Technische Universiteit
Eindhoven
University of Technology

Faculty of Mechanical Engineering

Internship report
Numerical Methods for 1D Polymeric Membrane Model

Quartile 1: 2018 - 2019

Supervisors:	Dr. Miguel Sierra Aznar
	Prof. J.Y. Chen
	Prof. J.A. Oijen

Student:	C.E.A.G. van Gool
ID number:	0885992

Eindhoven, February 9, 2019

Acknowledgement

First of all I would like to thank Miguel Sierra Aznar to make it possible to do my internship at UC Berkeley. Thereby I am really grateful to Charles Scudiere, James Urban and Maricarmen Thomsen for helping me out on both work and social fields. Last but not least I would like to express my gratitude towards my graduation Professor Jeroen van Oijen of the University of Technology Eindhoven, who offered me the opportunity to do an internship at UC Berkeley, by introducing Miguel Sierra Aznar.

Nomenclature

Symbol	Unit	Description
γ		selectivity
δ	[-]	partial derivative
η	[-]	efficiency
θ	[-]	stage-cut
κ	[-]	specific heat ratio
ρ	kg/m^3	density
ϕ		a general property appearing in the general transport equation
Γ		diffusion/conduction coefficient
a	[-]	coefficient of the differencing scheme
L	[m]	length
$\dot{m}$	kg/s	mass flow rate
p	Pa	pressure
r	[-]	pressure ratio
t	s	time
u	m/s	velocity
$\dot{w}$		source/sink term
x	m	distance
D		diffusion conductance
D		diffusivity
F		convective mass flux
MW	kg/mol	molar weight
N	[-]	number of grid cells
Pe		Péclet number
R_u		universal gas constant
T	K	temperature
X	[-]	mole fraction
Y	[-]	mass fraction

Subscript	Description
e	east face
w	west face
E	east cell centre
W	west cell centre
k	species k
mix	mixture

Superscript	Description
f	feed
p	permeate
r	retentate
in	inlet
$\circ$	conditions at current time step

Contents

1 Introduction

As the world population keeps on growing, so does the need for energy. The increasing demand of energy requires more sustainable and efficient energy production methods as the amount of air pollution must decrease which is stated by the Paris climate accord. So the challenge at this moment is not only generating more power, it is also about generating energy with less pollution or even better, without any pollution. The use of power technologies that integrate carbon capture such as the Allam Cycle and the Argon Power Cycle (APC) are gaining interest. In the case of the latter, polymeric membrane separation is a promising technology for cost-effective integration of carbon capture onto large size internal combustion engines used today to compensate intermittent renewable sources.
The APC is an innovative technology which either improves thermodynamic efficiency and reduces emission. This is achieved by replacing air with argon as working fluid. Other noble gases such as helium could be used as well, but argon is preferred due to its abundance, non-toxicity and relatively low cost compared to other gases. The increase in thermal efficiency mostly relies on the high specific heat ratio of mono-atomic gases. The influence of the specific heat ratio on the thermal efficiency is clearly apparent in the *Otto-cycle* where the efficiency is described with:

$$\eta = 1 - \frac{1}{r^{\kappa - 1}} \tag{1.1}$$

Although hydrogen is the perfect fuel for the APC because it does not produce carbon dioxide, it suffers from knocking. Due to high heat capacity ratio, the auto-ignition temperature is reached at relatively low compression ratio r and this can lead to knocking. Methane offers a good solution to this problem.

The gasses of CO_2, H_2O and Ar should be separated from the exhaust stream to achieve a closed loop combustion of methane. The first separation stage is done using a condensation unit through which water is removed from the system. The innovation in this work consists of the use of membrane separation. The remaining CO_2 and Ar are separated using a set of membranes and compressors. This results in a high purity stream of CO_2 (95%), which may be injected in CO_2 pipelines for further use in Enhanced Oil Recovery (EOR) or in chemical processes. [1]

