> .....	22
3.2 <i>Model implementation</i> .....	24
3.2.1 <i>Discretisation method</i> .....	24
3.2.2 <i>Implementation in gPROMS and ACM</i> .....	26
3.3 <i>Solving the custom model</i> .....	28
3.4 <i>Model comparison</i> .....	29
<i>4 Software interoperability: current status</i> .....	33
4.1 <i>Background information</i> .....	33
4.1.1 <i>CAPE-OPEN standards (gPROMS – Aspen Plus)</i> .....	33
4.1.2 <i>Aspen Tech interfaces (ACM – Aspen Plus)</i> .....	35
4.2 <i>Work processes for interfacing</i> .....	36
4.3 <i>Current status</i> .....	37
4.3.1 <i>Physical property interface</i> .....	37
4.3.2 <i>Unit model interface</i> .....	39
4.3.3 <i>Conclusion</i> .....	43
<i>5 Custom model performance in Aspen Plus</i> .....	45
5.1 <i>Custom model performance</i> .....	45
5.2 <i>Application to custom membrane model</i> .....	46
5.2.1 <i>Initialisation structure</i> .....	47
5.2.2 <i>Case study to test robustness initialisation structure</i> .....	49
5.3 <i>Initialisation structure in Aspen Plus</i> .....	52
<i>6 Conclusion and recommendations</i> .....	53

6.1	Conclusion .....	53
6.2	Future research.....	55
Literature list .....		57
Nomenclature.....		59
Appendix A Mathematical model for gas separation membrane unit.....		61
A.1	Equations in main model (for membrane geometry and characteristics) .....	61
A.2	Plug flow model for flow through fibre and shell side .....	63
A.3	Flow through membrane (transport model) .....	65
Appendix B Printout custom membrane model in gPROMS and ACM .....		66
B.1	Model in gPROMS.....	66
B.2	Model in ACM .....	82
Appendix C Specification of parameters and variables for membrane model .....		96
Appendix D Results membrane model for comparison.....		97
Appendix E Work processes interfacing .....		99
E.1	Work process physical property package interfacing Aspen Plus → gPROMS ....	99
E.2	Work process physical property package interfacing Aspen Plus → ACM.....	102
E.3	Work process unit model interfacing gPROMS → Aspen Plus.....	108
E.4	Work process unit model interfacing ACM → Aspen Plus .....	117
Appendix F Automation initialisation structure in equation oriented modelling tools ..		125
Appendix G Definition base case.....		126
Appendix H Results case study initialisation structure.....		127

# 1 Introduction

*This chapter gives a general introduction to the research performed in this thesis. After presenting background information which led to the initialisation of this project, the objectives are described. Next, the scope of this thesis is elaborated by describing the research done.*

## 1.1 Background and objectives

Higher safety standards, more stringent environmental regulations, better management of energy and raw materials, and increased product quality requirements have caused the structure of chemical processes to become increasingly complex and has tightened the process operating limits. To satisfy this complex set of requirements, the designer of a chemical plant needs to have a proper understanding of the physical and chemical phenomena, as well as the behaviour of the process equipment involved. Process simulation (and optimization) has become a powerful tool to support the process design work processes and is now a critical skill for every process design engineer.

Simulation tools for chemical design processes, which are relevant for this study, can be divided into two categories, namely sequential modular (or block oriented) flowsheet simulators and equation oriented process modelling tools.

### *Sequential modular flowsheet simulators*

The most well known simulation tools in this category are Aspen Plus (from Aspen Tech Inc.) and PRO/II (from Simulation Science Inc.). The basic idea behind these simulators is that the user selects from a model library the models which represent the unit operations (a so-called block) in the process at study. The process flowsheet is developed by (graphically) connecting the blocks. The connections represent the process material and/or energy streams. The user also needs to specify the parameters to be used by the block models (e.g. like the number of stages in a distillation column, feed stage location, etc.), as well as the thermodynamic property methods, feed stream data, and operating conditions. The user does not need to provide any mathematical equations to describe the behaviour of the process flowsheet and its unit operations. When the process flowsheet is completed, the user can run the flowsheet model, and mass and energy balance data, as well as some equipment specifications, are returned by the program to the user. This type of process simulation is applied for continuous, steady-state simulations.

### *Equation oriented process modelling tools*

The most well-known equation oriented process modelling tools are gPROMS (from Process Systems Enterprise Ltd.) and Aspen Custom Modeler (from Aspen Tech Inc.). The idea behind this class of tools is to provide a generic process modelling environment in which the user (model developer) can focus on the mathematical equations required to describe the process (unit) at study. The numerical solution of the resulting set of equations is taken care of by the modelling tool and is transparent to the user.

This equation oriented environment can be used for steady-state and dynamic process simulation and optimization, as well as for parameter estimation and experimental design. The use in industry is however limited compared to sequential modular flowsheet

simulations, due to the effort and engineering skills required to develop custom process models. Also, equation oriented process modelling is a much more recent development (last decade) than sequential modular flowsheet simulation, which has been in place now for more than 20 years.

Sequential modular flowsheet simulator environments are widely used by many process engineers, because of its ease of use and robustness to handle large scale process simulation problems (large number of unit operations, process streams and chemical components). However, sequential modular flowsheet simulators have its limitations as well. One of the important limitations is the type and number of models of process unit operations available in the model libraries. Most models in these libraries are models of process unit operations using idealized hydrodynamics (like well-mixed) and thermodynamic equilibrium between the fluid phases present in the process unit. Rate-based models, like multi-phase reactor models, membrane models, crystallizer models, etc. are often lacking.

On the other hand, equation oriented process modelling tools offer the opportunity to develop custom models of process unit operations, without the burden of putting an effort on the numerical solution methods for the model equations. Therefore, it would be a good idea to combine these two simulation environments. Equation oriented process modelling tools can be used excellently to develop models of process unit operations which do not exist or do not exist in the preferred level of detail in the model libraries of sequential modular flowsheet simulators. After development, the custom model can then be used as a user model block inside a sequential modular flowsheet simulator.

Besides the limitation of the type and number of models available in the model libraries of sequential modular flowsheet simulators, several other reasons can be adduced for the usage of custom models of process unit operations in flowsheet simulators.

Embedding of custom models in a sequential modular flowsheet simulator enhances consistent process simulation. The current practice for simulating a unit operation present in a process flowsheet, whose model do not exists in the model library is to use a shortcut method to be able to mimic the mass and energy balances for this unit. In a complete other software package the unit operation is simulated in more detail on basis of the stream information obtained from the sequential modular flowsheet simulator. The simulation of the process flowsheet and the unit operation in different software tools have to run parallel and errors can easily be made. Not to mention the difference in calculation of thermodynamic and physical properties in these tools.

Furthermore, during process development alternatives can quickly and precisely be evaluated and optimized, if unit models with relevant level of detail are available in sequential modular flowsheet simulators. Especially for new technology detailed unit models are required to predict the process performance. (Academic) research groups can often provide detailed models of process unit operations, which are developed in equation oriented modelling tools.

The use of a custom model of process unit operation, developed in equation oriented modelling tools, within a sequential modular flowsheet simulator, requires interfacing between the different software programs. In the early nineties the idea was created by

academic institutions as well as by industry that an open interface for transferring data between process simulation software of various origins would be favourable in order to improve consistent modelling. Several CAPE-OPEN (Computer Aided Process Engineering) projects were initiated to develop standards and prove the possibilities for open interfaces to transfer models of process unit operations, thermodynamic and physical property packages, and even numerical solvers, between the various process simulation tools. This results finally in the establishment of CO-LaN. The CAPE-OPEN Laboratories Network, CO-LaN, is a neutral and academic association supporting and promoting the use of CAPE-OPEN standards [1]. As the major process simulation software suppliers were involved in the CAPE-OPEN projects, implementation of CAPE-OPEN standards in commercial software was straightforward. Presently, CAPE-OPEN compliant versions of process simulation software are released. The first applications of interfacing in process simulation are appearing.

The recent development and implementation of open interfaces for process simulation tools resulted in the main objective of this thesis. The main objective is to:

***Improve the ability to use custom models of process unit operations, developed in equation oriented modelling tools, within a sequential modular flowsheet simulator.***

The focus is on rate-based distributed parameter custom models. The majority of process unit operations not available in the model libraries of sequential modular flowsheet simulators can be described more precisely by mathematical models containing variables which are a function of the spatial position in the unit. As mentioned before, almost all models in the sequential modular flowsheet simulators are thermodynamic equilibrium based models. The combination of distributed parameter and rate-based characteristics is expected to add (numerical) complexity to the custom model and its interfacing.

The existence of open interfaces for software interoperability is not the only criteria for using a custom model within a sequential modular flowsheet simulation. For successful implementation, the custom model also has to be applied just like any other model of a process unit operation available in the libraries of sequential modular flowsheet simulators. As like these models, the custom model needs to be able to handle:

- any list and number of components,
- any physical property method,
- a large range of operation conditions, and
- various equipment or design specifications.

As a consequence, a custom model of a process unit operation developed in equation oriented modelling tools needs to be generic and robust. The interface functionalities should also offer the possibility to future users to call this model from a library.



To refine the main objective several sub-objectives are defined, which are:

- *The development of a generic distributed parameter model of a gas separation membrane unit:*
  - *to examine the current status of software interoperability,*
  - *to investigate the key requirements for custom model performance within a sequential modular flowsheet simulator.*
- *The development of work processes for future users by supporting them in using the open interfaces between process simulation software related to physical property packages and custom models of process unit operations.*

## **1.2 Scope of this thesis**

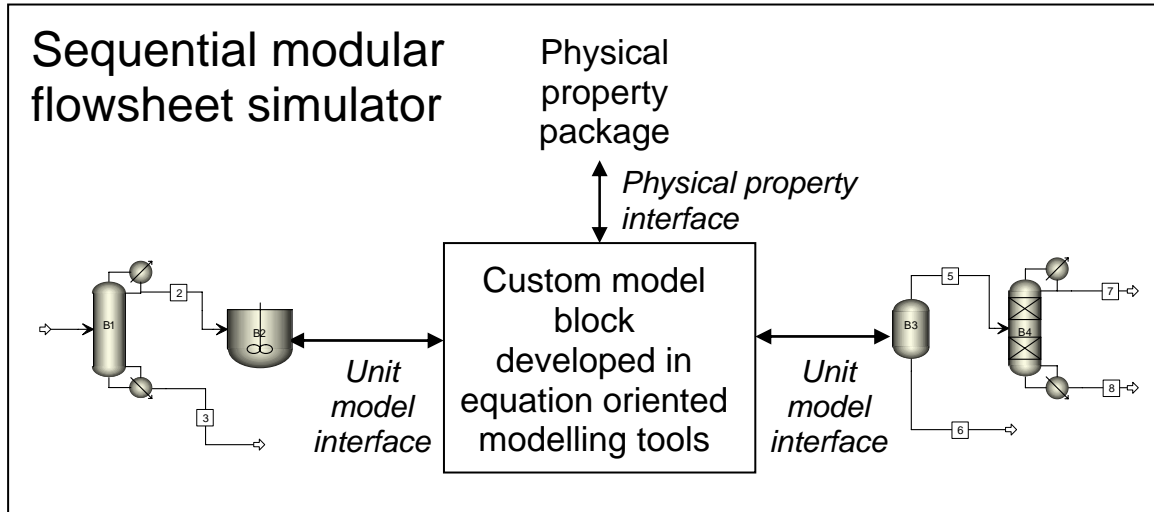
The use of a custom model in a sequential modular flowsheet simulator requires interfacing with equation oriented modelling tools. In equation oriented modelling tools no physical property methods and databanks are standard available for the calculation of thermodynamic and physical properties. Therefore, during the development of a custom model, assumptions should be made in order to calculate the properties or ideally for consistent modelling a physical property package from the sequential modular simulator has to be used. The latter one requires the interface of a physical property package between the sequential modular flowsheet simulator and the equation oriented modelling tool (the physical property interface). Moreover, when the development of the custom model is finished, the model itself should be placed in the sequential modular flowsheet simulator. This requires an interface of the custom model between the equation oriented modelling tool and the sequential modular flowsheet simulator (the unit model interface). A schematic picture of the physical property interface and the unit model interface is shown in Figure 1.1.

The research, described in this thesis is carried out at Dow Benelux B.V.. The Dow Chemical Company has standardized its process simulation environment based on the Aspen Engineering Suite (from Aspen Tech Inc.). This Suite includes the sequential modular simulator Aspen Plus and the equation oriented modelling tool Aspen Custom Modeler (ACM). Also, gPROMS (from Process Systems Enterprise Ltd.) is available within Dow and used for equation oriented modelling.

Hence, this research is concentrated on these three simulation tools: Aspen Plus, ACM and gPROMS. The following versions of the software packages have been used:

- Aspen PLUS : version 12.1  
version 2004.1
- ACM : version 12.1  
version 2004.1
- gPROMS : version 2.3  
version 3.0 (alpha test version)

To interface gPROMS and Aspen Plus, the CAPE-OPEN standards have been utilized. For interfacing ACM and Aspen Plus, default Aspen Tech interfaces have been used.



**Figure 1.1: Custom model block of a process unit operation in a sequential modular flowsheet simulator**

To study the main objective described above, a model of a gas separation membrane unit has been developed in both gPROMS and ACM. A membrane unit has several modelling characteristics (rate-based and distributed parameter), which makes it attractive to use as a case study. Moreover, a model of a membrane module does not (yet) exist in the Aspen Plus model library. Given the continuously increasing cost of oil and gas, the separation / purification of gases at low capital and energy cost is becoming a major issue. The same is valid for the increasing treatment requirements of process vent streams before emitting them into the air. A membrane gas separation unit is a potential technology for these applications. The availability of rigorous process models inside a sequential modular flowsheet simulator will help to study the applicability of this technology in certain cases.

After presenting in Chapter 2 the most important theory of gas separation membrane processes, the development of the custom membrane model is described in detail in Chapter 3. The developed custom model is also compared with experimental data and existing simulations from literature.

Chapter 4 reported the current status of software interoperability obtained from interfacing the developed custom membrane model with Aspen Plus. In this chapter also work processes are developed to guide future users on interfacing:

- a physical property package from Aspen Plus with gPROMS/ACM
- a custom model of a process unit operation, developed in gPROMS/ACM, with Aspen Plus

In Chapter 5 the key requirements for custom model performance in sequential modular flowsheet simulations are further explained. Based on the current functionalities available

for unit model interfacing a new approach for custom model initialisation is proposed. A potential solution on basis of the developed custom membrane model is presented.

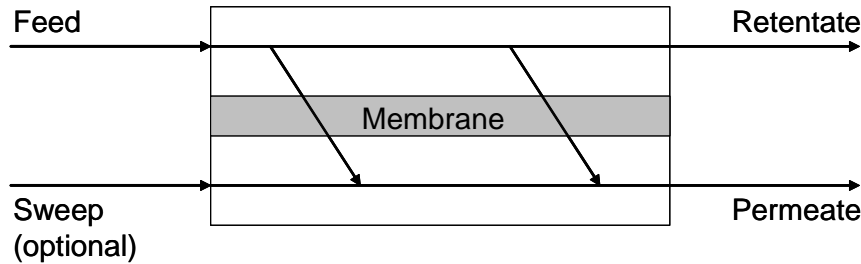
Finally, in Chapter 6 conclusions are given for the research results obtained and possible directions for future work are discussed.

## 2 Theory of gas separation membrane processes

*The aim of this chapter is to familiarise the reader with membranes processes for gas separation. First, a general introduction to membrane processes is given. In the next section an overview is given of commercial available membrane modules used for gas separation. Thereafter different transport mechanisms which describe the flux of components through the membrane are discussed. In the last section a review of existing simulations of gas separation membrane units in literature is presented.*

### 2.1 Membrane processes

There are many membrane processes, based on different separation mechanisms and they can cover the broad size range from particles to molecules. In spite of these various differences, all membrane processes have one thing in common, i.e., the membrane. The membrane is at the heart of every membrane process and can be considered as a perm-selective barrier between two phases. The feed enters on one side of the membrane and is divided into two streams; the retentate and permeate stream, as shown in Figure 2.1. The sweep stream, a liquid or gas, is sometimes used to improve the removal of components from the feed stream.



**Figure 2.1: General membrane process**

The membrane has the ability to transport one component more readily than others. The transport rate of a component through a membrane is determined by its permeability in the membrane and by the driving force across the membrane. The general relation for the transport rate of component  $i$  through a membrane is therefore given by:

$$J_i = Q_i \frac{d(\text{driving\_force})_i}{dz} = \left( \frac{Q_i}{\delta^m} \right) \cdot \Delta(\text{driving\_force})_i$$

$$= \bar{Q}_i \cdot \Delta(\text{driving\_force})_i \quad (2.1)$$

where  $\bar{Q}_i$  is the permeance, which is defined as the ratio of  $Q_i$ , the permeability, to  $\delta^m$ , the membrane thickness [2]. The driving force is a chemical or an electrical potential difference across the membrane [3].

Large scale membrane gas separation processes emerged during the 1980s by the development of synthetic polymeric membranes [4]. Gas separation is the separation of gaseous fluids by using a pressure difference. Usually a sweep gas is not used, but the permeate side of the membrane is maintained at a much lower pressure than the retentate side. Membrane gas separation is an area in which currently considerable research is done, and the number of applications is expanding rapidly. Major current applications of gas separation membranes are the separation of hydrogen from nitrogen, argon and methane in ammonia plants; the production of nitrogen from air; and the separation of carbon dioxide from methane in natural gas operations [4]. Gas permeation must still compete with classical separation operations like distillation at cryogenic conditions, absorption and pressure-swing adsorption.

For process design of membrane gas separation units, there are three factors that determine the performance of the system [5]:

- The pressure ratio between the retentate and permeate side of the membrane. This ratio is dependent on the operation conditions and on the configuration of the retentate and permeate stream around the membrane. Section 2.2 describes the different membrane modules used for gas separation.
- The membrane selectivity, which is the ratio of the permeabilities of the different components in the feed mixture. The membrane selectivity is determined by the choice of the membrane material. In Section 2.3 an overview is given of the various models used to estimate permeabilities from experimental data and to describe the transport of components through the membrane.
- The stage cut, a trade off between recovery and purity. The stage cut is defined by the ratio of the permeate flowrate and the feed flowrate. At low stage cuts a concentrated permeate stream is obtained, but the key component is only modestly removed from the feed stream. At high stage cuts almost complete removal of the key component from the feed stream is obtained. However the permeate stream is only slightly more enriched than the original feed stream, because the concentration of the less permeable components increases. The stage cut can be adapted during the design of a membrane system by varying the membrane area or during its operation by changing the flowrate of the feed stream.

## **2.2 Membrane modules**

For commercial applications the membrane has to be suitably housed in a unit. As many different membrane processes exist, also a large number of membrane modules are available commercially. Gas separation membranes are typically formed into spiral-wound or hollow fibre modules, because of their high packing density (membrane surface area per unit volume) compared to other modules [4].

### ***Spiral wound membrane module***

Spiral wound membrane modules are produced from flat sheets. A laminate, consisting of two membrane sheets separated by spacers for the flow of the feed and permeate, is wound around a central collection tube to form a module that is inserted into a pressure vessel. The feed flows axial through the cylindrical module parallel along the central pipe

whereas the permeate flows radial toward the central collection pipe, see Figure 2.2. A typical spiral-wound module is 0.1 to 0.3 meter in diameter and 3 meter long [2].

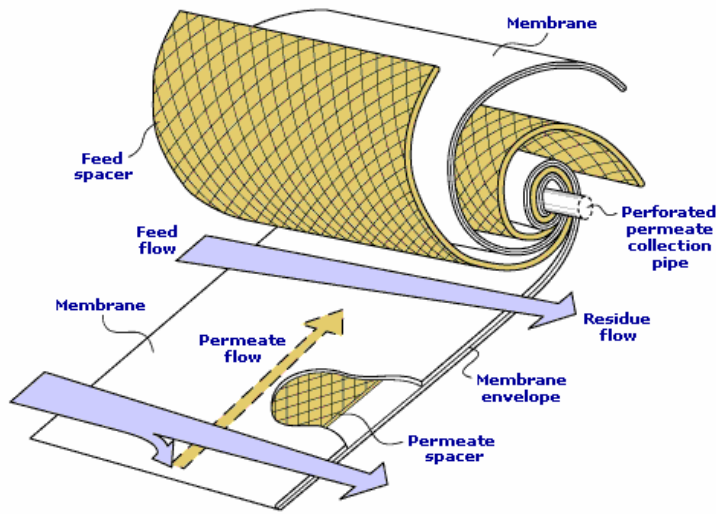


Figure 2.2: Spiral wound membrane module [6]

### *Hollow fibre membrane module*

Hollow-fibre membranes consist of a large number of hollow, hair-like fibres bundled together either a U-shape or straight-forward configuration and housed in a vessel. Feed gas may be introduced on the fibre side or on the shell side of the module. Usually the high-pressure feed gas is directed through the shell side, and the permeate stream which is enriched with the more permeable components is withdrawn from the inside of the fibres through the openings on the fibre tube sheet. It is also possible that the permeate material is collected in the shell side of the hollow fibre module, see figure 2.3. A commercial module might be 1 meter long and 0.1 to 0.25 meter in diameter and contain more than one million hollow fibres [2]. The idealized flow patterns in a hollow fibre module can be co-current as well as counter-current.

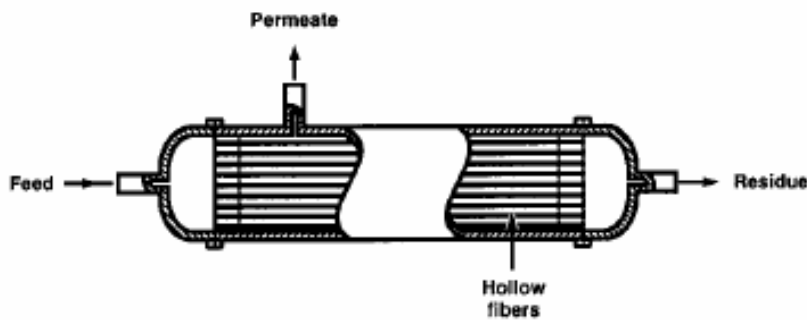


Figure 2.3: Hollow fibre membrane module where the feed is flowing through the fibres [4]

The packing density of a hollow fibre membrane module ( $500 - 9,000 \text{ m}^2/\text{m}^3$ ) is much larger than of a spiral wound membrane module ( $200\text{-}800 \text{ m}^2/\text{m}^3$ ) [2]. As a consequence

the production costs of a hollow fibre membrane module is much less than the production costs of an equivalent spiral wound module [4].

However, the hollow fibre membrane module also has several disadvantages [5]. The membrane permeability in hollow fibre modules is often lower than in spiral wound design. A reason is the difference in thickness of the active membrane layer (in spiral wound modules 500 – 5000 Å and in hollow fibre design 0.1 – 1 µm [2]). The transport rate (Equation 2.1) is reverse proportional to the membrane thickness. This off-sets some of the advantage that derives from the high packing density for hollow fibre design.

Besides, the pressure drop in the fibres can become seriously and the hollow fibre module is more severe for plugging and fouling than the spiral wound module. The latter disadvantage can be rejected, since for gaseous feed stream particulates and other potential fouling materials can be completely and economically removed before it enters the membrane module [4].

Today most of the gas separation membranes are formed into hollow fibre modules, with perhaps fewer than 20% being formed into spiral wound modules [4]. Based on the need to minimize production costs and thus also minimize the membrane area, it appears that hollow fibre membrane modules will eventually dominate the gas separation field completely. As a result, the model of a gas separation membrane unit, which has been developed in this thesis, is a model of a hollow fibre membrane unit.

### ***2.3 Overview of membrane transport mechanisms***

The separative character of a membrane is determined by the difference in transport rate of the components through the membrane. Therefore, the transport through the membrane is one of the fundamental aspects to understand and to improve the process performance. Various models are developed over the years to describe the transport rate. The choice of a transport model depends greatly on the separation under consideration, the membrane used and the purpose of the transport model.

The structure of a membrane can be macro-porous, micro-porous, or dense (non-porous). Only micro-porous or dense membranes are selective. However, macro-porous membranes are widely used to support thin micro-porous and dense membranes when significant pressure differences across the membrane exist. Although micro-porous membranes are topics of considerable research interest, all current commercial gas separations are based on dense polymer membranes, sometimes supported by a macro-porous layer (which is then called an asymmetric membrane) [5].

The various models, which do exist in literature to describe the transport of components through a membrane, are based on theoretical or phenomenological fundamentals. Theoretical models make use of molecular parameters, derived from thermodynamic and physical relations. Phenomenological models are based upon a theoretical background, but their parameters have no fundamental meaning anymore. It is possible to divide the most relevant models to describe the transport of components through a dense membrane into solution-diffusion models, thermodynamics of irreversible processes and the Maxwell-Stefan theory. In Table 2.1 a short overview is given of these models.

**Table 2.1: Overview models to describe transport of components through a dense membrane**

General relation: $J_i = Q_i \frac{d(driving\_force)_i}{dz} = \left( \frac{Q_i}{\delta^m} \right) \cdot \Delta(driving\_force)_i = \bar{Q}_i \cdot \Delta(driving\_force)_i$		
<b>Transport model</b>	<b>Characteristics</b>	<b>Derivation and equation for gas separation</b>
Solution-diffusion model [7]	<ul style="list-style-type: none"> <li>Phenomenological model</li> <li>Multi-component system</li> </ul>	<p>Permeability: product of sorption coefficient (Henry's law) and diffusion coefficient (Fick's law). Permeability coefficient is function of temperature given by Arrhenius expression:</p> $Q_i = Q_{i0} e^{-\frac{E_{a_i}}{R} \left( \frac{1}{T} - \frac{1}{T_0} \right)}$ <p>Driving force: gradient in chemical potential. For gas separation simplified to the difference in partial pressure.</p> $J_i = \frac{Q_i}{\delta^m} \Delta p_i$
Thermodynamics of irreversible processes (TIP) [8]	<ul style="list-style-type: none"> <li>Phenomenological model</li> <li>Multi-component system</li> </ul>	<p>Permeability: matrix with on the diagonal the permeability coefficients and on the off-diagonal the cross term coefficients.</p> <p>Driving force: gradient in chemical potential (or for gas separation gradient in partial pressure). A flux of a component can be caused by any other driving force in addition to its own conjugated force.</p> $J_i = \sum_j (Q_{ij} \cdot (driving\_force)_j)$
Maxwell-Stefan theory [9]	<ul style="list-style-type: none"> <li>Theoretical model</li> <li>Ternary system</li> </ul>	<p>Permeability: incorporated in the friction coefficient, <math>\xi</math></p> <p>Driving force: The driving force on a component is equal to the friction with other components and expressed as a linear function of the velocities (or fluxes):</p> $(driving\_force)_i = \sum \xi_{i,j} (x_j u_i - x_i u_j) + \xi_{i,M} u_i$ <p>As a result the fluxes are given implicitly. The component fraction is the fraction in the membrane. Therefore also a correct solution model has to be used.</p>



In literature there is not a clear consensus which is the best transport model to be used for gas separation through dense membranes. For engineering science and process design, phenomenological models are usually used to describe the transport of components through the membrane. Theoretical models often require complex experiments to determine the fundamental parameters, such as the friction coefficients in the Maxwell-Stefan equation. As a result, the Maxwell-Stefan equations are only used for binary or ternary systems. The model of the membrane unit developed in this thesis has to be applicable for multi-component systems. Since existing literature hardly provides cross term coefficients for the TIP-equations, the solution-diffusion model has been applied to describe the transport of components through the membrane. Besides, the solution-diffusion model is widely accepted to predict transport of components through dense membranes.

The transport rate of components through the membrane can also be influenced by other effects, such as concentration polarisation. Concentration polarisation is the build-up or depletion of species in the boundary layer or film close to the membrane due to mass transfer resistance. However, for gas separation membrane processes concentration polarisation is usually neglected. In Section 3.1 an explanation is given for this assumption.

## ***2.4 Literature review of existing membrane unit simulations***

In literature a limited number of rigorous mathematical models are published for gas separation membrane units. J.I. Marriott [13] and M.H.M. Chowdhury [18] give an extensive review of mathematical models available in literature. Most of the models deal with binary systems and a few of them deal with ternary or multi-component gas separation systems. Also different solution methods for the model equations have been used [18]. Marriott concluded that a rigorous general approach to membrane unit simulations has not yet been published. Most of the existing work has limited applications due to simplifying assumptions [13]. Besides the various simplifications and solution methods, also several attempts have been made to implement a custom model of a membrane gas separation unit into a sequential modular flowsheet simulator. In Table 2.2 a summary is given of the most recent publications concerning the development of a custom model for a multi-component gas separation membrane unit and its interfacing with flowsheet simulators.

Table 2.2 starts with the description of the research performed by C.Y. Pan [10]. Although it was published in 1986, it is included in this table, since different modifications and approaches based on Pan's model have been proposed over the years. Very recently, M.H.M. Chowdhury [18] stated that Pan's [10] model is widely accepted as the most practical representation of multi-component gas separation in hollow fibre asymmetric membranes. For this reason Chowdhury [18] presents a similar model solved with a new developed numerical algorithm in FORTRAN. J.I. Marriott [13] [14] [15] used the experimental data of Pan's [10] work to validate the developed model for gas separation applications. Marriott [13] considered Pan's [10] model as too specific, and developed more detailed models for hollow fibre as well as for spiral wound membrane modules in gPROMS.

**Table 2.2: Overview of mathematical models for gas separation membrane units in literature**

<b>Year &amp; Author</b>	<b>Description &amp; Solution method</b>	<b>Balance equations</b>	<b>Membrane characteristics</b>	<b>Physical properties</b>	<b>Model interfacing</b>
1986 C.Y. Pan [10]	Model for multi-component gas separation by high-flux hollow fibre membranes, either co- or counter-current, feed enters on shell side. Trial and error shooting method is developed to solve the model.	Differential balance for mass, pressure drop is described by Hagen-Poiseuille equation	Solution-diffusion theory: for asymmetric membranes the driving force is defined as the difference between partial pressure of the feed side and partial pressure inside the membrane (component fraction in membrane and pressure of permeate side). No concentration polarisation.	Viscosity calculation is unknown.	Model is not interfaced.
1996 R. Rautenbach et al. [11]	Cross flow models for multi-component gas separation, reverse osmosis and pervaporation developed in FORTRAN. Solution method is unknown.	Differential balances for mass, momentum and energy, assuming plug flow at the feed side and unhindered permeate flow. Pressure drop is neglected.	Solution-diffusion theory: driving force is the difference between partial pressure of the feed and permeate side. Concentration polarisation is neglected.	Utilization of physical property models and data bases of Aspen Plus is possible.	FORTTRAN routine incorporated in Aspen Plus.
1999 S. Tessendorf et al. [12]	Models for multi-component gas separation in spiral wound (cross-current) and hollow fibre (counter-current) membrane configuration. An algorithm based on the orthogonal collocation method to solve the boundary value problems is developed.	Differential balance for mass. Pressure drop can be described by Hagen-Poiseuille equation.	Solution-diffusion theory: driving force is difference in partial pressure, permeability is the product of Fick's diffusion coefficient and Henry's law solubility coefficient.	Unknown.	Implemented in the equation oriented environment OPTISIM.
2001 J.I. Marriott [13], [14], [15]	Dynamic models for multi-component gas separation, reverse osmosis and pervaporation in hollow fibre and spiral wound membrane configuration, developed in gPROMS. The preferred discretisation method for steady state calculations is the orthogonal collocation method (provided by gPROMS).	1-D (plug flow) and 2-D dynamic differential balances for mass, momentum and energy. Case for gas separation process: 1-D models for retentate and permeate side.	Emphasis is not on generic model for transport. Concentration polarisation for 1-D flow models is described by mass resistance in a film layer. Case for gas separation: same transport model as C.Y. Pan [10] is used, concentration polarisation is not taken into account.	Multiflash is used to calculate the properties used in the balance equations	Model is not interfaced.
2002 R.A. Davis [16]	Model for multi-component gas separation and pervaporation in spiral wound and hollow fibre membrane configuration solved by an iterative, trial and error shooting method in HYSYS.	Differential balance for mass; pressure drop is neglected.	Solution-diffusion theory: the driving force is described by the logarithmic-mean average of trans-membrane partial pressure. Concentration polarisation is neglected.	No thermodynamic and physical properties are used in the model equations.	Model developed in HYSYS intrinsic spreadsheet functionality.
2003 T. Brinkmann et al. [17]	Cross flow model for vapour permeation developed in Aspen Custom Modeler (ACM).	Mass balance: cross flow model assuming plug flow at feed side and unhindered permeate flow. Pressure drop is neglected.	Solution-diffusion theory: driving force is the difference in partial pressure; free-volume model to describe diffusion. Concentration polarisation is taken into account in the boundary layer on the feed side of the membrane.	An Aspen Plus physical property package is used.	The ACM model is compatible with Aspen Plus, but not tested.
2005 M.H.M. Chowdhury et al. [18]	Model for multi-component gas separation by high-flux hollow fibre membranes, either co- or counter-current, developed in FORTRAN. Solved as an initial value problem, solution algorithm also developed in FORTRAN.	Based on C.Y. Pan [10]	Based on C.Y. Pan [10]	The Aspen Plus component property databank is used to calculate the gas mixture viscosity with the Wilke method.	FORTTRAN routine incorporated in Aspen Plus.

In this thesis the custom model of a hollow fibre membrane module developed by Marriott [13] is adapted. The model exists of three sub-models, two which describe the flow on either side of the membrane and a third model which characterises the separative properties of the membrane. For gas separation the flows on both sides of the membrane were simulated by one-dimensional mass, momentum and energy balances along the axial length of the membrane unit. Marriott used a physical property package from Multiflash (from Infochem Computer Services Ltd.) to calculate the thermodynamic and physical properties of the membrane unit.

The other models which have been developed for gas separation membrane units (Table 2.2) include many simplifications, usually in order to solve the model with an own developed numerical algorithm. Since numerical methods are included in equations oriented modelling tools to solve custom models, these simplifications are not needed anymore. Marriott [13] developed the membrane model in gPROMS. The custom membrane model described in this thesis has been developed in gPROMS as well as ACM.

Other simplifications are made to calculate the thermodynamic and physical properties for gas separation membrane units. If an external property package can be used, assumptions, such as the ideal gas law, are also not needed anymore. Both Marriott [13] and Brinkmann [17] used an external physical property package. For the custom model developed in this thesis, a physical property package from Aspen Plus is used.

In addition, the custom membrane model developed in this thesis is exported to be used in Aspen Plus. Rautenbach [11] and Chowdhury [18] created a custom membrane model for implementation in an Aspen Plus simulation. However, their models are developed in FORTRAN and needed a solution algorithm. Brinkmann [17] only mentioned that the developed custom model can be exported to Aspen Plus, but it seems that the export functionality was not tested.

### 3 Custom model of a gas separation membrane unit

*In this chapter the description, implementation and validation of the developed custom model for a multi-component gas separation membrane unit is presented. First, the model objective is given and its structure and assumptions are discussed. In the next section the implementation of the set of model equations in gPROMS and ACM is described; including the differences in implementation between both equation oriented modelling tools. In the last section the results, calculated by the developed custom model, are compared with experimental data and existing models.*

#### 3.1 Model description

Before giving a comprehensive description of the model developed for a gas separation membrane unit, the modelling objective is presented.

##### 3.1.1 Modelling objective

The custom model of a gas separation membrane unit, which is developed in gPROMS and ACM, serves as a case study to test the current status of software interfaces and custom model performance in Aspen Plus. A model of a membrane unit is not (yet) available in the Aspen Plus model library and consequently the membrane unit is imitated by a separation block, when present in a process flowsheet. For consistent process simulation it is required to have a membrane model available in Aspen Plus. Besides, for successful implementation of the exported custom membrane model, the model should have the ability to be used like any other model of a process unit operation present in the library of Aspen Plus, as described in Section 1.1.

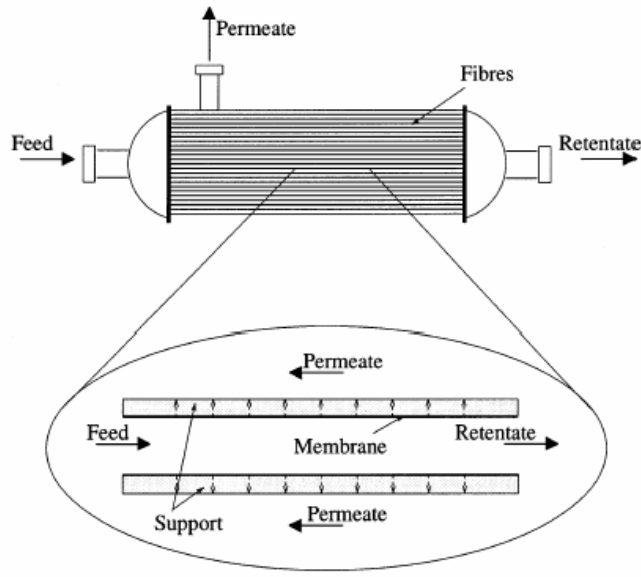
As a result, the objective in this chapter is to develop a generic custom model of a gas separation membrane unit in gPROMS and ACM, which can later on be exported for use in Aspen Plus, as like the other unit models available in the Aspen Plus model library.

In Chapter 2 it is discussed that the model is developed for a hollow fibre membrane module for which the transport through the membrane is described by the solution-diffusion theory. The model Marriott [13] developed recently in gPROMS is adapted.

