



# Summer Fellowship Report

On

## Custom Unit Operations in DWSIM using Scilab

Submitted by

**S. Pranaav**

Under the guidance of

**Prof.Kannan M. Moudgalya**
Chemical Engineering Department
IIT Bombay

July 24, 2018

# Acknowledgment

I wish to express our profound gratitude to our internship guide Dr. Kannan Moudgalya, Professor, Department of Chemical Engineering, IIT Bombay for his constant support and supervision throughout the internship. We have reaped benefits from his wisdom, guidance and patience all along. I would also like to thank my professor Dr.P.R.Naren, SASTRA University, my mentors, Priyam Nayak, A.S. Rahul and Pravin Dalve, FOSSEE Team, IIT-Bombay for their timely help and guidance.

# Contents

# Chapter 1

# Introduction

*DWSIM* is a free and open source steady state chemical process simulator. It follows the sequential modular approach. *DWSIM* has more than fifteen thermodynamic property packages built into it along with basic unit operations that can be used to build a process flowsheet. Furthermore, a user can also develop custom unit operations in *DWSIM* using Scilab scripts. This feature helps to develop unit operations that are otherwise not inherently available in *DWSIM*. This enhances the workability of the user to incorporate various features in *DWSIM* according to the needs of the user. This increasses the versatility of *DWSIM* to be used in commercial process industries without the need for any proprietary tool. However, Scilab must neccessarily be installed to run these custom models.

# Chapter 2

# Tools required for Custom Unit Operations Modelling

## ChemSep

ChemSep is a CAPE-OPEN technology that helps model distillation, absorbtion and extraction columns. It also has various properties of many commonly used compounds. Thus, it can be used extensively in Chemical Process simulators. DWSIM uses ChemSep LITE database to find out the properties of many of the compounds. It is installed by default while installing DWSIM.

## Scilab unit Operations

Scilab unit operations is a CAPE-OPEN tool that allows the user to implement unit operations based on calculations entered in Scilab. This tool is developed and maintained by AMSTERCHEM corporation. This plugin must be installed in addition to DWSIM in order to build new unit operations. This tool allows users to type the script in Scilab. It must be noted that Scilab versions only above 5.5.4 can be used for custom modelling. Working in Scilab is easy and the users can create new models easily. It can be downloaded from `https://www.amsterchem.com/scilabunitop.html`. Academic and individual users can obtain a non-commercial license also from the same website.

## Scilab Thermo Import

In order to obtain the properties of the stream from the flowsheet, Scilab Thermo Import must be used. Scilab Thermo Import must be installed on the computer. This software is also maintained by AMSTERCHEM corporation. Various properties of the compounds can be obtained like compound constants, single phase propertues, two phase properties etc. It can be downloaded from `https://www.amsterchem.com/scilabthermo.html`. Academic and individual users can obtain a non-commercial license also from the same website.

## CAPE OPEN Property Package

CAPE-OPEN Property Package or COPP in short, is used to build the underlying thermodynamics model based on the list of compounds present in ChemSep Database. The users can create a new COPP from the list of available compounds according to their requirements in the flowsheet. This can then be accessed by the using Scilab Thermo Import. This program is available only if ChemSep has already been installed.

# Chapter 3

# Guidelines while using Scilab CAPE-OPEN Unit Operations

- Scilab must necessarily be installed.
- The number of inlet and output ports must be specified as soon as the unit operation is created.
- Parameters if used, must be specified with default,minimum and maximum values.
- A list of functions to be used are mentioned in the website: `https://www.amsterchem.com/scilabunitophelp.php` for further reference.
- The code must be tested in the dialog box before running the flowsheet. Any errors arising can be rectified in this step itself.

# Chapter 4

# Custom Modelling of a Mixer

## 4.1 Objective

The objective is to develop a model that can mix two or more material streams to obtain one new material stream and calculate its properties. This particular model mixes 3 material streams. The result is then compared with that obtained from DWSIM's default stream mixer.