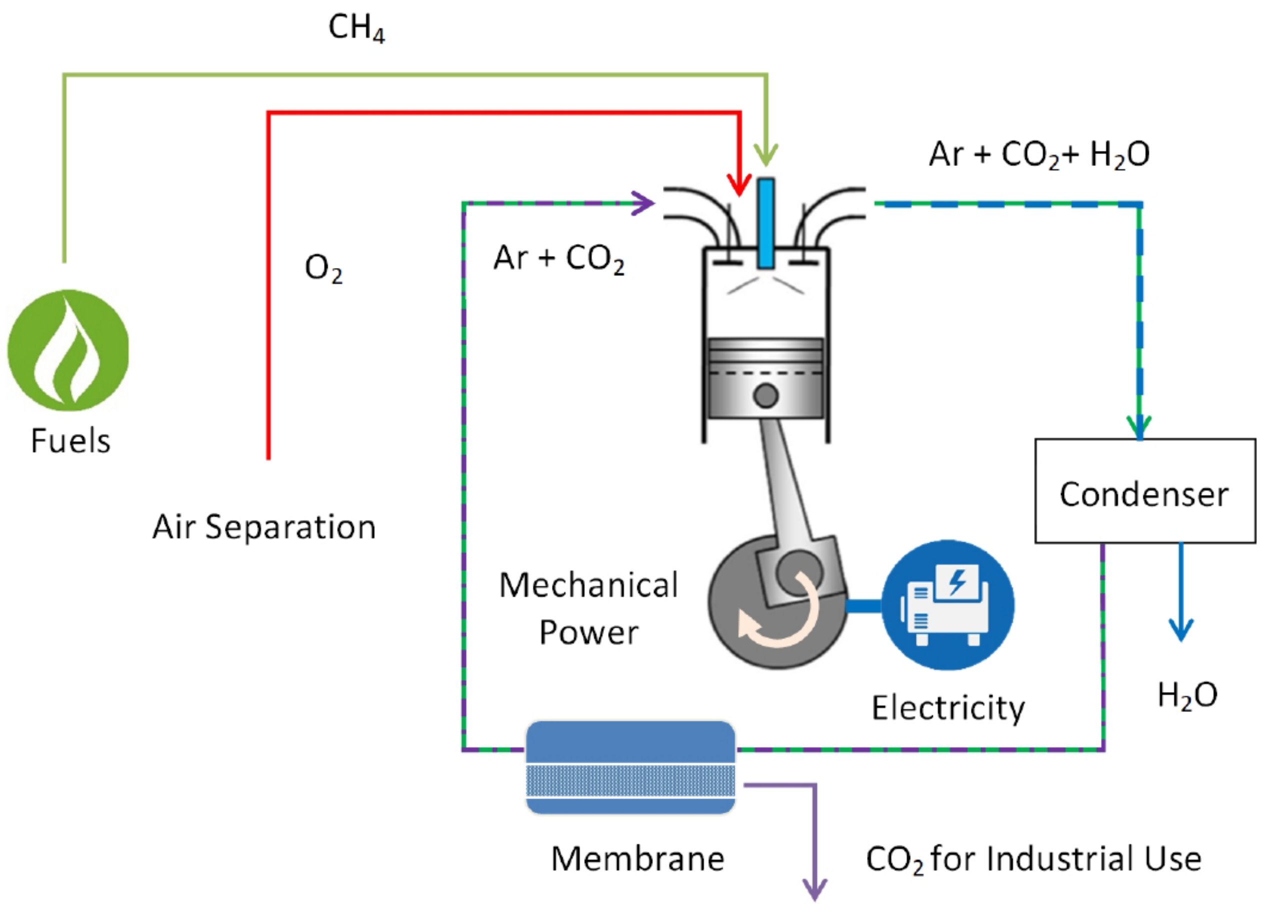


Figure 1.1: *Basic process diagram of APC in a methane energy storage scheme [1]*

To investigate the process, a one dimensional model which accounts for mass and species conservation is developed and solved using Finite Volume Method (FVM) in MATLAB®. The target is to develop a simplified code for transient APC systems.

A non-polluting powerplant would provide several benefits, but the most important one is that there is less need for flexibility as it could operate without time constraints from the government.

2 Theory

2.1 Membrane gas separation theory

Hollow fiber membrane systems have somewhat the same configuration as tube and shell heat exchangers, as can be seen in 2.1. The feed enters at one side of the membrane unit and flows through the tube or shell. While flowing through the tube, fluid will be permeated through the membrane walls. The remaining fluid leaving the membrane at the opposite end is called the retentate. The fluid that has permeated, the permeate, leaves the system through an outlet passage at either end of the membrane, depending on whether the configuration is counter-current (CC) or co-current (CO).