##### 3.1.2 Model structure and assumptions

In Figure 3.1 a schematic picture is given of a hollow fibre membrane module; for which the feed stream enters the module on the shell side. It is decided to develop the custom model of the gas separation membrane unit in a hierarchical structure using sub-models in order to:

- make a clear distinction between generic and specific (transport of components through the membrane) model parts,
- be able to develop the custom membrane model more easily for various options in equipment and design specifications, and
- enhance the re-usability of (parts of) the custom model for future users.



**Figure 3.1: Flow pattern in a hollow fibre membrane module (feed flows through the fibre side) [14]**

The natural structure of the membrane unit in Figure 3.1 has been exploited for its decomposition into sub-structures. The hollow fibre membrane unit is decomposed into three sections; namely the flow through:

1. (all) fibres
2. shell side, and
3. the membrane.

Before developing the mathematical models for each sub-structure, several assumptions are made to simplify the behaviour of the membrane unit. The assumptions are:

- Considering each fibre individually is impossible and therefore the fibre bundle is treated as a continuous radial symmetric porous medium. The fluid in the shell side of the module flows around each fibre. Each fibre in the module has identical specifications and the total flow through the fibres is distributed equally.
- The flow of gaseous fluid through the shell side of the module is assumed to flow parallel to the fibres. Cross current (radial) effects for the flow through the shell side are not considered, because this makes the complete model of the gas separation membrane unit much more complicated. The transport of components through the membrane depends not only on the axial position, but also on the radial position, and therefore every fibre must be considered separately depending on its radial position in the module shell.
- The pressure drop occurring at the flow entries and exits are neglected. The pressure drops if known can easily be incorporated as parameters in the model.
- It is assumed that there is no hold-up of gaseous material in the membrane itself; the residence time compared to the flow through the fibres and shell side is very small. Therefore the membrane can be considered as an interface between the flows (permeate and retentate) on either side of the membrane.

- Energy (enthalpy) which is carried by the flow of components through the membrane is taken into account. The temperature of the membrane is equal to the temperature of the retentate side.
- The flow through the shell side as well as through the fibres is laminar (Reynolds number is 0 – 5). However, for both flows plug flow is assumed and temperature variations in radial direction are neglected.
- Back-mixing of mass and heat in axial direction is neglected. The Peclet number is large for low Reynolds numbers, as such for the flow through the fibres and shell side of the module, and thus little back-mixing will occur.
- Concentration polarisation for gas separation membranes is neglected. Usually for gas separation processes, the transport of components through the membrane is slower than the radial diffusion of the gaseous components (determined by the molecular diffusion coefficient) in the fluid of the flow through the shell and fibre side. As a result, the transport through the membrane is the rate limiting factor.
- The axial pressure drop inside each fibre is described by a friction factor in the momentum balance. The friction factor is derived from the Hagen-Poiseuille relation (used for laminar flow through pipes). The axial pressure drop for the shell side is also considered by applying a similar relation characterized using a friction parameter.

### 3.1.3 Model description

The assumptions made in the previous paragraph for a hollow fibre gas separation membrane unit result in a set of model equations which are practically similar as the model Marriott's [13] developed recently (see Section 2.4). The equations used for the custom membrane model are presented in Appendix A.

The flows through the shell side and (each) fibre are described by one-dimensional balance equations for mass (mole), energy and momentum. Since for the shell side and fibres plug flow is assumed, the same set of model equations can be used for both sub-models. The differential balance equations contain accumulation, convection and transport terms. Eventually the custom membrane model will be part of an Aspen Plus flowsheet simulation, therefore:

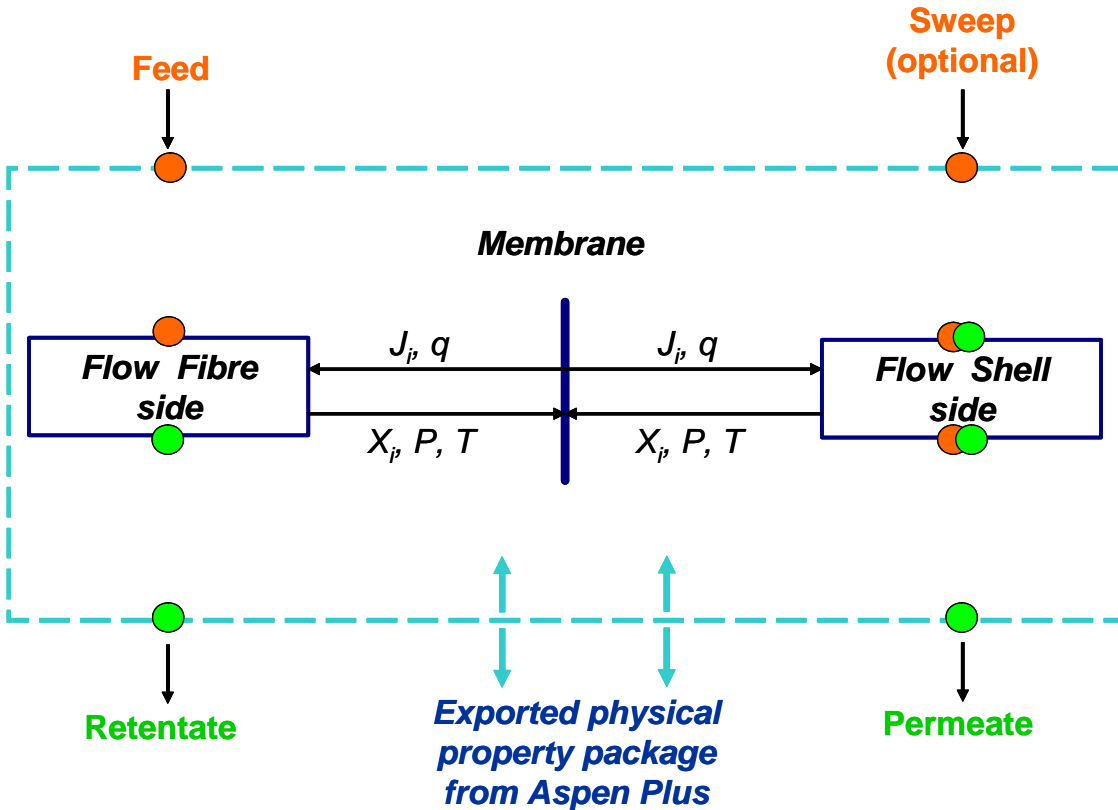
- The accumulation terms in the balance equations are set to zero (steady state), since Aspen Plus simulations are limited to continuous, steady state problems.
- The boundary conditions of the differential balance equations are given by the conditions of the inlet streams to the membrane module (the feed and sweep stream).

The sub-model for the flow through the membrane contains a constitutive equation relating the difference in partial pressure between the flows on both sides of the membrane to a mass and energy flux through the membrane. Furthermore, the Arrhenius relation is used to describe the permeability coefficient for the transport of components as a function of the temperature.

The model also includes variables to calculate or specify the characteristics of a hollow fibre membrane unit, such as the stage cut, pressure ratio and packing density.

Since the custom membrane model has to be fully embedded in an Aspen Plus flowsheet simulation, the model makes use of an exported physical property package from Aspen Plus to calculate thermodynamic and physical properties. The existence of physical property interfaces between Aspen Plus and gPROMS/ACM makes this possible.

The sub-models for the flow through fibre, shell and membrane have to be connected to present the complete model of a gas separation membrane unit. The main model contains equations to connect the ports, defined within the sub-models, to each other. Via the port connections information is passed on from one sub-model to another. The ports of the sub-models describing the flow through fibre and shell side pass on the component fractions, temperature and pressure (on every discretisation node) to the sub-model which describe the transport of components through the membrane. This sub-model on its turn calculates on basis of the information it gets the mass and energy fluxes through the membrane and returns these values to the two sub-models for flow through fibre and shell side, as shown in Figure 3.2.



**Figure 3.2: The structural description of the developed custom membrane model for gas separation**

One of the criteria for the performance of an exported custom model is that the model has to be used for different equipment or design specifications in Aspen Plus. To be able to choose an equipment or design specification in Aspen Plus, the option for the specification should already be embedded in the custom model during its development in the equation oriented modelling tools.

The developed custom membrane model includes several options for equipment and design specifications; the user is able to make a choice on:

- **Feed side** - The feed stream can enter on the shell side or on fibre side of the hollow fibre membrane module. This option is embedded in the custom model by connecting the feed stream to the port of the correct sub-model. This is schematic shown in Figure 3.2, the feed stream can be connected to the port of the sub-model describing the flow through the fibre side or to the port of the sub-model describing the flow through the shell side. The sweep stream is then connected to the other sub-model.
- **Mode of operation** - The flow pattern in a hollow fibre module can be co-current or counter-current. The direction of the sub-model for the flow through the shell side can be inverted, as shown also in Figure 3.2. For counter-current operation, the feed or sweep stream is connected to the port at the end of the shell sub-model. Also the discretisation method for this sub-model has to be changed (see Section 3.2).
- **Usage of a sweep stream** – A sweep stream is usually not used for gas separation processes. However, a sweep stream if present can easily be connected to the custom model, as like the feed stream. If the sweep stream is not present, the boundary conditions for the balance equations, which are otherwise given by the sweep stream, have to be specified by the user (see Section 3.3).
- **Method of simulation** – To obtain a set of model equations which can be solved (not under- or over-specified), values for variables and parameters have to be specified. There are several options conceivable for the developed custom membrane model. The most straightforward option is to assign values for the variables describing the geometry of the unit (rating method). On basis of the conditions of the feed (and sweep) stream and the geometry data, the outlet conditions for the retentate and permeate stream can be calculated. However, the geometry is not always known and for these situations the custom membrane model can also be used (design method). A more detailed description of the rating and design method is given in Section 3.3.
- **Type of membrane** – The type of membrane used in a hollow fibre module can be symmetric or asymmetric. For both types the solution-diffusion theory can be applied, however the definition of the driving force is a bit different (see Section 3.4).

## 3.2 Model implementation

The set of model equations for a gas separation membrane unit, presented in the previous section, have been implemented in gProms version 2.3.6 and Aspen Custom Modeler (ACM) 2004.1 version 13.2. Printouts of the model equations written in gProms and ACM are shown in Appendix B.

### 3.2.1 Discretisation method

Distributed models are usually solved by discretising the spatial domain. In equation oriented modelling tools several discretisation methods are available; finite difference



methods (backward, forward and centred) and orthogonal collocation on finite elements methods. The user has to specify the discretisation method (type, order and number of discretisation intervals).

Marriott [13] preferred using the orthogonal collocation method in gPROMS to solve the distributed membrane model. According to Marriott [13] the finite difference method is less accurate than the orthogonal collocation on finite elements. However, for the model of the gas separation membrane unit developed in this thesis, finite difference methods are used to discretise the model variables. The orthogonal collocation method is not applicable for solving the custom membrane model at large stage cuts. If more than  $\pm 50\%$  of the feed stream permeates through the membrane the model does not converge anymore. This is not the case if a finite difference method is used. Besides, it is recommended [19] to use finite difference methods for purely or strongly convective systems. The balance equations for the sub-models describing the flow through the shell side and fibres do not contain diffusive or dispersive terms, and can therefore be considered as a convective system.

Generally for finite difference methods, one-sided finite differences should be used taken opposite to the direction of the flow. For a set of model equations containing a momentum balance, the pressure have to be discretised using one-sided finite differences taken in the direction of the flow, while all the other variables are discretised using one-sided finite differences opposite to the direction of the flow [19].

The developed custom model is solved for steady state applications and therefore all the boundary conditions, including the pressure, can be given at the inlet of the flow model. As a result, the pressure is discretised with the same finite difference method as for the other variables.

For co-current operation, the flows through the fibres and shell side are both from left ( $z = 0$ ) to right ( $z = L$ ), and the backward finite difference method is used. For counter-current operation, the flow through the fibres is from left to right (backward difference method), while the flow through the shell side is from right to left for which a forward finite difference method has to be used.

The accuracy of the finite difference method can be enhanced by increasing the number of discretisation intervals. To demonstrate this, the number of discretisation intervals for the developed custom membrane model is increased. The accuracy can be defined by the error in the total molar balance:

$$error\% = \left( 1 - \frac{F_{feed}}{F_{permeate} + F_{retentate}} \right) \cdot 100\% \quad (3.1)$$

The model used to calculate the error as a function of the number of discretisation intervals is a counter-current membrane system (geometry and permeability data is given in Section 3.4). In Table 3.1 the effect of the number of discretisation intervals on the total molar balance error is given. As expected, the error is reduced if the number of discretisation intervals is increased. However, by increasing the number of discretisation intervals the model size is also increased. For a reasonable accuracy of the model, the number of discretisation intervals is set to 50.

**Table 3.1: Molar balance error as function of the number of discretisation intervals**

Number of discretisation intervals	Feed stream ( $10^{-5}$ kmol/hr)	Retentate ( $10^{-5}$ kmol/hr)	Permeate ( $10^{-5}$ kmol/hr)	Error (%)
10	8.00000	3.7272	4.4143	1.738
30	8.00000	3.7089	4.3584	0.834
50	8.00000	3.6918	4.3574	0.610
70	8.00000	3.6822	4.3570	0.488
90	8.00000	3.6761	4.3569	0.411

### 3.2.2 Implementation in gPROMS and ACM

In Figure 3.3 and Figure 3.4 the graphical user interfaces of gPROMS and ACM together with the implemented membrane models are shown. During the development of the hierarchical membrane model in gPROMS and ACM small differences has been detected between both equation oriented modelling tools. A glance at the printouts in Appendix B shows the difference in writing the (distributed) model equations in gPROMS and ACM. Some other differences which have been encountered are:

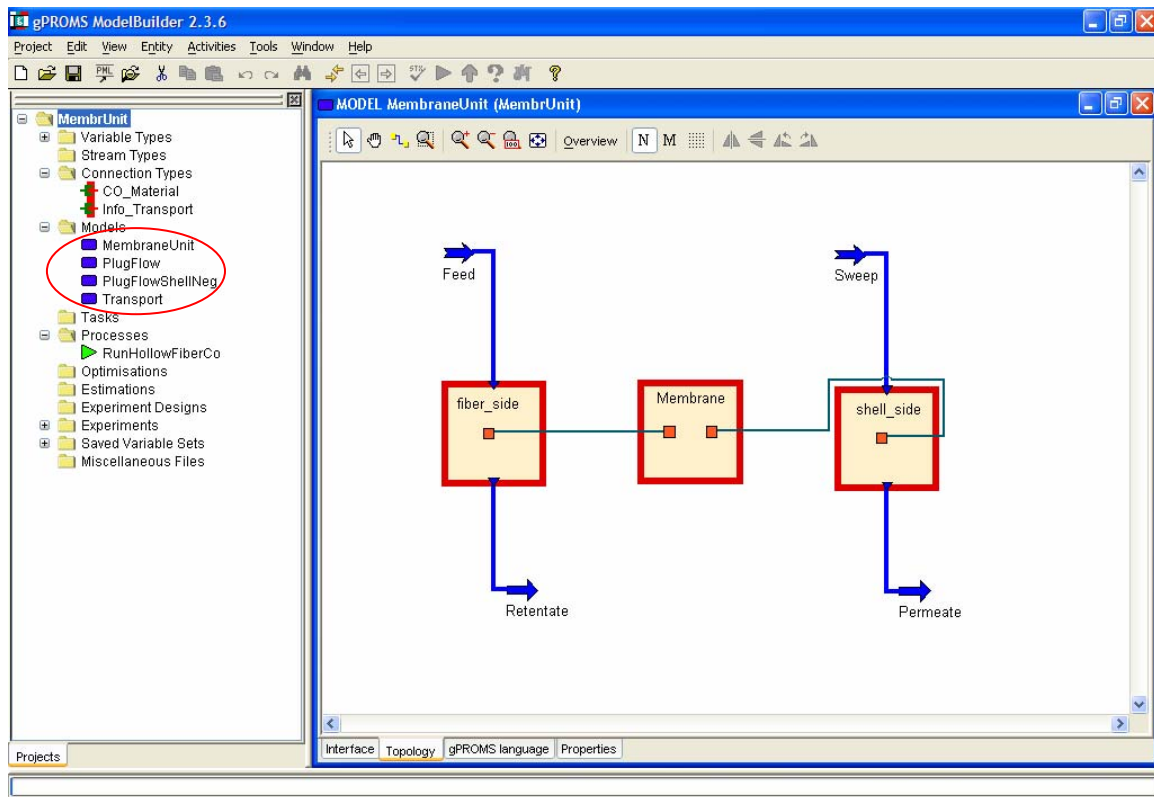
➤ ***Unit of measurement***

ACM uses metric units (base units of measurements) to solve the model equations. The user must ensure that the model equations are written consistently in the base units of measurements. However, the user can choose the unit of measurements of the variables which are displayed in the graphical user interface. This means that values shown on plots and tables are automatically displayed in the current unit of measurements and values you enter into the forms are automatically converted to the base units of measurement. In gPROMS the user can write the set of model equations in an own chosen unit system, but must ensure that the input variables and parameters are also in these units. Since later on the custom membrane model will be exported, it is recommended to use in gPROMS the same unit system as defined by the CAPE-OPEN standards for unit model interfacing (see paragraph 4.3.2)

➤ ***Connection of sub-models***

For connection of the sub-models, variables (information) have to be passed on from one model to the other and visa versa, as explained earlier (see Figure 3.2). These variables are dependent on the axial position in the membrane module. However in ACM it is not possible to define distributed variables in ports and the port is an input port or an output port. This is in contrast to the connection types for ports in gPROMS, in which distributed variables can be defined and the direction can be chosen as inlet, outlet or bidirectional.

To connect the sub-models of the custom membrane model in ACM, the variables in the ports are defined by an array which is on its turn defined by the number of discretisation intervals. Also two ports are defined, one containing the variables (information) temperature, pressure and molar fraction and another one containing the variables for the molar and energy flux through the membrane.



**Figure 3.3: Model implementation: user interface of gPROMS 2.3.6**

➤ ***Definition of discretisation method of sub-models in main model***

As described in the previous paragraph, for simulating a counter-current flow pattern the discretisation (finite difference) methods for the sub-models describing the flow through fibre and shell side are not the same. However, the set of equations is for both sub-models equal and therefore it is preferred to adapt only the discretisation method for the two sub-models in the main model. To switch between co-current and counter-current run mode a selector ( $\rightarrow$  case) in gPROMS or a selector parameter ( $\rightarrow$  if then else) in ACM is defined. However, in gPROMS it is not possible to assign the method for discretisation in a case structure. As a result an extra sub-model is created (see Figure 3.3, PlugFlowShellNeg) which must be used if the counter-current flow pattern is simulated. This has also consequences for the exported custom model from gPROMS. Two custom models have to be exported, one for the co-current and another one for the counter-current run mode.

➤ ***Definition of the various equipment and design specifications***

For each option of an equipment or design specification created for the membrane model, as described in the previous section, a selector ( $\rightarrow$  case) in gPROMS or a selector parameter ( $\rightarrow$  if then else) in ACM is defined. Some of these selectors are used in different sub-models. For the custom model in gPROMS the selector can not be defined globally and therefore the future user cannot simply change the value of one selector, but has to take care that the same selectors defined separately in the sub-models are also adapted.

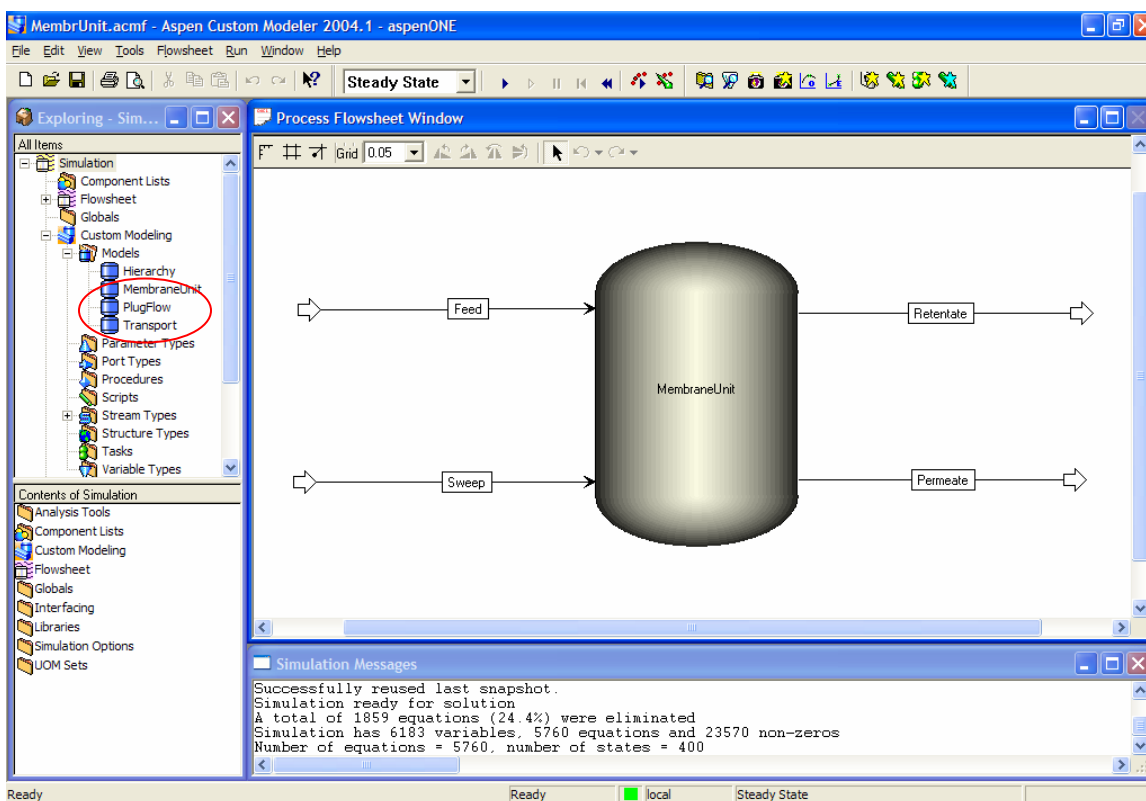


Figure 3.4: Model implementation: user interface Aspen Custom Modeler 2004.1/13.2

### 3.3 Solving the custom model

In order to solve the custom membrane model developed in gPROMS and ACM values of some parameters and variables have to be specified. As described in Section 3.1 two types of simulation methods for the developed custom membrane model are defined, namely the rating method and the design method. The rating method can be used for simulating the membrane unit, for which the geometry is known. On basis of the feed conditions and the values specified for the geometry parameters the conditions for the permeate and retentate streams are calculated. The design method can be used if the exact geometry data (number of fibres, shell diameter, etc.) is not available, but only a membrane area is known. The design method can also be used in Aspen Plus for problems for which the membrane area has to be determined. Consequently, a design specification has to be applied. For example the preferred stage cut or fraction of a component in the permeate stream has to be specified and the required membrane area will be calculated.

In Appendix C a list is given of parameters and variables which have to be specified for the rating and design method. Of course it is possible to specify other variables, as long as a set of model equations is created, which can be solved. In Table 3.2 a summary of Appendix C is given, showing the specified variables which are different for the rating and design method.

**Table 3.2: Different variables specified for rating and design method**

Rating method	Design method
<ul style="list-style-type: none"><li>• Inner diameter of a fibre</li><li>• Outer diameter of a fibre</li><li>• Shell diameter of module</li><li>• Number of fibres</li><li>• Length of module</li></ul>	<ul style="list-style-type: none"><li>• Packing density of module</li><li>• Ratio of cross sectional area fibre and shell side</li><li>• Membrane thickness</li><li>• Membrane area</li><li>• Constant to specify velocity of the feed stream</li></ul>

Since the membrane model is a rate-based model, the volume of the unit has to be known for simulations. For the design method two extra variables are defined to predict the volume of the unit, the ratio of cross sectional areas and a constant to specify the velocity of the feed stream.

As described also in Appendix C, boundary conditions have to be given to solve the custom membrane model. At the inlets of the sub-models describing the flow through shell and fibre side the flowrate, pressure, temperature and component fractions have to be specified. For the exported custom membrane model in Aspen Plus, these boundary conditions are specified by the conditions of the feed and sweep stream connected to the model. In equation oriented modelling tools, the Aspen Plus streams are imitated by specifying values for the variables in the inlet ports of the custom model.

Not always a sweep stream is present, and the boundary conditions have to be specified by the user. The boundary conditions for the temperature and component fraction of the ‘fictional’ sweep stream are set equal to the values of these variables in the feed stream. The flowrate of the ‘fictional’ sweep stream is set to a very small value, almost zero and the pressure is defined by the operating conditions specified for the membrane unit.

After the specification of variables and boundary conditions a set of model equations for a gas separation membrane unit is created which can be solved. This set cannot be solved directly from the default values defined for the variables used, but several intermediate modelling steps are required to approach the final result. This is more clearly described in Chapter 5.

### **3.4 Model comparison**

In spite of the differences reported in Section 3.2 between gPROMS and ACM, both equation oriented modelling tools give exactly the same results for the simulation of the gas separation membrane unit.

The developed model of a gas separation membrane unit has also been compared to the simulations executed by Marriott [13] and Chowdhury [18]. Marriott [13] and Chowdhury [18] based their model validation on experimental data reported by Pan [10] for hydrogen recovery from simulated purge gas of an ammonia plant. Pan [10] performed experiments on an asymmetric hollow-fibre membrane module, see Table 3.3 for the geometric parameters of the module. The high-pressure feed flowed through the shell side of the module, and the permeate stream was withdrawn from the fibre openings on the tube (fibre) side. For asymmetric membranes the transport of components through the membrane is given by [13]:

$$J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,M} P_F - X_{i,S} P_S) \quad (3.2)$$

Instead of using the partial pressure of the permeate stream in the fibres, the component fraction inside the membrane is multiplied by the pressure of the permeate stream. Pan [10] has stated that the porous supporting layer of the membrane prevents mixing of local permeate fluxes; the downstream concentration is independent of the bulk concentration of the permeate stream. Operation conditions and permeance parameters are given in Table 3.4. The Arrhenius equation for the temperature dependence of the permeability coefficient presented in the custom model is left out of consideration for this comparison. Since the geometry is known for this system, the ‘rating’ method is used.

**Table 3.3: Geometric parameters [10]**

Number of fibres	20
Fibre length (cm)	15
Inner diameter ( $\mu m$ )	80
Outer diameter ( $\mu m$ )	200
Shell diameter (mm)	21

**Table 3.4: Operation conditions and permeance [10]**

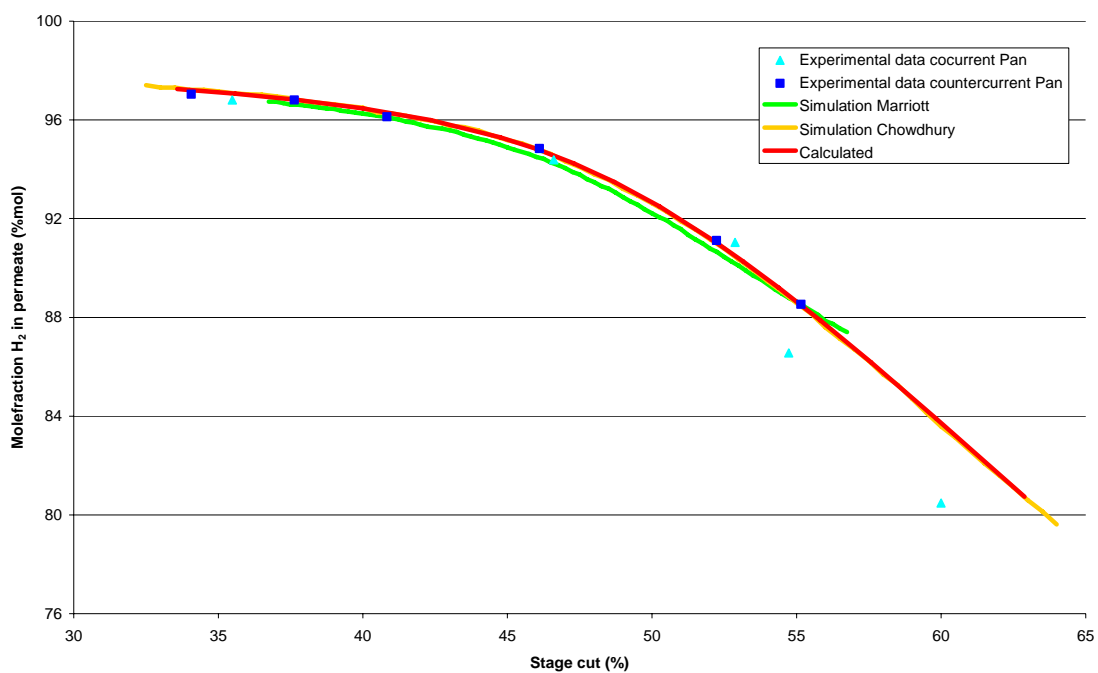
Feed temperature (K)	298
Feed pressure (bar)	69.64
Feed molar fraction H <sub>2</sub> (mol/mol)	0.5178
Feed molar fraction N <sub>2</sub> (mol/mol)	0.2469
Feed molar fraction CH <sub>4</sub> (mol/mol)	0.1957
Feed molar fraction Ar (mol/mol)	0.0396
Permeate pressure (bar)	11.23
Permeance H <sub>2</sub> ( $10^{-10}$ mol/m <sup>2</sup> sPa)	284
Permeance N <sub>2</sub> ( $10^{-10}$ mol/m <sup>2</sup> sPa)	2.95
Permeance CH <sub>4</sub> ( $10^{-10}$ mol/m <sup>2</sup> sPa)	2.84
Permeance Ar ( $10^{-10}$ mol/m <sup>2</sup> sPa)	7.70

Marriott [13] and Chowdhury [18] reported graphs showing the hydrogen purity in the permeate stream as a function of the total product recovery (stage cut). To compare the developed model with their graphs, the stage cut has been varied by changing the value of the feed flowrate. In Appendix D the results obtained are given for the component fractions in the permeate stream at different feed flowrates. Figure 3.5 and Figure 3.6 shows the stage cut versus the molar fraction of the components in the permeate stream.

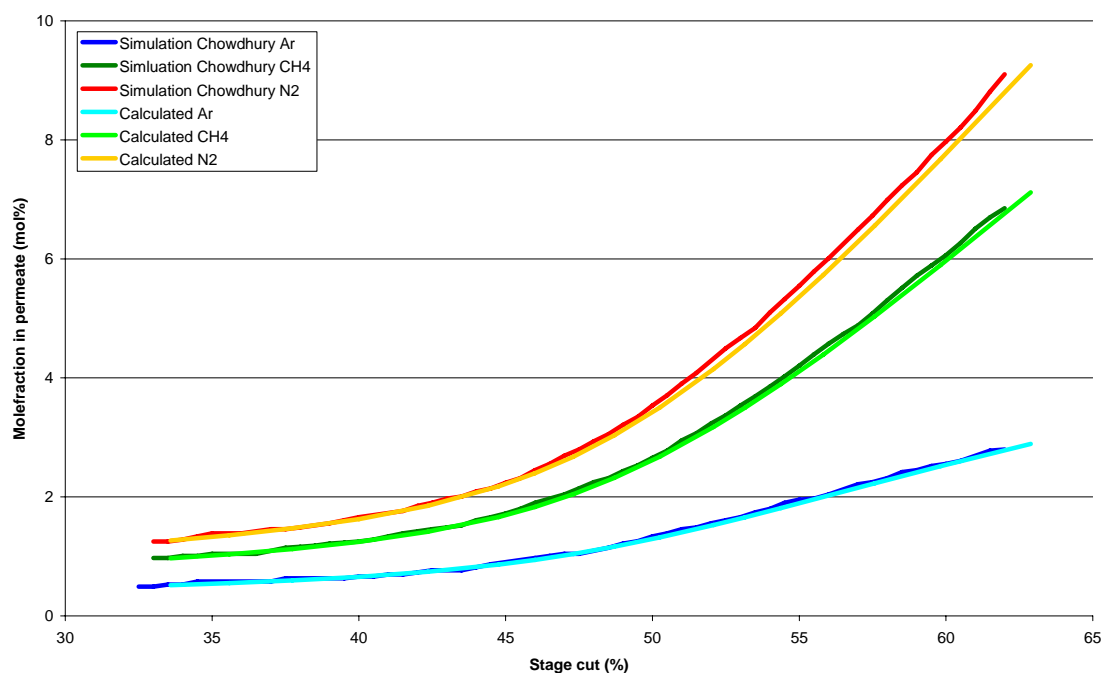
There are small differences at large stage cuts between Chowdhury’s [18] simulation and the calculated results, see Figure 3.6. A reason for these small differences can be ascribed to the accuracy of determining the results of the graphs Chowdhury [18] presented. The graphs are scanned and with a software script tables are created from the graphs. Summing up the scanned molar fractions of the components in the permeate stream reported by Chowdhury gives a total fraction larger than 1.

Figure 3.5 also shows a small difference between Marriott’s [13] simulation and the simulation performed with the developed custom membrane model. This is probably due to the difference in discretisation method Marriott [13] used for solving the set of model equations, as described already in paragraph 3.2.1.

Despite the small difference, it can be concluded that the custom model of a gas separation membrane unit, developed in this chapter, gives comparable results as the custom models created by Marriott [13] and Chowdhury [18].



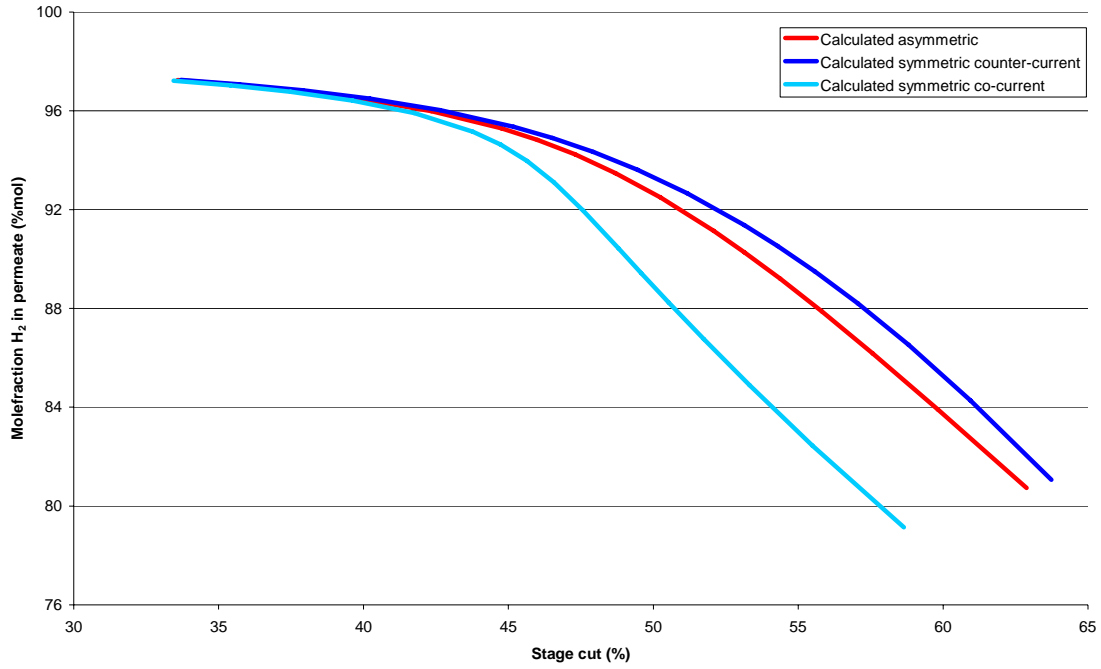
**Figure 3.5: Stage cut versus molar fraction  $H_2$  in the permeate stream**



**Figure 3.6: Stage cut versus the molar fraction of the other components in the permeate stream**

In Figure 3.7 also the results for the simulation of a symmetric membrane are shown. For an asymmetric membrane there is hardly any difference between co-current and countercurrent operation. For a symmetric membrane there is a difference, since the partial

pressure of the retentate side as well as of the permeate side are taken into account for calculating the transport of components through the membrane. As expected, the fraction of the key component ( $H_2$ ) in the permeate stream is larger for counter-current operation mode than for co-current operation.



**Figure 3.7: Stage cut versus molar fraction  $H_2$  in permeate stream for symmetric and asymmetric membrane**



## 4 Software interoperability: current status

*In this chapter the current status of software interoperability for process simulation is examined. The first section gives background information for the standards available for interfacing. Thereafter a work process for future users is developed to support them on interfacing physical property packages and custom models of process unit operations. In the last section several issues in the field of software interoperability are reported which were encountered during the development and implementation of the custom membrane model in Aspen Plus.*

### 4.1 Background information

As already discussed in Chapter 1, within the Dow Chemical Company they aim to centre their process simulation and design activities around Aspen Plus (from Aspen Tech). For models of process unit operations not (yet) available in the Aspen Plus model library, user models can be embedded in the simulations. Various software tools can create these user models, for example FORTRAN, Microsoft Excel (from Microsoft Corp.) and Aspen Custom Modeler (from Aspen Tech Inc.). Equation oriented software tools however have the advantage that numerical solvers are already incorporated in the program and that the user only needs to concentrate on writing the set of model equations to describe the process unit operation.

The emphasis in this thesis is on the equation oriented software tools gPROMS (from Process Systems Enterprise Ltd.) and Aspen Custom Modeler (from Aspen Tech Inc.). For exporting physical property packages from Aspen Plus to gPROMS and for interfacing custom models developed in gPROMS with Aspen Plus the CAPE-OPEN standards (CAPE = computer-aided process engineering) are recently available. Aspen Tech developed for the interfacing of physical property packages and custom models between ACM and Aspen Plus its own standards.

Since Aspen Plus and ACM are both available in the Aspen Engineering Suite (from Aspen Tech Inc.) released by the same software vendor, it is expected that a custom model of a process unit operation developed in ACM can more readily be implemented in Aspen Plus, than a custom model developed in gPROMS and exported as a CAPE-OPEN compatible model.