## 4.2 Assumptions

- Steady state.
- No loss of energy to surroundings.

## 4.3 Equations used

### 4.3.1 Total individual flow balance

$$F = \Sigma f_i \tag{4.1}$$

### 4.3.2 Total mole balance

$$totF = \Sigma F \tag{4.2}$$

### 4.3.3 Pressure of outlet

$$P = \left(\frac{\Sigma p_i}{n}\right) \tag{4.3}$$

### 4.3.4 Enthalpy balance

$$H = \left(\frac{\Sigma h_i f_i}{n}\right) \tag{4.4}$$

## 4.4 Degree of freedom analysis

| Head | Details | Numbers |
|---|---|---|
| Variables | $f_i$, $p_i$, $h_i$, F, P, H, totF, n | 14 |
| Known | $f_i$, $p_i$, $h_i$, n | 9 |
| Unknown | F, P, H, totF | 4 |
| Number of independent equations | 4 | 4 |
| Degrees of freedom | - | 0 |

## 4.5 Special functions used in the code

### 4.5.1 getFeedProp

This function is used to get properties like molar flow rate, temperature, pressure etc. from the material streams.

### 4.5.2 setFeedProp

This function is used to store properties like molar flow rate, temperature, pressure etc. after calculation into the outlet material streams.

## 4.6 Scilab Code

```
//==============

// Basic Material Stream Mixer

//==============
// Input section
//==============
f1=getFeedProp(1,"flow")
f2=getFeedProp(2,"flow")
f3=getFeedProp(3,"flow")
```

```
h1=getFeedProp(1,"enthalpy")
h2=getFeedProp(2,"enthalpy")
h3=getFeedProp(3,"enthalpy")

p1=getFeedProp(1,"pressure")
p2=getFeedProp(2,"pressure")
p3=getFeedProp(3,"pressure")

//==============
// Calculation section
//==============

p=(p1+p2+p3)/3 // Average pressure in Pa
f=f1+f2+f3 // Total individual molar flow in mol/s
totF=sum(f) // Total Flow in mol/s
h=((h1*f1)+(h2*f2)+(h3*f3))/f // Enthalpy in kJ/kmol

//==============
// Output section
//==============

setProduct(1,totF,f/totF,"pressure",p,"enthalpy",h)

//==============
```

## 4.7 Results

| Master Property Table | | | | | | |
|---|---|---|---|---|---|---|
| Object | Feed1 | Feed2 | Feed3 | OutletUO | OutletMix | |
| Temperature | 300 | 298.15 | 298.15 | 299.748 | 298.465 | K |
| Pressure | 101325 | 101325 | 101325 | 101325 | 101325 | Pa |
| Mass Flow | 1.5 | 1 | 6 | 8.5 | 8.5 | kg/s |
| Molar Flow | 46.8136 | 35.9274 | 200.415 | 283.155 | 283.155 | mol/s |
| Mixture Molar Enthalpy | -41268.3 | -40781.1 | -39330.1 | -39708.4 | -39834.6 | kJ/kmol |
| Molar Fraction (Mixture) / Water | 0.333333 | 0.4 | 0.2 | 0.24742 | 0.24742 | |
| Molar Fraction (Mixture) / Methanol | 0.333333 | 0.5 | 0.75 | 0.649393 | 0.649393 | |
| Molar Fraction (Mixture) / Ethanol | 0.333333 | 0.1 | 0.05 | 0.103187 | 0.103187 | |

Figure 4.1: Property table

## 4.8 Nomenclature

### 4.8.1 Latin

| | |
|---|---|
| f | Molar flow rates of individual components mol/s |
| F | Molar flow of outlet stream mol/s |
| h | Enthalpies of the streams kJ/kmol |
| H | Enthalpy of outlet stream kJ/kmol |
| n | Number of feeds |
| p | Pressure of the streams Pa |
| P | Pressure of outlet stream Pa |
| totF | Total molar flow mol/s |

### 4.8.2 Subscript letters

| | |
|---|---|
| i | Index of feed |

## 4.9 Additional Notes

The outlet enthalpies vary slightly from that obtained from DWSIM's mixer. This is because the unit operation uses molar enthalpies instead of mass enthalpies.