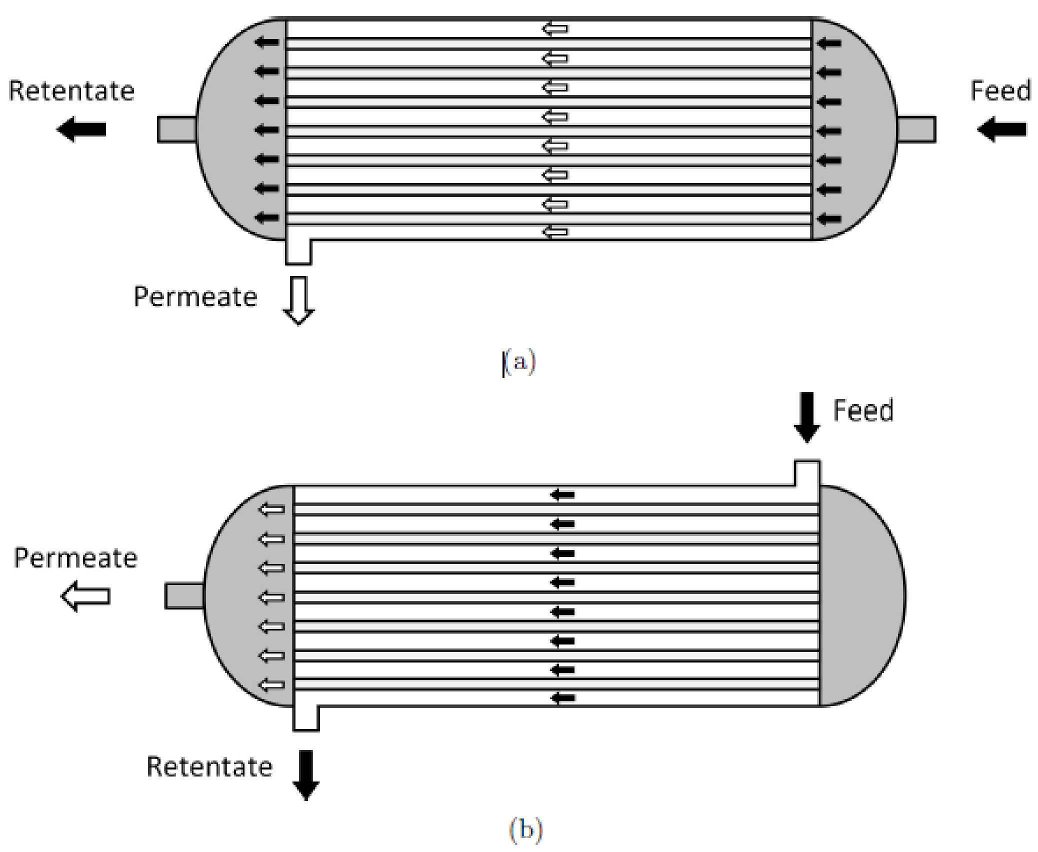


Figure 2.1: *Representation of a hollow figer membrane setup; black arrows indicate fee/retentate flow, and white arrows indicate permeate flow: (a) bore feed, (b) shell feed [1]*

The mass-flux over the membrane is controlled by P_k, the permeability of species k. The selectivity is given by the ratio of the permeabilities over the lowest permeability $\gamma_k = P_k/min\{P_k\}$. Low permeabilities lead typically to high selectivities and vice versa.
The physics related to gas separation is complicated since it depends on:

1. the mass transfer mechanism across the membrane layer.
2. the flow pattern on the retentate and permeate side, which depends on; the pressure, velocity, species mass fraction and temperature.

2.2 Discretization schemes

In this section the central differencing scheme (CD), the upwind differencing scheme and the hybrid differencing scheme for a convection diffusion problem will be discussed as well as some properties of discretisation schemes.