In the next two paragraphs background information is presented of the development and current status of CAPE-OPEN standards for interfacing and Aspen Tech interfaces.

#### 4.1.1 CAPE-OPEN standards (gPROMS – Aspen Plus)

The tools available for process simulation can be seen as process modelling environments which support the construction of a process model and allow the user to perform different tasks such as process simulation and optimisation. The process modelling environments can make use of various process modelling components, such as thermodynamic and physical properties, models of unit operations, numerical solvers and kinetic systems [1]. The different tools for process simulation have different strengths and weaknesses in software components. To obtain the best simulation and design results more than one simulator is usually required. This makes chemical process simulation a slow and costly

process. Furthermore, the advances made by research institutes and in-house specialities cannot readily be embedded in process modelling environments.

In the early nineties the idea was created to combine process modelling components of various origins in one process modelling environment for faster and consistent process simulation. Two projects were initiated, partly funded by the European Community, and required the collaboration of users (operating companies), software vendors and academic institutions [20]:

- CAPE-OPEN project (1997 - 1999): Software interoperability requires an agreed set of interface standards. As a consequence, the objective of this project was to develop open interface specifications enabling interoperability of process simulation components, and to demonstrate their viability through working prototypes.
- Global CAPE-OPEN project (1999 - 2002): This project promoted the first commercial releases of CAPE-OPEN compliant process modelling environments and components.

The outcome of these two projects led among other things in 2001 to the establishment of the Co-LaN organisation [21]. The Co-LaN organisation is an internationally recognised user-driven organisation for testing and management of the CAPE-OPEN standards. The Dow Chemical Company is also one of the members of Co-LaN.



**Figure 4.1: CO-LaN Logo [21]**

Also major process simulation software suppliers were involved in the CAPE-OPEN projects; therefore CAPE-OPEN (CO) compliant versions of simulation software were released recently or will be released in the near future. The versions of Aspen Plus used for this project (see section 4.2) are fully CO compliant. The current version of gPROMS is CO compliant for thermodynamic and physical properties modelling components. The 'Export to CAPE-OPEN' functionality for developed custom models is not available in this version. PSE provided gPROMS alpha version 3.0 for this project, which allows the export of gPROMS models to CO compliant packages. In Figure 4.2 the work flow for exporting custom models in gPROMS is shown.

Based on the release of CO compliant software packages, the first applications are beginning to appear. The article by H. Pingen et al. [1] published earlier this year give a short overview of industrial demonstrations on the functionality of the CAPE-OPEN standards. An example related to this project is the development of a rate-based absorber model by M. Pons in 2004 [23]. The absorber model is developed in gPROMS (alpha

version 3.0) using the Aspen Plus physical properties and afterwards it is used an integrated unit operation in an Aspen Plus flowsheet.

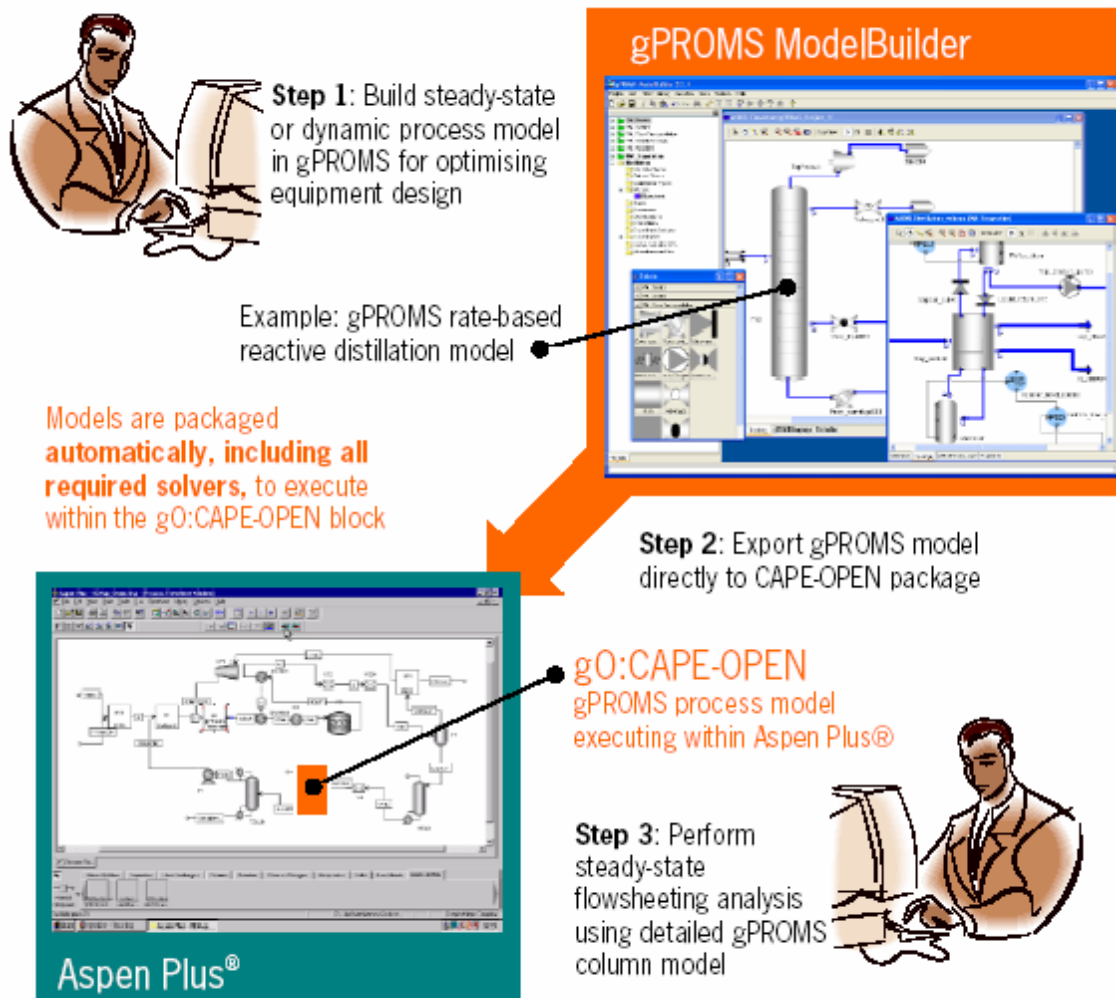


Figure 4.2: Brochure of gO:CAPE-OPEN functionality in gPROMS [22]

#### 4.1.2 Aspen Tech interfaces (ACM – Aspen Plus)

The various simulation software packages available in the Aspen Engineering Suite (from Aspen Tech Inc.) are highly integrated with each other. To be brief, this contribution only focuses on the integration of Aspen Custom Modeler (ACM) and Aspen Plus.

In Figure 4.3 a schematic representation is given of the usage of a physical property package (property definition file) from Aspen Plus in ACM. Property information saved by an Aspen Plus simulation can be used in ACM for property calculations. For further explanation the reader is referred to the work process for physical property interfacing developed in Section 4.2.

## ComponentList with Physical Properties

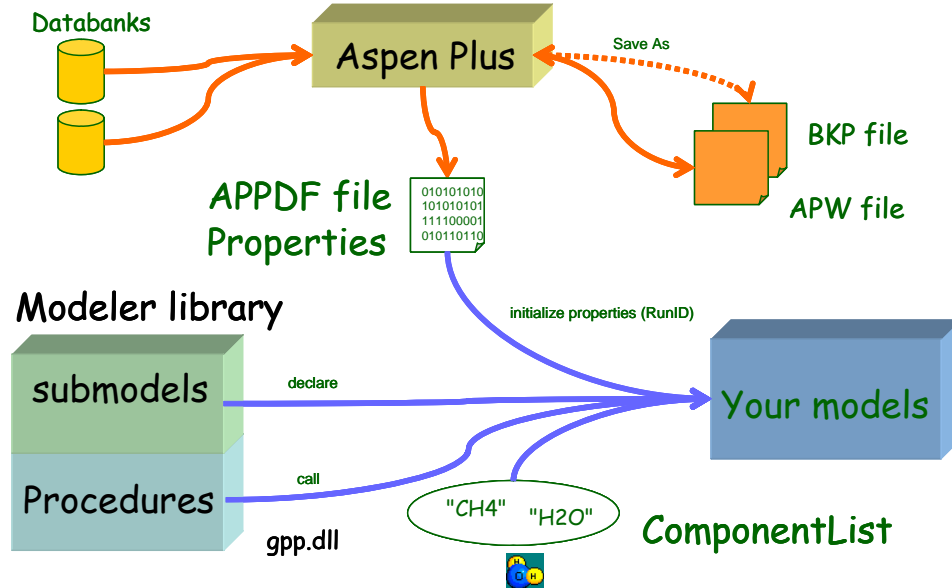


Figure 4.3: Integration of physical property package from Aspen Plus in ACM [24]

Two functionalities are available in ACM to export a custom model for implementation in Aspen Plus, namely flowsheet export and model export. The option in ACM for flowsheet export was created in the past; at the time that the functionality of model export didn't yet exist. An exported flowsheet embedded in Aspen Plus is less flexible than an exported model. For an exported flowsheet it is impossible to change the list of components, the physical property package and parameters. Therefore it is recommended to use the functionality of model export to create an ACM user model in Aspen Plus [25]. In Aspen Plus numerical solvers are included to calculate the results of the exported ACM model block. This is in contradiction to the gPROMS CAPE-OPEN user block (see Figure 4.2), where the solver is included in the exported model package.

Several attempts have been made to use an ACM user block in Aspen Plus. The software installation files of ACM contain an example of the VAM reactor. This example simulates a gas phase tubular reactor which is part of a process for producing vinyl acetate monomer (VAM) and shows the ability to use ACM to create custom models for usage in Aspen Plus. Recently J.A. Jara [26] developed a custom model for a multi-tubular catalytic fixed-bed reactor in ACM and exported it into the flowsheet simulator. Possible issues encountered during model interfacing are not reported.

### 4.2 Work processes for interfacing

In this project work processes are developed to support future users on interfacing physical property packages and custom models of process unit operations. Together with the work processes, a plug flow model of a tube is developed in both gPROMS and ACM and interfaced with Aspen Plus, to show the usage of the interfaces.

In Appendix E work processes are created for:

- Physical property interfacing
  - Aspen Plus 12.1  $\Rightarrow$  gPROMS 2.3 (and 3.0)
  - Aspen Plus 2004.1/13.2  $\Rightarrow$  Aspen Custom Modeler 2004.1/13.2
- Unit model interfacing
  - gPROMS 3.0  $\Rightarrow$  Aspen Plus 12.1
  - Aspen Custom Modeler 2004.1/13.2  $\Rightarrow$  Aspen Plus 2004.1/13.2

Exporting models from ACM requires a C++ compiler installed on the computer for creating \*.msi files. However, the installation (creating a \*.dll file) and the use of an exported custom model in Aspen Plus does not require a compiler. Since Aspen Plus version 2004.1/13.2 has the advantage that solvers are included to handle ACM custom models which do make use of group decomposition to converge, this version is used to develop the work processes.

As already explained in the previous section, Process Systems Enterprise Ltd. provided gPROMS alpha version 3.0 for this project, to export gPROMS models to CAPE-OPEN packages (create a \*.gCO file).

Due to differences in interpretation of the CAPE-OPEN standards, a physical property package from Aspen Plus 2004.1 can not (yet) be used inside a gPROMS simulation and an exported CO-compliant gPROMS model can not (yet) be embedded in Aspen Plus 2004.1. Consequently, Aspen Plus 12.1 is used to develop the work processes for the gPROMS/Aspen Plus interfaces.

### **4.3 Current status**

In this section the work flow and issues for interfacing are reported. They are encountered during the development of the custom membrane model and its export to and implementation in Aspen Plus. The same versions of process simulation software as for the development of the work processes are used. The procedures and issues are divided into cases which have to do with physical property interfacing and with unit model interfacing.

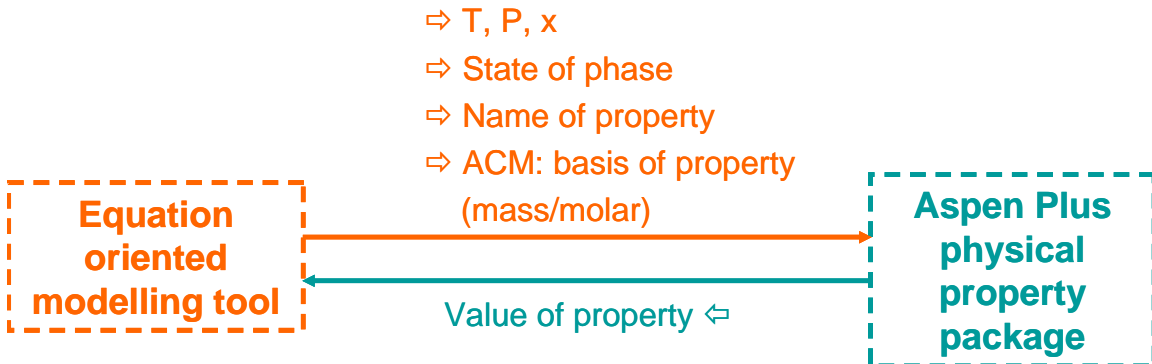
#### **4.3.1 Physical property interface**

Work flow and issues encountered during the development of the membrane model in equation oriented modelling tools, where property calculations are performed by using an Aspen Plus physical property package:

##### **➤ Information exchange**

To call a thermodynamic or physical property from the package, information needs to be exchanged via the interface. In Figure 4.4 a schematic representation is given of the physical property package interface, from a user point of view. The custom model has to give information on the conditions (temperature, pressure and composition) of the phase for which the property will be calculated. Together with this information, the user has to include the state of the phase (gas, liquid or solid) and have to specify which property (or procedure) is needed (see also the work processes for physical property interfacing in

Appendix E). As a result the physical property package returns the value of the thermodynamic or physical property.



**Figure 4.4: Physical property package interface information exchange**

In Table 4.1 the units are given of the variables which are exchanged between the custom model and the physical property package. The property can be calculated on mass or molar basis. For the CAPE-OPEN interface the properties are returned on molar basis. In ACM the property is calculated on molar - or mass basis depending on the chosen procedure name to call the property.

**Table 4.1: Units defined for variables exchanged between model and physical property package**

	<b>CAPE-OPEN interface</b>	<b>Aspen Tech interface</b>
Temperature	<i>Kelvin</i>	<i>Celsius</i>
Pressure	<i>Pascal</i>	<i>Bar</i>
Component fraction	<i>mol/mol</i>	<i>kmol/kmol</i>
Basis of property	<i>molar</i>	<i>molar or mass</i>

In both equation oriented modelling tools the component fraction is an array containing the number of components. The order of the components in the array is important. The CAPE-OPEN compliant physical property package linked to gPROMS expects to get the components in the component fraction array in the same order as the order which is defined in Aspen Plus when creating the package. In ACM, the order of the components is the order of the components defined in the component list (normally alphabetical order), which is not necessary the same as in Aspen Plus. The name of the components in the component list of ACM is equal to the name entered for the components in Aspen Plus when creating the property definition file.

### ➤ **Reference state enthalpy**

The custom model should use the same reference state for the enthalpy calculations as the physical property package. Some physical property packages include the heat of formation in the enthalpy values, whereas others don't. Consistency is required to avoid erroneous results. Therefore care should be taken how the enthalpy reference state is defined in the physical property package used. The enthalpy values in physical property packages created by Aspen Plus do include the heat of formation. As a consequence for reacting systems the heat of reaction term do not have to be considered explicitly in the

energy balance. In ACM the reference state for enthalpy can be adapted (see ACM help file on KBASE).

➤ ***Incorporation of two or more physical property packages***

The structure of the developed custom membrane model can perfectly be used to define two different physical property packages for the flow through the shell and fibre side of the module. In gPROMS the second physical property package can be activated as done for the first property package (see the work process in Appendix E.1). In ACM a second ‘Componentlist’ needs to be defined besides the default ‘Componentlist’ containing the first property package (see ACM help file for more information).

➤ ***Pure component properties***

The calculation of pure component properties besides the calculation of the property of a mixture of these components is difficult to perform. For the calculation of the properties of pure components, a component fraction array needs to be defined for each component separately. For example, if a mixture contains 3 components, a array should be defined for the component fractions in the mixture ([0.1,0.4,0.5]), and 3 arrays for each component separately ([1,0,0]), [0,1,0], [0,0,1]).

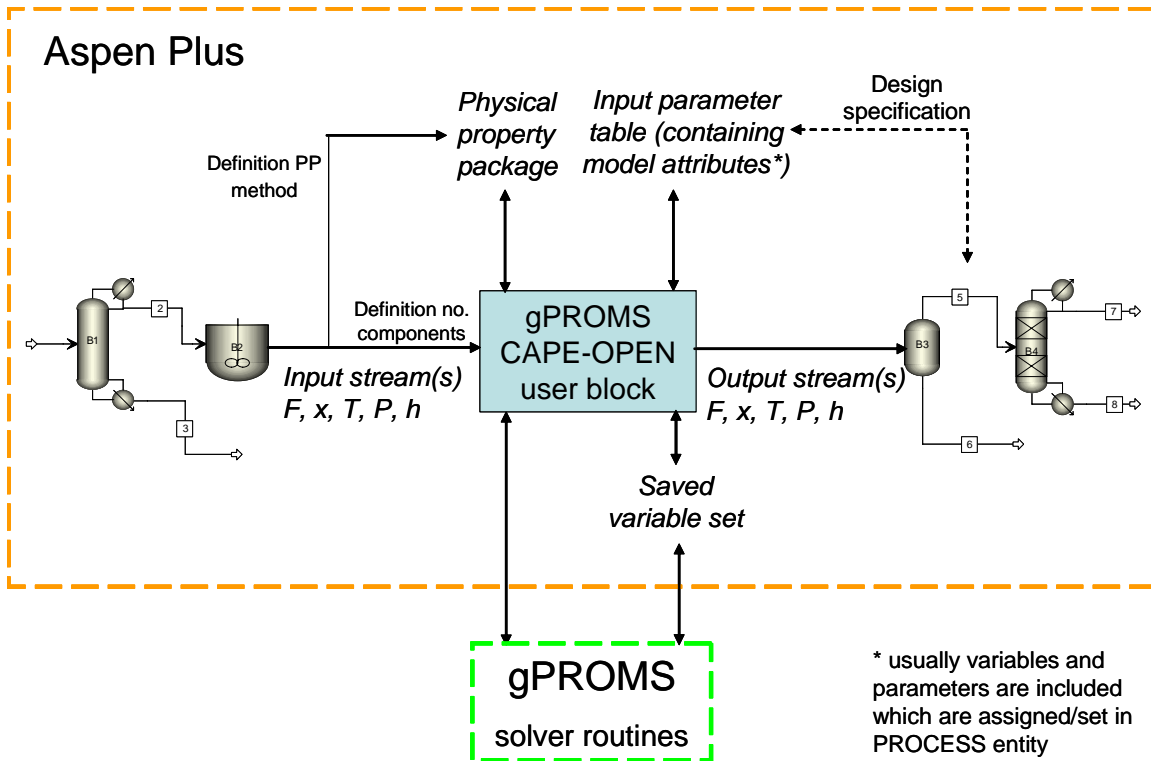
#### **4.3.2 Unit model interface**

In Figure 4.5 and Figure 4.6 schematic pictures are given from a user point of view, showing the connections and exchange of information between the exported custom model and Aspen Plus. The work flow and issues encountered when embedding the exported custom membrane model in Aspen Plus can be summarized as follows:

➤ ***Port connection***

In Aspen Plus model blocks are connected by streams representing the flow of mass or energy. As like the other model blocks, Aspen Plus streams can be connected to the ports of the custom model (see Figure 4.5 and Figure 4.6). The variables representing an Aspen Plus material stream are in SI units. In Table 4.2 the units of these variables are given. The ACM user block in Aspen Plus calculates in metric units (see also paragraph 3.2.2) instead of in SI units, therefore the units of the variables in the port have to be converted, which is done automatically. Also the CAPE-OPEN standards defined for unit model interfacing are different from the base unit of measurements in Aspen Plus. The port connection between Aspen Plus and the gPROMS CAPE-OPEN model block also automatically converts the units into the units defined by the CAPE-OPEN standards and vice versa.

For the custom model developed in gPROMS it is recommended to write the model equations in the same units as defined by the CAPE-OPEN standards for unit model interfacing. Software developers from Process System Enterprise Ltd. are creating an option within the ‘Export to CAPE-OPEN’ functionality, where the user can choose between the basis (mass/molar) of the variables in the port of the custom model block [27].



**Figure 4.5: gPROMS CAPE-OPEN user block in Aspen Plus**

Aspen Plus, like other sequential modular flowsheet simulators, solves a flowsheet by calculating for each model block the outlet conditions on basis of the information in the feed stream(s). This applies also for the custom user block. Before exporting the custom model, the model of the process unit operation in an equation oriented tool should be able to solve with specified values for the variables defined in the input ports representing the Aspen Plus stream. If differential balance equations are used in the custom model (as for the custom membrane model), the boundary conditions are preferred to be defined by the variables in the input ports.

In ACM it is possible to define a condition statement, which applies if a port is not connected with a stream in Aspen Plus. For example, for the developed custom membrane model this statement can be used to define the boundary conditions for the ‘fictional’ sweep stream, if such a stream is not connected to the user block.

**Table 4.2: Units of variables in stream of Aspen Plus and port of custom model block**

	<i>Aspen Plus base unit of measurement</i>	<i>CAPE-OPEN interface</i>	<i>ACM base unit of measurement</i>
Flowrate	<i>kmol/s</i>	<i>mol/s</i>	<i>kmol/hr</i>
Temperature	<i>Kelvin</i>	<i>Kelvin</i>	<i>Celsius</i>
Pressure	<i>Pascal</i>	<i>Pascal</i>	<i>Bar</i>
Specific volume	<i>m<sup>3</sup>/kmol</i>	<i>*** Not exchanged ***</i>	<i>m<sup>3</sup>/kmol</i>
Specific enthalpy	<i>J/kmol</i>	<i>J/mol</i>	<i>GJ/kmol</i>
Component fraction	<i>kmol/kmol</i>	<i>mol/mol</i>	<i>kmol/kmol</i>



➤ **Link with physical property package**

In Figure 4.5 and Figure 4.6 it is shown that the gPROMS CAPE-OPEN user block and the ACM user block have different ways to define the property method used by the custom model in Aspen Plus. For a gPROMS CAPE-OPEN block the physical property method is defined by the stream connected to one of the input ports. The physical property method for an ACM custom model is defined in the user block itself, as for all other model blocks in the model library of Aspen Plus.

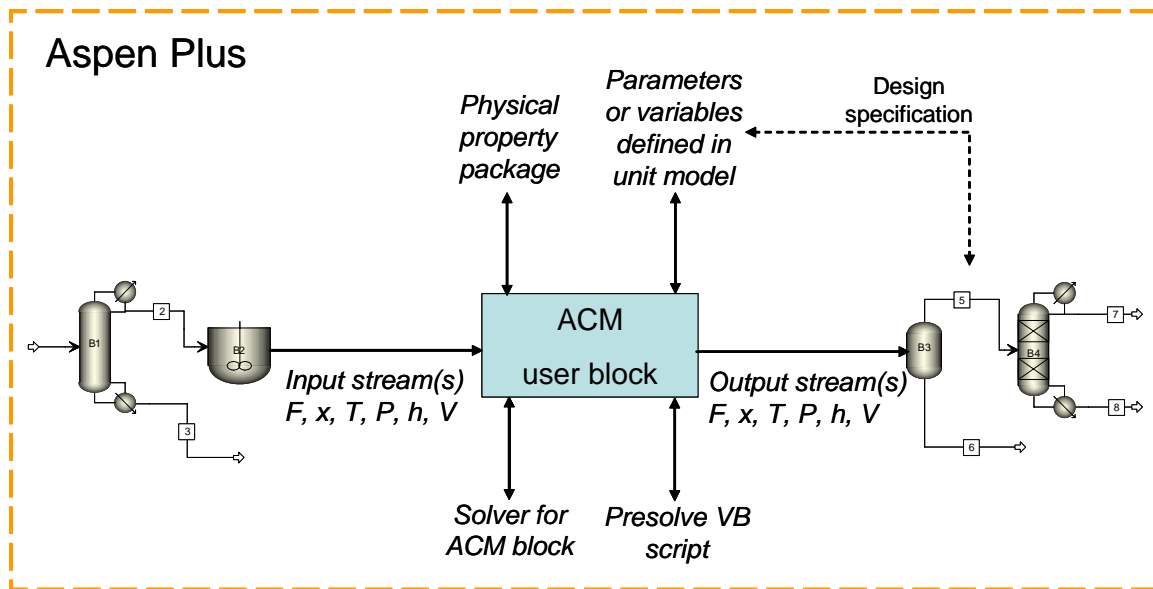


Figure 4.6: ACM user block in Aspen Plus

➤ **Solvers for custom model block in Aspen Plus**

As already reported in Section 4.1 the numerical solvers for an ACM user block are embedded in Aspen Plus, in contradiction to the solvers used for a gPROMS CAPE-OPEN user block, which are solvers of gPROMS included in the model package (see also Figure 4.5 and 4.6). When running a simulation containing a gPROMS CAPE-OPEN block, gPROMS runs on the background to provide its solver routines to the block.

In Aspen Plus version 2004.1/13.2 the user can choose between seven different solvers for an ACM user block, namely SPARSE, DMO, LSSQP, NSOLVE, DSPARSE, DLSSQP and DDMO. For the developed plug flow model and membrane model it is arbitrary which solver to choose. The developed custom plug flow model needs the DMO-solver to give correct results. For the custom membrane model it is recommended to use the DSPARSE-solver [25]. However the results generated in ACM for the membrane model are not equal to the results obtained in Aspen Plus (using the same design specifications and operating conditions). A reason is that solvers in Aspen Plus for an ACM user block are not the same solvers used in ACM. In Aspen Plus, the non-linear solver is used for all groups (non-linear, linear and explicit) of equations, regardless of their type. Aspen Tech support provided a workaround to force the solver in Aspen Plus to take at least one iteration step, which give more accurate results for the exported custom membrane model [25].

Besides, the solvers used in Aspen Plus to calculate the results for an ACM block (using block decomposition) are far from optimal; the time required for the solve the exported custom membrane model in Aspen plus is in the order of minutes.

#### ➤ ***Initialisation set***

As mentioned in Section 3.3 the set of model equations created for the custom membrane model cannot be solved in once, but several intermediate modelling steps are required to approach the final results. To make it possible that also the exported model will converge in Aspen Plus, an initialisation set is exported from ACM and gPROMS together with the custom model. This initialisation set contains the values of all variables near the values of the final result for a specific application.

In gPROMS a ‘saved variable set’ is included in the exported model package as described in the work process for unit model interfacing given in Appendix E.3. For the custom membrane model developed in ACM a visual basic script (PreSolve) is created. This PreSolve script in Aspen Plus runs just before the actual run solving the equations, and set the values of all variables to the values defined in the script (see also the work process for unit model interfacing in Appendix E.4).

As a consequence, the exported custom model can only be used for a very specific application in Aspen Plus. This means also that the criterion given in Section 1.1, stating that the exported custom model should be generic and robust, is not fulfilled. In Section 5.2 a possible solution for this problem is given.

#### ➤ ***Design specification***

Variables and parameters defined in an ACM - or a gPROMS CAPE-OPEN - user block can be connected to other parameters or variables of other model of process unit operations in a sequential modular flowsheet simulator. A well-known example is the design specification in Aspen Plus. For the developed plug flow model, it has been tested to vary a variable of the user block by specifying an outlet condition. For an ACM user block, all the variables which are originally defined in ACM can be chosen. For a gPROMS CAPE-OPEN user block only those variables can be used which are defined as a model attribute in the input parameter table. Currently it is only possible to included variables in the model attribute table in gPROMS (alpha 3.0). In the final release also parameters can be included in the model attribute table [27].

#### ➤ ***Change value for selector parameter***

The custom membrane model is developed in such a way that the model can be used to simulate both co-current and counter-current flow patterns. Also other equipment and design specification are incorporated in the membrane model, as described in Section 3.1. The user can choose the design specification by giving the appropriate value as a description to the selector in gPROMS or the selector parameter in ACM. To develop a generic custom membrane model which can be used for various applications, it is preferred to have this functionality also available in Aspen Plus.

For an ACM custom model in Aspen Plus it is possible to change the value of a selector parameter. For a CAPE-OPEN user block this is not (yet) possible, in spite of the fact that the selector is incorporated as a model attribute in the input parameter table of the gPROMS CAPE-OPEN user block.

➤ ***Mass balance check***

For every block in a sequential modular flowsheet simulator a mass balance check is executed. Since more complicated models can make use of differential equations and numerical techniques are used to solve these equations, a small numerical error can occur in the total mass balance of a custom model block. Therefore it should be prevented that an error is generated in a sequential modular flowsheet simulator if small errors occur in the total mass balance of a custom model.

➤ ***Displaying results of user model block in Aspen Plus***

Besides the variables in the output streams of the gPROMS CAPE-OPEN user block, no other results of the custom model are displayed in Aspen Plus. Especially for distributed parameter models it is useful to show the profiles of some other variables. Therefore it is preferred if a functionality is available inside Aspen Plus to display the results of a (ACM and gPROMS CAPE-OPEN) user model block in graphs and tables.

***Software programme issues for a gPROMS CAPE-OPEN Unit Object block:***

- Only model entity of gPROMS project is exported. Model specifications in the process entity have to be moved to the model entity if possible. The variables which are assigned in the process entity are recommended to incorporate them in the input parameter table.
- It is possible to incorporate only one gPROMS CAPE-OPEN user model in an Aspen Plus flowsheet simulation. This has to do with the fact that the software architecture of gPROMS does not allow the run of two projects simultaneously.
- It is not possible to rename the gPROMS CAPE-OPEN block in the Aspen Plus flowsheet.
- After restarting an Aspen Plus simulation containing a gPROMS CAPE-OPEN block, the \*.gCO file has to be opened again and the results of the block are not restored.
- If input parameters (in the exported model attribute table) are changed in the gPROMS CAPE-OPEN block, Aspen Plus does not consider this as a change in the process flowsheet; 'input changed' is not displayed.
- The reconnection of streams to a gPROMS CAPE-OPEN block not possible.
- Variables defined in the input parameter table of the gPROMS CAPE-OPEN model block, which are defined in gPROMS as an array of the number of components, are not given as an array in Aspen Plus.

***Software programme issues for an ACM user block:***

- Reset of variables in ACM user block not possible

### **4.3.3 Conclusion**

In the previous paragraphs several issues are reported, which were detected during the development of the custom membrane model in equation oriented modelling tools and its implementation as user model block in Aspen Plus. The most important issues encountered are:

- The exported custom membrane model can only be used for a specific application, due to the fact that an initialisation set have to be included in the exported model package to be able to converge the custom model in Aspen Plus.
- Solvers in Aspen Plus for ACM user blocks are far from optimal (slow and not robust). Besides it is arbitrary which solver to choose for which application.
- Aspen Plus 2004.1/13.2 and gPROMS 2.3.6 and alpha 3.0 are not compatible (for both physical property package and unit model interfacing).
- Only one gPROMS CAPE-OPEN user model block can be embedded in an Aspen Plus simulation.

Before starting with this project it was expected that a custom model of a process unit operation developed in ACM can more readily be implemented in Aspen Plus, than a custom model developed in gPROMS and exported as a CAPE-OPEN compatible model. However, due to the results obtained in this section, it is difficult to compare the CAPE-OPEN standards for interfacing with the Aspen Tech interfaces. Both interfaces have its advantages and disadvantages, and they aren't fully developed yet. As many issues for software interoperability are encountered, it can be concluded that the software packages aren't ready yet for industrial application of unit model interfacing. The use of a physical property packages from Aspen Plus in equation oriented modelling tools is better developed and practical applicable.

## 5 Custom model performance in Aspen Plus

*In this chapter the requirements are described and examined for custom models embedded in a sequential modular flowsheet simulator. In the first section the requirements are defined. Based on the implemented functionality for unit model interfacing (see Chapter 4) an improved approach for model initialisation was required and a potential solution is presented. This approach is tested in equation oriented modelling tools for the custom membrane model described before. In the last section potential (interface) functionalities are discussed so that the proposed initialisation approach can be used for exported custom models in an Aspen Plus flowsheet simulation.*

### 5.1 Custom model performance

The recent implementation of interface functionalities in process simulation tools enables the usage of a custom model of a process unit operation developed in equation oriented modelling tools within a sequential modular flowsheet simulator. Besides the availability of open interfaces another criterion is mentioned in Section 1.1 for successful implementation of a custom model in a sequential modular flowsheet simulator, namely the exported custom model should have the same performance as the models of process unit operations available in the model library of these programs.

Custom models of process unit operations are usually developed for a specific application and a specific range of operating conditions. The model is often build from scratch and extended step by step till the model objective is satisfied and explicable results are obtained. Custom models are usually hard to initialise if the set of equations has to converge from default values defined for the variables, or in case the system deviates from the conditions or application for which the model has been developed originally. This leads to a practice that frequently only the person who developed the custom model knows how to use and adapt it.

Since it is often not easy to initialise a custom model, the software interface functionalities provide the option to include an initialisation set with the model export. As described in Section 4.3, this initialisation set contains a set of values for all variables which approach the final results for a specific condition. The exact application however is not always known if an exported custom model is used for process design in flowsheet simulators. It can happen that the exported custom model is used for another application than for which it has been developed originally, and has therefore problems to be initialised properly. If possible, the custom model and its initialisation set needs to be adapted then in the equation oriented modelling tool and exported again. However, it can take a lot of time and effort to adapt the custom model for a new application. The users of a sequential modular flowsheet simulator do not always have sufficient knowledge of the custom model and/or the equation oriented modelling tools to make the required modifications themselves.

Preferably, the custom models needed in a process flowsheet simulation can be used just like the models which are available in the library of sequential modular flowsheet simulators. Consequently, the exported custom model should be able to:

- handle a wide range of process and feed conditions,
- be used for different equipment and design specifications,
- handle any number of components as well as different sets of components, and
- handle any physical property method as available in the physical property package used by the sequential modular flowsheet simulation software.

With other words, the custom model has to be generic, robust and proper initialisation should be guaranteed for all cases. The use of an initialisation set – based on a specific (different) application - is not sufficient for this purpose.

Initialisation of a custom model over a broad range of operation conditions, as well as for different applications, needs a structured approach. However, in open literature hardly any information is found about the initialisation of custom models for process unit operations. In some articles, issues mentioned are:

- “At the time of model development it is useful to be able to use parameters and results of less detailed models. This will allow the use of consistent parameter values and can simplify the initialisation of complex models [28]”.
- “The different versions of a model being built during a modelling project for model initialisation, model refinement, or different application context need to be documented together with their interrelationships [29]”.
- “Reasonable initial values are provided by hand or computed, in order to improve initialisation and convergence of later simulation runs [30]”.

It seems that the initialisation of a custom model is more an engineering practice/art, than that a systematic (academic) methodology is available.

A possible approach to initialise a custom model of a process unit operation properly is to approach the final model step by step. These initialisation steps (or levels) have to be chosen in such a way that the custom model can be applied for a larger range of applications. By embedding the initialisation steps in the custom model itself, before exporting it to a sequential modular flowsheet simulator, the future user of the exported model will not be concerned with model initialisation. All is already taken care of by the model developer. To shown this approach, an initialisation structure for the developed custom membrane model is proposed and presented in the next section.

## **5.2 Application to custom membrane model**

In Chapter 3 a start is already made to develop a generic custom model for a gas separation membrane unit in sequential modular flowsheet simulators. The custom model developed can be used for different equipment and design specifications, see Section 3.1. The proposed initialisation structure should be able to initialise the set of model equations properly for all these specifications. Besides also changes in system and process conditions have to be intercepted by the initialisation structure.

### 5.2.1 Initialisation structure

In Table 5.1 a possible initialisation structure for the developed custom membrane model is presented. It should be mentioned that the number of variables and equations in each step are equal. For each modelling step a short explanation is given:

➤ **Level 1**

The first level of the initialisation structure is used to set the values of the variables equal to the inlet conditions. There is no flow of components through the membrane.

➤ **Level 2**

A pressure and temperature profile along the axial length of the membrane unit does not exist in this level. Consequently, the component fractions used in the equation to describe the transport of components through the membrane are the only variables in this equation which are no constants.

Especially when the pressure drop for the flows through shell and fibre side is large, this level is required to initialise the custom membrane model. The equation for the mass transport through asymmetric membranes is equal to the equation for symmetric membranes.

➤ **Level 3**

The pressure drops for the flow through the shell side and fibres are taken into account from this level on. The energy balance equation to calculate a temperature profile along the axial length is still left out of consideration. The equation which describes the transport of components through the membrane is for symmetric membranes equal to the equation used in level 2. For asymmetric membranes the equation for the transport through the membrane is given as function of the component fraction inside the membrane, as explained in Section 3.4.

➤ **Level 4**

In level 4 the differential balance equation for energy in the sub-models describing the flow through fibre and shell side is added. The energy balance is not used in an earlier level, because the custom membrane model has difficulties to converge if both pressure and temperature differ too much from the final solution.

➤ **Level 5**

The permeability coefficient used in the equation describing the transport of components through the membrane is a function of temperature.

A selector in gPROMS or a selector parameter in ACM is created which contains five values defined for level 1 to 5. The results of each level have to be saved and loaded for the run of the next level. To go to the next level, the value of the selector can be changed manually, however it is also possible to run the 5 levels after each other without the intervention of the user. In gPROMS a SCHEDULE in the PROCESS entity is created and in ACM a visual basic script is defined. In Appendix F the implementation of these automated simulation processes in gPROMS and ACM is given.