# Chapter 5

# Custom Modelling of a Generic Mixer

## 5.1 Objective

The objective is to develop a model that can mix two or more material streams to obtain one new material stream and calculate its properties. This model can mix any number of streams with any number of components. The result is then compared with that obtained from DWSIM's default stream mixer.

## 5.2 Assumptions

- Steady state.
- No loss of energy to surroundings.

## 5.3 Equations used

### 5.3.1 Total individual flow balance

$$F = \Sigma f_i \tag{5.1}$$

### 5.3.2 Total mole balance

$$totF = \Sigma F \tag{5.2}$$

### 5.3.3 Pressure of outlet

$$P = \left(\frac{\Sigma p_i}{n}\right) \tag{5.3}$$

### 5.3.4 Enthalpy balance

$$H = \left(\frac{\Sigma h_i f_i}{n}\right) \tag{5.4}$$

## 5.4 Degree of freedom analysis

| Head | Details | Numbers |
|---|---|---|
| Variables | $f_i$, $p_i$, $h_i$, F, P, H, totF, N, Nc | 15 |
| Known | $f_i$, $p_i$, $h_i$, n, Nc | 11 |
| Unknown | F, P, H, totF | 4 |
| Number of independent equations | 4 | 4 |
| Degrees of freedom | - | 0 |

## 5.5 Special functions used in the code

### 5.5.1 getFeedProp

This function is used to get properties like molar flow rate, temperature, pressure etc. from the material streams.

### 5.5.2 getNumberOfFeeds

This function returns an integer value that contains the number of input streams input into the particular unit operation.

### 5.5.3 getParameter

This function is used to get the value of parameters that have been input before the execution of the program.

### 5.5.4 setFeedProp

This function is used to store properties like molar flow rate, temperature, pressure etc. after calculation into the outlet material streams.

## 5.6 Scilab Code

```
//==============

//Generic Mixer

//==============
// Input section
//==============

// To get number of streams
N=getNumberOfFeeds()
N1=double(N)

// To get number of components
Nc=getParameter("Ncomp")

// To get feed  properties
for  i=1:N
f(i ,:) =getFeedProp(i,"flow")
h(i)=getFeedProp(i,"enthalpy")
p(i)=getFeedProp(i,"pressure")
T(i)=getFeedProp(i,"temperature")
end;

F=[]
H=[]

//==============
// Calculation  section
//==============

for  i=1:Nc
F(i ,:) =sum(f(:,i))
end

P=0
```

```
for i=1:N
totF=sum(f) // Total molar flow in mol/s
H(i,:)=(h(i)*f(i,:)) // Individual molar enthalpy in kJ/kmol
P=P+(p(i))
end;

for i=1:N
H1(:,i)=sum(H(:,i)) // Total molar enthalpy in kJ/kmol
end

//===============
// Output section
//===============

setProduct(1,totF,F/totF,"pressure",P/N1,"enthalpy",H1/F')

//===============
```

## 5.7 Results

| Master Property Table | | | | | | | |
|---|---|---|---|---|---|---|---|
| Object | Feed1 | Feed2 | Feed3 | Feed4 | MixerOutlet | OutputUO | |
| Temperature | 298.15 | 350 | 325 | 325 | 330.706 | 331.319 | K |
| Molar Flow | 31.9074 | 77.7307 | 34.2037 | 24.7169 | 168.559 | 168.559 | mol/s |
| Mixture Molar Enthalpy | -43017.6 | -39028.8 | -38049.3 | -40155.4 | -39750.3 | -39686.7 | kJ/kmol |
| Molar Fraction (Mixture) / Water | 0.5 | 0.75 | 0.35 | 0.25 | 0.54819 | 0.54819 | |
| Molar Fraction (Mixture) / Ethanol | 0.25 | 0.1 | 0.15 | 0.15 | 0.145872 | 0.145872 | |
| Molar Fraction (Mixture) / 1-propanol | 0.1 | 0.1 | 0 | 0.35 | 0.116367 | 0.116367 | |
| Molar Fraction (Mixture) / Methanol | 0.15 | 0.05 | 0.5 | 0.25 | 0.18957 | 0.18957 | |

