

Development of an open source membrane steady state simulator

Shahrul Azman B Hj Zainal Abidin

*PETRONAS Research Scientific and Services Sdn. Bhd.
Lot 3288 & 3289 Off Jalan Ayer Itam,
Kawasan Institusi Bangi,
43000 Kajang, Selangor,
Malaysia*

Abstract

A membrane steady state simulator based on an open source concept is discussed. It is believe that this way of developing software will overcome weaknesses posed by available commercial membrane simulators. The application provides a cost effective state of the art tool for membrane modeling and object oriented design and its usage allows for easy incorporation of specific developments either open source or proprietary. This paper presents the components require to produce an open source membrane steady state simulator. This approach of development differs significantly from a close black-box approach. Several membrane models using the simulator will be elaborated and compared with literature results.

1. Conventional software practices

The term “conventional software practice” refers to companies that operate in a business model where the source code of their applications is not released to the users. This conventional approach results in several disadvantages and limitations namely

- i) Software code development is constrained within the developers
- ii) Application enhancement is based on the developer’s interpretation of the need of their current users.
- iii) Users have to wait for new releases
- iv) Customization of the product can only be made by the company that develop the software

In the area of membrane applications, it can be said that available commercial simulators such as WINFlows or MEMSYS, tend to provide a “total solution” that involves a great deal of specialized or proprietary technologies and scientific knowledge. These technologies range from highly specific mathematical modeling approach to user-friendly graphical interfaces. User-friendly GUI results in software that must invest resources in many areas depending only on their employees and strategic partners to incorporate new developments into their products (1).

A membrane research unit that focus in a small niche of membrane technology may find it difficult to exploit their business because they depend on other available

commercial membrane simulator to link their products and this may pose either technical or economic restrictions.

2. Open source software (OSS)

Development of open source software has grown as a near optimal solution for the development of robust and reliable applications using minimal formal resources. The key aspect of open source is that it releases the code to the users usually through the Internet at no cost, allowing use, test and rapid development by accepting fixes, changes or areas of improvements.

The term “Open Source” is not controlled or owned by any individual or corporation. Membrane programmers from all over the world can donate their source code to a membrane research unit that willing to be a caretaker for the membrane open source code foundation. The foundation is a non-profit organization to promote the development of open source membrane simulator. The foundation will publish membrane source code through the Internet and can be accessed freely by membrane researchers throughout the world.

3. Advantages of membrane open source simulator (MOSS)

The open and free nature of MOSS stimulates the involvement of membrane programmers and researchers with many different needs and servicing of varied interest niches that would be ignored by the large commercial simulation companies. MOSS also benefits from rapid releases and prompt feedback because is not constrained by time zones or work hours. Having access to the code permits quick bug fixes that become available to membrane fraternity as soon as they are finished.

An important aspect of these types of project is that the application does not necessarily depend on the involvement of specific persons or corporations. Anyone can have accessed to the source code and if the caretaker side of a project goes away, another one can take over.

In membrane technology, it should be noted that some developments might involve crucial technology that a company cannot release to the public domain for strategic or economic reasons. Therefore, an open source membrane simulator must be designed and licensed in such a way that it contemplates the addition of proprietary as well as open source membrane applications.

4. Features

Some of the most recommended, important and distinctive features of membrane simulator are:

- 1) The simulator is written in the Python programming language that is open source software and can be obtained at no cost via the Internet. It is object oriented, it is very high level, has a clean syntax and it is essentially multi-platform.

- 2) The building blocks of the simulator were created with well-defined interfaces thus allowing parallel and decentralized development. A clear independence from user interfaces and thermodynamic method providers is emphasized.
- 3) It provides all kind of flow patterns and configurations and supports multi-component separation for liquid and gas phases.
- 4) Different thermodynamic providers can be accessed within the same flow sheet.

5. Basic design of the simulator

The membrane simulator on its own provides an interface to simulation services such as a flow sheet object, unit operations and thermodynamic administrator.