**Table 5.1: Initialisation structure for custom membrane model**

Level no.	Balance equations of flow through fibres and shell side	Transport of components through membrane	
		Symmetric membrane	Asymmetric membrane
<b>1</b>	Mass: $F_i = F_{in}$ in ACM: $\frac{\partial F_i}{\partial z} = 0$ Momentum: $P_i = P_{in}$ Energy: $T_i = T_{in}$	$J_{i,S} = 0$	$J_{i,S} = 0$
<b>2</b>	Mass: $\frac{\partial c_i}{\partial t} = -\frac{1}{L} \frac{\partial F_i}{\partial z} + \beta J_i = 0$ Momentum: $P_i = P_{in}$ Energy: $T_i = T_{in}$	$J_{i,S} = \frac{Q_{0i}}{\delta^m} (X_{i,F} P_F - X_{i,S} P_S)$	$J_{i,S} = \frac{Q_{0i}}{\delta^m} (X_{i,F} P_F - X_{i,S} P_S)$
<b>3</b>	Mass: $\frac{\partial c_i}{\partial t} = -\frac{1}{L} \frac{\partial F_i}{\partial z} + \beta J_i = 0$ Momentum: $\frac{\partial M_{holdup}}{\partial t} = -\frac{1}{L} \frac{\partial P}{\partial z} - \frac{1}{L} \frac{\partial M_{flux}}{\partial z} - f_v = 0$ Energy: $T_i = T_{in}$	As level 2	Feed enters shell side: $J_{i,S} = \frac{Q_{0i}}{\delta^m} (X_{i,M} P_F - X_{i,S} P_S)$ Feed enters fibre side: $J_{i,S} = \frac{Q_{0i}}{\delta^m} (X_{i,F} P_F - X_{i,M} P_S)$
<b>4</b>	Mass: $\frac{\partial c_i}{\partial t} = -\frac{1}{L} \frac{\partial F_i}{\partial z} + \beta J_i = 0$ Momentum: $\frac{\partial M_{holdup}}{\partial t} = -\frac{1}{L} \frac{\partial P}{\partial z} - \frac{1}{L} \frac{\partial M_{flux}}{\partial z} - f_v = 0$ Energy: $\frac{\partial U}{\partial t} = -\frac{1}{L} \frac{\partial E}{\partial z} + \beta q = 0$	As level 2	AS level 3
<b>5</b>	As level 5	$J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,F} P_F - X_{i,S} P_S)$	Feed enters shell side: $J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,M} P_F - X_{i,S} P_S)$ Feed enters fibre side: $J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,F} P_F - X_{i,M} P_S)$



### 5.2.2 Case study to test robustness initialisation structure

The robustness of the initialisation structure developed in the previous paragraph for the custom membrane model has been tested. First a base case (counter-current, feed enters on shell side and symmetric membrane type) is defined in Appendix G. Variations compared to the base case are applied. The tested variations are divided into discrete changes, module size changes and operation changes.

#### *Discrete changes*

In Section 3.1 several options are created within the custom model for the variation in equipment and design specifications of the gas separation membrane unit. In Table H.1 and Table H.2 of Appendix H the test results are given for a variation in one of the specifications compared to the base case defined. Also the number and set of component is changed.

Table 5.2 shows the initialisation level which is required for a particular simulation. Without this level the model will not converge. For all the discrete changes which have been tested, level 3 is the critical initialisation level, except if the model is used to simulate a system for an asymmetric membrane type than level 2 is required.

**Table 5.2: Initialisation level required for tested simulation (rating or design)**

<b><i>Variation of the ...</i></b>	<b><i>Change to</i></b>	<b><i>Rating</i></b>	<b><i>Design</i></b>
Base case		3	3
Operation mode	Co-current	3	3
Feed side	Fibre	3	3
Sweep stream	With	3	3
Membrane type	Asymmetric	2	2
Number & set of components	8	3	3

These tests demonstrate that the set of model equations cannot be solved if both pressure and temperature profiles along the axial length of the membrane unit are taken into account from the first initialisation level on.

For an asymmetric membrane the driving force for the transport of components through the membrane is different than for symmetric membranes (equation 3.2). The fraction of components inside the membrane material is included in this equation. The set of model equations has difficulties to converge if no initial guess is given for this fraction. It is chosen to use the transport equation of a symmetric membrane to initialise the simulation of an asymmetric membrane.

It should be mentioned that for each test the number of discretisation intervals is increased to examine if the critical initialisation level is indeed necessary to initialise the set of model equation or can be left out due to a better accuracy of the numerical method. Nevertheless, all the critical initialisation levels are still required even if the number of discretisation intervals is increased.

#### *Module size changes*

For different applications the membrane area has to be adapted often. When the rating method for simulation is used, the membrane area is dependent on various equipment specifications, but can among other things be changed by adapting the length of the module. In Table H.3 of Appendix H the results are given for the simulation (with the

rating method) of a membrane unit for which the length varies from very small to larger values. If the length of the module is larger than 1.5 meters the model and its initialisation structure are not able anymore to converge. Two reasons can be given for this observation:

- For ideal membrane systems, it is expected that the driving force for transport through the membrane approaches zero if the length of the membrane area is increased to infinity. However, the set of model equations do not reach equilibrium for large systems, since a driving force for transport of component through the membrane is always presented; the pressure drop of the permeate side (inside fibres) decreases more than the pressure drop of the retentate side (shell side). Increasing the length (or membrane area) causes that the pressure difference between permeate side and retentate side will also increase and equilibrium is never reached. Tests performed for a co-current system demonstrate this, see Table H.4 in Appendix H. It is important to mention that hollow fibre membrane units are commercial available till one meter.
- However, for the counter-current systems tested (Table H.3 of Appendix H) such a large pressure drop is not reached. So, another reason is more likely for the failure in simulation of large membrane areas for counter-current systems. It is believed that counter-current systems have difficulties to converge due to the way of solving the set of model equations. It is tried to approach the final solution of the simulation with very small steps for the transport of component through the membrane, but even then the set of model equations cannot be solved. Increasing the number of discretisation intervals helps, but if the length is increased afterwards, the model fails again. An exact reason for this failure cannot be given, however this proves that the proposed initialisation structure is not responsible for the fact that the set of model equations cannot be solved for large membrane areas.

It seems that the maximum membrane area which can be simulated for the defined base case is depended on the flowrate of the feed stream. To examine this dependence the maximum membrane area is determined for different flowrates of the feed stream using the design method. In Table 5.3 the results are given. It shows that the model is only capable to converge if the stage cut is smaller than 70%. Based on the results obtained it is recommended to use the developed custom membrane model for simulation of membrane systems where the stage cut is smaller than 65 %.

**Table 5.3: Maximum membrane area and stage cut for different flowrates of the feed stream**

<b>Flowrate feed (mol/s)</b>	<b>Max. membrane area (m<sup>2</sup>)</b>	<b>Max. length (m)</b>	<b>Max. stage cut (%)</b>
0.028	2	2.0	67.2
0.139	10	2.0	67.2
0.208	15	2.0	67.2
0.278	21	2.1	68.1
0.247	26	2.1	67.9
0.417	31	2.1	67.8

### ***Operation changes***

If an exported custom model is placed in a flowsheet simulation, the conditions of the feed stream to the membrane unit often vary. In Table H.5, Table H.6 and Table H.7 of Appendix H the results are given if respectively the pressure ratio, the component fractions and temperature of feed stream are changed compared to the base case. In Table 5.4 the results are summarized by giving the lower and upper bound for which the operation conditions have been tested and the accompanying required initialisation level.

**Table 5.4: Operation changes, lower- and upper bound and critical initialisation level**

	<b><i>Lower bound</i></b>	<b><i>Critical initialisation level</i></b>	<b><i>Upper bound</i></b>	<b><i>Critical initialisation level</i></b>
<b><i>Pressure ratio</i></b>	1 (Feed = 70 bar, Sweep = 69.5 bar)	3	35 (Feed = 70 bar, Sweep = 2 bar)	2
<b><i>Component fraction</i></b>	0	3	0.75	3
<b><i>Temperature</i></b>	275 K	3	400 K	3

➤ ***Pressure ratio of feed and sweep stream***

Since the driving force for the transport of a component through the membrane is determined by the partial pressure difference between shell and fibre side, it has been tested if the model and the initialisation structure can handle different pressure ratios (defined as the ratio of the pressure of the feed stream to the pressure of the sweep stream). The upper bound given in Table 5.4 is imposed by the pressure drop inside the fibres. A pressure lower than 2 bar cannot be specified, because the pressure of the permeate stream will then be lower than zero. It is noticed that because of the relatively large pressure drop, level 2 is required to initialise the set of model equations for large pressure ratios.

➤ ***Component fraction in feed stream***

The lower bound of the component fraction in the feed stream is determined physically. These tests proved that also trace components can be taken into account by the custom membrane model and its initialisation structure. The upper bound is restricted by the value for transport of component through the membrane. If the component fraction is further increased the flux through the membrane becomes larger than the feed stream, resulting in an infeasible system. For as well as the lower bound as the upper bound level 3 is the critical initialisation level.

➤ ***Temperature of feed stream***

Due to the Arrhenius relation used to describe the temperature dependence of the permeability coefficient, the transport of components through the membrane increases if the temperature is risen. The maximum temperature which can be solved by the model is bounded by the stage cut. For the membrane system tested the temperature of the feed stream can be increased till 400 K. Above this temperature the stage cut hits its bound (~70%) and cannot be solve anymore.

It can be concluded from the test performed that the proposed initialisation structure can be used to initialise the custom membrane model for a larger variety of specifications and operating conditions than if the initialisation structure is not used.

Furthermore, it appears that the set of model equation cannot be used to simulate membrane processes for which the stage cut is larger than 65%. More tests have to determine the exact constraint of the stage cut and its dependence with other variables. Also the reason for this constraint must be investigated.

### **5.3 Initialisation structure in Aspen Plus**

The custom membrane model including the proposed initialisation structure is exported to Aspen Plus. The functionality of running the 5 levels of the initialisation structure after each other without the intervention of the user is not available for both an ACM and a gPROMS CAPE-OPEN user block in Aspen Plus. The 'run' statement in a visual basic script cannot be used for a 'PreSolve' script in Aspen Plus and the SCHEDULE in the PROCESS entity of a gPROMS project cannot be exported together with the custom model.

Consequently, for the exported custom membrane model the selector (parameter) must be changed manually. However, for a gPROMS CAPE-OPEN block it is not possible to change the value of a selector (see Section 4.3). Thus, only for the ACM user block the proposed initialisation structure in Aspen Plus is tested. Together with the provided workaround for the solver (also described in Section 4.3) the ACM user block, containing the membrane model and its initialisation structure, is able to give the same results as in ACM without the need of an exported initialisation set for a specific application. Due to the numerical solver used for the ACM custom membrane model in Aspen Plus, which is not optimal, it takes more than half an hour to go through the 5 levels of the initialisation structure until the complete model is converged (for the membrane model in ACM it takes less than one minute).

To be able to use a similar kind of initialisation structure for custom models in Aspen Plus, the following functionalities are preferred:

- Automation of the initialisation structure, in order to run the different levels of the structure after each other without the intervention of the user.
- A 'smart' solver for the custom model. Once an exported custom model is converged in a flowsheet simulation with the help of the complete initialisation structure, possibly not all initialisation levels are needed anymore for a next simulation run or during iterations for flowsheet model convergence (recycle loops, design specifications, optimization of flowsheet, etc.). To save time for flowsheet convergence, the 'smart solver of the custom model should be able to determine the initialisation level to start with and be able to go back one or more levels if the initialisation fails.

## 6 Conclusion and recommendations

*This chapter concludes the work that has been presented in this thesis. Also recommendations for future research are suggested.*

### 6.1 Conclusion

Equation oriented process modelling tools (gPROMS and Aspen Custom Modeler) have been used in this project to create a custom model of a gas separation membrane unit operation. This custom model is exported and interfaced with the Aspen Plus flowsheet simulator. The objective is that the custom membrane model can be used just like the library models available in Aspen Plus: i.e they can be used for a variety of applications, different components sets, any physical property method, and over a broad range of operating conditions. Several conclusions can be drawn from the research performed:

#### *Development of generic custom membrane model in gPROMS and ACM*

The custom model for a gas separation membrane unit has been developed for a hollow fibre membrane module. In gPROMS as well as in ACM the foundation has been laid for a generic and robust custom membrane model in Aspen Plus, due to the following software or model features:

➤ *Hierarchical model structure*

The custom membrane model has been developed by connecting three sub-models with each other. The sub-models represent the flow through shell side, through fibre side and through the membrane. The flow through fibre and shell side is assumed to be a plug flow described by one-dimensional balance equations. The flow through the membrane is described by a transport equation using the solution-diffusion theory.

This hierarchical model structure enhances the creation of several build-in equipment and design specifications for which a user can choose the type of application. Due to the hierarchical structure, the options for the flow pattern (co- or counter-current), the side where the feed stream enters (shell – or fibre side) and the type of membrane (symmetric or asymmetric) are incorporated easily in the custom membrane model by making use of selectors.

➤ *Usage external physical property package from Aspen Plus*

To be able to choose any physical property method in Aspen Plus for the calculation of thermodynamic and physical properties of the membrane unit, the calls to a physical property package has been incorporated in the custom model developed in the equation oriented modelling tools. The interface functionalities for physical property packages made it practically possible to develop the custom membrane model in gPROMS and ACM with an external package from Aspen Plus.

➤ *Incorporated model initialisation structure*

To be able to use the custom model for a broad range of operation conditions, as well as for different applications, proper initialisation should be guaranteed for all cases. The custom membrane model should be able to converge from the default

values defined for all the variables used. The language of equation oriented modelling tools has been used to set up an automated initialisation structure for the developed custom membrane model. This structure contains several initialisation levels in which the complexity of the model is increased step by step. The number of variables and equations however stays the same for each level. Tests proved that the proposed initialisation structure for the developed custom membrane model enables proper initialisation of the set of model equations for different applications and a broad range of process conditions.

### ***Interfacing and embedding of custom membrane model in Aspen Plus***

The developed custom membrane model has been used to examine the software interface functionalities and its performance in Aspen Plus.

Issues concerning the interface functionalities which were encountered during this project are:

- *For gPROMS CAPE-OPEN user block*
  - It is possible to incorporate only 1 gPROMS CAPE-OPEN user block simultaneously in an Aspen Plus simulation, because of a limitation in the software architecture of gPROMS (multiple “processes” can not run at the same time)
  - Due to a different interpretation of the CAPE-OPEN standards, Aspen Plus version 2004.1 and gPROMS version 2.3.6 or alpha 3.0 are incompatible.
- *ACM user block*
  - The numerical solver of ACM is not exported together with the set of model equations, in contradiction to an exported gPROMS model package. The solvers for an ACM user block are embedded in Aspen Plus. However these solvers are not optimal; it is arbitrary which solver to choose for a particular custom model, the solvers are very slow and not robust.

The implemented features described above for a generic custom membrane model are not completely supported if the model is exported for use in Aspen Plus. The software functionalities required to use the automated initialisation structure (Visual Basic script in ACM and SCHEDULE section of PROCESS entity in gPROMS) are not supported by the exported custom model in Aspen Plus. Therefore the values defined for the selector of the initialisation structure must be adapted manually. Besides, for a gPROMS CAPE-OPEN user block it is not possible to change values of selectors defined in a gPROMS model, therefore the build-in options for equipment and design specification, and the initialisation level cannot be changed in Aspen Plus.

Due to the issues encountered, it is clear that the software functionalities for unit model interfacing, defined by CAPE-OPEN standards as well as by integrated Aspen Tech standards, are still in its infancy. These functionalities are not sufficiently developed yet to be used for large scale industrial practice.

## 6.2 Future research

Related directions for future research in the field of custom (membrane) modelling and interface functionalities of process simulation software are now discussed.

### *Custom model of gas separation membrane unit*

- The custom model of a gas separation membrane unit developed in this project has served as a case study to improve the ability for unit model interfacing. Nevertheless realistic simulation results are obtained comparable with simulations reported in literature. To use this model for process design and simulation activities more test needs to be performed to investigate the results given by the model. Preferably the simulation results need to be compared with industrial operating hollow fibre membrane units.  
Also the exact boundaries for which the custom membrane model is capable to give feasible results has to be determined. From the test performed in this project it was observed that the custom membrane model can only be used if the stage cut is smaller than 65%.
- Moreover, the developed custom membrane model can be extended. Other transport models as well as concentration polarisation can be incorporated in the custom model. The model can also be applied to other separations such as pervaporation.  
Besides process simulation, the custom membrane model can be adapted to be used for process control, parameter estimation and optimization activities.
- The sub-model developed for the flow of fluid through the fibre and shell side of the hollow fibre module (plug flow) can easily be used to develop models for other process unit operations, such as adsorption columns and plug flow reactors.

### *Interface functionalities*

- This project shows the importance of testing new software functionalities as a user. Although the functionalities for software interoperability do exist in the latest releases of process simulation tools, they are not sufficient to be used on a large scale yet. As a result of the issues reported in this thesis, software vendors are informed about the defects and shortcomings of the open interface functionalities. Continuous testing as a user is required to exert pressure on software vendors to improve the open interface functionalities for their products.
- The development of a generic and robust custom model of a process unit operation requires another way of thinking for model developers than the current practice of custom modelling. Among other things, initialisation of the custom model has to be taken into account during its development. Since a practical methodology for initialisation is not found in literature, a potential solution is proposed consisting of a step by step initialisation structure. However, more research towards a structured methodology for model initialisation is recommended to support model developers to set up a generic and robust custom model of a process unit operation in equation oriented modelling tools.

- Besides a methodology to develop generic custom models with proper initialisation, software tools should support the functionalities which are necessary to embed such a custom model in sequential modular flowsheet simulators. Software vendors, academic institutions and the Co-LaN organisation need to continue to work together to find practical solutions.



## Literature list

- [1] H. Pingen, *Breakthrough in process simulation through CAPE-OPEN interoperability standard*, NPT Procestechologie, 1, febr. 2005, 11-13
- [2] J.D. Seader, E.J. Henley, *Separation process principles*, Chapter 14, John Wiley & Sons Inc, 1998
- [3] M. Mulder, *Basic principles of membrane technology*, Kluwer Academic Publishers, Dordrecht, 1991
- [4] R.W. Baker, *Future directions of membrane gas separation technology*, Ind. Eng. Chem. Res. Vol. 41, pp. 1393-1411, 2002
- [5] R.W. Baker, *Membrane Technology and Applications*, Second Edition, John Wiley & Sons, Ltd., 2004
- [6] <http://www.mtrinc.com/Pages/FAQ/faqs.html>, Membrane Technology and Research Inc., visit site on September 15<sup>th</sup> 2005
- [7] J.G. Wijmans, R.W. Baker, *The solution-diffusion model: a review*, Journal of Membrane Science, Vol. 107, pp. 1-21, 1995
- [8] S.T. Hwang, *Non-equilibrium thermodynamics of membrane transport*, AIChE Journal, Vol. 50, No.4, pp. 862-870, 2004
- [9] J.A. Wesselingh and R. Krishna, *Mass transfer in multi-component mixtures*, Delft University Press, 2000
- [10] C.Y. Pan, *Gas separation by high-flux, asymmetric hollow-fiber membrane*, AIChE Journal, Vol. 32, No. 12, pp. 2020-2027, 1986
- [11] R. Rautenbach, R. Knauf, A. Struck, J. Vier, *Simulation and design of membrane plants with AspenPlus*, Chem. Eng. Technol., Vol. 19, pp. 391-397, 1996
- [12] S. Tessendorf, R. Gani, M.L. Michelsen, *Modeling, simulation and optimization of membrane-based gas separation systems*, Chemical Engineering Science, Vol. 54, pp. 943-955, 1999
- [13] J.I. Marriott, *Detailed modelling and optimal design of membrane separation systems*, PhD Thesis, University of London, 2001
- [14] J.I. Marriott, E. Sorensen, I.D.L. Bogle, *Detailed mathematical modelling of membrane modules*, Computers and Chemical Engineering, Vol. 25, pp. 693-700, 2001
- [15] J.I. Marriott, E. Sorensen, *A general approach to modelling membrane modules*, Chemical Engineering Science, Vol. 58, pp. 4975-4990, 2003
- [16] R.A. Davis, *Simple gas permeation and pervaporation membrane unit operation models for process simulators*, Chem. Eng. Technol., Vol. 25, No. 7, pp. 717 – 722, 2002
- [17] T. Brinkmann, M. Dijkstra, K. Ebert, K. Ohlrogge, *Improved simulation of a vapour permeation module*, Journal of Chemical Technology and Biotechnology, Vol. 78, pp. 332-337, 2003

- [18] M.H.M. Chowdhury, X. Fend, P. Douglas, E. Croiset, *A new numerical approach for a detailed multicomponent gas separation model and AspenPlus simulation*, Chem. Eng. Technol., Vol. 28, No. 7, pp. 773-782
- [19] gPROMS website: “What is an appropriate discretisation method for a distributed system?”
- [20] P.S. Banks, K.A. Irons, M.R. Woodman, *Interoperability of process simulation software*, Oil & Gas Science and Technology – Rev. IFP, Vol. 60, No. 4, pp. 607-616, 2005
- [21] <http://www.colan.org>, Official website of The CAPE-OPEN Laboratories Network, visited: 30 September 2005
- [22] <http://www.psenterprise.com/gproms/brochures/gocapeopen.pdf>, *gO:CAPE-OPEN, the gPROMS object for CAPE-OPEN*, visited: 30 September 2005
- [23] Presentation given by Michel Pons, *Running a gPROMS reactive absorption model inside Aspen Plus*, PSE-User Meeting April 20-21, 2004
- [24] DOW ACM/Reactions Toolkit, Training Manual, 2005 AspenTech, Aspen Technology, Inc.
- [25] Personal communication with E. Lejeune (Aspen Technology Inc.)
- [26] J.A. Jara, A. Garea, J.A. Irabien, *Simulation of o-xylene oxidation into phthalic anhydride: rigorous multi-tubular catalytic reactor modelling and exportation into the process flowsheet*, European Symposium on Computer Aided Process Engineering (ESCAPE) – 15, L. Puigjaner and A. Espuna (Editors), Elsevier Science B.V., 2005
- [27] Personal communication with D. Rethinam (Process Systems Enterprise Ltd.)
- [28] M. Eggersmann, L. von Wedel, W. Marquardt, *Management and reuse of mathematical models in the industrial design process*, Chem. Eng. Technol., Vol. 27, No. 1, pp. 13-22, 2004
- [29] M. Jarke, W. Marquardt, *Design and evaluation of computer-aided process modelling tools*, In: J.F. Davis, G. Stephanopoulos, V. Venkatasubramanian (Eds.): Intelligent systems in process engineering, AIChE Symposium, Ser. 312, Vol. 92, pp. 97-109, 1996, Found on: <http://www.lpt.rwth-aachen.de/Publication/Techreport.php>
- [30] B.A. Foss, B. Lohmann, W. Marquardt, *A field study of the industrial modeling process*, J. Proc. Cont., Vol. 8, No. 5&6, pp. 325-338, 1998

## Nomenclature

$A_{cross}$	Cross sectional area	$m^2$
$A_{membrane}$	Membrane area	$m^2$
$c_i$	Molar concentration of component $i$	$mol/m^3$
$D_{inner}$	Inner diameter fibre	$m$
$D_{outer}$	Outer diameter fibre	$m$
$D_{shell}$	Inner shell diameter module	$m$
$E$	Energy flux	$J/s$
$E_{a_i}$	Activation energy of component $i$	$J/mol$
$F_i$	Molar flux of component $i$	$mol/m^2/s$
$F_{feed}$	Molar flowrate feed stream	$mol/s$
$F_{permeate}$	Molar flowrate permeate stream	$mol/s$
$F_{total}$	Total molar flux	$mol/m^2/s$
$f_v$	Friction factor	$Pa/m = kg/m^2s^2$
$H$	Specific molar enthalpy	$J/mol$
$J_i$	Molar or mass flux of component $i$ through membrane	$mol/m^2s$ or $kg/m^2s$
$K$	Frictional parameter	$m^2$
$L$	Module length	$m$
$M_{flux}$	Momentum flux	$kg/m^2/s \cdot m/s$
$M_{holdup}$	Momentum holdup	$kg/m^3 \cdot m/s$
$M_{weight}$	Molar weight	$10^{-3} kg/mol$
$N$	Number of fibres	-
$p_i$	Partial pressure of component $i$	$Pa$ or $bar$
$P$	Pressure	$Pa$ or $bar$
$P_{feed}$	Pressure feed stream	$Pa$ or $bar$
$P_{sweep}$	Pressure sweep stream	$Pa$ or $bar$
$q$	Energy flux through membrane	$J/m^2s$
$Q_i$	Permeability of component $i$	<i>Various</i>
$\overline{Q}_i$	Permeance of component $i$	<i>Various</i>
$Q_{i0}$	Permeability of component $i$ at standard temperature	<i>Various</i>
$Q_{i,j}$	Matrix of phenomenological permeabilities	<i>Various</i>
$R$	Ideal gas constant	$J/mol/K$
$t$	Time	$s$

$T$	Temperature	$K$
$T_0$	Standard temperature	$K$
$T_{retentate}$	Temperature retentate stream	$K$
$u_i$ or $u_j$	Velocity of component $i$ or $j$	$m/s$
$U$	Internal energy	$J/m^3$
$v$	Velocity	$m/s$
$X_i$	Molar fraction of component $i$	$mol/mol$
$z$	Axial distance along module	$m$

### ***Greek symbols***

$\beta$	Geometric factor	$l/m$
$\delta^m$	Membrane thickness	$m$
$\phi$	Pressure ratio	-
$\varphi$	Stage cut	-
$\lambda$	Ratio cross sectional area fibre and shell side	-
$\mu$	Viscosity	$Pa \cdot s$
$\rho$	Molar density of mixture	$mol/m^3$
$\rho_{unit}$	Packing density	$m^2/m^3$
$\xi_{i,j}$	Friction coefficient between $i$ and $j$	$Ns/mol/m$
$\xi_{i,M}$	Friction coefficient between $i$ and membrane	$Ns/mol/m$

### ***Subscripts***

$F$	Fibre side	-
$i$	Component $i$	-
$j$	Component $j$	-
$M$	Inside membrane	-
$S$	Shell side	-

## Appendix A Mathematical model for gas separation membrane unit

### A.1 Equations in main model (for membrane geometry and characteristics)

#### Geometric variables

##### *Volume of fibre and shell side*

$$V_F = A_{cross,F} L \quad (\text{A.1})$$

$$V_S = A_{cross,S} L \quad (\text{A.2})$$

##### *Geometric parameter for fibre and shell side*

$$\beta_F = \frac{A_{membrane}}{N \cdot V_F} \quad (\text{A.3})$$

$$\beta_S = \frac{A_{membrane}}{V_S} \quad (\text{A.4})$$

#### Membrane characteristics

##### *Stage-cut*

$$\varphi = \frac{F_{permeate}}{F_{feed}} \quad (\text{A.5})$$

##### *Pressure ratio*

$$\phi = \frac{P_{feed}}{P_{sweep}} \quad (\text{A.6})$$

##### *Packing density*

$$\rho_{unit} = \frac{A_{membrane}}{(A_{cross,S} + A_{cross,F}) L} \quad (\text{A.7})$$

##### *Ratio cross sectional area fibre and shell side*

$$\lambda = \frac{A_{cross,F} \cdot N}{A_{cross,S}} \quad (\text{A.8})$$

### **Friction factors**

$$f_{v,F} = \frac{\mu_F v_F}{K_F} \quad \text{with } K_F = \frac{D_{inner}^2}{32} \quad (\text{A.9})$$

$$f_{v,S} = \frac{\mu_S v_S}{K_S} \quad \text{with } K_S = 1 \cdot 10^{-11} \quad (\text{A.10})$$

### **For rating method**

#### **Cross sectional area**

$$A_{cross,F} = \frac{1}{4} \pi D_{inner}^2 \quad (\text{A.11})$$

$$A_{cross,S} = \frac{1}{4} \pi D_{shell}^2 - N \frac{1}{4} \pi D_{outer}^2 \quad (\text{A.12})$$

#### **Membrane area and thickness**

$$A_{membrane} = N \pi D_{inner} L \quad (\text{A.13})$$

$$\delta^m = D_{outer} - D_{inner} \quad (\text{A.14})$$

### **For design method (fictional)**

$$N = 1 \quad (\text{A.15})$$

$$D_{inner} = 100 \cdot 10^{-6} \quad (\text{A.16})$$

$$D_{outer} = 1 \quad (\text{A.17})$$

$$D_{shell} = 1 \quad (\text{A.18})$$

## ***A.2 Plug flow model for flow through fibre and shell side***

### **Component balance**

$$\frac{\partial c_i}{\partial t} = -\frac{1}{L} \frac{\partial F_i}{\partial z} + \beta J_i \quad z \in (0,1] \text{ or } z \in [0,1) \quad (\text{A.19})$$

where:

$$F_{total} = \sum F_i \quad (\text{A.20})$$

$$F_i = v c_i \quad (\text{A.21})$$

$$F_i = X_i F_{total} \quad (\text{A.22})$$

$$\rho = \sum c_i \quad (\text{A.23})$$

$$\rho = f(T, P, X_i) \quad (\text{A.24})$$

### **Momentum balance**

$$\frac{\partial M_{holdup}}{\partial t} = -\frac{1}{L} \frac{\partial P}{\partial z} - \frac{1}{L} \frac{\partial M_{flux}}{\partial z} - f_v \quad z \in (0,1] \text{ or } z \in [0,1) \quad (\text{A.25})$$

where:

$$M_{holdup} = \rho M_{weight} v \quad (\text{A.26})$$

$$M_{flux} = F_{total} M_{weight} v \quad (\text{A.27})$$

$$\mu = f(T, P, X_i) \quad (\text{A.28})$$

### **Energy balance**

$$\frac{\partial U}{\partial t} = -\frac{1}{L} \frac{\partial E}{\partial z} + \beta q \quad z \in (0,1] \text{ or } z \in [0,1) \quad (\text{A.29})$$

where:

$$U = \rho H - P \quad (\text{A.30})$$

$$E = F_{total}H \quad (\text{A.31})$$

$$H = f(T, P, X_i) \quad (\text{A.32})$$

**Boundary conditions**

$$F_{total,in} = F_{total}\big|_{z=0} \quad \text{or} \quad F_{total,in} = F_{total}\big|_{z=1} \quad (\text{A.29})$$

$$X_{i,in} = X_i\big|_{z=0} \quad \text{or} \quad X_{i,in} = X_i\big|_{z=1} \quad (\text{A.30})$$

$$P_{in} = P\big|_{z=0} \quad \text{or} \quad P_{in} = P\big|_{z=1} \quad (\text{A.31})$$

$$T_{in} = T\big|_{z=0} \quad \text{or} \quad T_{in} = T\big|_{z=1} \quad (\text{A.32})$$



### **A.3 Flow through membrane (transport model)**

#### **Transport of components through membrane**

*For symmetric membranes*

$$J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,F} P_F - X_{i,S} P_S) \quad (\text{A.33})$$

*For asymmetric membranes*

$$\text{Feed = shell: } J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,M} P_F - X_{i,S} P_S) \quad (\text{A.34})$$

$$\text{Feed = fibre: } J_{i,S} = \frac{Q_i}{\delta^m} (X_{i,F} P_F - X_{i,M} P_S) \quad (\text{A.35})$$

where:

$$J_{i,F} = -J_{i,S} \quad (\text{A.36})$$

$$X_{i,M} = \frac{J_i}{\sum_i J_i} \quad (\text{A.37})$$

$$Q_i = Q_{i0} e^{-\frac{E_{a_i}}{R} \left( \frac{1}{T_{retentate}} - \frac{1}{T_0} \right)} \quad (\text{A.38})$$

#### **Energy flux through membrane**

$$q_S = \sum_i J_{i,S} H_S \quad (\text{A.39})$$

$$q_F = -\sum_i J_{i,S} H_F \quad (\text{A.40})$$

where:

$$H_S = f(T_{retentate}, P_S, X_{i,M}) \quad (\text{A.41})$$

$$H_F = f(T_{retentate}, P_F, X_{i,M}) \quad (\text{A.42})$$

## Appendix B Printout custom membrane model in gPROMS and ACM

### B.1 Model in gPROMS

#### Main model membrane unit

```
*** MAIN MODEL: HOLLOW FIBER MEMBRANE UNIT ***
```

```
#=====
```

```
PARAMETER
```

```
# *** Global parameters ***
```

phys_prop	AS FOREIGN_OBJECT "CapeOpenThermo"	# Physical property package
no_components	AS INTEGER	# Number of components
R	AS REAL DEFAULT 8.31433	# Gas constant [8.314 J/mol/K]
pi	AS REAL DEFAULT 3.14159	# [3.14159]

```
#-----
```

```
# *** Parameters for discretisation method ***
```

no_of_nodes	AS INTEGER DEFAULT 50	# Number of nodes
-------------	-----------------------	-------------------

```
#-----
```

```
# *** Scaling factors ***
```

flow_scale	AS REAL DEFAULT 1e-4
pressure_scale	AS REAL DEFAULT 1e5
energy_scale	AS REAL DEFAULT 1e6

```
#=====
```

```
DISTRIBUTION_DOMAIN
```

axial	AS (0:1)
-------	----------

```
#=====
```

```
UNIT
```

fiber_side	AS PlugFlow
shell_side	AS PlugFlowShellNeg
membrane	AS Transport

```
#=====
```

```
PORT
```

feed	AS CO_Material	# Inlet stream of membrane module
sweep	AS CO_Material	# Inlet stream of membrane module
retentate	AS CO_Material	# Outlet stream of membrane module
permeate	AS CO_Material	# Outlet stream of membrane module

```
#=====
```

```
VARIABLE
```

```
# *** Geometric variables ***
```

length	AS length	# Length of fibers [m]
inner_diameter	AS length	# Inner diameter of 1 fiber [m]
outer_diameter	AS length	# Outer diameter of 1 fiber [m]

```

shell_diameter      AS length      # (Inner) shell diameter of membrane module [m]
no_of_fibers        AS no_type     # Number of fibers in membrane module [-]
membrane_area       AS area        # Total membrane area of module [m2]
membrane_thickness  AS length      # Membrane thickness [m]
#-----
# *** Membrane characteristics ***
stage_cut           AS no_type     # Stage cut (flowrate permeate / flowrate feed)
pressure_ratio      AS no_type     # Pressure ratio (pressure feed / pressure sweep)
packing_density     AS no_type     # Packing density (membrane area / total volume) [m2/m3]
ratio_cross_area    AS no_type     # Ratio cross area fiber to shell side
#-----
# *** Friction constants ***
shell_fr_constant   AS no_type     # Friction parameter for shell side
fiber_fr_constant   AS no_type     # Friction parameter for fiber side
#-----
# *** Auxiliary variable ***
feed_rho            AS molar_density # Molar density of feed stream [mol/m3]
constant_velocity   AS no_type     # Constant for velocity of feed stream [m/s]
#=====
SELECTOR
run_mode            AS ( CoCurrent, CounterCurrent )
feed_side           AS ( Shell, Fiber )
sim_mode            AS ( Level_1, Level_2, Level_3, Level_4, Level_5 )
membr_type          AS ( Asymmetric , Symmetric )
method              AS ( Rating, Design )
#=====
SET
no_components       := phys_prop.NumberOfComponents;
axial               := [BFDm, 2, no_of_nodes];
#=====
TOPOLOGY
CASE feed_side OF
  WHEN shell:
    feed              = shell_side.input;
    shell_side.output = retentate;
    sweep             = fiber_side.input;
    fiber_side.output = permeate;
  WHEN fiber:
    feed              = fiber_side.input;
    fiber_side.output = retentate;
    sweep             = shell_side.input;
    shell_side.output = permeate;
END
fiber_side.info_transport = membrane.fiber_info;
membrane.shell_info      = shell_side.info_transport;
#=====

```

# EQUATION

```

# *** Geometric variables ***
# Volume fiber and shell side
  fiber_side.volume = fiber_side.cross_area * length;
  shell_side.volume = shell_side.cross_area * length;
# Geometric parameter B for fiber and shell side (ratio of membrane surface area to volume)
  fiber_side.B = membrane_area / no_of_fibers / fiber_side.volume;
  shell_side.B = membrane_area / shell_side.volume;
#-----
# *** Membrane characteristics ***
  stage_cut = permeate.mass_flowrate / feed.mass_flowrate;
  pressure_ratio = feed.pressure / sweep.pressure;
  (shell_side.cross_area + fiber_side.cross_area) * length * packing_density = membrane_area;
  fiber_side.cross_area * no_of_fibers = shell_side.cross_area * ratio_cross_area;
#-----
# *** Friction factors ***
  fiber_fr_constant = inner_diameter^2/32;
  shell_fr_constant = 1e-11;
  FOR z:=0 TO 1 DO
    fiber_side.fr_factor(z) * fiber_fr_constant = fiber_side.viscosity(z) * fiber_side.velocity(z);
    shell_side.fr_factor(z) * shell_fr_constant = shell_side.viscosity(z) * shell_side.velocity(z);
  END
#-----
# *** Auxiliary equations to specify velocity for feed stream ***
  feed_rho = phys_prop.VapourDensity(feed.temperature,feed.pressure,feed.mass_fraction);
  CASE feed_side OF
    WHEN fiber:
      feed.mass_flowrate = feed_rho * constant_velocity * no_of_fibers * fiber_side.cross_area;
    WHEN shell:
      feed.mass_flowrate = feed_rho * constant_velocity * shell_side.cross_area;
  END
#-----
# *** Equations for rating or design method ***
  CASE method OF
    WHEN rating:
      fiber_side.cross_area = 1/4 * pi * inner_diameter^2;
      shell_side.cross_area = 1/4 * pi * shell_diameter^2 - no_of_fibers * 1/4 * pi * outer_diameter^2 ;
      membrane_area = no_of_fibers * pi * inner_diameter * length;
      membrane_thickness = outer_diameter - inner_diameter;
    WHEN design:
      no_of_fibers = 1;
      inner_diameter = 100e-6;
      outer_diameter = 1;
      shell_diameter = 1;
  END
#-----