Figure 5.1: Property table

## 5.8 Nomenclature

### 5.8.1 Latin

| | |
|---|---|
| f | Molar flow rates of individual components mol/s |
| F | Molar flow of outlet stream mol/s |
| h | Enthalpies of the streams kJ/kmol |
| H | Enthalpy of outlet stream kJ/kmol |
| N | Number of feed streams |
| Nc | Number of components |
| p | Pressure of the streams Pa |
| P | Pressure of outlet stream Pa |
| totF | Total molar flow mol/s |

### 5.8.2 Subscript letters

| | |
|---|---|
| i | Index of feed |
| j | Index of component |

## 5.9 Additional Notes

The outlet enthalpies vary slightly from that obtained from DWSIM's mixer. This is because the unit operation uses molar enthalpies instead of mass enthalpies.

# Chapter 6

# Crystalliser

## 6.1 Objective

The objective is to develop a crystalliser model. This model simulates the crystallisation of a certain salt from a solvent. It then computes the output mass flow rates of the mother liquor and crystals obtained.

## 6.2 Assumptions

- Steady state.
- No loss of energy to surroundings.

## 6.3 Equations used

### 6.3.1 Overall Mass Balance

$$F = E + M + C \tag{6.1}$$

### 6.3.2 Salt Balance

$$Fx_{\mathrm{f}} = Mx_{\mathrm{m}} + Cx_{\mathrm{c}} \tag{6.2}$$

## 6.4 Degree of freedom analysis

| Head | Details | Numbers |
|---|---|---|
| Variables | F, M, C, E, $x_f$, $x_m$, $x_c$ | 7 |
| Known | F, E, $x_f$, $x_m$, $x_c$ | 5 |
| Unknown | M, C | 2 |
| Number of independent equations | 2 | 2 |
| Degrees of freedom | - | 0 |

## 6.5 Special functions used in the code

### 6.5.1 getFeedProp

This function is used to get properties like molar flow rate, temperature, pressure etc. from the material streams.

### 6.5.2 getParameter

This function is used to get the value of parameters that have been input before the execution of the program.

### 6.5.3 logMessage

This function is used to display any messages that need to be displayed to the user. This displays the message in the information toolbox.

### 6.5.4 setFeedProp

This function is used to store properties like molar flow rate, temperature, pressure etc. after calculation into the outlet material streams.

## 6.6 Scilab Code

```
// To find the composition of mother liquor, crystalliser and
   amount of water vapourised.

// The crystal is NaCl with no water of crystallisation

XF=getFeedProp(1,"fraction")
```

```
T=getFeedProp(1, "temperature")
P=getFeedProp(1, "pressure")
FF=getFeedProp(1,"totalFlow")

// Overall Mass balance
// F= C + M + E
// Salt balance
// Fxf=Cxc+Mxm

// Molecular weight of salt
MW=getParameter('MolWt of salt')


// Percentage of water evaporated
per=getParameter('Percentage water evaporated')
avgm=(18*XF(1))+ (MW*XF(2))

//Flow rate of solution in kg/s
Fm=(FF*avgm)/1000

// Amount of salt in solution
S=XF(2)*FF*MW/1000
// Amount of water in solution
W=Fm-S

// Amount of water evaporated
E=W*per/100

// To find the amount of crystal and mother liquor
// Solving via simultaneous linear equations

// Solubility in mother liquor
sol=getParameter('Solubility in mother liquor')
A=[1,1;1,(sol/100)]

B=[-Fm+E;-S]
```

```
x=linsolve(A,B)

C=x(1)
M=x(2)

// Output to streams
// setProduct(productIndex,totalFlow,moleFraction,propName1,
    propVal1,propName2,propVal2)

// Evaporator stream
setProduct(1,(E*1000/18),[1,0],'temperature',370,'pressure',P)

// Mother Liquor stream

// To find mole fraction of mother liquor stream

s=(sol/100)*M
xm(2)=(s/MW)/((s/MW)+((M-s)/18))
xm(1)=1-xm(2)

avgm=(xm(1)*18)+(xm(2)*MW)
setProduct(2,M*1000/avgm,xm,'temperature',T,'pressure',P)

// Crystal stream

setProduct(3,C*1000/MW,[0,1],'temperature',T,'pressure',P)

if (C<0 | M<0 | E<0)
then
logMessage("Note: Negative solution was found");
end
```