Consider the one dimensional steady convection-diffusion equation for a general property ϕ in the absence of sources:

$$\frac{d}{dx}(\rho u\phi) = \frac{d}{dx}(\Gamma\frac{d\phi}{dx}) \tag{2.1}$$

The discretised convection-diffusion equation for the central, upwind and hybrid differencing schemes of a one-dimensional problem can be written as:

$$a_P\phi_P = a_W\phi_W + a_E\phi_E \tag{2.2}$$

with the coefficient of volume-cell 'P' as

$$a_P = a_W + a_E + (F_e - F_w) \tag{2.3}$$

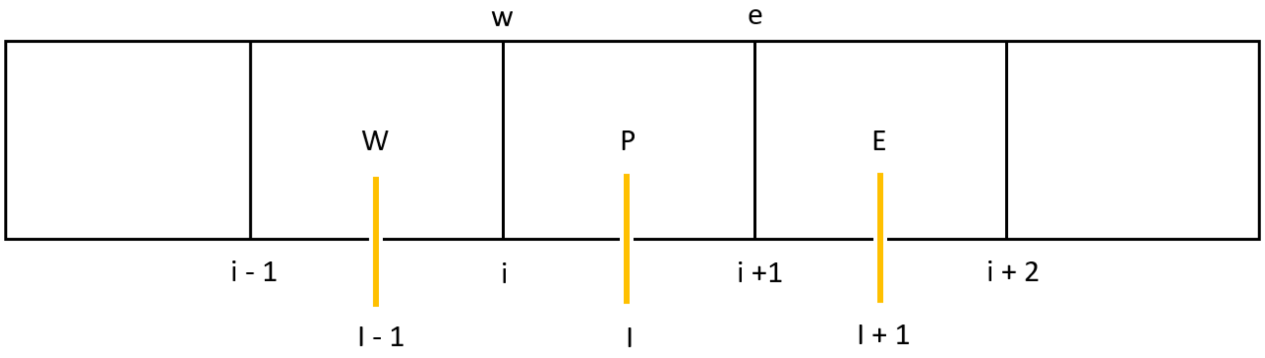


Figure 2.2: *Discretised staggered grid.*

The neighbour coefficients for these schemes are

Scheme	a_W	a_E
Central differencing	$D_w + \frac{F_w}{2}$	$D_e - \frac{F_e}{2}$
Upwind differencing	$D_w + max\left[F_w, , 0\right]$	$D_e + max\left[0, -F_e\right]$
Hybrid differencing	$max\left[F_w, \left(D_w + \frac{F_w}{2}\right), 0\right]$	$max\left[-F_e, \left(D_e - \frac{F_e}{2}\right), 0\right]$

In Figure 2.2 a schematic representation of the cell volumes is shown.

The boundary conditions enter the discretised equations via source terms. Their treatment is specific to each discretisation scheme.

Discretisation schemes that possess conservativeness, boundedness, and transportiveness give physically realistic results and stable iterative solutions. The central differencing method lacks transportiveness and gives unrealistic solutions for large values of the cell Péclet number. Hence it is not suitable for general-purpose convection-diffusion problems. Upwind and hybrid differencing schemes both possess conservativeness, boundedness and transportiveness and are highly stable, but suffer from false diffusion for multi-dimensional flows if the velocity vector is not parallel to one of the co-ordinate directions. In this work the hybrid scheme is preferred due to its stability and it has second order accuracy in terms of Taylor series for low Pe numbers, $Pe < 2$.

2.3 Numerical solver

The discretization of equations is discussed in the previous section. This resulted in a system of linear algebraic equations which needs to be solved. The size and complexity of the system depends on the number of grid points, the discretisation method and the dimensionality of the problem. There exist two families of solution techniques for linear algebraic equations: **direct methods** and **indirect** or **iterative methods.** This section only deals with the tri-diagonal matrix algorithm (TDMA), a direct method, and the Gauss-Seidel point-iterative method.

2.3.1 Gaus-Seidel

The Gaus-Seidel method is a point-iterative algorithm and easy to implement. One condition for the iteration process to be convergent is that the matrix must be diagonally dominant. The finite volume method yields diagonally dominant systems as part of the discretisation process, so this aspect does not require special attention. The main disadvantage of this method is that the convergence rate can be slow when the system of equations is large.

2.3.2 TDMA

The tri-diagonal matrix algorithm is a direct method for one-dimensional situations, but it can be applied iteratively to solve multi-dimensional problems. It consists of a forward elimination and back-substitution stage. This algorithm requires only a minimum amount of storage and is computationally inexpensive.