```

```

*** Connection between plug flow models and transport model***
# For fiber side:
fiber_side.info_transport.flux_mol_membr = fiber_side.flux_mol_membr;
fiber_side.info_transport.fraction_mol = fiber_side.fraction_mol;
fiber_side.info_transport.energy_flux_membr = fiber_side.energy_flux_membr;
fiber_side.info_transport.temperature = fiber_side.temperature;
fiber_side.info_transport.pressure = fiber_side.pressure;
# For shell side:
shell_side.info_transport.flux_mol_membr = shell_side.flux_mol_membr;
shell_side.info_transport.fraction_mol = shell_side.fraction_mol;
shell_side.info_transport.energy_flux_membr = shell_side.energy_flux_membr;
shell_side.info_transport.temperature = shell_side.temperature;
shell_side.info_transport.pressure = shell_side.pressure;
# -----
*** Connection between model ports and plug flow models***
# For fiber side
# Port connections at z = 0
(1/flow_scale) * fiber_side.flux_mol_total(0) * fiber_side.cross_area * no_of_fibers = (1/flow_scale) * fiber_side.input.mass_flowrate;
fiber_side.fraction_mol(0) = fiber_side.input.mass_fraction;
fiber_side.temperature(0) = fiber_side.input.temperature;
fiber_side.pressure(0) = fiber_side.input.pressure;
fiber_side.spec_enth_mol(0) = fiber_side.input.mass_specific_enthalpy;
# Port connections at z = 1
fiber_side.output.mass_flowrate = fiber_side.input.mass_flowrate - no_of_fibers * ( fiber_side.flux_mol_total(0) - fiber_side.flux_mol_total(1) ) * fiber_side.cross_area;
fiber_side.output.mass_fraction = fiber_side.fraction_mol(1);
fiber_side.output.temperature = fiber_side.temperature(1);
fiber_side.output.mass_specific_enthalpy = fiber_side.spec_enth_mol(1);
fiber_side.output.pressure = fiber_side.pressure(1);
# For shell side
CASE run_mode OF
  WHEN CoCurrent:
    # Port connections at z = 0
    (1/flow_scale) * shell_side.flux_mol_total(0) * shell_side.cross_area = (1/flow_scale) * shell_side.input.mass_flowrate;
    shell_side.fraction_mol(0) = shell_side.input.mass_fraction;
    shell_side.temperature(0) = shell_side.input.temperature;
    shell_side.pressure(0) = shell_side.input.pressure;
    shell_side.spec_enth_mol(0) = shell_side.input.mass_specific_enthalpy;
    # Port connections at z = 1
    shell_side.output.mass_flowrate = shell_side.flux_mol_total(1) * shell_side.cross_area;
    shell_side.output.mass_fraction = shell_side.fraction_mol(1);
    shell_side.output.temperature = shell_side.temperature(1);
    shell_side.output.mass_specific_enthalpy = shell_side.spec_enth_mol(1);
    shell_side.output.pressure = shell_side.pressure(1);
  WHEN CounterCurrent:
    # Port connections at z = 1
    - (1/flow_scale) * shell_side.flux_mol_total(1) * shell_side.cross_area = (1/flow_scale) * shell_side.input.mass_flowrate;

```

```

shell_side.fraction_mol(1) = shell_side.input.mass_fraction;
shell_side.temperature(1) = shell_side.input.temperature;
shell_side.pressure(1) = shell_side.input.pressure;
shell_side.spec_enth_mol(1) = shell_side.input.mass_specific_enthalpy;
# Port connections at z = 0
shell_side.output.mass_flowrate = - shell_side.flux_mol_total(0) * shell_side.cross_area;
shell_side.output.mass_fraction = shell_side.fraction_mol(0);
shell_side.output.temperature = shell_side.temperature(0);
shell_side.output.mass_specific_enthalpy = shell_side.spec_enth_mol(0);
shell_side.output.pressure = shell_side.pressure(0);
END
#-----
# *** Connection of variables ***
shell_side.length      = length;
fiber_side.length      = length;
membrane.membrane_thickness = membrane_thickness;
#=====

```

### *Sub-model flow through fibre and shell side (for co-current)*

```

# *** PLUG FLOW MODEL ***
#=====
PARAMETER
# *** Global parameters ***
  phys_prop      AS FOREIGN_OBJECT "CapeOpenThermo"  # Physical property package
  no_components  AS INTEGER                        # Number of components
  R              AS REAL                          # Gas constant [8.314 J/mol/K]
  pi             AS REAL                          # [3.14159]
#-----
# *** Parameters for discretisation method ***
  no_of_nodes    AS INTEGER                        # Number of nodes
#-----
# *** Scaling factors ***
  flow_scale     AS REAL
  pressure_scale AS REAL
  energy_scale   AS REAL
#=====
DISTRIBUTION_DOMAIN
  axial          AS (0:1)
#=====
PORT
  input          AS CO_MATERIAL
  output         AS CO_MATERIAL
  info_transport AS INFO_TRANSPORT
#=====
VARIABLE
# *** Geometric variables ***
  length      AS length      # Length of fibers [m]
  B           AS no_type     # Geometric fractor [1/m]
  cross_area  AS area        # Total cross sectional area of 1 side [m2]
  volume      AS volume      # Total volume of 1 side [m3]
#-----
# *** Variables for mass balance ***
  conc_mol    AS DISTRIBUTION(no_components,axial)  OF molar_concentration  # Molar concentration of component i [mol/m3]
  flux_mol    AS DISTRIBUTION(no_components,axial)  OF molar_flowrate       # Molar flux of component i [mol/m2/s]
  flux_mol_total AS DISTRIBUTION(axial)             OF molar_flowrate       # Molar flux [mol/m2/s]
  fraction_mol AS DISTRIBUTION(no_components,axial)  OF molar_fraction       # Molefraction [mol/mol]
  flux_mol_membr AS DISTRIBUTION(no_components,axial) OF no_type              # Molar flux component i through membr [mol/m2/s]
#-----
# *** Variables for momentum balance ***
  pressure      AS DISTRIBUTION(axial)  OF pressure  # Pressure [N/m2]
  velocity      AS DISTRIBUTION(axial)  OF velocity  # Velocity [m/s]
  dens_mol      AS DISTRIBUTION(axial)  OF molar_density  # Molar density [mol/m3]
  momentum_holdup AS DISTRIBUTION(axial) OF momentum_holdup # Holdup of momentum [kg/m2/s]

```

momentum_flux	AS DISTRIBUTION(axial)	OF momentum_flow	# Momentum flux [kgm/s2]
viscosity	AS DISTRIBUTION(axial)	OF viscosity	# Viscosity [Pa.s] = [kg/m/s]
fr_factor	AS DISTRIBUTION(axial)	OF no_type	# Friction factor [Pa/m = kg/m2/s2]

---

```

*** Variables for energy balance ***
temperature      AS DISTRIBUTION(axial)      OF temperature      # Temperature [K]
spec_enth_mol    AS DISTRIBUTION(axial)      OF molar_specific_enthalpy # Molar specific enthalpy [J/mol]
energy_holdup    AS DISTRIBUTION(axial)      OF volume_energy    # Holdup of energy [J/m3]
energy_flux      AS DISTRIBUTION(axial)      OF energy_rate      # Energy flux [J/s]
energy_flux_membr AS DISTRIBUTION(axial)      OF heat_flux        # Energy flux through membr [J/m2/s]

```

---

```

*** Auxiliary variables ***
mol_weight      AS DISTRIBUTION(no_components,axial)  OF molecular_weight # Molecular weight of component i [kg/kmol]
mol_weight_mix  AS DISTRIBUTION(axial)                OF molecular_weight # Molecular weight of mixture [kg/kmol]

```

---

```

SELECTOR
sim_mode        AS ( Level_1, Level_2, Level_3, Level_4, Level_5 )

```

---

```

EQUATION
*** Balance equations ***
FOR z:=0|+ TO 1 DO
  (1/flow_scale) * flux_mol_total(z) = (1/flow_scale) * SIGMA(flux_mol(z));
  CASE sim_mode OF
    WHEN level_1:
      flux_mol(z) = flux_mol(0);
      pressure(z) = pressure(0);
      temperature(z) = temperature(0);
    WHEN level_2:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      pressure(z) = pressure(0);
      temperature(z) = temperature(0);
    WHEN level_3:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      temperature(z) = temperature(0);
    WHEN level_4:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      0 = - 1/length * PARTIAL(energy_flux(z),axial) + B * energy_flux_membr(z);
    WHEN level_5:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      0 = - 1/length * PARTIAL(energy_flux(z),axial) + B * energy_flux_membr(z);
  END
END

```

---

```

END

```

---



```

*** Auxiliary equations ***
FOR z:=0| TO 1| DO
  (1/flow_scale) * flux_mol(z) = (1/flow_scale) * velocity(z) * conc_mol(z);
  (1/flow_scale) * flux_mol(z) = (1/flow_scale) * fraction_mol(z) * flux_mol_total(z);
  dens_mol(z) = SIGMA(conc_mol(z));
  (1/pressure_scale) * momentum_holdup(z) = (1/pressure_scale) * dens_mol(z) * mol_weight_mix(z) * 1e-3 * velocity(z);
  (1/pressure_scale) * momentum_flux(z) = (1/pressure_scale) * flux_mol_total(z) * mol_weight_mix(z) * 1e-3 * velocity(z);
  (1/energy_scale) * energy_holdup(z) = (1/energy_scale) * ( dens_mol(z) * spec_enth_mol(z) - pressure(z) );
  (1/energy_scale) * energy_flux(z) = (1/energy_scale) * flux_mol_total(z) * spec_enth_mol(z);
END
# -----
*** Call for physical properties ***
FOR z:=0 TO 1 DO
  mol_weight(z) = phys_prop.MolecularWeight();
  mol_weight_mix(z) = SIGMA(mol_weight(z) * fraction_mol(z));
  dens_mol(z) = phys_prop.VapourDensity(temperature(z),pressure(z),fraction_mol(z));
  viscosity(z) = phys_prop.VapourViscosity(temperature(z),pressure(z),fraction_mol(z));
END
FOR z:=0| TO 1| DO
  spec_enth_mol(z) = phys_prop.VapourEnthalpy(temperature(z),pressure(z),fraction_mol(z));
END
# =====

```

### *Sub-model flow through shell side (for counter-current)*

```

# *** PLUG FLOW MODEL SHELL NEG ***
#=====
PARAMETER
# *** Global parameters ***
  phys_prop      AS FOREIGN_OBJECT "CapeOpenThermo"  # Physical property package
  no_components  AS INTEGER                          # Number of components
  R              AS REAL                             # Gas constant [8.314 J/mol/K]
  pi             AS REAL                             # [3.14159]
#-----
# *** Parameters for discretisation method ***
  no_of_nodes    AS INTEGER                          # Number of nodes
#-----
# *** Scaling factors ***
  flow_scale     AS REAL
  pressure_scale AS REAL
  energy_scale   AS REAL
#=====
DISTRIBUTION_DOMAIN
  axial          AS (0:1)
#=====
PORT
  input          AS CO_MATERIAL
  output         AS CO_MATERIAL
  info_transport AS INFO_TRANSPORT
#=====
VARIABLE
# *** Geometric variables ***
  length      AS length      # Length of fibers [m]
  B           AS no_type     # Geometric fractor [1/m]
  cross_area  AS area        # Total cross sectional area of 1 side [m2]
  volume      AS volume      # Total volume of 1 side [m3]
#-----
# *** Variables for mass balance ***
  conc_mol    AS DISTRIBUTION(no_components,axial)  OF molar_concentration  # Molar concentration of component i [mol/m3]
  flux_mol    AS DISTRIBUTION(no_components,axial)  OF molar_flowrate       # Molar flux of component i [mol/m2/s]
  flux_mol_total AS DISTRIBUTION(axial)             OF molar_flowrate       # Molar flux [mol/m2/s]
  fraction_mol AS DISTRIBUTION(no_components,axial)  OF molar_fraction       # Molefraction [mol/mol]
  flux_mol_membr AS DISTRIBUTION(no_components,axial) OF no_type              # Molar flux component i through membr [mol/m2/s]
#-----
# *** Variables for momentum balance ***
  pressure      AS DISTRIBUTION(axial)  OF pressure  # Pressure [N/m2]
  velocity      AS DISTRIBUTION(axial)  OF velocity  # Velocity [m/s]
  dens_mol      AS DISTRIBUTION(axial)  OF molar_density  # Molar density [mol/m3]
  momentum_holdup AS DISTRIBUTION(axial) OF momentum_holdup # Holdup of momentum [kg/m2/s]

```

momentum_flux	AS DISTRIBUTION(axial)	OF momentum_flow	# Momentum flux [kgm/s2]
viscosity	AS DISTRIBUTION(axial)	OF viscosity	# Viscosity [Pa.s] = [kg/m/s]
fr_factor	AS DISTRIBUTION(axial)	OF no_type	# Friction factor [Pa/m = kg/m2/s2]

---

```

# *** Variables for energy balance ***
temperature      AS DISTRIBUTION(axial)      OF temperature      # Temperature [K]
spec_enth_mol    AS DISTRIBUTION(axial)      OF molar_specific_enthalpy # Molar specific enthalpy [J/mol]
energy_holdup    AS DISTRIBUTION(axial)      OF volume_energy    # Holdup of energy [J/m3]
energy_flux      AS DISTRIBUTION(axial)      OF energy_rate       # Energy flux [J/s]
energy_flux_membr AS DISTRIBUTION(axial)      OF heat_flux         # Energy flux through membr [J/m2/s]

```

---

```

# *** Auxiliary variables ***
mol_weight      AS DISTRIBUTION(no_components,axial)  OF molecular_weight  # Molecular weight of component i [kg/kmol]
mol_weight_mix  AS DISTRIBUTION(axial)                OF molecular_weight  # Molecular weight of mixture [kg/kmol]

```

---

```

SELECTOR
sim_mode        AS ( Level_1, Level_2, Level_3, Level_4, Level_5 )

```

---

```

SET
axial := [FFDM, 2, no_of_nodes];

```

---

```

EQUATION
# *** Balance equations ***
FOR z:=0| TO 1|- DO
  (1/flow_scale) * flux_mol_total(z) = (1/flow_scale) * SIGMA(flux_mol(z));
  CASE sim_mode OF
    WHEN level_1:
      flux_mol(z) = flux_mol(1);
      pressure(z) = pressure(1);
      temperature(z) = temperature(1);
    WHEN level_2:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      pressure(z) = pressure(1);
      temperature(z) = temperature(1);
    WHEN level_3:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      temperature(z) = temperature(1);
    WHEN level_4:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      0 = - 1/length * PARTIAL(energy_flux(z),axial) + B * energy_flux_membr(z);
    WHEN level_5:
      0 = - 1/length * PARTIAL(flux_mol(z),axial) + B * flux_mol_membr(z);
      0 = - 1/length * PARTIAL(pressure(z),axial) - 1/length * PARTIAL(momentum_flux(z),axial) - fr_factor(z);
      0 = - 1/length * PARTIAL(energy_flux(z),axial) + B * energy_flux_membr(z);

```

```

END
END
# -----
# *** Auxiliary equations ***
FOR z:=0 TO 1 DO
  (1/flow_scale) * flux_mol(z) = (1/flow_scale) * velocity(z) * conc_mol(z);
  (1/flow_scale) * flux_mol(z) = (1/flow_scale) * fraction_mol(z) * flux_mol_total(z);
  dens_mol(z) = SIGMA(conc_mol(z));
  (1/pressure_scale) * momentum_holdup(z) = (1/pressure_scale) * dens_mol(z) * mol_weight_mix(z) * 1e-3 * velocity(z);
  (1/pressure_scale) * momentum_flux(z) = (1/pressure_scale) * flux_mol_total(z) * mol_weight_mix(z) * 1e-3 * velocity(z);
  (1/energy_scale) * energy_holdup(z) = (1/energy_scale) * ( dens_mol(z) * spec_enth_mol(z) - pressure(z) );
  (1/energy_scale) * energy_flux(z) = (1/energy_scale) * flux_mol_total(z) * spec_enth_mol(z);
END
# -----
# *** Call for physical properties ***
FOR z:=0 TO 1 DO
  mol_weight(z) = phys_prop.MolecularWeight();
  mol_weight_mix(z) = SIGMA(mol_weight(z) * fraction_mol(z));
  dens_mol(z) = phys_prop.VapourDensity(temperature(z),pressure(z),fraction_mol(z));
  viscosity(z) = phys_prop.VapourViscosity(temperature(z),pressure(z),fraction_mol(z));
END
FOR z:=0 TO 1 DO
  spec_enth_mol(z) = phys_prop.VapourEnthalpy(temperature(z),pressure(z),fraction_mol(z));
END
# =====

```

### *Sub-model flow/transport of components through the membrane*

```

*** TRANSPORT MODEL ***
#=====
PARAMETER
# *** Global parameters ***
  phys_prop      AS FOREIGN_OBJECT "CapeOpenThermo"    # Physical property package
  no_components   AS INTEGER                          # Number of components
  R              AS REAL                              # Gas constant [8.314 J/mol/K]
  pi             AS REAL                              # [3.14159]
#-----
# *** Parameters for discretisation method ***
  no_of_nodes     AS INTEGER                          # Number of nodes
#-----
# *** Scaling factors ***
  flow_scale      AS REAL
  pressure_scale  AS REAL
  energy_scale    AS REAL
#=====
DISTRIBUTION_DOMAIN
  axial          AS (0:1)
#=====
PORT
  fiber_info      AS info_transport
  shell_info      AS info_transport
#=====
VARIABLE
# *** Geometric variables ***
  membrane_thickness AS length
#-----
# *** Variables for separative character of the membrane ***
  permeability0     AS ARRAY(no_components)           OF no_type      # Permeability [...]
  activation_energy  AS ARRAY(no_components)           OF no_type      # Activation energy for permeability
  temperature0      AS no_type                        # Standard temperature
  permeability       AS DISTRIBUTION(no_components,axial) OF permeability # Permeability [...]
#-----
# *** Auxiliary variables
  fraction_mol_membr AS DISTRIBUTION(no_components,axial) OF molar_fraction # Molar fraction of component i in membrane [mol/mol]
  spec_enth_mol_fiber AS DISTRIBUTION(axial)              OF molar_specific_enthalpy # Specific molar enthalpy of components in membrane [J/mol]
  spec_enth_mol_shell AS DISTRIBUTION(axial)              OF molar_specific_enthalpy # Specific molar enthalpy
#=====
SELECTOR
  feed_side      AS ( Shell, Fiber )
  sim_mode       AS ( Level_1, Level_2, Level_3, Level_4, Level_5 )
  membr_type     AS ( Asymmetric , Symmetric )
#=====

```

EQUATION

FOR z:= 0 TO 1 DO

```
#-----
# *** Component fractions in membrane ***
fraction_mol_membr(z) * SIGMA ( ABS(shell_info.flux_mol_membr(z)) ) = ABS (shell_info.flux_mol_membr(z));
#-----
# *** Enthalpy of mass flux through membrane and permeability (Arrhenius funtion) ***
CASE feed_side OF
  WHEN shell:
    (1/1e-14) * permeability(z) = (1/1e-14) * permeability0 * exp( - activation_energy / R * ( 1 / shell_info.temperature(z) - 1 / temperature0 ));
    spec_enth_mol_fiber(z) = phys_prop.VapourEnthalpy(shell_info.temperature(z),fiber_info.pressure(z),fraction_mol_membr(z));
    spec_enth_mol_shell(z) = phys_prop.VapourEnthalpy(shell_info.temperature(z),shell_info.pressure(z),fraction_mol_membr(z));
  WHEN fiber:
    (1/1e-14) * permeability(z) = (1/1e-14) * permeability0 * exp( - activation_energy / R * ( 1 / fiber_info.temperature(z) - 1 / temperature0 ));
    spec_enth_mol_fiber(z) = phys_prop.VapourEnthalpy(fiber_info.temperature(z),fiber_info.pressure(z),fraction_mol_membr(z));
    spec_enth_mol_shell(z) = phys_prop.VapourEnthalpy(fiber_info.temperature(z),shell_info.pressure(z),fraction_mol_membr(z));
END
#-----
# *** Molar and energy flux through membrane: connection between two sides ***
(1/flow_scale) * fiber_info.flux_mol_membr(z) = - (1/flow_scale) * shell_info.flux_mol_membr(z);
#-----
# *** Molar flux through membrane ***
CASE sim_mode OF
  WHEN level_1:
    shell_info.flux_mol_membr(z) = 0;
  WHEN level_2:
    (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) - shell_info.fraction_mol(z)
    * shell_info.pressure(z));
  WHEN level_3:
    CASE membr_type OF
      WHEN Symmetric:
        (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
        shell_info.fraction_mol(z) * shell_info.pressure(z));
      WHEN Asymmetric:
        CASE feed_side OF
          WHEN shell:
            (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fraction_mol_membr(z) * fiber_info.pressure(z) -
            shell_info.fraction_mol(z) * shell_info.pressure(z));
          WHEN fiber:
            (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
            fraction_mol_membr(z) * shell_info.pressure(z));
        END
      END
  WHEN level_4:
    CASE membr_type OF
      WHEN Symmetric:
```

```

        (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
        shell_info.fraction_mol(z) * shell_info.pressure(z));
    WHEN Asymmetric:
        CASE feed_side OF
            WHEN shell:
                (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fraction_mol_membr(z) * fiber_info.pressure(z) -
                shell_info.fraction_mol(z) * shell_info.pressure(z));
            WHEN fiber:
                (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability0 / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
                fraction_mol_membr(z) * shell_info.pressure(z));
        END
    END
    WHEN level_5:
        CASE membr_type OF
            WHEN Symmetric:
                (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability(z) / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
                shell_info.fraction_mol(z) * shell_info.pressure(z));
            WHEN Asymmetric:
                CASE feed_side OF
                    WHEN shell:
                        (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability(z) / membrane_thickness * ( fraction_mol_membr(z) * fiber_info.pressure(z) -
                        shell_info.fraction_mol(z) * shell_info.pressure(z));
                    WHEN fiber:
                        (1/flow_scale) * shell_info.flux_mol_membr(z) = (1/flow_scale) * permeability(z) / membrane_thickness * ( fiber_info.fraction_mol(z) * fiber_info.pressure(z) -
                        fraction_mol_membr(z) * shell_info.pressure(z));
                END
            END
        END
    END
    # -----
    # *** Energy flux through membrane ***
    shell_info.energy_flux_membr(z) = SIGMA(shell_info.flux_mol_membr(z)) * spec_enth_mol_shell(z);
    fiber_info.energy_flux_membr(z) = -SIGMA(shell_info.flux_mol_membr(z)) * spec_enth_mol_fiber(z);
    END
    # =====

```

## ***PROCESS entity***

UNIT

MembraneUnit

AS MembraneUnit

SET

MembraneUnit.phys\_prop := "CapeOpenThermo::(PS)ATCOProperties.COPropertySystem.1<CompMixt>";

ASSIGN

WITHIN MembraneUnit DO

Feed.mass\_flowrate := 0.1/3.6;  
Feed.mass\_fraction := [0.50,0.25,0.20,0.05];  
Feed.Temperature := 298;  
Feed.Pressure := 70e5;

Sweep.mass\_flowrate := 1e-10/3.6;  
Sweep.mass\_fraction := [0.50,0.25,0.20,0.05];  
Sweep.Temperature := 298;  
Sweep.Pressure := 10e5;

membrane.permeability0(1) := 3.408e-12;  
membrane.permeability0(2) := 3.54e-14;  
membrane.permeability0(3) := 3.408e-14;  
membrane.permeability0(4) := 9.24e-14;  
membrane.activation\_energy(1) := 10000;  
membrane.activation\_energy(2) := 10000;  
membrane.activation\_energy(3) := 10000;  
membrane.activation\_energy(4) := 10000;  
membrane.temperature0 := 298.15;

# Rating method

#inner\_diameter := 80e-6;  
#outer\_diameter := 200e-6;  
#shell\_diameter := 0.075;  
#no\_of\_fibers := 50000;  
#length := 0.5;

# Design method

packing\_density := 4500;  
ratio\_cross\_area := 0.1;  
membrane\_thickness := 100e-6;  
membrane\_area := 2;  
constant\_velocity := 0.05;

END

SELECTOR



```

MembraneUnit.method           := MembraneUnit.Design;
MembraneUnit.run_mode         := MembraneUnit.CounterCurrent;
MembraneUnit.feed_side        := MembraneUnit.shell;
MembraneUnit.membrane.feed_side := MembraneUnit.membrane.shell;
MembraneUnit.membrane.membr_type := MembraneUnit.membrane.symmetric;
MembraneUnit.shell_side.sim_mode := MembraneUnit.Shell_side.level_1;
MembraneUnit.fiber_side.sim_mode := MembraneUnit.Fiber_side.level_1;
MembraneUnit.membrane.sim_mode  := MembraneUnit.membrane.level_1;

SCHEDULE
SEQUENCE
  SWITCH
    MembraneUnit.shell_side.sim_mode := MembraneUnit.shell_side.level_2;
    MembraneUnit.fiber_side.sim_mode := MembraneUnit.fiber_side.level_2;
    MembraneUnit.membrane.sim_mode  := MembraneUnit.membrane.level_2;
  END
  SWITCH
    MembraneUnit.shell_side.sim_mode := MembraneUnit.shell_side.level_3;
    MembraneUnit.fiber_side.sim_mode := MembraneUnit.fiber_side.level_3;
    MembraneUnit.membrane.sim_mode  := MembraneUnit.membrane.level_3;
  END
  SWITCH
    MembraneUnit.shell_side.sim_mode := MembraneUnit.shell_side.level_4;
    MembraneUnit.fiber_side.sim_mode := MembraneUnit.fiber_side.level_4;
    MembraneUnit.membrane.sim_mode  := MembraneUnit.membrane.level_4;
  END
  SWITCH
    MembraneUnit.shell_side.sim_mode := MembraneUnit.shell_side.level_5;
    MembraneUnit.fiber_side.sim_mode := MembraneUnit.fiber_side.level_5;
    MembraneUnit.membrane.sim_mode  := MembraneUnit.membrane.level_5;
  END
END

```

## B.2 Model in ACM

### Main model membrane unit

```
Model MembraneUnit
//=====
// PARAMETER DEFINITION
// *** Global parameters ***
      R                                as global RealParameter;    // Gas constant [8.314 J/mol/K]
      pi                              as global RealParameter;    // [3.14159]
//-----
// *** Parameters for discretisation method ***
      no_of_nodes                     as global IntegerParameter;  // Number of nodes
//-----
// *** Scaling factors ***
      flow_scale                      as global RealParameter;    // Scaling factor
//-----
// *** Unit conversion factors ***
      hr_to_s                        as global RealParameter;    // Conversion factor time: hours to seconds
      bar_to_Pa                      as global RealParameter;    // Conversion factor pressure: bar to Pascal
      cP_to_Pa.s                    as global RealParameter;    // Conversion factor viscosity: cP to Pa.s
      GJ_to_J                        as global RealParameter;    // Conversion factor energy: GJ to J
//-----
// *** Defintion of parameters for run mode, feed side and simulation mode ***
      run_mode                       as global RunMode;          // Parameter to change the operation mode: "CoCurrent" or "CounterCurrent"
      feed_side                      as global FeedSide;         // Parameter to change the feed side: "Shell" or "Fiber"
      sim_mode                       as global SimMode;          // Parameter to change level of model: "Level_1" to "Level_5"
      membr_type                    as global MembrType;         // Parameter to change membrane type: "Symmetric" or "Asymmetric"
      method                        as global Meth;              // Parameter to change method: "Rating" or "Design"
//=====
// PORT DEFINITION
      feed                          as input MoleFractionPort;    // Inlet stream of membrane module
      sweep                        as input MoleFractionPort;    // Inlet stream of membrane module
      retentate                    as output MoleFractionPort;    // Outlet stream of membrane module
      permeate                     as output MoleFractionPort;    // Outlet stream of membrane module
//=====
// DISTRIBUTION DOMAIN DEFINITION
      axial                        as LengthDomain (Length:1, SpacingPreference: 1/no_of_nodes, DiscretizationMethod: "BFD2", HighestOrderDerivative: 1);
//=====
// DOMAIN SET DEFINITION
      Nset                        as global IntegerSet;
//=====
// UNIT DEFINITION
```

```

        membrane                as Transport;
        fiber_side               as Plugflow;
        shell_side               as Plugflow;
//=====
// VARIABLE DEFINITION
// *** Geometric variables ***
        length                  as length;           // Length of fibers [m]
        inner_diameter          as length;           // Inner diameter of 1 fiber [m]
        outer_diameter          as length;           // Outer diameter of 1 fiber [m]
        shell_diameter          as length;           // (Inner) shell diameter of membrane module [m]
        no_of_fibers            as notype;           // Number of fibers in membrane module [-]
        membrane_area           as area;             // Total membrane area of module [m2]
        membrane_thickness      as length;           // Membrane thickness [m]
//-----
// *** Membrane characteristics ***
        stage_cut               as notype;           // Stage cut (flowrate permeate / flowrate feed)
        pressure_ratio           as notype;           // Pressure ratio (pressure feed / pressure sweep) [m2/m3]
        packing_density         as notype;           // Packing density (membrane area / total volume)
        ratio_cross_area        as notype;           // Ratio cross area fiber to shell side
//-----
// *** Friction constants ***
        shell_fr_constant       as notype;           // Friction parameter for shell side
        fiber_fr_constant       as notype;           // Friction parameter for fiber side
//-----
// *** Auxiliary variables ***
        sweep_rho               as dens_mol;         // Molar density of sweep stream [mol/m3]
        feed_rho                as dens_mol;         // Molar density of feed stream [mol/m3]
        constant_velocity       as notype;           // Constant for velocity of feed stream [m/s]
//=====
// EQUATIONS
// *** Topology: connection of the submodels ***
        If feed_side == "Shell" then
            LINK feed              AND shell_side.import;
            LINK sweep             AND fiber_side.import;
            LINK retentate         AND shell_side.export;
            LINK permeate          AND fiber_side.export;
        elseif feed_side == "Fiber" then
            LINK feed              AND fiber_side.import;
            LINK sweep             AND shell_side.import;
            LINK retentate         AND fiber_side.export;
            LINK permeate          AND shell_side.export;
        Endif

        CONNECT fiber_side.info_transport AND membrane.fiber_info;
        CONNECT fiber_side.result_transport AND membrane.fiber_result;
        CONNECT shell_side.info_transport AND membrane.shell_info;

```

```

CONNECT shell_side.result_transport          AND membrane.shell_result;
//-----
// *** Boundary conditions if sweep stream is not connected to membrane model block ***
    If not sweep.isconnected then
        sweep.F.spec                        : fixed;
        sweep.F.value                       : 1e-10;
        sweep.P.spec                        : fixed;
        sweep.P.value                       : 11.23;
        sweep.T.spec                        : fixed;
        sweep.T.value                       : feed.T.value;
        sweep.Z(componentlist).spec         : fixed;
        sweep.Z(componentlist).value        : feed.F.value;
        call(sweep.h) = pEnth_Mol_Vap (sweep.T,sweep.P,sweep.Z);
        call(sweep_rho) = pDens_Mol_Vap (sweep.T,sweep.P,sweep.Z);
        sweep.V * sweep_rho = 1;
    Endif
//-----
// *** Equations for geometric variables ***
    // Volume shell and fibre side
        fiber_side.volume_ = fiber_side.cross_area * length;
        shell_side.volume_ = shell_side.cross_area * length;
    // Geometric parameter B (ratio of membrane surface area to volume)
        fiber_side.B = membrane_area / no_of_fibers / fiber_side.volume_;
        shell_side.B = membrane_area / shell_side.volume_;
//-----
// *** Membrane characteristics ***
    stage_cut = permeate.F / feed.F;
    pressure_ratio = feed.P / sweep.P;
    (shell_side.cross_area + fiber_side.cross_area) * length * packing_density = membrane_area;
    fiber_side.cross_area * no_of_fibers = shell_side.cross_area * ratio_cross_area;
//-----
// *** Fiction factors
    fiber_fr_constant = inner_diameter^2 / 32;
    shell_fr_constant = 1e-11;
    For z in Nset do
        fiber_side.fr_factor(z) * fiber_fr_constant = fiber_side.viscosity_(z) * cP_to_Pas * fiber_side.velocity_(z);
        shell_side.fr_factor(z) * shell_fr_constant = shell_side.viscosity_(z) * cP_to_Pas * shell_side.velocity_(z);
    Endfor
//-----
// *** Auxiliary equations to specify velocity for feed stream ***
    call(feed_rho) = pDens_Mol_Vap (feed.T,feed.P,feed.Z);
    If feed_side == "Fiber" then
        feed.F = feed_rho * constant_velocity * hr_to_s * fiber_side.cross_area;
    elseif feed_side == "Shell" then
        feed.F = feed_rho * constant_velocity * hr_to_s * shell_side.cross_area;
    Endif

```

```

//-----
// *** Equations for rating or design method
    If method == "Rating" then
        inner_diameter.spec      : fixed;
        inner_diameter.value     : 80e-6;
        outer_diameter.spec      : fixed;
        outer_diameter.value     : 200e-6;
        shell_diameter.spec      : fixed;
        shell_diameter.value     : 0.075;
        no_of_fibers.spec        : fixed;
        no_of_fibers.value       : 50000;
        length.spec              : fixed;
        length.value              : 0.5;

        fiber_side.cross_area = 1/4 * pi * inner_diameter^2;
        shell_side.cross_area = 1/4 * pi * shell_diameter^2 - no_of_fibers * 1/4 * pi * outer_diameter^2;
        membrane_area = no_of_fibers * pi * inner_diameter * length;
        membrane_thickness = outer_diameter - inner_diameter;

    elseif method == "Design" then
        packing_density.spec      : fixed;
        packing_density.value     : 4500;
        ratio_cross_area.spec     : fixed;
        ratio_cross_area.value    : 0.1;
        membrane_thickness.spec   : fixed;
        membrane_thickness.value  : 100e-6;
        membrane_area.spec       : fixed;
        membrane_area.value      : 5;
        constant_velocity         : fixed;
        constant_velocity         : 0.05;

        no_of_fibers.spec        : fixed;
        no_of_fibers.value       : 1;
        inner_diameter.spec      : fixed;
        inner_diameter.value     : 100e-6;
        outer_diameter.spec      : fixed;
        outer_diameter.value     : 1;
        shell_diameter.spec      : fixed;
        shell_diameter.value     : 1;

    Endif
//-----
// *** Connection between plug flow models and transport model***
    For z in Nset do
        // for fiber side:
        fiber_side.fraction_mol(componentlist)(z) = fiber_side.info_transport.fraction_mol(componentlist)(z);
        fiber_side.pressure_(z) = fiber_side.info_transport.pressure_(z);
    Endfor

```

```

        fiber_side.temperature_(z) = fiber_side.info_transport.temperature_(z);
        (1/flow_scale) * fiber_side.flux_mol_membr(componentlist)(z) = (1/flow_scale) * fiber_side.result_transport.flux_mol_membr(componentlist)(z);
        (1/flow_scale) * fiber_side.energy_flux_membr(z) = (1/flow_scale) * fiber_side.result_transport.energy_flux_membr(z);
    // for shell side:
        shell_side.fraction_mol(componentlist)(z) = shell_side.info_transport.fraction_mol(componentlist)(z);
        shell_side.pressure_(z) = shell_side.info_transport.pressure_(z);
        shell_side.temperature_(z) = shell_side.info_transport.temperature_(z);
        (1/flow_scale) * shell_side.flux_mol_membr(componentlist)(z) = (1/flow_scale) * shell_side.result_transport.flux_mol_membr(componentlist)(z);
        (1/flow_scale) * shell_side.energy_flux_membr(z) = (1/flow_scale) * shell_side.result_transport.energy_flux_membr(z);

Endfor
//-----
// *** Connection between model ports and plug flow models ***
// for fiber side
    fiber_side.axial.DiscretizationMethod : "BFD2" ;
    fiber_side.axial.SpacingPreference: 1/no_of_nodes;
    fiber_side.NsetMod : [axial.interior + axial.endnode];
    // Port connections at z = 0
        (1/flow_scale) * fiber_side.import.F = (1/flow_scale) * fiber_side.flux_mol_total.value(0) * fiber_side.cross_area * no_of_fibers ;
        fiber_side.import.Z = fiber_side.fraction_mol.value(0);
        fiber_side.import.T = fiber_side.temperature_.value(0);
        fiber_side.import.P = fiber_side.pressure_.value(0);
        fiber_side.import.h = fiber_side.spec_enth_mol.value(0);
        fiber_side.import.V * fiber_side.dens_mol_.value(0) = 1;
    // Port connections at z = 1
        (1/flow_scale) * fiber_side.export.F = (1/flow_scale) * (fiber_side.import.F - no_of_fibers * (fiber_side.flux_mol_total.value(0) -
        fiber_side.flux_mol_total.value(axial.endnode))) * fiber_side.cross_area;
        fiber_side.export.Z = fiber_side.fraction_mol.value(axial.endnode);
        fiber_side.export.T = fiber_side.temperature_.value(axial.endnode);
        fiber_side.export.P = fiber_side.pressure_.value(axial.endnode);
        fiber_side.export.h = fiber_side.spec_enth_mol.value(axial.endnode);
        fiber_side.export.V * fiber_side.dens_mol_.value(axial.endnode) = 1;

// for shell side
    If run_mode == "CoCurrent" then
        shell_side.axial.DiscretizationMethod: "BFD2" ;
        shell_side.axial.SpacingPreference: 1/no_of_nodes;
        shell_side.NsetMod : [axial.interior + axial.endnode];
        // Port connections at z = 0
            (1/flow_scale) * shell_side.import.F = (1/flow_scale) * shell_side.flux_mol_total.value(0) * shell_side.cross_area;
            shell_side.import.Z = shell_side.fraction_mol.value(0);
            shell_side.import.T = shell_side.temperature_.value(0);
            shell_side.import.P = shell_side.pressure_.value(0);
            shell_side.import.h = shell_side.spec_enth_mol.value(0);
            shell_side.import.V * shell_side.dens_mol_.value(0) = 1;
        // Port connections at z = 1
            (1/flow_scale) * shell_side.export.F = (1/flow_scale) * shell_side.flux_mol_total.value(axial.endnode) * shell_side.cross_area;
            shell_side.export.Z = shell_side.fraction_mol.value(axial.endnode);