## 6.7 Results

| Master Property Table | | | | | |
|---|---|---|---|---|---|
| Object | Feed | Evaporated | MotherLiquor | Crystal | |
| Mass Flow | 7500 | 792 | 5607.92 | 1100.08 | kg/s |
| Molar Flow | 331077 | 43963.4 | 268289 | 18824.2 | mol/s |
| Molar Fraction (Mixture) / Water | 0.88526 | 1 | 0.928571 | 0 | |
| Molar Fraction (Mixture) / Sodium Chloride | 0.11474 | 0 | 0.0714286 | 1 | |

Figure 6.1: Property table

## 6.8 Nomenclature

### 6.8.1 Latin

C Mass flow of crystals in kg/s
E Mass flow of evaporated solvent in kg/s
F Mass flow of feed in kg/s
M Mass flow of mother liquor in kg/s
$x_c$ Mole fraction of salt in crystals
$x_f$ Mole fraction of salt in feed
$x_m$ Mole fraction of salt in mother liquor

# Chapter 7

# Plate Type Heat Exchanger

## 7.1 Objective

The objective is to develop a plate type heat exchanger model. This model cools a hot fluid and heats up the cooling fluid and works like a shell and tube heat exchanger. The outlet temperatures, pressure drops and heat duty are calculated from given inlet temperatures and design specifications using scilab script.

## 7.2 Assumptions

- Steady state
- No loss of energy to surroundings.

## 7.3 Equations used

### 7.3.1 Prandtl Number

$$Pr = \frac{Cp\mu}{k} \tag{7.1}$$

### 7.3.2 Cross sectional area

$$C_{area} = PsPw \tag{7.2}$$

### 7.3.3 Equivalent diameter

$$d_e = 2Ps \tag{7.3}$$

### 7.3.4 Port area

$$p_{area} = \frac{\pi d^2}{4} \tag{7.4}$$

### 7.3.5 Flow velocity

$$u = \frac{m}{C_{area}\rho No} \tag{7.5}$$

### 7.3.6 Reynolds Number

$$Re = \frac{d_e u \rho}{\mu} \tag{7.6}$$

### 7.3.7 Nusselt Number

$$Nu = 0.26 Re^{0.65} Pr^{0.4} \tag{7.7}$$

### 7.3.8 Heat transfer coefficient

$$h = \frac{Nuk}{d_e} \tag{7.8}$$

### 7.3.9 Overall heat transfer coefficient

$$\frac{1}{U} = \frac{1}{h_1} + \frac{1}{h_2} + \frac{1}{F_h} + \frac{1}{F_w} + \frac{th}{k_{Ti}} \tag{7.9}$$

### 7.3.10 Friction Factor

$$J_f = 0.6 Re^{-0.3} \tag{7.10}$$

### 7.3.11 Plate pressure drop

$$\Delta P_d = 8 J_f \left(\frac{Lp}{d_e}\right)\left(\frac{\rho u^2}{2}\right) \tag{7.11}$$