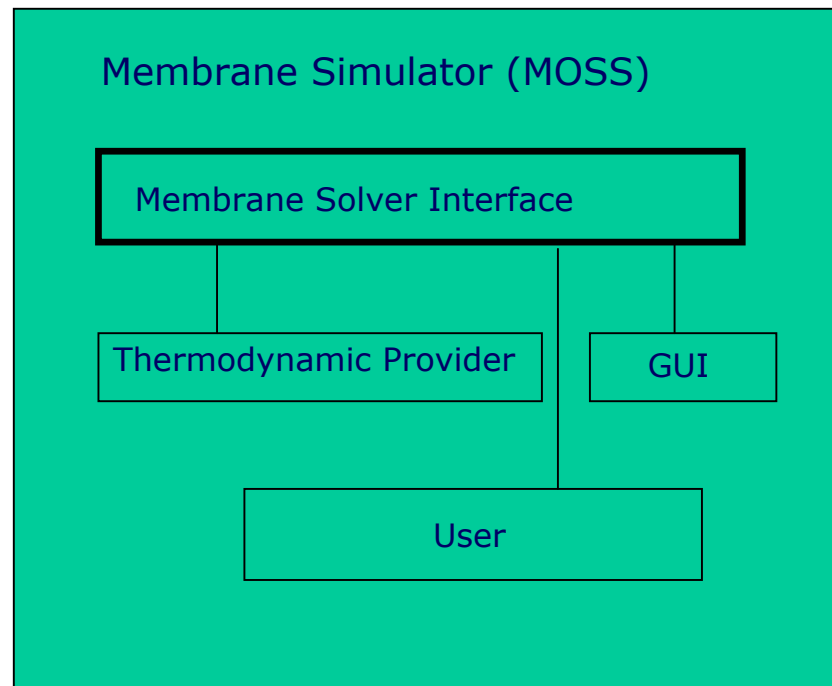


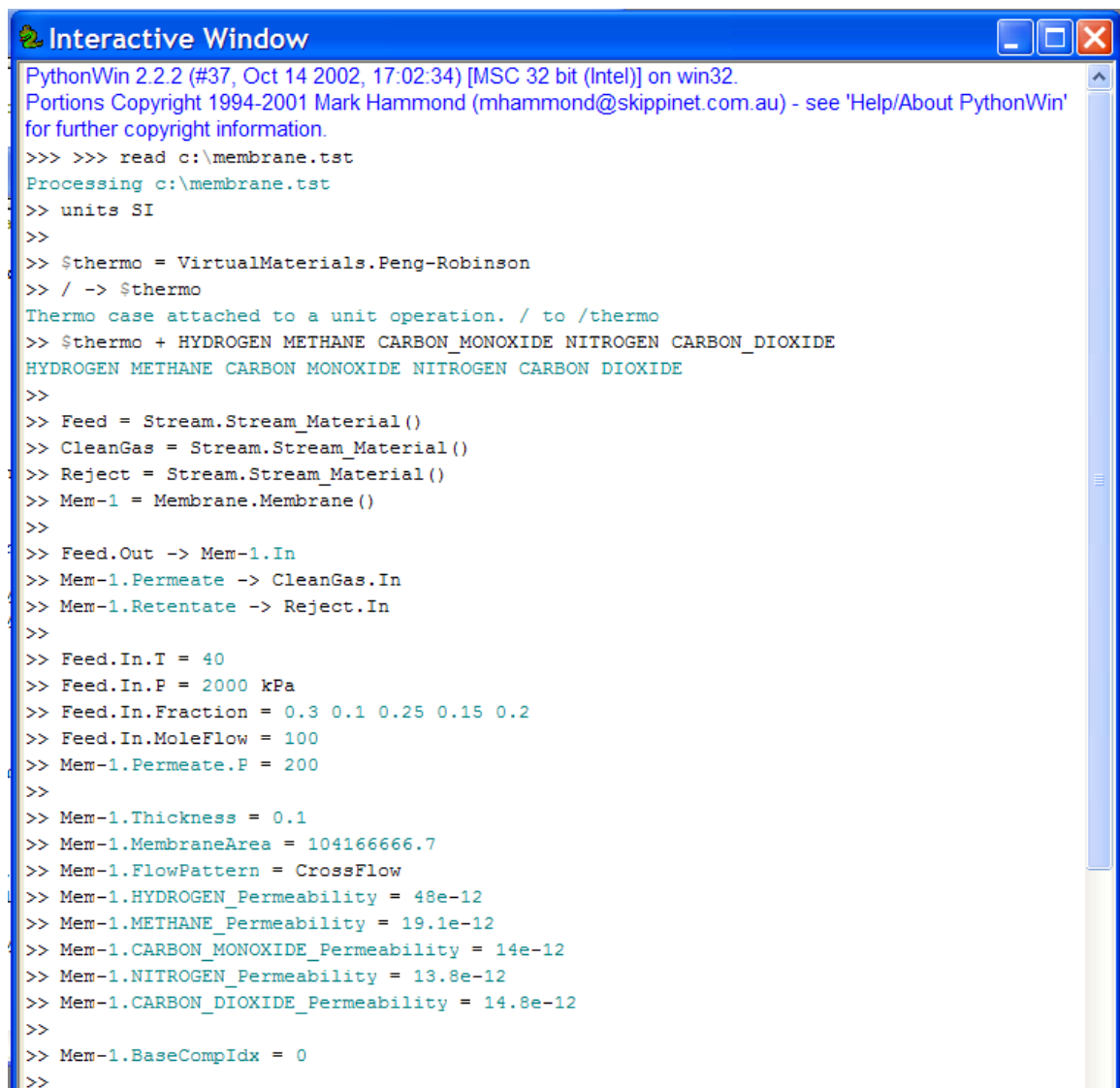
Figure 1: Basic architecture for Membrane Open Source Simulator (MOSS)