```

```

        shell_side.export.T = shell_side.temperature_.value(axial.endnode);
        shell_side.export.P = shell_side.pressure_.value(axial.endnode);
        shell_side.export.h = shell_side.spec_enth_mol.value(axial.endnode);
        shell_side.export.V * shell_side.dens_mol_.value(axial.endnode) = 1;
elseif run_mode == "Countercurrent" then
    shell_side.axial.DiscretizationMethod: "FFD2" ;
    shell_side.axial.SpacingPreference: 1/no_of_nodes;
    shell_side.NsetMod : [0 + axial.interior];
    // Port connections at z = 1
        (1/flow_scale) * shell_side.import.F = - (1/flow_scale) * shell_side.flux_mol_total.value(axial.endnode) * shell_side.cross_area;
        shell_side.import.Z = shell_side.fraction_mol.value(axial.endnode);
        shell_side.import.T = shell_side.temperature_.value(axial.endnode);
        shell_side.import.P = shell_side.pressure_.value(axial.endnode);
        shell_side.import.h = shell_side.spec_enth_mol.value(axial.endnode);
        shell_side.import.V * shell_side.dens_mol_.value(axial.endnode) = 1;
    // Port connections at z = 0
        (1/flow_scale) * shell_side.export.F = - (1/flow_scale) * shell_side.flux_mol_total.value(0) * shell_side.cross_area;
        shell_side.export.Z = shell_side.fraction_mol.value(0);
        shell_side.export.T = shell_side.temperature_.value(0);
        shell_side.export.P = shell_side.pressure_.value(0);
        shell_side.export.h = shell_side.spec_enth_mol.value(0);
        shell_side.export.V * shell_side.dens_mol_.value(0) = 1;

Endif;
//-----
// *** Connection of variables ***
    shell_side.length = length;
    fiber_side.length = length;
    membrane.membrane_thickness = membrane_thickness;
//-----
End

```

### *Sub-model flow through fibre and shell side*

```

Model PlugFlow
//=====
// PARAMETER DEFINITION
// *** Global parameters ***
      R                                as global RealParameter;      // Gas constant [8.314 J/mol/K]
      pi                               as global RealParameter;      // [3.14159]
//-----
// *** Parameters for discretisation method ***
      no_of_nodes                      as global IntegerParameter;    // Number of nodes
//-----
// *** Scaling factors ***
      flow_scale                       as global RealParameter;      // Scaling factor
//-----
// *** Unit conversion factors ***
      hr_to_s                         as global RealParameter;      // Conversion factor time: hours to seconds
      bar_to_Pa                       as global RealParameter;      // Conversion factor pressure: bar to Pascal
      cP_to_Pas                       as global RealParameter;      // Conversion factor viscosity: cP to Pa.s
      GJ_to_J                         as global RealParameter;      // Conversion factor energy: GJ to J
//-----
// *** Defintion of simulation mode ***
      sim_mode                        as global SimMode;              // Parameter to change level of model
//=====
// PORT DEFINITION (cannot define port with bi-direction)
      import                          as input MoleFractionPort;
      export                          as output MoleFractionPort;
      info_transport                  as output InfoTransport;
      result_transport                as input ResultTransport;
//=====
// DISTRIBUTION DOMAIN DEFINITION
      axial                          as LengthDomain;
//=====
// DOMAIN SET DEFINITION
      Nset                           as global IntegerSet;
      NsetMod                        as IntegerSet;
//=====
// VARIABLE DEFINITION
// *** Geometric variables ***
      length                         as length;                      // Length of fibers [m]
      B                             as notype;                     // Geometric factor [1/m]
      cross_area                     as area;                      // Total cross sectional area of 1 side [m2]
      volume_                        as volume;                    // Total volume of 1 side [m3]
//-----
// *** Variables for mass balance ***
      conc_mol(componentlist)        as Distribution1D (XDomain is axial) of conc_mole (initial); // Molar concentration of component i [kmol/m3]

```



```

flux_mol(componentlist)      as Distribution1D (XDomain is axial)    of notype;          // Molar flux of component i [kmol/m2/hr]
flux_mol_total               as Distribution1D (XDomain is axial)    of notype;          // Molar flux [kmol/m2/hr]
fraction_mol(componentlist)  as Distribution1D (XDomain is axial)    of molefraction;    // Molefraction [kmol/kmol]
flux_mol_membr(componentlist) as Distribution1D (XDomain is axial)    of notype;          // Molar flux component i through membr [kmol/m2/hr]
//-----
// *** Variables for momentum balance ***
pressure_                    as Distribution1D (XDomain is axial)    of pressure;        // Pressure [bar]
velocity_                    as Distribution1D (XDomain is axial)    of velocity_rev;    // Velocity [m/s]
dens_mol_                    as Distribution1D (XDomain is axial)    of dens_mol;        // Molar density [kmol/m3]
momentum_holdup              as Distribution1D (XDomain is axial)    of notype (initial); // Holdup of momentum [kg/m3*m/s]
momentum_flux                as Distribution1D (XDomain is axial)    of notype;          // Momentum flux [kg/m2/hr*m/s]
viscosity_                   as Distribution1D (XDomain is axial)    of viscosity;        // Viscosity [0.001 * cP = Pa*s]
fr_factor                    as Distribution1D (XDomain is axial)    of notype;          // Friction factor [Pa/m = kg/m2/s2]
//-----
// *** Variables for energy balance ***
temperature_                 as Distribution1D (XDomain is axial)    of temperature;     // Temperature [C]
spec_enth_mol                as Distribution1D (XDomain is axial)    of enth_mol;        // Molar specific enthalpy [GJ/kmol]
energy_holdup                as Distribution1D (XDomain is axial)    of notype (initial); // Holdup of energy [GJ/m3]
energy_flux                  as Distribution1D (XDomain is axial)    of notype;          // Energy flux [GJ/m2/hr]
energy_flux_membr            as Distribution1D (XDomain is axial)    of notype;          // Energy flux through membr [GJ/m2/hr]
//-----
// *** Auxiliary variables ***
mol_weight_mix               as Distribution1D (XDomain is axial)    of molweight;       // Molecular weight of mixture [kg/kmol]
//=====
// EQUATIONS
// *** Balance equations ***
For z in NsetMod do
  flux_mol_total(z) = SIGMA(flux_mol(Componentlist)(z));
  If sim_mode == "level_1" then
    flux_mol(componentlist)(z).ddx = 0;
    pressure_(z) = import.P;
    temperature_(z) = import.T;
  elseif sim_mode == "level_2" then
    (1/flow_scale) * $conc_mol(componentlist)(z) = (1/flow_scale) * (- 1/length * flux_mol(componentlist)(z).ddx + B * flux_mol_membr(componentlist)(z));
    pressure_(z) = import.P;
    temperature_(z) = import.T;
  elseif sim_mode == "level_3" then
    (1/flow_scale) * $conc_mol(componentlist)(z) = (1/flow_scale) * (- 1/length * flux_mol(componentlist)(z).ddx + B * flux_mol_membr(componentlist)(z));
    $momentum_holdup(z) = - bar_to_Pa * hr_to_s * 1/length * pressure_(z).ddx - 1/length * momentum_flux(z).ddx - hr_to_s * fr_factor(z);
    temperature_(z) = import.T;
  elseif sim_mode == "level_4" then
    (1/flow_scale) * $conc_mol(componentlist)(z) = (1/flow_scale) * (- 1/length * flux_mol(componentlist)(z).ddx + B * flux_mol_membr(componentlist)(z));
    $momentum_holdup(z) = - bar_to_Pa * hr_to_s * 1/length * pressure_(z).ddx - 1/length * momentum_flux(z).ddx - hr_to_s * fr_factor(z);
    $energy_holdup(z) = - 1/length * energy_flux(z).ddx + B * energy_flux_membr(z);
  elseif sim_mode == "level_5" then
    (1/flow_scale) * $conc_mol(componentlist)(z) = (1/flow_scale) * (- 1/length * flux_mol(componentlist)(z).ddx + B * flux_mol_membr(componentlist)(z));

```

```

$momentum_holdup(z) = - bar_to_Pa * hr_to_s * 1/length * pressure_(z).ddx - 1/length * momentum_flux(z).ddx - hr_to_s * fr_factor(z);
$energy_holdup(z) = - 1/length * energy_flux(z).ddx + B * energy_flux_membr(z);

Endif

Endfor

//-----
// *** Auxiliary equations ***
For z in Nset do
    flux_mol(componentlist)(z) = hr_to_s * velocity_(z) * conc_mol(componentlist)(z);
    flux_mol(componentlist)(z) = fraction_mol(componentlist)(z) * flux_mol_total(z);
    dens_mol_(z) = SIGMA(conc_mol(componentlist)(z));
    momentum_holdup(z) = dens_mol_(z) * mol_weight_mix(z) * velocity_(z);
    momentum_flux(z) = mol_weight_mix(z) * abs(flux_mol_total(z)) * velocity_(z);
    energy_holdup(z) = dens_mol_(z) * spec_enth_mol(z) - bar_to_Pa * 1/GJ_to_J * pressure_(z);
    energy_flux(z) = flux_mol_total(z) * spec_enth_mol(z);

Endfor

//-----
// *** Call for physical properties ***
For z in Nset do
    call(mol_weight_mix(z)) = pMolWeight(fraction_mol.value(z));
    call(viscosity_(z)) = pVisc_Vap(temperature_.value(z),pressure_.value(z),fraction_mol.value(z));

Endfor
For z in Nset do
    call(spec_enth_mol(z)) = pEnth_Mol_Vap(temperature_.value(z),pressure_.value(z),fraction_mol.value(z));
    call(dens_mol_(z)) = pDens_Mol_Vap(temperature_.value(z),pressure_.value(z),fraction_mol.value(z));

Endfor

//=====
End

```

### *Sub-model flow/transport of components through the membrane*

Model Transport

```
//=====
// PARAMETER DEFINITION
// *** Global parameters ***
      R                                as global RealParameter;      // Gas constant [8.314 J/mol/K]
      pi                               as global RealParameter;      // [3.14159]
//-----
// *** Parameters for discretisation method ***
      no_of_nodes                      as global IntegerParameter;    // Number of nodes
//-----
// *** Scaling factors ***
      flow_scale                      as global RealParameter;        // Scaling factor
//-----
// *** Unit conversion factors ***
      hr_to_s                         as global RealParameter;        // Conversion factor time: hours to seconds
      bar_to_Pa                       as global RealParameter;        // Conversion factor pressure: bar to Pascal
      cP_to_Pas                       as global RealParameter;        // Conversion factor viscosity: cP to Pa.s
      GJ_to_J                         as global RealParameter;        // Conversion factor energy: GJ to J
//-----
// *** Defintion of parameters for feed side and simulation mode ***
      feed_side                       as global FeedSide;             //
      sim_mode                        as global SimMode;               // Parameter to change level of model
      membr_type                      as global MembrType;             // Parameter to change membrane type: "Symmetric" or "Asymmetric"
//-----
// *** Parameters for separative character of the membrane ***
      permeability0(componentlist)    as RealParameter;              // Permeability at T0
      permeability0(componentlist)    : 1e-14;
      activation_energy(componentlist) as RealParameter;              // Activation energy [J/mol]
      activation_energy(componentlist) : 10000;
      temperature0                    as RealParameter;              // Standard temperature [C]
      temperature0                    : 25;
//=====
// PORT DEFINITION
      shell_info                      as input  InfoTransport;
      shell_result                    as output ResultTransport;
      fiber_info                      as input  InfoTransport;
      fiber_result                    as output ResultTransport;
//=====
// DISTRIBUTION DOMAIN DEFINITION
      axial                          as external LengthDomain;
//=====
// DOMAIN SET DEFINITION
      Nset                           as global IntegerSet;
//=====
```

```

// VARIABLE DEFINITION
// *** Geometric variables
membrane_thickness          as area;                      // Membrane thickness [m]
//-----
// *** Variable for separative character of the membrane ***
permeability(componentlist) as Distribution1D(XDomain is axial) of notype;    // Permeability
//-----
// *** Auxiliary variables
fraction_mol_membr(componentlist) as Distribution1D(XDomain is axial) of molefraction; // Molar fraction of component i in membrane [kmol/kmol]
spec_enth_mol_fiber          as Distribution1D(XDomain is axial) of enth_mol;    // Specific molar enthalpy of flux trough membr [GJ/mol]
spec_enth_mol_shell          as Distribution1D(XDomain is axial) of enth_mol;    // Specific molar enthalpy of flux trough membr [GJ/mol]
//=====
// EQUATIONS
For z in 0 + axial.interior + axial.endnode do
    // *** Component fractions in membrane and enthalpy of mass flux ***
    fraction_mol_membr(componentlist)(z) * SIGMA (ABS(shell_result.flux_mol_membr(componentlist)(z)) ) = ABS (shell_result.flux_mol_membr(componentlist)(z)) ;
//-----
    // *** Permeability (Arrhenius funtion) ***
    If feed_side == "Shell" then
        (1/1e-14) * permeability(componentlist)(z) = (1/1e-14) * permeability0(componentlist) * exp ( - activation_energy(componentlist) / R * ( 1 /
        (shell_info.temperature_(z) + 273.15) - 1 / (temperature0 + 273.15) ) );
        call(spec_enth_mol_fiber(z)) = pEnth_Mol_Vap(shell_info.temperature_(z),fiber_info.pressure_(z),fraction_mol_membr(componentlist)(z));
        call(spec_enth_mol_shell(z)) = pEnth_Mol_Vap(shell_info.temperature_(z),shell_info.pressure_(z),fraction_mol_membr(componentlist)(z));
    elseif feed_side == "Fiber" then
        (1/1e-14) * permeability(componentlist)(z) = (1/1e-14) * permeability0(componentlist) * exp ( - activation_energy(componentlist) / R * ( 1 /
        (fiber_info.temperature_(z) + 273.15) - 1 / (temperature0 + 273.15) ) );
        call(spec_enth_mol_fiber(z)) = pEnth_Mol_Vap(fiber_info.temperature_(z),fiber_info.pressure_(z),fraction_mol_membr(componentlist)(z));
        call(spec_enth_mol_shell(z)) = pEnth_Mol_Vap(fiber_info.temperature_(z),shell_info.pressure_(z),fraction_mol_membr(componentlist)(z));
    Endif
//-----
    // *** Molar and energy flux through membrane: connection between two sides ***
    (1/flow_scale) * Fiber_result.Flux_mol_membr(componentlist)(z) = - (1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z);
//-----
    // *** Molar flux through membrane ***
    If sim_mode == "Level_1" then
        Shell_result.Flux_mol_membr(componentlist)(z) = 0;
    elseif sim_mode == "Level_2" then
        (1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) / membrane_thickness *
        1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) - shell_info.fraction_mol(componentlist)(z) * shell_info.pressure_(z));
    elseif sim_mode == "Level_3" then
        If membr_type == "Symmetric" then
            (1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
            membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) - shell_info.fraction_mol(componentlist)(z) *
            shell_info.pressure_(z));
        elseif membr_type == "Asymmetric" then
            If feed_side == "Shell" then

```

```

(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
membrane_thickness * 1e5 * (fraction_mol_membr(componentlist)(z) * fiber_info.pressure_(z) -
shell_info.fraction_mol(componentlist)(z) * shell_info.pressure_(z));
elseif feed_side == "Fiber" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) -
fraction_mol_membr(componentlist)(z) * shell_info.pressure_(z));
Endif
Endif
elseif sim_mode == "Level_4" then
If membr_type == "Symmetric" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) - shell_info.fraction_mol(componentlist)(z) *
shell_info.pressure_(z));
elseif membr_type == "Asymmetric" then
If feed_side == "Shell" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
membrane_thickness * 1e5 * (fraction_mol_membr(componentlist)(z) * fiber_info.pressure_(z) -
shell_info.fraction_mol(componentlist)(z) * shell_info.pressure_(z));
elseif feed_side == "Fiber" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability0(componentlist) /
membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) -
fraction_mol_membr(componentlist)(z) * shell_info.pressure_(z));
Endif
Endif
elseif sim_mode == "Level_5" then
If membr_type == "Symmetric" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability(componentlist)(z) /
membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) - shell_info.fraction_mol(componentlist)(z) *
shell_info.pressure_(z));
elseif membr_type == "Asymmetric" then
If feed_side == "Shell" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability(componentlist)(z) /
membrane_thickness * 1e5 * (fraction_mol_membr(componentlist)(z) * fiber_info.pressure_(z) -
shell_info.fraction_mol(componentlist)(z) * shell_info.pressure_(z));
elseif feed_side == "Fiber" then
(1/flow_scale) * Shell_result.Flux_mol_membr(componentlist)(z) = (1/flow_scale) * 3600 * 1e-3 * permeability(componentlist)(z) /
membrane_thickness * 1e5 * (fiber_info.fraction_mol(componentlist)(z) * fiber_info.pressure_(z) -
fraction_mol_membr(componentlist)(z) * shell_info.pressure_(z));
Endif
Endif
Endif

//-----
// *** Energy flux through membrane ***
(1/flow_scale) * Shell_result.energy_flux_membr(z) = (1/flow_scale) * SIGMA(shell_result.flux_mol_membr(componentlist)(z)) * spec_enth_mol_shell(z);
(1/flow_scale) * Fiber_result.energy_flux_membr(z) = - (1/flow_scale) * SIGMA(shell_result.flux_mol_membr(componentlist)(z)) * spec_enth_mol_fiber(z);

```

```
Endfor
//=====
End
```

## ***Visual Basic Script***

```
global.sim_mode.value = "level_1"  
application.simulation.runmode = "steady state"  
application.simulation.run(true)  
application.msg "**** Simulation level_1 model completed ****"
```

```
global.sim_mode.value = "level_2"  
application.simulation.runmode = "steady state"  
application.simulation.run(true)  
application.msg "**** Simulation level_2 model completed ****"
```

```
global.sim_mode.value = "level_3"  
application.simulation.runmode = "steady state"  
application.simulation.run(true)  
application.msg "**** Simulation level_3 model completed ****"
```

```
global.sim_mode.value = "level_4"  
application.simulation.runmode = "steady state"  
application.simulation.run(true)  
application.msg "**** Simulation level_4 model completed ****"
```

```
global.sim_mode.value = "level_5"  
application.simulation.runmode = "steady state"  
application.simulation.run(true)  
application.msg "**** Simulation level_5 model completed ****"
```

## Appendix C Specification of parameters and variables for membrane model

The values of the following parameters should be specified:

- Physical property package
- Numerical discretisation methods
- Number of discretisation intervals

The following variables must be specified

- |   |    |   |
|---|----|---|
| <ul style="list-style-type: none"><li>• Number of fibres, <math>N</math></li><li>• Fibre inner diameter, <math>D_{inner}</math></li><li>• Fibre outer diameter, <math>D_{outer}</math></li><li>• Module length, <math>L</math></li><li>• Shell diameter, <math>D_{shell}</math></li></ul> | or | <ul style="list-style-type: none"><li>• Packing density, <math>\rho_{unit}</math></li><li>• Ratio cross sectional areas, <math>\lambda</math></li><li>• Membrane thickness, <math>\delta^m</math></li><li>• Membrane area, <math>A_{membrane}</math></li><li>• Constant for velocity of feed stream</li></ul> |
|---|----|---|
- 
- Permeability at standard temperature,  $Q_{i0}$
  - Standard temperature,  $T_0$
  - Activation energy,  $E_{ai}$

The following boundary conditions need to be specified:

- Feed and sweep flowrate
- Feed and sweep composition
- Feed and sweep pressure
- Feed and sweep temperature



## Appendix D Results membrane model for comparison

**Table D.1: Results counter-current membrane unit with asymmetric membrane**

Feed (kmol/hr)	Permeate (kmol/hr)	Stage cut (%)	Molar fraction of components in permeate (%)			
			H <sub>2</sub>	Ar	CH <sub>4</sub>	N <sub>2</sub>
4.00E-05	2.515E-05	62.88	80.74	2.89	7.12	9.26
5.00E-05	2.991E-05	59.82	83.91	2.52	5.90	7.68
6.00E-05	3.455E-05	57.58	86.18	2.23	5.03	6.56
7.00E-05	3.907E-05	55.82	87.88	2.00	4.39	5.73
8.00E-05	4.350E-05	54.38	89.21	1.81	3.90	5.08
9.00E-05	4.783E-05	53.15	90.27	1.66	3.50	4.57
1.00E-04	5.208E-05	52.08	91.15	1.53	3.18	4.15
1.20E-04	6.030E-05	50.26	92.49	1.32	2.68	3.50
1.40E-04	6.818E-05	48.70	93.48	1.16	2.32	3.03
1.60E-04	7.568E-05	47.31	94.24	1.04	2.05	2.68
1.80E-04	8.280E-05	46.00	94.83	0.94	1.83	2.40
2.00E-04	8.951E-05	44.76	95.30	0.86	1.66	2.18
2.40E-04	1.016E-04	42.35	95.99	0.74	1.41	1.85
2.80E-04	1.120E-04	40.00	96.47	0.66	1.24	1.63
3.20E-04	1.207E-04	37.74	96.81	0.60	1.12	1.47
3.60E-04	1.281E-04	35.59	97.06	0.55	1.04	1.36
4.00E-04	1.343E-04	33.59	97.25	0.52	0.97	1.27

**Table D.2: Results counter-current membrane unit with symmetric membrane**

Feed (kmol/hr)	Permeate (kmol/hr)	Stage cut (%)	Molar fraction of components in permeate (%)			
			H <sub>2</sub>	Ar	CH <sub>4</sub>	N <sub>2</sub>
4.00E-05	2.550E-05	63.74	81.06	2.77	7.03	9.14
5.00E-05	3.047E-05	60.95	84.28	2.39	5.80	7.54
6.00E-05	3.528E-05	58.81	86.54	2.11	4.93	6.42
7.00E-05	3.994E-05	57.06	88.21	1.89	4.30	5.60
8.00E-05	4.446E-05	55.58	89.50	1.72	3.81	4.97
9.00E-05	4.886E-05	54.29	90.53	1.57	3.43	4.47
1.00E-04	5.315E-05	53.15	91.37	1.45	3.11	4.06
1.20E-04	6.140E-05	51.16	92.67	1.27	2.63	3.44
1.40E-04	6.924E-05	49.46	93.61	1.12	2.28	2.98
1.60E-04	7.669E-05	47.93	94.34	1.01	2.02	2.64
1.80E-04	8.373E-05	46.52	94.90	0.92	1.81	2.37
2.00E-04	9.037E-05	45.18	95.36	0.84	1.65	2.15
2.40E-04	1.023E-04	42.64	96.03	0.73	1.40	1.84
2.80E-04	1.126E-04	40.22	96.50	0.65	1.24	1.62
3.20E-04	1.213E-04	37.91	96.83	0.59	1.12	1.46
3.60E-04	1.286E-04	35.73	97.07	0.55	1.03	1.35
4.00E-04	1.348E-04	33.70	97.26	0.51	0.96	1.26

**Table D.3: Results co-current membrane unit with symmetric membrane**

Feed (kmol/hr)	Permeate (kmol/hr)	Stage cut (%)	Molar fraction of components in permeate (%)			
			H <sub>2</sub>	Ar	CH <sub>4</sub>	N <sub>2</sub>
4.00E-05	2.346E-05	58.65	79.14	3.31	7.62	9.93
5.00E-05	2.774E-05	55.49	82.46	2.90	6.36	8.29
6.00E-05	3.199E-05	53.32	84.89	2.56	5.44	7.11
7.00E-05	3.622E-05	51.74	86.76	2.29	4.75	6.20
8.00E-05	4.043E-05	50.54	88.24	2.06	4.20	5.50
9.00E-05	4.462E-05	49.58	89.44	1.87	3.77	4.92
1.00E-04	4.880E-05	48.80	90.43	1.71	3.41	4.45
1.20E-04	5.708E-05	47.57	91.98	1.45	2.85	3.73
1.40E-04	6.520E-05	46.57	93.11	1.26	2.44	3.19
1.60E-04	7.304E-05	45.65	93.97	1.11	2.13	2.79
1.80E-04	8.051E-05	44.73	94.64	0.99	1.89	2.48
2.00E-04	8.756E-05	43.78	95.16	0.90	1.71	2.24
2.40E-04	1.002E-04	41.75	95.91	0.76	1.44	1.88
2.80E-04	1.109E-04	39.62	96.42	0.67	1.26	1.65
3.20E-04	1.199E-04	37.48	96.77	0.61	1.13	1.49
3.60E-04	1.275E-04	35.40	97.03	0.56	1.04	1.37
4.00E-04	1.338E-04	33.45	97.23	0.52	0.97	1.28

## Appendix E Work processes interfacing

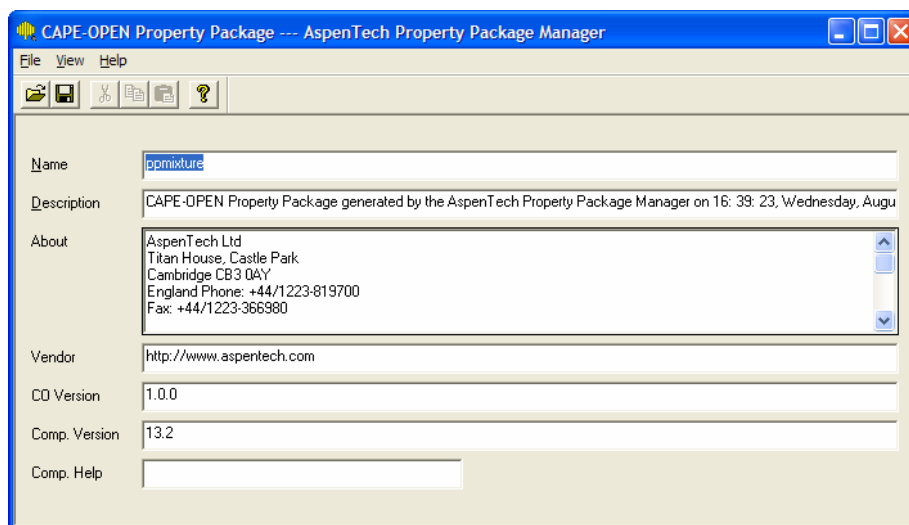
### ***E.1 Work process physical property package interfacing Aspen Plus → gPROMS***

- *Aspen Plus 12.1.12 (Sept. 2005: not possible yet to use Aspen Plus version 12.1.13 and Aspen Plus 2004.1/13.2 because of a bug in software/interfaces)*
- *gPROMS version 2.3 or alpha version 3.0*

#### What to do in Aspen Plus?

To use a physical property package from Aspen Plus in gPROMS a \*.cota file should be created. Steps to take in Aspen Properties or Aspen Plus are:

1. Open or create an \*.bkp file in Aspen Plus (when no \*.bkp file exists, the runtime of the simulation can be set to Properties Plus in Data Browser / Setup / Specifications / Global / Run Type → Properties Plus).
2. Select Tools → Export CAPE-OPEN Package.  
The following window is displayed:



3. Select File → Save as or click on the save button in the toolbar of this window.
4. Save the \*.cota file in the same folder as you are going to store your gPROMS project file.
5. Close Aspen Plus.

## What to do in gPROMS?

### ***Incorporation of \*.cota file in project file:***

1. Enter in MODEL entity – PARAMETER section:  
*ParameterName* AS FOREIGN\_OBJECT “CapeOpenThermo”
2. Enter in PROCESS entity – SET section:  
ParameterPath :=  
“CapeOpenThermo::(PS)ATCOProperties.COPropertySystem.1<NamePPfile>”;

For example:

```
Model entity  
PARAMETER  
phys_prop AS FOREIGN_OBJECT "CapeOpenThermo"
```

```
Process entity  
SET  
unit1.phys_prop :=  
“(PS)ATCOProperties.COPropertySystem.1<PPmixture>”
```

### ***Calling for a physical property from the package:***

In the EQUATION section of the MODEL entity of your gPROMS project you can set up an equation to call for a physical property. As input for the physical property calculation temperature (in Kelvin), pressure (in Pascal) and molar fraction (mol/mol) of a phase have to be given. The property package returns its answer also on molar basis (/mol).

For example, if you want to calculate the enthalpy of a liquid phase, in the EQUATION section you should write:

```
enthalpy = phys_prop.LiquidEnthalpy (temperature, pressure, fraction_mol);
```

In Table E.1 a more comprehensive list is given of all the properties which can be called using the CAPE-OPEN interface. Before the property name, the state of the system should be given with the one of these words: Liquid, Vapour or Overall.

The number of components is not stated in this list but can also be called by using the property name “NumberOfComponents”, like:

```
no_comp = phys_prop.NumberOfComponents;
```

**The order of the components in the array defined for the molar fraction of the system is of importance. The physical property package expects to get the component in the molar fraction array in the same order as the order defined in Aspen Plus when creating the package.**

**Table E.1: Thermodynamic and physical properties defined in CAPE-OPEN interface**

+++ Properties +++

ACTIVITYCOEFFICIENT  
DENSITY  
DIFFUSIONCOEFFICIENT  
DIFFUSIONCOEFFICIENT.DMOLES  
DIFFUSIONCOEFFICIENT.DMOLFRACTION  
DIFFUSIONCOEFFICIENT.DTEMPERATURE  
ENTHALPY  
ENTHALPY.DMOLES  
ENTHALPY.DMOLFRACTION  
ENTHALPY.DPRESSURE  
ENTHALPY.DTEMPERATURE  
ENTROPY  
ENTROPY.DMOLES  
ENTROPY.DMOLFRACTION  
ENTROPY.DPRESSURE  
ENTROPY.DTEMPERATURE  
FUGACITYCOEFFICIENT  
FUGACITYCOEFFICIENT.DMOLES  
FUGACITYCOEFFICIENT.DMOLFRACTION  
FUGACITYCOEFFICIENT.DPRESSURE  
FUGACITYCOEFFICIENT.DTEMPERATURE  
GIBBSFREEENERGY  
GIBBSFREEENERGY.DMOLES  
GIBBSFREEENERGY.DMOLFRACTION  
GIBBSFREEENERGY.DPRESSURE  
GIBBSFREEENERGY.DTEMPERATURE  
HEATCAPACITY  
MOLECULARWEIGHT  
NORMALBOILINGPOINT  
PH  
SOLUBILITYINDEX  
SURFACETENSION  
SURFACETENSION.DMOLES  
SURFACETENSION.DMOLFRACTION  
SURFACETENSION.DTEMPERATURE  
THERMALCONDUCTIVITY  
THERMALCONDUCTIVITY.DMOLES  
THERMALCONDUCTIVITY.DMOLFRACTION  
THERMALCONDUCTIVITY.DTEMPERATURE  
VAPORPRESSURE  
VISCOSITY  
VISCOSITY.DMOLES  
VISCOSITY.DMOLFRACTION  
VISCOSITY.DPRESSURE  
VISCOSITY.DTEMPERATURE  
VOLUME  
VOLUME.DMOLES  
VOLUME.DMOLFRACTION  
VOLUME.DPRESSURE  
VOLUME.DTEMPERATURE

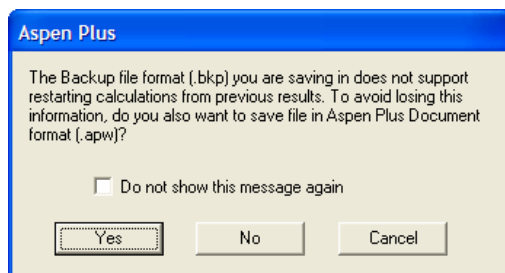
## ***E.2 Work process physical property package interfacing Aspen Plus → ACM***

- *Aspen Plus 2004.1 version 13.2 (interface also possible with Aspen Plus version 12.1)*
- *ACM 2004.1 version 13.2 (interface also possible with ACM version 12.1)*

### *What to do in Aspen Plus?*

To use a physical property package from Aspen Plus in ACM an \*.appdf file should be created. Steps to take in Aspen Plus are:

1. Open or create a \*.bkp file in Aspen Plus (when no \*.bkp file exists, the runtime of the simulation can be set to Properties Plus in Data Browser / Setup / Specifications / Global / Run Type → Properties Plus).
2. After running a simulation, save the \*.bkp file, automatically is asked if also the \*.apw file has to be saved. Click 'Yes'.

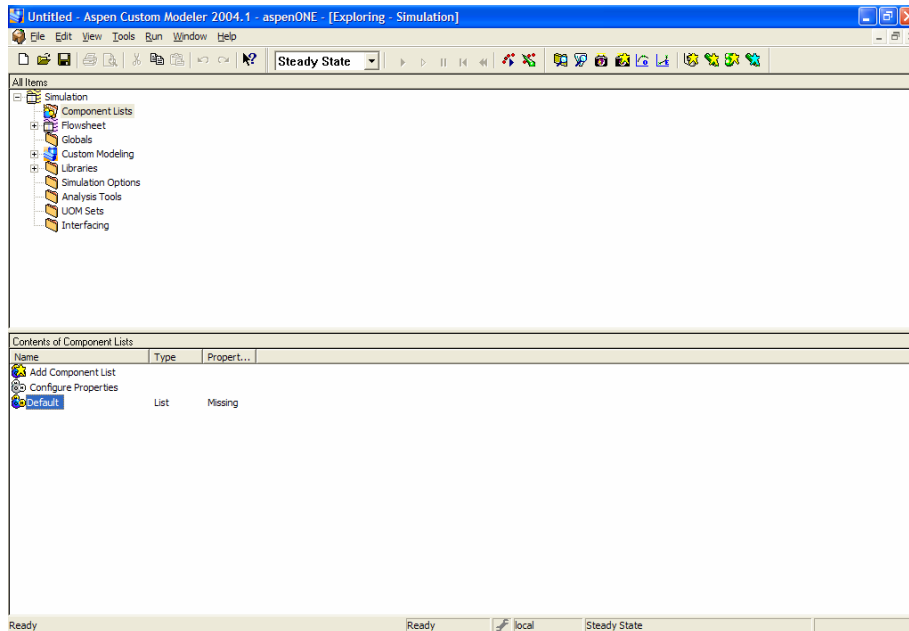


3. Together with the \*.bkp file and the \*.apw file, an \*.appdf file is saved in the same directory. The files created by Aspen Plus should be located in the same directory as you are going to save the ACM file.
4. Close Aspen Plus.

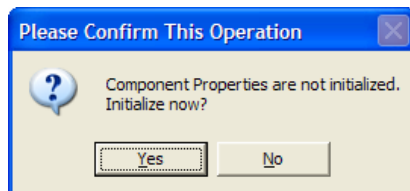
## What to do in ACM?

### ***Incorporation of \*.appdf file in ACM model:***

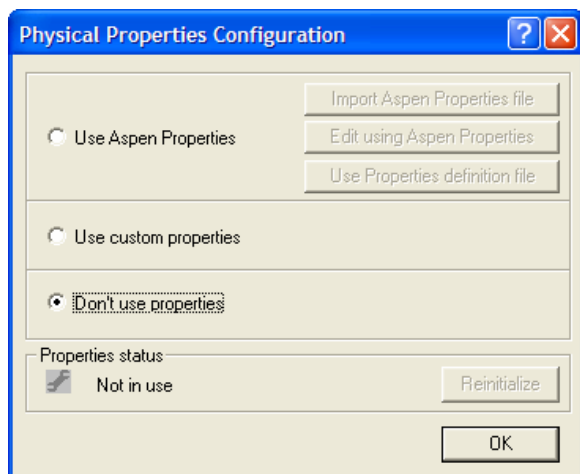
1. Select 'Component lists' from the Exploring-Simulation pane 'All items'.
2. Double click on 'Default' in the Exploring-Simulation pane 'Contents of Component Lists'.



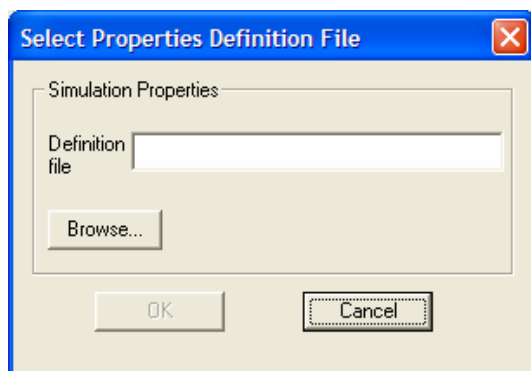
3. The next window is displayed. Click 'Yes':



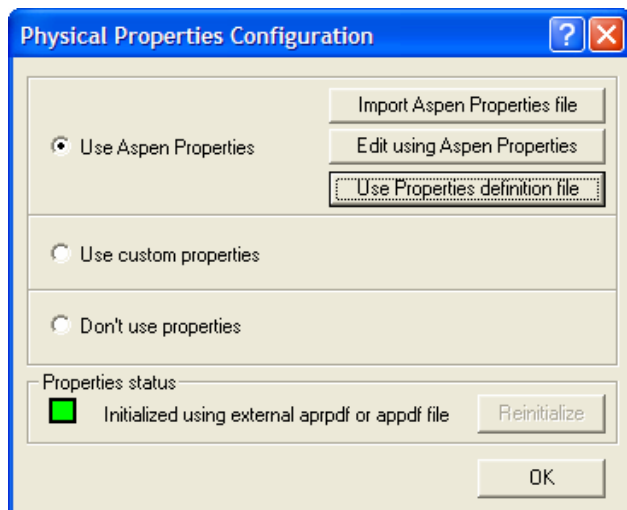
4. The 'physical property configuration' window is displayed. Select 'Use Aspen Properties' and then 'Use Properties definition file'.