### 7.3.12 Port velocity

$$u_p = \frac{m p_{area}}{\rho} \tag{7.12}$$

### 7.3.13 Pressure drop due to port loss

$$\Delta P_t = 1.3\rho \left( \frac{{u_p}^2}{2} \right) \tag{7.13}$$

### 7.3.14 Total pressure drop

$$\Delta P = \Delta P_t + \Delta P_d \tag{7.14}$$

### 7.3.15 Heat capacity

$$C = mCp \tag{7.15}$$

### 7.3.16 Maximum heat transfer

$$Q_{max} = C_{min}(T_{hi} - T_{ci}) \tag{7.16}$$

### 7.3.17 Number of transfer units

$$NTU = \frac{UA}{C_{min}} \tag{7.17}$$

### 7.3.18 Heat capacity ratio

$$C_r = \frac{C_{min}}{C_{max}} \tag{7.18}$$

### 7.3.19 Effectiveness

$$\epsilon = \frac{1 - exp(-NTU(1 - Cr))}{1 - Cr(1 - exp(-NTU(1 - Cr)))} \tag{7.19}$$

### 7.3.20 Actual heat transfer

$$Q_{act} = \epsilon Q_{max} \tag{7.20}$$

### 7.3.21 Outlet hot stream temperature

$$T_{ho} = T_{hi} - \left( \frac{Q_{act}}{C_h} \right) \tag{7.21}$$

### 7.3.22 Outlet cold stream temperature

$$T_{co} = T_{ci} + \left(\frac{Q_{act}}{C_c}\right) \tag{7.22}$$

## 7.4 Special functions used in the code

### 7.4.1 capeOpen1PhaseProp

This function can be used to single phase properties of a given system at specified conditions of temperature and mole fractions.

### 7.4.2 capeOpenCompoundConstant

This function can be used to get constant values (eg. molecular weight, acentric factor)for the given compounds.

### 7.4.3 capeOpenPackageManagers

This function lists all the CAPE compliant packages.

### 7.4.4 capeOpenPackages

This function lists all the specific packages available in the particular package.

### 7.4.5 capeOpenSetBasis

This function sets the basis to either 'mass' or 'moles' basis.

### 7.4.6 getFeedProp

This function is used to get properties like molar flow rate, temperature, pressure etc. from the material streams.

### 7.4.7 getParameter

This function is used to get the value of parameters that have been input before the execution of the program.

### 7.4.8 setFeedProp

This function is used to store properties like molar flow rate, temperature, pressure etc. after calculation into the outlet material streams.