Figure 1 shows that the user can communicate to Membrane Solver Interface directly to the user where Graphical User Interface (GUI) and Thermodynamic provider are interfaced. Using Microsoft Visio for different type and flow pattern in membrane operation and Visual Basic for communication can develop GUI. Thermodynamic provider is required to cater for physical properties changes that contribute to the calculation of energy required per unit membrane system.

6. Basic object and the first simulation

This example creates a feed stream called Feed and creates two product streams called CleanGas and Reject. The Feed stream is fed into a membrane unit operation and calculation were performed for the separation of H₂, CH₄, CO, N₂ and CO₂ mixture through a micro-porous glass membrane. Brubaker and Kammermeyer (3) determined the permeability of gases for the membrane used. The permeability data used are shown in figure 2. The calculation conditions and the numerical results are shown in figure 3 and 4.

Calculations were performed for five kinds of flow pattern namely perfect mixing, one side mixing, cross current, co-current and counter current. The mathematical modeling and example were based on Shindo, Hakuta and Yoshitome method (4). Virtual Material Group (2) Peng-Robinson property package was used to determine Feed, CleanGas and Reject streams physical property values.



```
PythonWin 2.2.2 (#37, Oct 14 2002, 17:02:34) [MSC 32 bit (Intel)] on win32.
Portions Copyright 1994-2001 Mark Hammond (mhammond@skippinet.com.au) - see 'Help/About PythonWin'
for further copyright information.
>>> >>> read c:\membrane.tst
Processing c:\membrane.tst
>> units SI
>>
>> $thermo = VirtualMaterials.Peng-Robinson
>> / -> $thermo
Thermo case attached to a unit operation. / to /thermo
>> $thermo + HYDROGEN METHANE CARBON_MONOXIDE NITROGEN CARBON_DIOXIDE
HYDROGEN METHANE CARBON MONOXIDE NITROGEN CARBON DIOXIDE
>>
>> Feed = Stream.Stream_Material()
>> CleanGas = Stream.Stream_Material()
>> Reject = Stream.Stream_Material()
>> Mem-1 = Membrane.Membrane()
>>
>> Feed.Out -> Mem-1.In
>> Mem-1.Permeate -> CleanGas.In
>> Mem-1.Retentate -> Reject.In
>>
>> Feed.In.T = 40
>> Feed.In.P = 2000 kPa
>> Feed.In.Fraction = 0.3 0.1 0.25 0.15 0.2
>> Feed.In.MoleFlow = 100
>> Mem-1.Permeate.P = 200
>>
>> Mem-1.Thickness = 0.1
>> Mem-1.MembraneArea = 104166666.7
>> Mem-1.FlowPattern = CrossFlow
>> Mem-1.HYDROGEN_Permeability = 48e-12
>> Mem-1.METHANE_Permeability = 19.1e-12
>> Mem-1.CARBON_MONOXIDE_Permeability = 14e-12
>> Mem-1.NITROGEN_Permeability = 13.8e-12
>> Mem-1.CARBON_DIOXIDE_Permeability = 14.8e-12
>>
>> Mem-1.BaseCompIdx = 0
>>
```

Figure 2: Feed, CleanGas, Reject Streams Creation and Permeability Values

```

Interactive Window
>> # Results
>> Feed.In
Port: /Feed.In + sim.solver.Ports.Port_Material
Connected to: None
VapFrac      = 1.0 =
T            = 40.0 * C
P           = 2000.0 * kPa
MoleFlow     = 100.0 * kgmole/h
MassFlow     = 2221.5457 = kg/h
VolumeFlow   = 128.283008093 = m3/hr
H            = 9035.84722308 = kJ/kmol
Energy       = 250995.756197 = W
MolecularWeight = 22.215457 =
ZFactor      = 0.985408542348 =
HYDROGEN     = 0.3 *
METHANE      = 0.1 *
CARBON MONOXIDE = 0.25 *
NITROGEN     = 0.15 *
CARBON DIOXIDE = 0.2 *