5. The next window is displayed. Browse for the definition file (the \*.appdf file) and click OK.

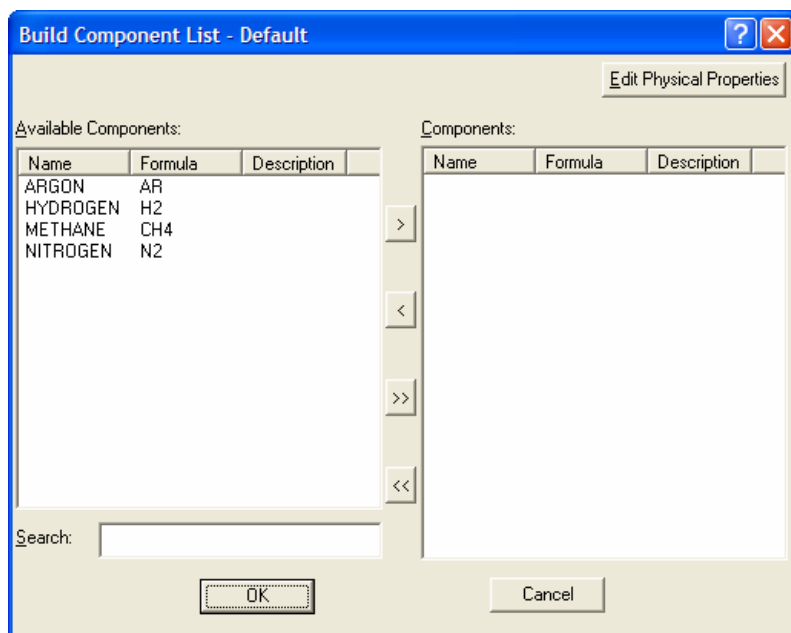


6. The former window appears again, check if the properties status is green. Click OK.

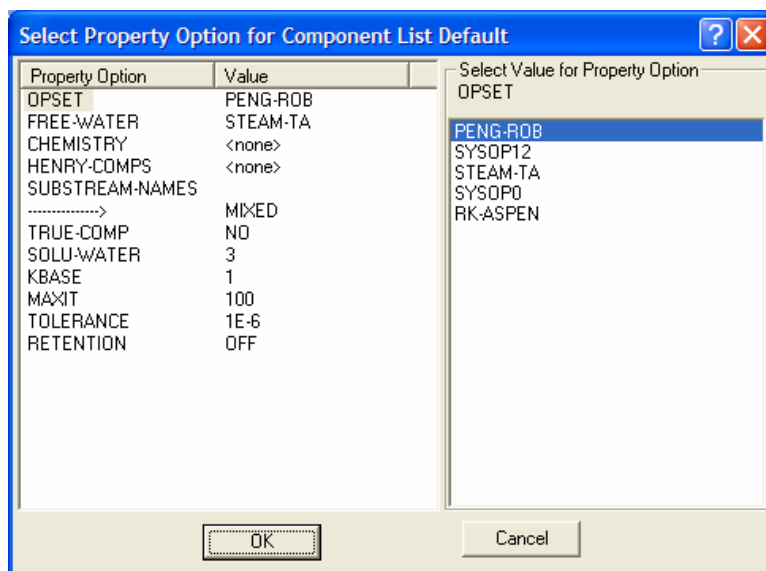




7. The next window is displayed. Select the components you want to use in the ACM model and bring them to the right pane.



8. To see or change the physical properties defined in the Default Component List, select 'Edit Physical Properties'. This window is displayed:

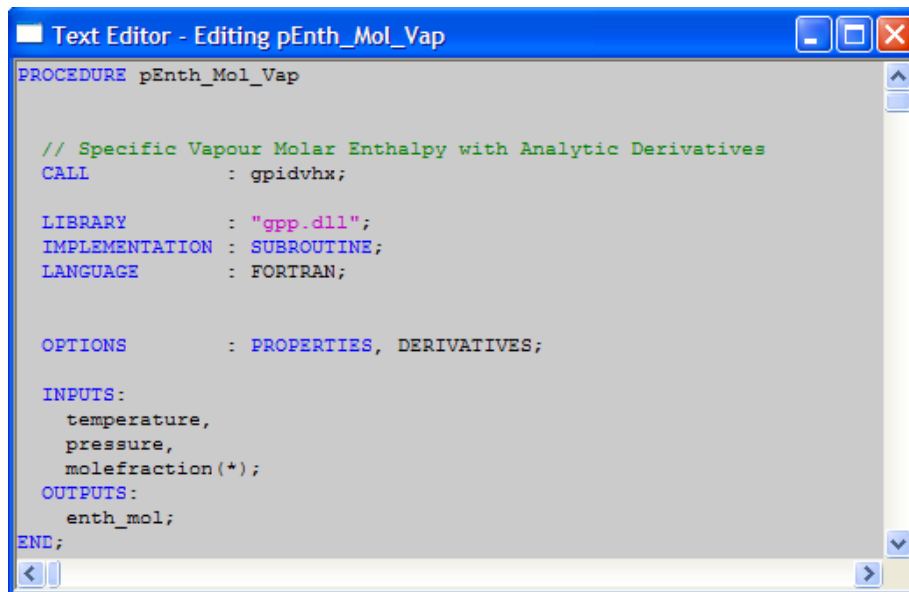


9. Click twice OK and the physical property package for the Default Component List in ACM is loaded successfully.

### *Calling for a physical property from the package:*

You can use procedure calls in models to use an external routine to make calculations. The procedure must be defined in a procedure definition. For a lot of thermodynamic and physical properties the procedures are already included in the library of Aspen Custom Modeler.

1. Select and expand 'Libraries' and then 'Modeler' from the Exploring-Simulation pane "All items".
2. Select Procedures. In the content pane a list of all predefined procedures are given.
3. By double click on one of the procedures a window will appear in which the syntax is written for this particular procedure.



```
PROCEDURE pEntH_Mol_Vap

// Specific Vapour Molar Enthalpy with Analytic Derivatives
CALL      : gpdivhx;

LIBRARY    : "gpp.dll";
IMPLEMENTATION : SUBROUTINE;
LANGUAGE   : FORTRAN;

OPTIONS    : PROPERTIES, DERIVATIVES;

INPUTS:
  temperature,
  pressure,
  molefraction(*);
OUTPUTS:
  enth_mol;
END;
```

4. In the text editor of the model the following line should be added:

CALL (OutputArgumentList) =  
      ProcedureName( InputArgumentList ) ComponentList;

Where:

- ProcedureName = Name of the procedure to use
- InputArgumentList = List of input variables in the same order as the procedure definition
- OutputArgumentList = List of output variables in the same order as the procedure definition
- ComponentList = Optional name of a component list that applies to this individual call. The component and thermodynamic properties associated with this component list are used in property calculation calls. A value is required only when you want to override the default component list.

For example

Call (enthalpy) = pEnth\_Mol\_Vap (temperature, pressure, fraction\_mol);

As input for the physical property calculation temperature (in Celsius), pressure (in bar) and molar fraction (kmol/kmol) of a phase have to be given. The physical property is returned on molar basis.

### **E.3 Work process unit model interfacing gPROMS → Aspen Plus**

- *Aspen Plus 12.1.12 (Sept. 2005: not (yet) possible to use Aspen Plus 2004.1/13.2 because of a bug in the port connection)*
- *gPROMS 3.0 (Sept. 2005: gPROMS 2.3.6 does not have the functionality to export CAPE-OPEN models. Version 3.0 is an alpha version of gPROMS, therefore the work process can be changed for the final release)*

**Prerequisite for exporting models from gPROMS: gPROMS version 3.0.0 and higher and the licenses gSRE\_4 and gSIM\_4.**

**If other versions of gPROMS are installed on the same machine, to be sure that no problems will occur:**

- **Check if the environment variable GPROMSHOME is set to the installation directory of the version you are going to use.**
- **Check if the %GPROMSHOME%\bin directory is included in the PATH environment variable.**
- **Open Windows command prompt (Start → Run → cmd). Go (with cd command) to installation directory\gPROMS\bin and type regsvr32 gPROMS\_COUnit.dll. The message 'DllRegisterServer in gPROMS\_COUnit.dll succeeded' is displayed.**

**Normally during the installation of gPROMS these things are automatically set correct.**

#### *How to export a model from gPROMS?*

This part of the work process is divided into model requirements and export functionality.

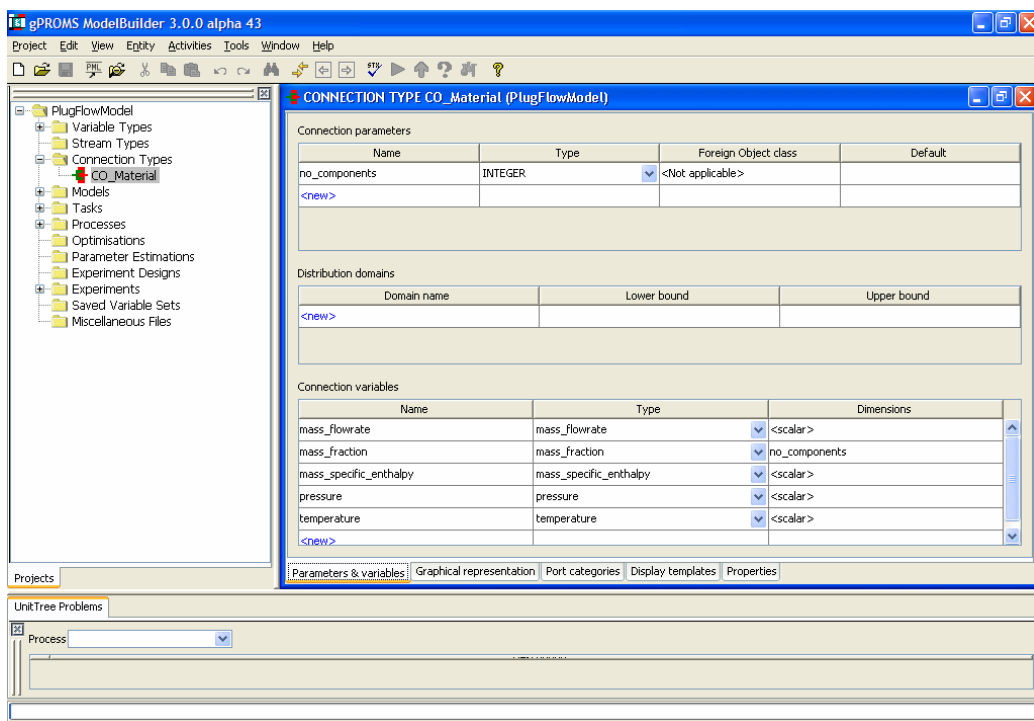
#### ***Model requirements***

##### **A. Port definition (obligatory)**

In order to link the gPROMS model within an Aspen Plus flowsheet simulation, the streams used in the model must conform the standard CAPE-OPEN definition.

gPROMS is supplied with a "CAPE-OPEN\_Unit" library (CAPE-OPEN\_Unit.gPJ in the directory PSE\gPROMS\ gOCAPEOPEN\examples) that contains a CAPE-OPEN material connection type, named "CO\_material", see figure below. To make your model CAPE-OPEN compliant, the connection type "CO\_material" have to be used for defining the input and output ports of the model you want to export. Streams from Aspen Plus can be connected to this connection type.

The variables in the "CO\_material" connection type/port are: flowrate [mol/s], fraction [mol/mol], pressure [Pa], temperature [K], specific enthalpy [J/mol]. The order of the variables defined in the connection type is not of importance. Note: the connection variables are on molar basis, and not on mass basis, as the CO\_material connection type indicates! PSE is going to correct this for the final release of gPROMS version 3.0.



Variables in the connection type/port with direction input are automatically set as 'get' when the model is exported. Variables in the connection type/port with direction output are automatically set as 'send'.

This means that values of the variables in the input port do not have to be calculated by the model, the model will receive these values from Aspen Plus. On the other hand the values of the variables in the output port have to be calculated by the model.

### B. Model specifications declared in PROCESS entity are not exported (obligatory)

The PROCESS section of the gPROMS project file is not exported; therefore you have to write everything what you want to export in the MODEL section. This also counts for the INITIALISE section. Initialisation of the model should be done in MODEL section instead of in PROCESS section. (This works only for version 3.0.0 or higher.)

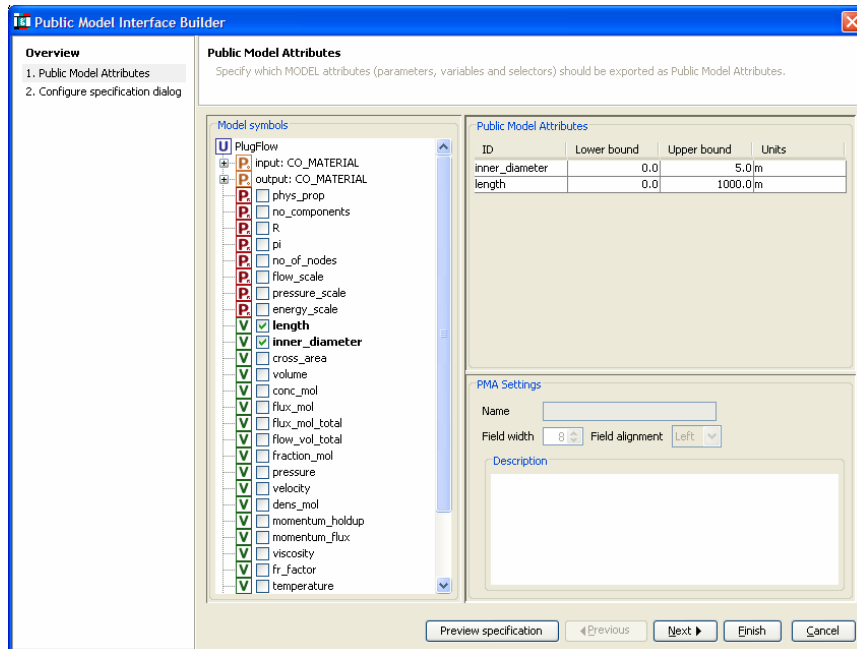
### C. Table with model attributes for export (optional)

gPROMS version 3.0 allows to have parameters or variables in its interface with CAPE-OPEN compliant process modelling environment. These parameters or variables can be adapted (manually or with a design specification) within the CAPE-OPEN flowsheet package (Aspen Plus). The gPROMS parameters or variables which are not defined in the interface as CAPE-OPEN parameters are not accessible in Aspen Plus. Currently, only gPROMS variables can be defined as CAPE-OPEN parameters. PSE is working on the option to include also gPROMS parameters as CAPE-OPEN parameters in the interface.

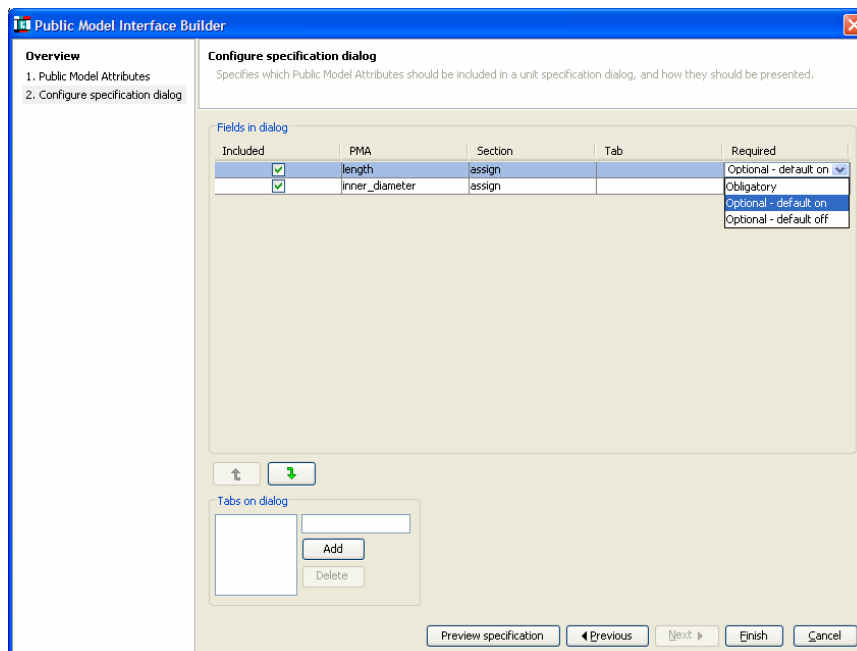
The following steps should be taken in order to create a table with model attributes which are available as CAPE-OPEN parameters in the interface:

1. Select the Interface tab at the bottom of your MODEL window.

- Click 'Edit specification...' in the left upper corner. This window is displayed:



- Select the model attributes (variables, parameters, selectors) you want to export as a table together with your model.
- Enter the lower and upper bound of the model attribute and its unit.
- Click 'Next' and select the model attributes again to be included in the table.



Depending on the definition of the model attribute (parameter/variable), and how they should be presented, choose for

- Section: Assign, Preset or Initial
- Required: Obligatory, Optional – default on or Optional – default off

6. Click 'Finish'.

#### **D. Saved variable set for export – initialisation (Optional)**

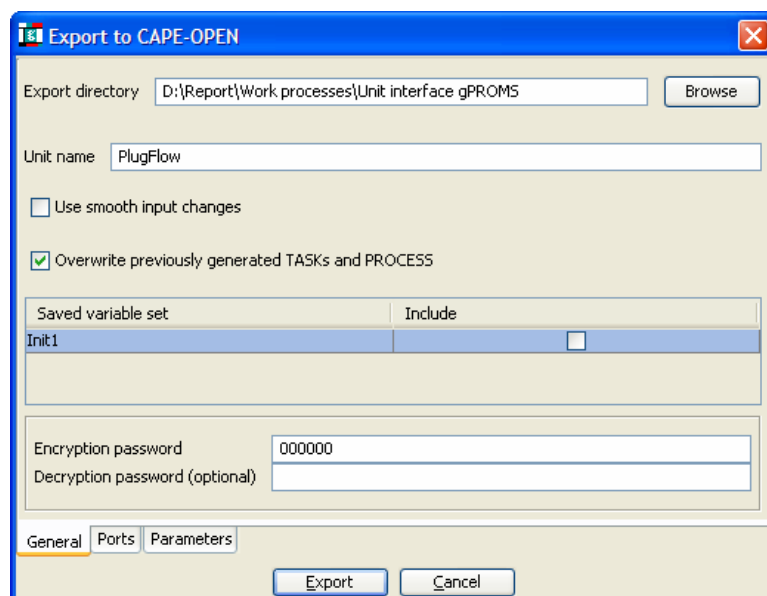
The following steps have to be done to create a saved variable (initialisation) set for your model:

1. In the 'Simulation activity' window, when starting a run, deselect 'Release model after execution'.
2. When the simulation is finished right click on the Execution output and select 'Create save variable set...'. The variable set is saved in the folder of the case in the project tree.
3. Make a copy of this text file by right clicking on the name and paste it in the 'Save variable set' of the project entity in the project tree.

## ***Export functionality***

By exporting a model from gPROMS, which is CAPE-OPEN compatible, a \*.gCO file is created. Steps to take for creating a \*.gCO file are:

1. Select the model entity in the project tree for the model you want to export. For models containing sub-models, select the main model entity.
2. Go to the Tools menu on the task bar of the gPROMS user interface and select Export to CAPE-OPEN.... This window is displayed:



In the export directory field you can change the directory where the \*.gCO file will be saved.

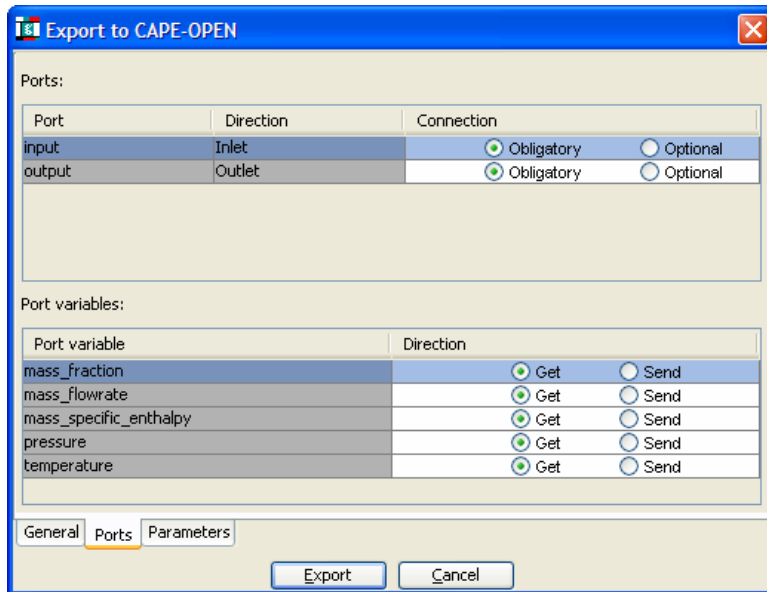
In the unit name field the name of the model which is exported is automatically displayed. Don't change this name!

If there are Saved variable sets created for the model, these can be exported together with the model by selecting Include.

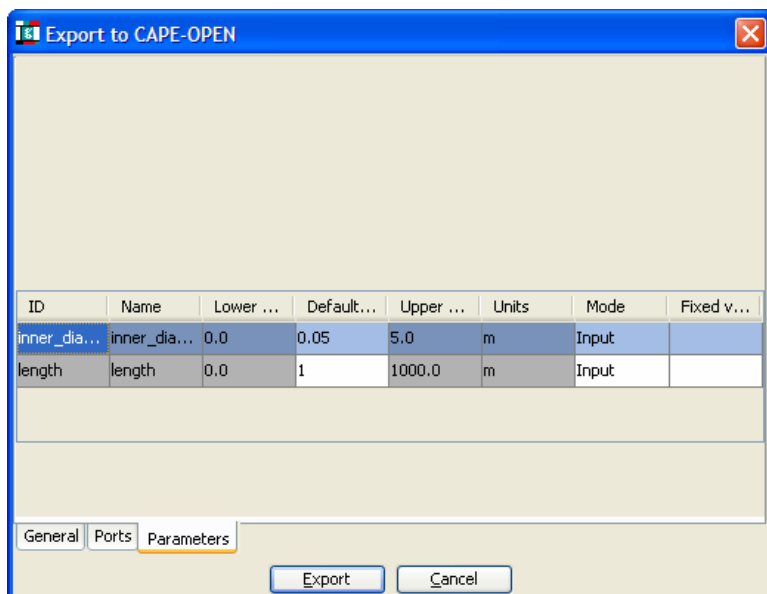
The exported file is encrypted in order to protect the model for unauthorised changes being made to it. An encryption password has to be entered.

3. Clicking on the Ports tap, shows this window below. In this window you can choose how the connection is established. For usage in an Aspen Plus simulation all the inlet variables have to set as "Get" and the outlet variables as 'Send'.





4. If you create a table with model attributes for exporting, the Parameter tab is also available:



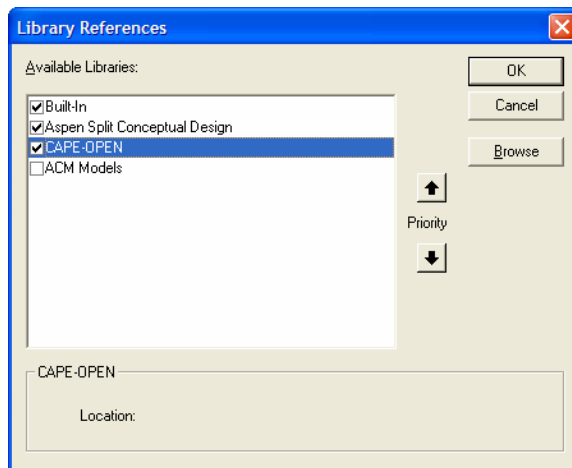
Don't enter Fixed values for the variables in this table. However, Default values for the variables can be entered here.

5. Click Export, and a \*.gCO file is created containing the model and some additional tasks in your project file.

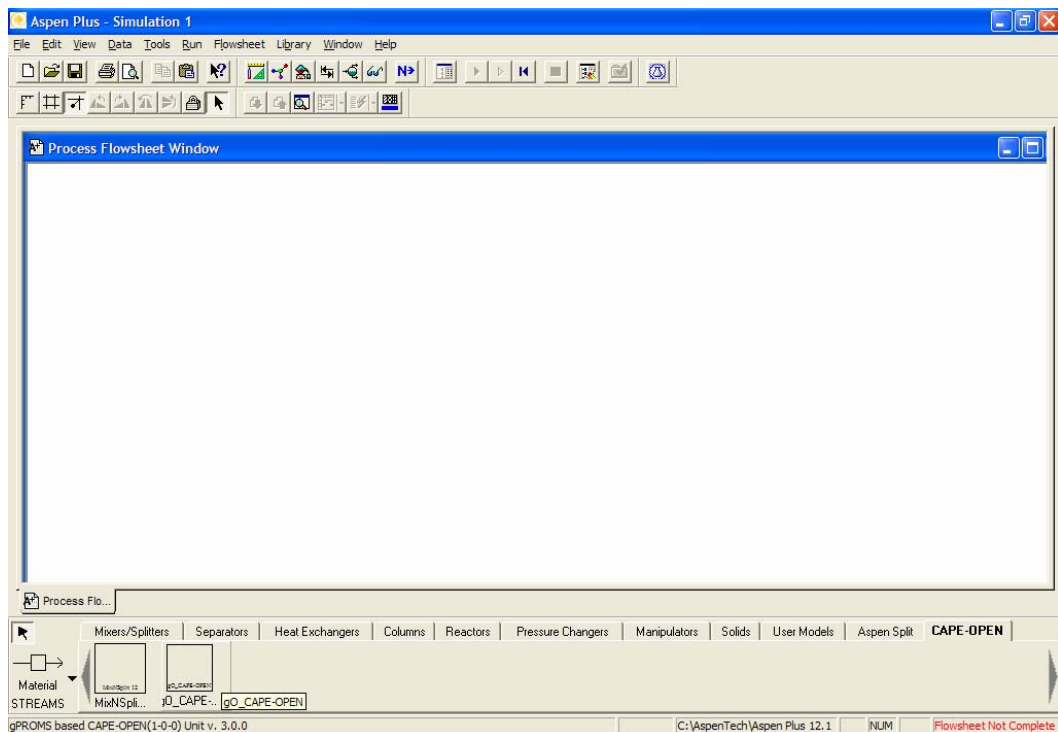
## How to import an exported gPROMS model (\*.gCO) in Aspen Plus?

Steps to follow are:

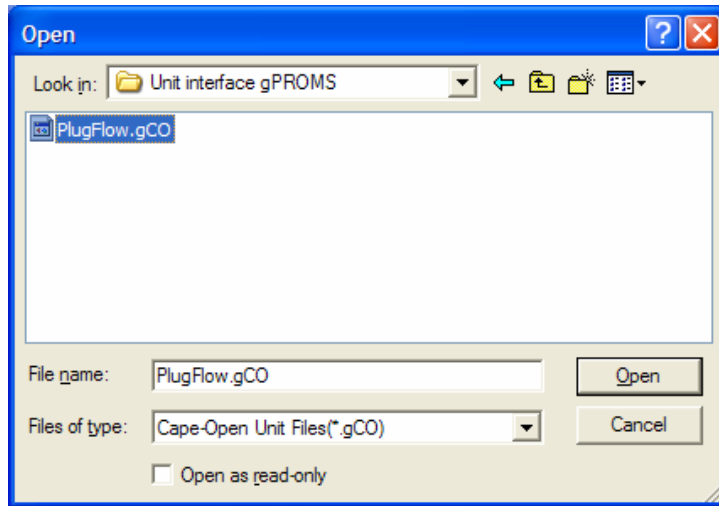
1. From the Library menu, click 'References'.
2. From the list of available libraries, select 'CAPE-OPEN' and click OK.



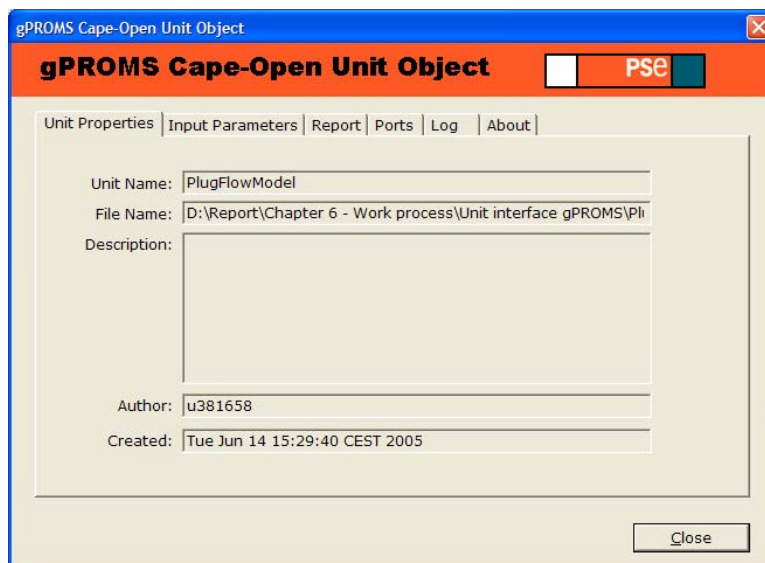
3. Select the CAPE-OPEN tab from the Model Library bar and drag the gO\_CAPE-OPEN block (gPROMS based CAPE-OPEN (1-0-0) Unit v.3.0.0) to the flowsheet.



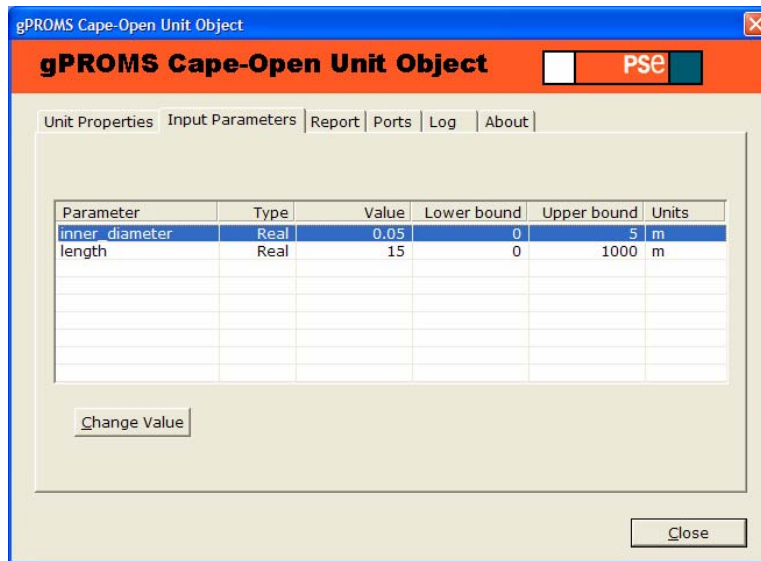
4. The next window will be displayed automatically (maybe hidden in the taskbar of Windows). Go to the directory containing the \*.gCO file, select it, and click OK.



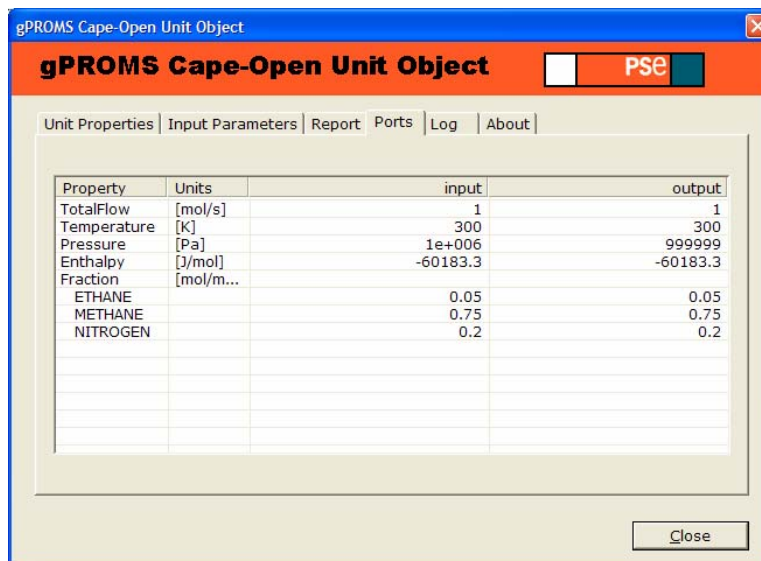
5. Doubling clicking on the CAPE-OPEN block in the Aspen Plus flowsheet results in the display of this window:



If model attributes (parameters/variables/etc.) are exported together with the model, the values of these attributes can be changed in the 'Input Parameters' tab:



6. As normal, streams have to be connected to the CAPE-OPEN block and the property method has to be defined in Aspen Plus.
7. After running the simulation the results of the CAPE-OPEN block are displayed among other things in the 'Ports' tab of the gPROMS CAPE-OPEN Unit Object block:



## ***E.4 Work process unit model interfacing ACM → Aspen Plus***

- *Aspen Plus 2004.1 version 13.2*
- *ACM 2004.1 version 13.2*

**Prerequisite for exporting models from ACM: a C++ compiler has to be installed on your computer! (Installation and running an exported model can be done without having a C++ compiler installed on your computer.)**

In the help content of ACM (“Using Aspen Custom Modeler → Exporting models to Aspen Plus and HYSYS”) an extended description is given of how to export models for usage in Aspen Plus. In this work process the highlights are summarized.

### ***How to export a model from ACM?***

This part of the work process is divided into model requirements and export functionality.

### ***Model requirements***

#### **A. Port definition (obligatory)**

Aspen Plus streams can only be connected to ports on exported models if the port types used in the model contain particular variables. For a port to be connected to a material stream in Aspen Plus it must contain at least the following variable definitions:

F as flow\_mol (Description: “Molar flow rate”)

T as temperature (Description: “Temperature”)

P as pressure (Description: “Pressure”)

V as vol\_mol (Description: “Molar volume”)

h as enth\_mol (Description: “Molar enthalpy”)

z (componentlist) as molefraction (Description: “Mole fractions”)

This port type (MoleFractionPort) is already predefined in the Modeler library and it is recommended to use this port type.

Variables in inlet ports are seen as fixed variables in Aspen Plus, whereas the variables in the outlet ports are seen as free variables, which need to be calculated by the ACM model.

#### **B. Specification of parameters and variables (optional)**

Only default values and specifications of parameters and variables are exported. Values of parameters or specifications (free/fixed/initial) of variables which are modified by the user in the AllVariables table are not exported together with the model. By modifying the specification and/or default value in the text editor of the model, these specification/values are taken along with the model export to Aspen Plus.

#### **C. Tables with model attributes for export (optional)**

Your model may own tables that you have defined to display particular sets of variables and parameters. These tables can be included in the exported model and displayed in Aspen Plus. By default ACM will include all tables belonging to a model in the exported

package. Use the Model Package Properties dialogs (see Export functionality) to exclude tables if necessary.

Aspen Plus distinguishes between input forms and results forms whereas ACM does not. Use the Model Package Properties dialogs (see Export functionality) to specify whether a form should be considered as an input form or a results form or both in Aspen Plus.

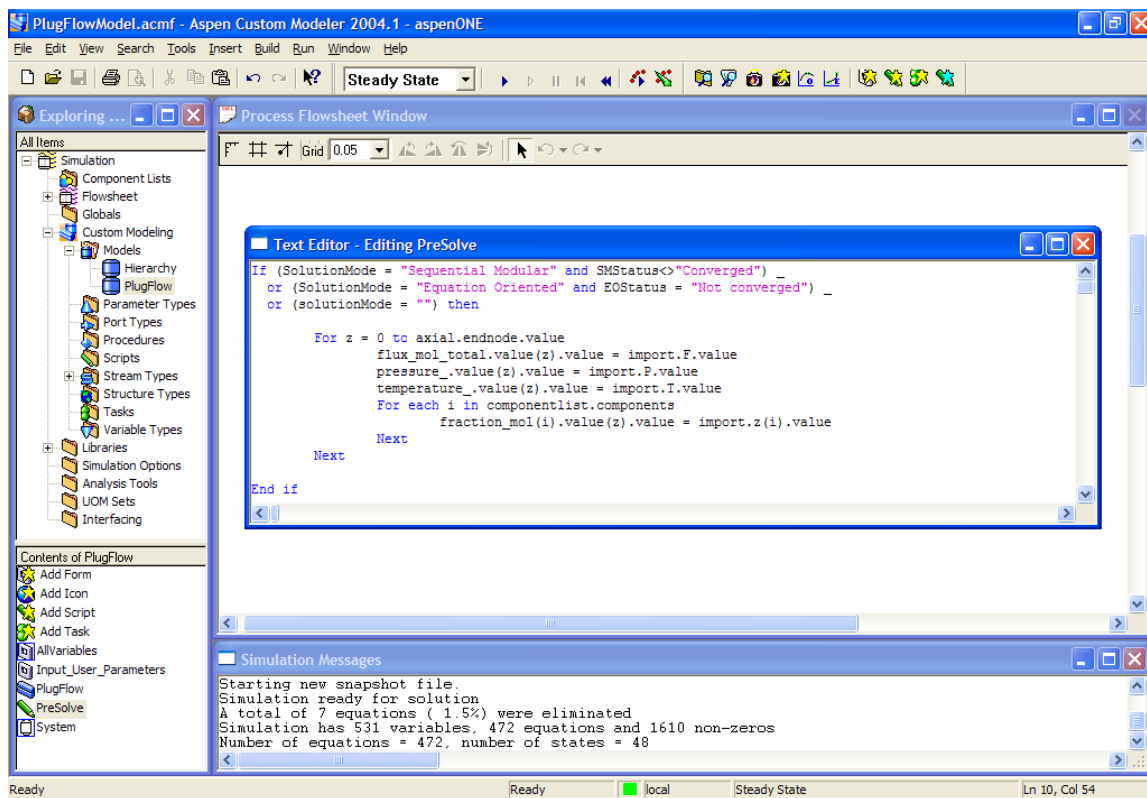
For creating new model tables, see the help content of ACM “Create a new model table”.