## 7.5 Scilab Code

```
// Design of a plate type heat exchanger with only input
   parameters
// Methanol and water used
clc
clear
//=========================
// Input Section
//=========================

// Flow rate in mol/s
M(1)=getFeedProp(1,"totalFlow")
M(2)=getFeedProp(2,"totalFlow")

// To link Scilab Thermo Import
mgrs=capeOpenPackageManagers()
cs=mgrs(3)
capeOpenPackages(cs)
capeOpenSetBasis("mass")
handle=capeOpenGetPackage(cs,"PHX")

MW=capeOpenCompoundConstant(handle,"molecularWeight")

//Mass flow rate in kg/s
m(1)=M(1)*MW(2)*1e-3
m(2)=M(2)*MW(1)*1e-3

x1=getFeedProp(1,"fraction")
x2=getFeedProp(2,"fraction")

// Pressure of the 2 streams
```

```
PP(1)=getFeedProp(1,"pressure")
PP(2)=getFeedProp(2,"pressure")

// Plate dimensions
Parea=getParameter('Plate area') // Area in m2
Pl=getParameter('Plate length')        // Length in m
Pw=getParameter('Plate width')     // Width in m
th=getParameter('thickness') // Thickness of plate in m
Ps=getParameter('Plate spacing')    // Plate spacing in m
portdia=getParameter('Port diameter') // Diameter of port in m
Area=getParameter('Area') // Total area in m2

// Temperatures of stream in Celsius

Tmi=getFeedProp(1,"temperature")-273
Twi=getFeedProp(2,"temperature")-273

// Properties

//Density
p1m=capeOpen1PhaseProp(handle,'density','liquid',Tmi+273,PP(1)
    ,x1) //kg/m3
p1w=capeOpen1PhaseProp(handle,'density','liquid',Twi+273,PP(2),
    x2) //kg/m3

//Viscosity
v1m=capeOpen1PhaseProp(handle,'viscosity','liquid',Tmi+273,PP
    (1),x1) // Pa s
v1w=capeOpen1PhaseProp(handle,'viscosity','liquid',Twi+273,PP
    (2),x2) // Pa s

//Thermal Conductivity
km=capeOpen1PhaseProp(handle,'thermalConductivity','liquid',
    Tmi+273,PP(1),x1) // W/mK
kw=capeOpen1PhaseProp(handle,'thermalConductivity','liquid',Twi
    +273,PP(2),x2) // W/mK
```

```
kTi=getParameter('k') // Thermal conductivity in W/mK of
    material

//Specific  Heat Capacity
Cpm=capeOpen1PhaseProp(handle,'heatCapacityCp','liquid',Tmi
    +273,PP(1),x1)/1e3//k J/kg K
Cpw=capeOpen1PhaseProp(handle,'heatCapacityCp','liquid',Twi
    +273,PP(2),x2)/1e3// kJ/kg K

//Number of Channels
Nchannel=60

//Number of Plates as specified  by the user
No=getParameter('No')

//=========================
// Calculation Section
//=========================

//Prandtl Number
Prm=Cpm*1e3*v1m/km
Prw=Cpw*1e3*v1w/kw

Carea=Ps*Pw //Cross sectional area in m^2
de=2*Ps       // Equivalent diameter in m
parea=(%pi/4)*portdia*portdia // Port area in m^2

// Methanol

// Heat transfer  coefficient

u(1)=m(1)/(Carea*p1m*No) // Velocity in m/s
Re(1)=p1m*u(1)*de/(v1m) // Reynolds Number
Nu(1)=0.26*((Re(1))^0.65)*(Prm^0.4) // Nusselt Number
h(1)=Nu(1)*km/de // Heat transfer coefficient in W/m^2K

// Brackish Water
```

```
// Heat transfer  coefficient

u(2)=m(2)/(Carea*p1w*No)// Velocity in m/s
Re(2)=p1w*u(2)*de/(v1w)// Reynolds Number
Nu(2)=0.26*((Re(2))^0.65)*(Prw^0.4) // Nusselt Number
h(2)=Nu(2)*kw/de // Heat transfer coefficient in W/m^2K

// Fouling factors

Fw=getParameter('Fw')//in W/m^2K
Fm=getParameter('Fm')//in W/m^2K

// Overall  coefficient

Uinv=(1/h(1))+(1/h(2))+(1/Fm)+(1/Fw)+(th/kTi)
U=1/Uinv //in W/m^2K

// Pressure drop

    // Methanol
    Jf(1)=0.6*(Re(1))^-0.3 // Friction factor
    Lp(1)=Pl*1 // Path length in m
    Pd(1)=8*Jf(1)*(Lp(1)/de)*p1m*u(1)*u(1)/2 // Plate pressure
        drop in Pa
    up(1)=(m(1)/p1m)/parea // Velocity through port in m/s
    Pt(1)=1.3*p1m*(up(1)^2)/2 // Pressure drop due to port loss
        in Pa
    PT(1)=Pt(1)+Pd(1) // Total Pressure Drop in Pa

    //Water
    Jf2=0.6*(Re(2))^-0.3// Friction factor
    Lp(2)=Pl*1// Path length in m
    Pd(2)=8*Jf2*(Lp(2)/de)*p1w*u(2)*u(2)/2// Plate pressure
        drop in Pa
    up(2)=(m(2)/p1w)/parea// Velocity through port in m/s
```

```
    Pt(2)=1.3*p1w*(up(2)^2)/2// Pressure drop due to port loss in
        Pa
    PT(2)=Pt(2)+Pd(2) // Total Pressure Drop in Pa

// Heat Duty calculations

Ch=Cpm*m(1)
Cc=Cpw*m(2)

Cmin=min(Ch,Cc)
Cmax=max(Ch,Cc)

Qmax=Cmin*(Tmi-Twi) // Maximum heat transferred in W

NTU= (U*Area)/(Cmin*1e3)// Number of transfer units

Cr=Cmin/Cmax // Heat capacity ratio

Eff=(1- %e^(-NTU*(1-Cr)))/(1-Cr*(%e^(-NTU*(1-Cr)))) //
    Effectiveness

Qact=Eff*Qmax // Actual heat transferred in W

// Outlet temperatures

Tmo=Tmi-(Qact/Ch)
Two=Twi+(Qact/Cc)

//=========================
// Output Section
//=========================

setProduct(1,M(1),x1,'temperature',Tmo+273, 'pressure', PP(1)-
    PT(1))
setProduct(2,M(2),x2,'temperature',Two+273, 'pressure', PP(2)-
    PT(2))
```