```

Figure 3: Feed Stream Condition

```

Interactive Window
>> CleanGas.Out
Port: /CleanGas.Out + sim.solver.Ports.Port_Material
Connected to: None
VapFrac      = 1.0 =
T            = 40.0 = C
P           = 200.0 = kPa
MoleFlow     = 41.2804147936 = kgmole/h
MassFlow     = 706.145501856 = kg/h
VolumeFlow   = 536.881799982 = m3/hr
H            = 9082.80686085 = kJ/kmol
Energy       = 104150.565196 = W
MolecularWeight = 17.1060660381 =
ZFactor      = 0.999037561556 =
HYDROGEN     = 0.470379389015 =
METHANE      = 0.0910842731129 =
CARBON MONOXIDE = 0.180586235679 =
NITROGEN     = 0.107143633695 =
CARBON DIOXIDE = 0.150806468497 =

>> Reject.Out
Port: /Reject.Out + sim.solver.Ports.Port_Material
Connected to: None
VapFrac      = 1.0 =
T            = 31.2518478707 = C
P           = 200.0 = kPa
MoleFlow     = 58.7195852064 = kgmole/h
MassFlow     = 1515.40019814 = kg/h
VolumeFlow   = 741.360066424 = m3/hr
H            = 9002.83416076 = kJ/kmol
Energy       = 146845.191001 = W
MolecularWeight = 25.8074063742 =
ZFactor      = 0.997696684715 =
HYDROGEN     = 0.18022170412 =
METHANE      = 0.106267838964 =
CARBON MONOXIDE = 0.298798522222 =
NITROGEN     = 0.180128424296 =
CARBON DIOXIDE = 0.234583510399 =

```

Figure 4: CleanGas and Reject Streams Condition

```
-""" Membrane modelling based on Y.Shindo, T.Hakuta and H.Yoshitome mathematical model.
    Iteration is using Runge-Kutta 4th Order method
    """

from sim.unitop import ComponentSplitter
from sim.solver.Variables import *
from sim.unitop import UnitOperations, Sensor, Stream
from sim.solver import Ports
import Numeric

# Parameters to be set by users through CLI or tst
BASE_COMP_PAR = 'BaseCompIdx'
FLOW_PATTERN_PAR = 'FlowPattern'

# Available flow patterns supported by this membrane model and can be set by users through CLI or tst
PERFECT_MIXING = 'PerfectMixing'
ONESIDE_MIXING = 'OneSideMixing'
CROSS_FLOW = 'CrossFlow'
CO_CURRENT = 'CoCurrent'
COUNTER_CURRENT = 'CounterCurrent'

PERMEATE_PORT = 'Permeate'
RETENTATE_PORT = 'Retentate'
MEMBRANEAREA_PORT = 'MembraneArea'
THICKNESS_PORT = 'Thickness'
STAGECUT_PORT = 'StageCut' # This will be calculated or specified as THI(theta)
TOTAL_SURFACE_PORT = 'TotalSurface' # Dimensionless

XF = {}
XC = {}
YP = {}
CQ = {}
QQ = {}
```

Figure 5: Example of Python Code for Membrane Unit Operation

Sample of Python language code written in Pythonwin environment is shown in figure 5. An open source GUI written in wxPython (5) or Visual Basic interface can also be made available to make it more presentable and user friendly to use.

7. Conclusion

Membrane Open Source Simulator (MOSS) cannot be called an open source success yet as it is a recent development and it's developing base is still in infant stage. It is expected that the application will naturally improve when involvement in the concept presented in this paper grows. More membrane mathematical modeling papers source code need to be included in Membrane Solver Interface administered by a non-profit organization / foundation.

References

- [1] R. Cota, M. Satyro, C. Morris, B. Svrcek, B. Young, Development of An Open Source Chemical Process Simulator, AIChE Spring Meeting 2003.
- [2] Virtual Material Group (VMG), <http://www.virtualmaterials.com>
- [3] D. W. Brubaker and K. Kammermeyers, Ind. Eng. Chem, 46, 733, 1952

[4] Y. Shindo, T. Hakuta, H. Yoshitome, Separation Science and Technology, 20(5&6), pp. 445-459, 1985.

[5] wxPython <http://www.wxpython.org>