#### D. Visual Basic script for export - Initialisation (optional)

Your model’s Visual Basic scripts will be exported with a model and can be run in Aspen Plus. Visual Basic scripts are primarily intended to help with model initialisation.

Scripts with the standard names: “Presolve”, “Postsolve” and “Init” will be run at specific times during process in Aspen Plus.

The “Presolve” script runs immediately before solving an Aspen Plus flowsheet. In this script starting values can be given to variables and parameters defined in the ACM model block. An example of a Visual Basic script for initialisation of a custom model is given in PlugFlowModel.acmf:

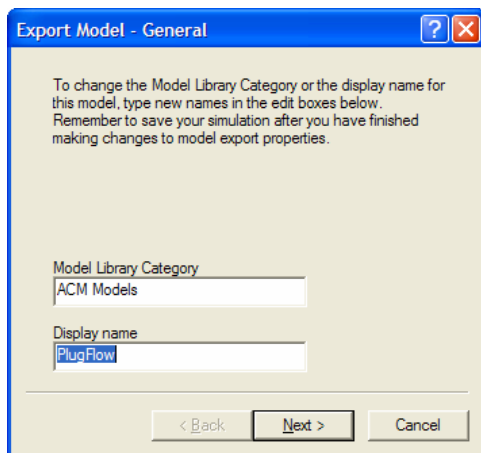


## *Export functionality*

### **A. Setting Model Package Properties**

Before exporting a model its properties can be modified by taking the following steps:

1. Right click on the model you want to export in the Explorer window and select the Model Package Properties menu item. This window is displayed:

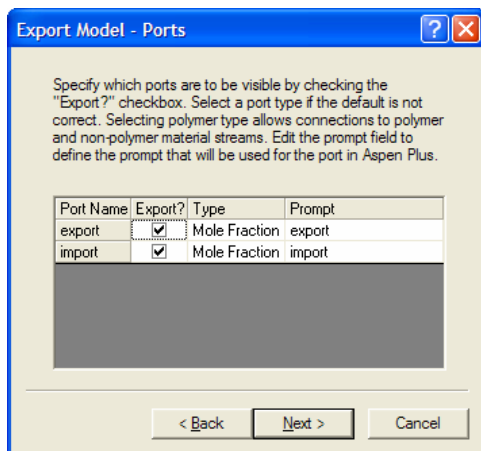


In the Model Library Category field it is possible to enter the name of the tab where you want the model to appear in Aspen Plus Model library (the standard category is 'ACM Models').

In the Display name field enter the name of the exported model.

Select 'Next'.

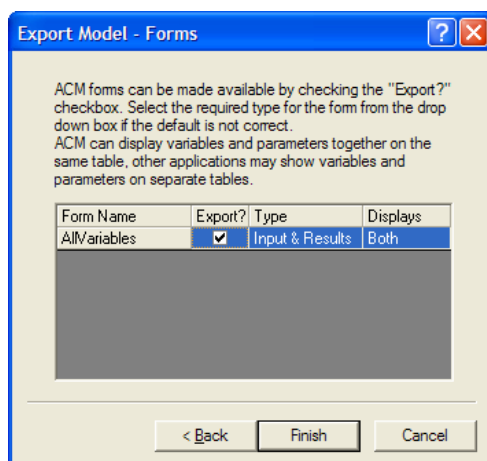
2. In the next window



you can control whether a port is accessible in the exported model and what its prompt should be.

Select 'Next'.

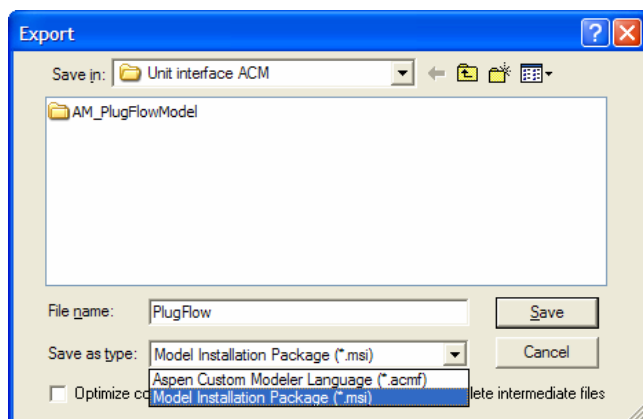
3. In the last window



you can control whether a form is included in the Model Package, whether it is an input, results or input & results form.  
Click 'Finish'.

## B. Export the model

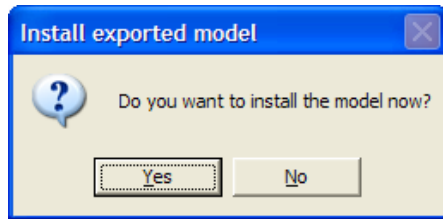
1. Right click on the model you want to export in the Explorer window and select the Export menu item.
2. On the Export dialog change the file type to \*.msi.



Navigate to the directory where you want ACM to create the Unit Operation Model. Select 'Save' to export the model.

3. When the Windows Installer Package for your model is complete Aspen Modeler will ask automatically if you want to install the model.

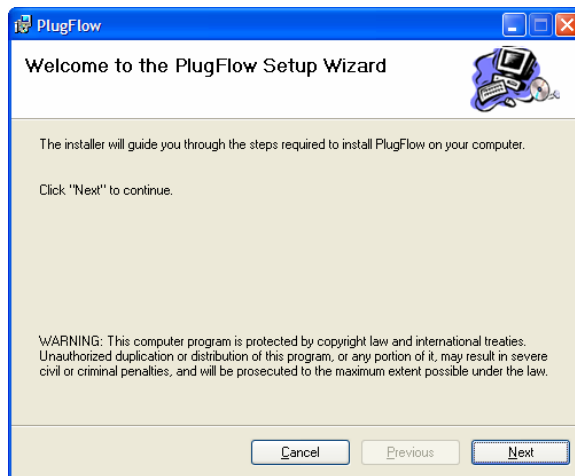




If you select 'Yes', follow the install instructions to install the model. If you select 'No' you can install the model manually later or on a different computer. Then, use the Windows Explorer to navigate to the directory where the \*.mis is saved and double click on it.

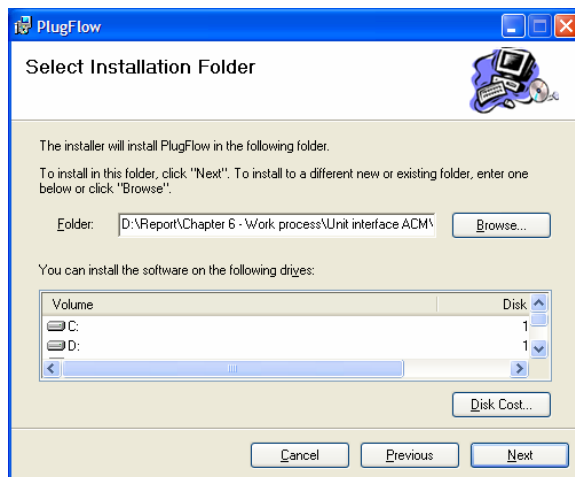
### C. Installing the model

1. The first window of the installation instruction looks as follows:



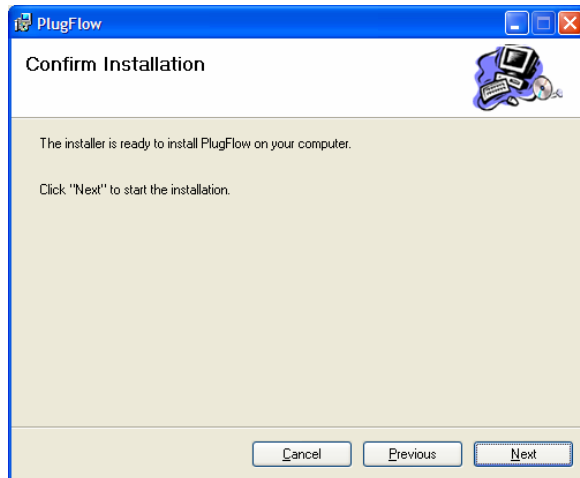
Select 'Next'.

2. In the next window

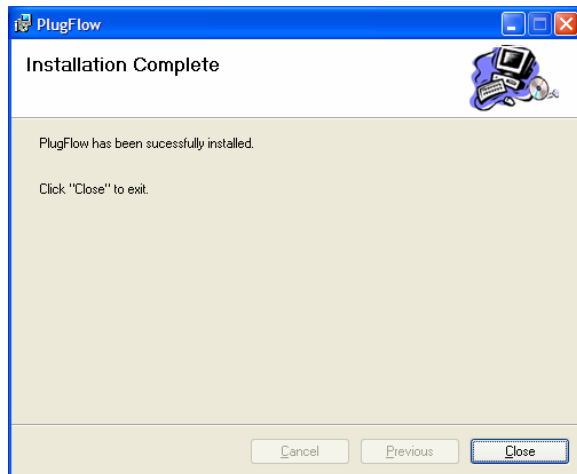


you can change the directory where the model is installed. The path of the directory isn't of importance.  
Select 'Next'.

3. This window is displayed and select 'Next' again.



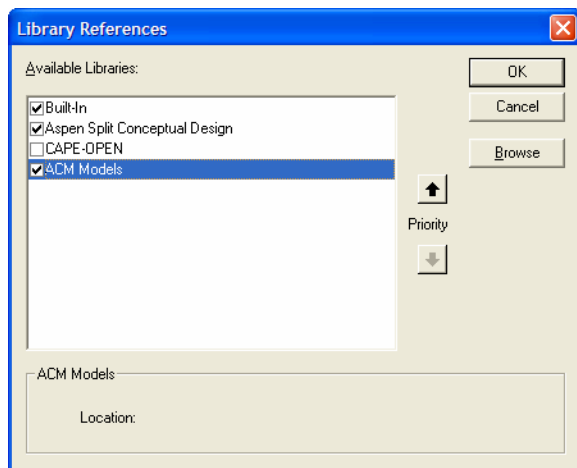
4. After the installation is completed, select 'Close' in the last window.



## How to import an exported ACM model in Aspen Plus?

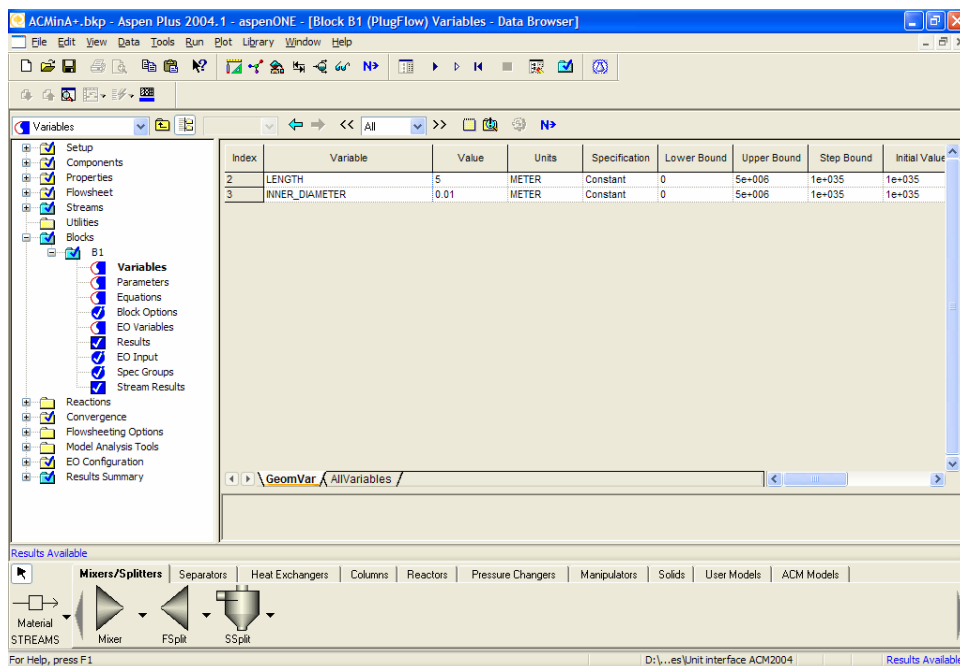
Steps to follow are:

1. From the Library menu, click 'References'.
2. From the list of available libraries, select 'ACM Models' and click OK.

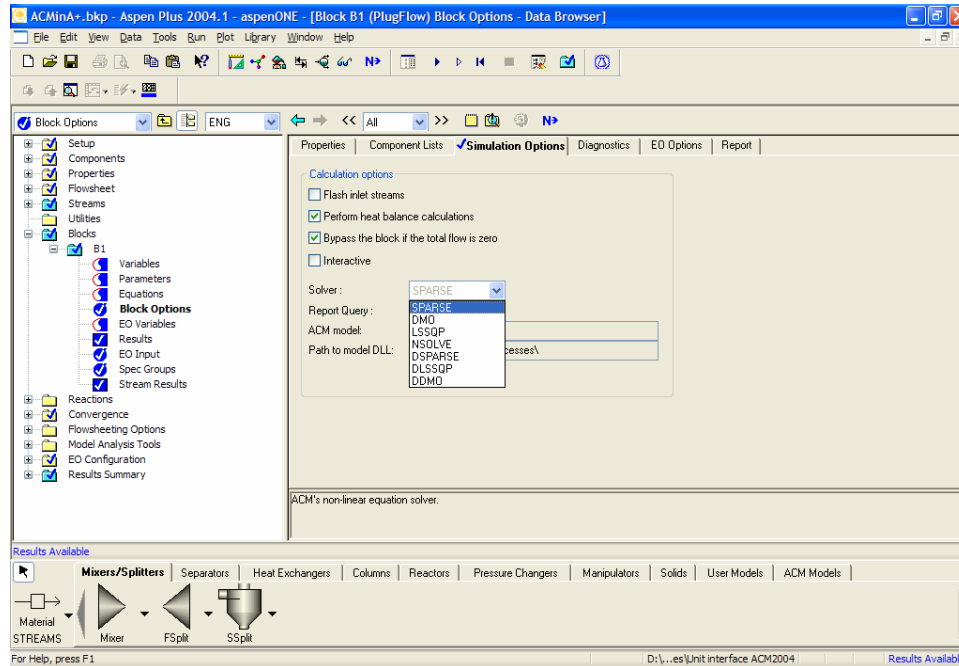


3. Select the ACM Models tab in the Model Library bar and drag the exported ACM model to the flowsheet.

As normal, streams can be connected to the ACM model block and parameters, property method and other options inside the block can be modified. If you exported tables together with the model, these can be displayed. In the block the values of defined variables and parameters in ACM can be changed.



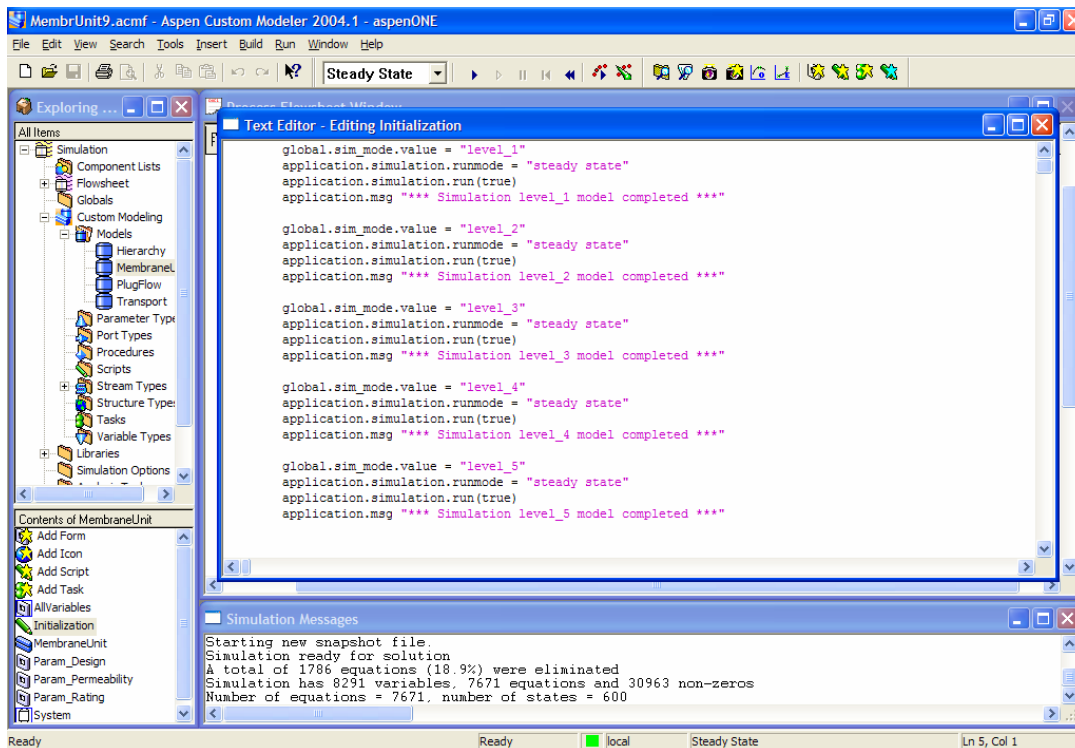
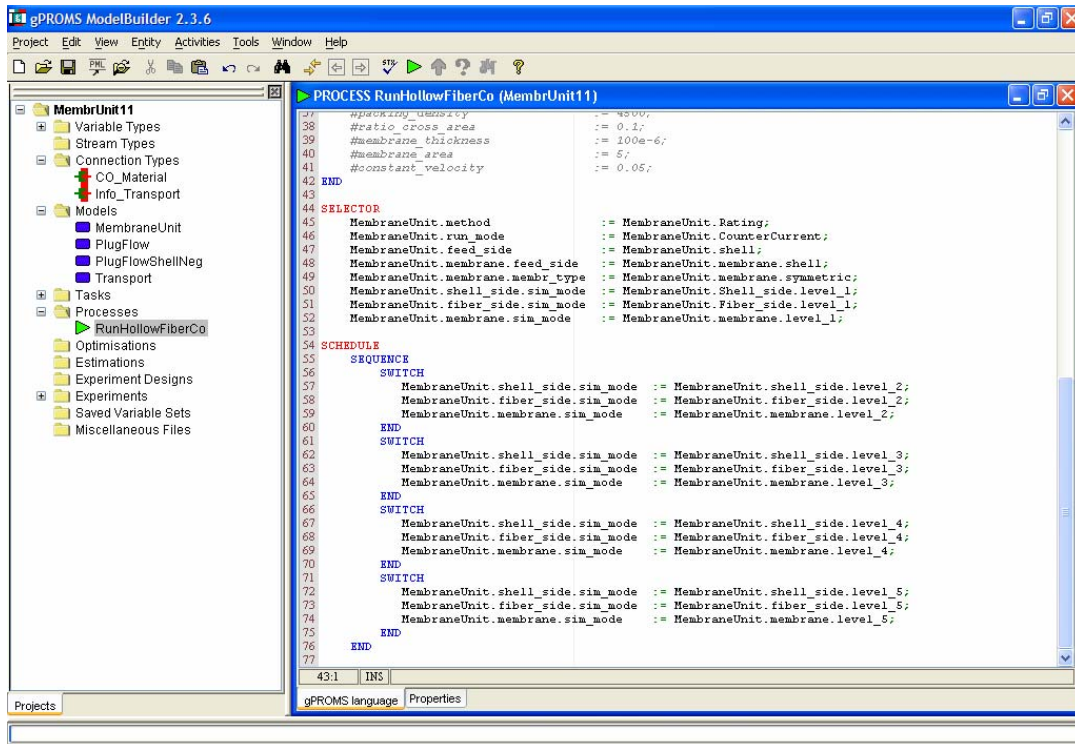
To solve the ACM block there are various solvers available in Aspen Plus. Select in block options the most appropriate solver for the model you developed. The letter D at the beginning of the name of the last three solvers stands for the ability of the solver to handle models which require block decomposition to converge.



It is for example also possible to put a design specification on a variable of the ACM block.

**Note: Before running an existing Aspen Plus simulation containing an ACM model block, the \*.dll file needs to be installed on the computer running Aspen Plus.**

## Appendix F Automation initialisation structure in equation oriented modelling tools



## Appendix G Definition base case

<b>Equipment &amp; system</b>	
Method	Rating
Run mode	Counter-current
Feed side	Shell
Sweep stream	Without
Membrane type	Symmetric
Number of components	4
Set of components	H2,N2,CH4,Ar

<b>Specifications</b>	
Length [m]	0.5
Inner diameter [m]	8.00E-05
Outer diameter [m]	2.00E-04
Shell diameter [m]	0.075
Number of fibres	50000
Physical property method	Peng Rob
Permeability0 [1e-14]	340.8;3.54;3.408;9.24
Activation energy [J/mol]	1E4;1E4;1E4;1E4
Standard temperature [K]	298.15
Number of discr. intervals	50

or

constant velocity [m/s]	0.05
packing density [m2/m3]	4500
ratio cross area	0.10
membrane area [m2]	5
membrane thickness [m]	1.00E-04

<b>Operation conditions</b>	
Feed flowrate [mol/s]	0.2083
Feed composition [mol/mol]	0.50;0.25;0.20;0.05
Feed pressure [bar]	70
Feed temperature [K]	298
Sweep flowrate [mol/s]	na
Sweep composition [mol/mol]	idem feed
Sweep pressure [bar]	10
Sweep temperature [K]	idem feed

## Appendix H Results case study initialisation structure

Table H.1: Discrete cases rating method

Change in	Base case	Operation mode	Feed side	Sweep stream	Membrane type	Components
<b>Equipment &amp; system</b>						
Method	Rating	Rating	Rating	Rating	Rating	Rating
Run mode	Counter-current	Co-current	Counter-current	Counter-current	Counter-current	Counter-current
Feed side	Shell	Shell	Fibre	Shell	Shell	Shell
Sweep stream	Without	Without	Without	With	Without	Without
Membrane type	Symmetric	Symmetric	Symmetric	Symmetric	Asymmetric	Symmetric
Number of components	4	4	4	4	4	8
Set of components	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar,O2,H2O,CO,CO2
<b>Specifications</b>						
Length [m]	0.5	0.5	0.5	0.5	0.5	0.5
Inner diameter [m]	8.00E-05	8.00E-05	8.00E-05	8.00E-05	8.00E-05	8.00E-05
Outer diameter [m]	2.00E-04	2.00E-04	2.00E-04	2.00E-04	2.00E-04	2.00E-04
Shell diameter [m]	0.075	0.075	0.075	0.075	0.075	0.075
Number of fibres	50000	50000	50000	50000	50000	50000
Physical property method	Peng Rob	Peng Rob	Peng Rob	Peng Rob	Peng Rob	Peng Rob
Permeability [1e-14]	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24;1;1;1
Number of discr. intervals	50	50	50	50	50	50
<b>Operation conditions</b>						
Feed flowrate [mol/s]	0.2083	0.2083	0.2083	0.2083	0.2083	0.2083
Feed composition [mol/mol]	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.3;0.25;0.2;0.05;0.05;0.05;0.05;0.05
Feed pressure [bar]	70	70	70	70	70	70
Feed temperature [K]	298	298	298	298	298	298
Sweep flowrate [mol/s]	na	na	na	0.0694	na	na
Sweep composition [mol/mol]	idem feed	idem feed	idem feed	0;0;0;1	idem feed	idem feed
Sweep pressure [bar]	10	10	10	10	10	10
Sweep temperature [K]	Idem feed	idem feed	idem feed	idem feed	idem feed	idem feed
<b>Results</b>						
Retentate flowrate [mol/s]	0.0967	0.1056	0.0967	0.0943	0.0989	0.1128
Retentate composition H2	0.046	0.127	0.046	0.000	0.067	0.015 - 0.083
N2	0.483	0.443	0.483	0.489	0.473	0.335 - 0.083
CH4	0.388	0.356	0.388	0.393	0.380	0.271 - 0.083
Ar	0.083	0.075	0.083	0.117	0.081	0.046 - 0.083
Retentate pressure [Pa]	6987582	6986983	6992973	6988186	6987276	6950771
Retentate temperature [K]	300.7	301.7	300.8	296.8	300.9	296.2
Permeate flowrate [mol/s]	0.1121	0.1028	0.1121	0.1835	0.1098	0.097
Permeate composition H2	0.894	0.883	0.894	0.568	0.892	0.644 - 0.010
N2	0.048	0.052	0.048	0.032	0.049	0.147 - 0.010
CH4	0.037	0.040	0.037	0.025	0.038	0.114 - 0.010
Ar	0.021	0.025	0.021	0.375	0.022	0.053 - 0.010
Permeate pressure [Pa]	991716	977965	985340	948347	991397	968238
Permeate temperature [K]	299.9	299.6	299.9	303.9	299.8	301.1
Stage cut	53.81	49.33	53.82	54.75	52.70	46.78
Get stuck on level	3	3	3	3	2	3

**Table H.2: Discrete cases design method**

Change in	Base case	Operation mode	Feed side	Sweep stream	Membrane type	Components
<b>Equipment &amp; system</b>						
Method	Design	Design	Design	Design	Design	Design
Run mode	Counter-current	Co-current	Counter-current	Counter-current	Counter-current	Counter-current
Feed side	Shell	Shell	Fibre	Shell	Shell	Shell
Sweep stream	Without	Without	Without	With	Without	Without
Membrane type	Symmetric	Symmetric	Symmetric	Symmetric	Asymmetric	Symmetric
Number of components	4	4	4	4	4	8
Set of components	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar	H2,N2,CH4,Ar,O2,H2O,CO,CO2
<b>Specifications</b>						
constant velocity [m/s]	0.05	0.05	0.50	0.05	0.05	0.05
packing density [m2/m3]	4500	4500	4500	4500	4500	4500
ratio cross area	0.10	0.10	0.10	0.10	0.10	0.10
membrane area [m]	5	5	5	5	5	5
membrane thickness [m]	1.00E-04	1.00E-04	1.00E-04	1.00E-04	1.00E-04	1.00E+00
Physical property method	Peng Rob	Peng Rob	Peng Rob	Peng Rob	Peng Rob	Peng Rob
Permeability [1e-14]	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24	340.8;3.54;3.408;9.24;1;1;1;1
Number of discr. intervals	50	50	50	50	50	51
<b>Operation conditions</b>						
Feed flowrate [mol/s]	0.2083	0.2083	0.2083	0.2083	0.2083	0.2083
Feed composition [mol/mol]	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.50;0.25;0.20;0.05	0.3;0.25;0.2;0.05;0.05;0.05;0.05;0.05
Feed pressure [Pa]	70	70	70	70	70	70
Feed temperature [K]	298	298	298	298	298	298
Sweep flowrate [mol/s]	na	na	na	0.0694	na	na
Sweep composition [mol/mol]	idem feed	idem feed	idem feed	0;0;0;1	idem feed	idem feed
Sweep pressure [Pa]	10	10	10	10	10	10
Sweep temperature [K]	idem feed	idem feed	idem feed	idem feed	idem feed	idem feed
<b>Results</b>						
Retentate flowrate [mol/s]	0.0977	0.1063	0.0976	0.0947	0.0999	0.1464
Retentate composition H2	0.050	0.127	0.049	8.429E-05	0.071	0.066 - 0.070
N2	0.481	0.442	0.481	0.490	0.470	0.331 - 0.070
CH4	0.386	0.355	0.386	0.394	0.378	0.265 - 0.070
Ar	0.083	0.075	0.084	0.116	0.081	0.060 - 0.070
Retentate pressure [Pa]	6967513	6966058	6989641	6969173	6966760	6951888
Retentate temperature [K]	300.9	301.6	300.9	297.2	300.8	302.3
Permeate flowrate [mol/s]	0.1110	0.1021	0.1111	0.1830	0.1087	0.0616
Permeate composition H2	0.898	0.888	0.898	0.569	0.896	0.853 - 0.004
N2	0.046	0.050	0.046	0.031	0.047	0.060 - 0.004
CH4	0.035	0.038	0.036	0.024	0.036	0.046 - 0.004
Ar	0.020	0.024	0.020	0.376	0.021	0.027 - 0.004
Permeate pressure [Pa]	987874	968230	961710	923429	987467	990405
Permeate temperature [K]	299.9	299.6	299.9	303.5	299.8	300.1
Stage cut	53.28	48.99	53.35	54.52	52.20	29.58
Required initialisation level	3	3	3	3	2	3



**Table H.3: Cases change in module length (rating method – counter-current)**

Length	0.01	0.05	0.5	1	1.3	1.5	1.7
Membrane area	0.13	0.63	6.28	12.57	16.34	18.85	21.36
<b>Results</b>							
Retentate flowrate	0.1995	0.1691	0.0967	0.0810	0.0733	0.0683	x
Retentate composition H2	0.478	0.389	0.046	0.022	0.026	0.029	x
N2	0.261	0.306	0.483	0.501	0.502	0.503	x
CH4	0.209	0.245	0.388	0.404	0.406	0.408	x
Ar	0.052	0.061	0.083	0.073	0.065	0.060	x
Retentate pressure	6999601.5	6998146	6987582	6978964	6974427	6971642	x
Retentate temperature	298.176	299.0	300.7	297.6	295.5	294.0	x
Permeate flowrate	0.009	0.0392	0.1121	0.1289	0.1368	0.1419	x
Permeate composition H2	0.983	0.979	0.894	0.807	0.761	0.733	x
N2	0.007	0.009	0.048	0.089	0.112	0.125	x
CH4	0.006	0.007	0.037	0.069	0.087	0.098	x
Ar	0.004	0.005	0.021	0.035	0.041	0.045	x
Permeate pressure	999979	999558	991716	982813	975624	969714	x
Permeate temperature	298.1	298.4	299.9	300.2	300.3	300.4	x
Stage cut	4.3	18.8	53.8	61.9	65.7	68.1	
Required initialisation level	3	3	3	3	3	3	2

**Table H.4: Cases change in module length (rating method – co-current)**

Length	0.01	1.7	2.5	3	3.5	4	4.2	4.4
Membrane area	0.126	21.36	31.42	37.70	43.98	50.27	52.78	55.29
<b>Results</b>								
Retentate flowrate	0.1995	0.0675	0.0446	0.0324	0.0296	0.0256	0.0233	x
Retentate composition H2	0.479	0.089	0.069	0.058	0.049	0.038	0.032	x
N2	0.261	0.476	0.493	0.502	0.507	0.514	0.517	x
CH4	0.209	0.388	0.407	0.418	0.424	0.432	0.436	x
Ar	0.052	0.047	0.030	0.022	0.020	0.017	0.015	x
Retentate pressure	6999602	6965780	6956709	6952864	6949423	6946412	6945388	x
Retentate temperature	298.2	301.8	302.0	300.3	290.2	283.2	281.5	x
Permeate flowrate	0.009	0.141	0.164	0.176	0.179	0.183	0.185	x
Permeate composition H2	0.983	0.697	0.617	0.582	0.575	0.565	0.559	x
N2	0.007	0.141	0.184	0.204	0.207	0.213	0.216	x
CH4	0.006	0.110	0.144	0.160	0.163	0.168	0.170	x
Ar	0.004	0.051	0.055	0.055	0.055	0.055	0.054	x
Permeate pressure	999979	873041	766643	681233	582356	457317	393743	x
Permeate temperature	298.1	299.9	300.1	300.6	302.9	304.1	304.1	x
Stage cut	4.3	67.6	78.6	84.4	85.8	87.7	88.8	
Required initialisation level	3	3	3	3	3	3	3	

**Table H.5: Continuous cases pressure ratio (rating method)**

Pressure ratio	1.0	1.0	1.2	3.5	7.0	14.0	23.3	35.0
Sweep pressure	69.5	69	60	20	10	5	3	2
<b>Results</b>								
Retentate flowrate [mol/s]	0.2083	0.2079	0.2027	0.1131	0.0967	0.0921	0.0910	0.0905
Retentate composition H2	0.500	0.500	0.496	0.164	0.046	0.012	0.005	0.003
N2	0.250	0.250	0.252	0.422	0.483	0.501	0.505	0.507
CH4	0.200	0.200	0.202	0.338	0.388	0.403	0.406	0.407
Ar	0.050	0.050	0.050	0.075	0.083	0.084	0.084	0.083
Retentate pressure [Pa]	6979689	6979706	6979911	6985803	6987582	6988211	6988403	6988487
Retentate temperature [K]	298.0	298.0	298.0	301.1	300.7	301.2	301.5	301.9
Permeate flowrate [mol/s]	0	3.500E-04	0.0046	0.0950	0.1121	0.1166	0.1176	0.1180
Permeate composition H2	0.501	0.507	0.581	0.898	0.894	0.888	0.884	0.882
N2	0.250	0.246	0.206	0.046	0.048	0.051	0.052	0.053
CH4	0.200	0.197	0.163	0.036	0.037	0.039	0.040	0.041
Ar	0.050	0.050	0.050	0.020	0.021	0.023	0.024	0.024
Permeate pressure [Pa]	6990001	6899991	5999858	1995750	991716	484244	273877	158969
Permeate temperature [K]	297.8	298.0	298.0	299.3	299.9	300.2	300.2	300.3
Stage cut	0.0	0.2	2.2	45.6	53.8	56.0	56.5	56.6
Required initialisation level	3	3	3	3	3	3	2	2

**Table H.6: Continuous cases composition feed stream (rating method)**

Feed component fraction H2	0	0.001	0.005	0.05	0.2	0.4	0.5	0.75	0.80
N2	0.75	0.749	0.745	0.7	0.55	0.35	0.25	0	0
CH4	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.15
Ar	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
<b>Results</b>									
Retentate flowrate [mol/s]	0.1966	0.1965	0.1962	0.1913	0.1639	0.1195	0.0967	0.040	x
Retentate composition H2	0.000	0.0006	0.003	0.027	0.055	0.053	0.046	0.033	x
N2	0.752	0.7513	0.748	0.719	0.653	0.557	0.483	0.000	x
CH4	0.201	0.2010	0.201	0.206	0.238	0.320	0.388	0.818	x
Ar	0.047	0.0470	0.047	0.048	0.054	0.070	0.083	0.149	x
Retentate pressure [Pa]	6977829	6977836	6977862	6978244	6980686	6985124	6987582	6994220	x
Retentate temperature [K]	298.0	298.0	298.0	298.1	299.4	301.0	300.7	296.7	x
Permeate flowrate [mol/s]	0.0117	0.0118	0.0121	0.0167	0.0441	0.0890	0.1121	0.1692	x
Permeate composition H2	0.000	0.006	0.032	0.298	0.734	0.867	0.894	0.923	x
N2	0.716	0.711	0.691	0.491	0.171	0.071	0.048	0.000	x
CH4	0.185	0.184	0.180	0.136	0.060	0.039	0.037	0.051	x
Ar	0.099	0.099	0.097	0.076	0.035	0.023	0.021	0.026	x
Permeate pressure [Pa]	997533	997520	997465	996765	994438	992373	991716	991293	x
Permeate temperature [K]	298.0	298.0	298.0	298.0	298.7	299.6	299.9	300.5	x
Stage cut	5.6	5.7	5.8	8.0	21.2	42.7	53.8	81.2	
Required initialisation level	3	3	3	3	3	3	3	3	

**Table H.7: Continuous cases temperature (rating method)**

Feed Temperature [K]	275	298	325	350	375	400	425
<b>Results</b>							
Retentate flowrate [mol/s]	0.1036	0.0967	0.0898	0.0840	0.0783	0.0724	x
Retentate composition H2	0.075	0.046	0.028	0.022	0.023	0.027	x
N2	0.467	0.483	0.494	0.499	0.501	0.502	x
CH4	0.375	0.388	0.398	0.402	0.405	0.406	x
Ar	0.084	0.083	0.080	0.076	0.071	0.064	x
Retentate pressure [Pa]	6988396	6987582	6986600	6985670	6984718	6983760	x
Retentate temperature [K]	280.1	300.7	325.9	349.1	372.2	395.1	x
Permeate flowrate [mol/s]	0.1048	0.1121	0.1195	0.1258	0.1317	0.1378	x
Permeate composition H2	0.921	0.894	0.859	0.825	0.790	0.755	x
N2	0.035	0.048	0.064	0.081	0.097	0.114	x
CH4	0.027	0.037	0.050	0.062	0.076	0.089	x
Ar	0.016	0.021	0.027	0.032	0.037	0.042	x
Permeate pressure [Pa]	992607	991716	990472	989009	987061	984408	x
Permeate temperature [K]	277.0	299.9	326.8	351.7	376.6	401.5	x
Stage cut	50.3	53.8	57.4	60.4	63.2	66.2	
Required initialisation level	3	3	3	3	3	3	