//==========

## 7.6 Results

| Master Property Table | | | | | |
|---|---|---|---|---|---|
| Object | Waterout | Waterin | Methout | Methin | |
| Temperature | 310.023 | 298.15 | 317.02 | 350 | K |
| Pressure | 24727.5 | 101325 | 84648.5 | 101325 | Pa |
| Mass Flow | 68.9 | 68.9 | 27.8 | 27.8 | kg/s |
| Molar Flow | 3824.59 | 3824.59 | 867.611 | 867.611 | mol/s |
| Mixture Molar Enthalpy | -43088.5 | -43976.2 | -35895.9 | 2405.39 | kJ/kmol |

Figure 7.1: Property table

## 7.7 Nomenclature

### 7.7.1 Latin

| | |
|---|---|
| A | Total area in $m^2$ |
| C | Heat capacity in J/K |
| $C_{area}$ | Cross sectional area in $m^2$ |
| $C_c$ | Cold fluid heat capacity in J/K |
| $C_h$ | Hot fluid heat capacity in J/K |
| $C_{max}$ | Maximum heat capacity in J/K |
| $C_{min}$ | Minimum heat capacity in J/K |
| Cp | Specific heat capacity in J/kg K |
| d | Port diameter in m |
| $d_e$ | Equivalent diameter in m |
| $F_c$ | Fouling factor in cold fluid in $m^2K/W$ |
| $F_h$ | Fouling factor in hot fluid in $m^2K/W$ |
| $h_1$ | Heat transfer coefficient of the hot fluid in $W/m^2K$ |
| $h_2$ | Heat transfer coefficient of the cold fluid in $W/m^2K$ |
| $J_f$ | Friction factor |
| k | Thermal conductivity in W/m K |
| $k_{Ti}$ | Thermal conductivity in W/m K |
| $L_p$ | Path length in m |
| m | Mass flow rate in kg/s |
| No | Number of plates |
| NTU | Number of Transfer Units |
| $p_{area}$ | Port area in $m^2$ |
| Pr | Prandtl Number |
| Ps | Plate spacing in m |
| Pw | Plate width in m |
| $Q_{act}$ | Actual heat transfer in W |
| $Q_{max}$ | Maximum heat transfer in W |
| Re | Reynolds Number |
| th | Plate thickness in m |
| T | Temperature in K |
| u | Velocity in m/s |
| $u_p$ | Port velocity in m/s |
| U | Overall heat transfer coefficient in $W/m^2K$ |

### 7.7.2 Greek

| | |
|---|---|
| $\Delta P$ | Total pressure drop in Pa |
| $\Delta P_d$ | Plate pressure drop in Pa |
| $\Delta P_t$ | Port pressure drop in Pa |
| $\epsilon$ | Effectiveness |
| $\mu$ | Viscosity in Pa s |
| $\pi$ | Ratio of circumference to diameter |
| $\rho$ | Density |

### 7.7.3 Subscript letters

| | |
|---|---|
| c | Cold fluid |
| h | Hot fluid |
| i | Input |
| o | Output |

## 7.8 Additional Notes

It is assumed in this model that the expression relating $\epsilon$ and NTU is same as that of a counter-current heat exchanger.