3 Membrane model

3.1 Governing Equations

The flow direction of the gas streams is parallel and in the same direction on both sides of the membrane. Figure 3.1 shows the schematic of the co-current flow in a membrane separator.

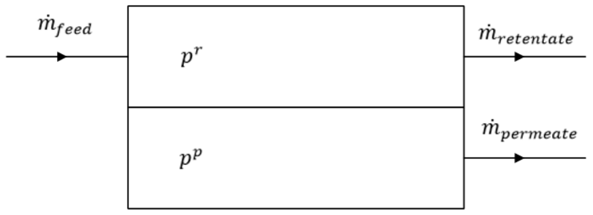


Figure 3.1: *Schematic of the co-current flow in a membrane separator.*

The feed with a specific mass flow frate $\dot{m}^f$ and composition enters the unit and is divided into two streams: $\dot{m}^p$ on the permeate side and $\dot{m}^r$ leaving on the retentate side. The flow is assumed to be ideal, frictionless, incompressible, isothermal and at constant velocity. The continuity and species continuity equations are presented in 1D Cartesian coordinates for the above assumptions. The continuity equation is defined as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0 \tag{3.1}$$

Note that for an incompressible fluid the density of the individual species does not change. However, the density of the mixture can change due to permeation which influences the fluid composition and is computed at the end of each time step according to:

$$\rho = \frac{p}{R_u T} \sum_{k=1}^{N} X_k (MW)_k \tag{3.2}$$

The species conservation equation at the retentate side is given by:

$$\frac{\partial (\rho Y_k^r)}{\partial t} + \frac{\partial}{\partial x} (\rho u Y_k^r) = \frac{\partial}{\partial x} \left(\rho D_k \, \frac{\partial Y_k^r}{\partial x} \right) + \dot{w}_k \tag{3.3}$$

with Y_k^r the mass fraction of specie k, ρ the density of the mixture, u the velocity of the fluid, D_k the diffusivity of specie k and $\dot{w}_k$ the source/sink term of specie k. The sink term is defined as:

$$\dot{w}_k = -P_k \Big[p^r MW_{mix}^r Y_k^r - p^p MW_{mix}^p Y_k^p \Big] \tag{3.4}$$

With P_k the permeability of specie k, p^r and p^p the pressure at retentate and permeate side respectively and MW_{mix} the molar weight of the mixture. Replacing Y_k^r by Y_k^p and flipping the sign of the sink term in Equation 3.4 gives the species conservation equation at the permeate side.

The stage cut θ, is a parameter which gives information on the performance of the membrane and is defined as:

$$\theta = 1 - \frac{\dot{m}^r}{\dot{m}^f} \tag{3.5}$$

3.2 Boundary conditions

The following boundary conditions are applied to the problem described above:

$$Permeate = \begin{cases} x = 0 \ : \quad u = 0 \\ x = L \ : \ \frac{\partial Y_k}{\partial x} = 0 \end{cases}$$

$$Retentate = \begin{cases} x = 0 \ : \quad u = u_{in} \ , \ Y_k = Y_{k,in} \\ x = L \ : \ \frac{\partial Y_k}{\partial x} = 0 \end{cases}$$

For the transient cases the feed fuel composition is a function of time, e.g. $Y_{in} = Y_{in}(t)$.

There is no need for pressure boundary conditions due to the assumption of constant pressure at retentate and permeate side.

3.3 Numerical method

The fully implicit method is used because of its superior stability. The implicit discretised form of Equation 3.3 is given by:

$$a_P Y^r_{k,P} = a_W Y^r_{k,W} + a_E Y^r_{k,E} + b \tag{3.6}$$

where

$$a_P = a_W + a_E + a^\circ_P + \Delta F \tag{3.7}$$

with

$$a^\circ_P = \frac{\rho^\circ_P \Delta x}{\Delta t} \tag{3.8}$$

and

$$b = \dot{w}_k \Delta x + a^\circ_P Y^{r,\circ}_{k,P} \tag{3.9}$$

In the equations above the superscript $\circ$ refers to the value of the current time step. The neighbour coefficients for the internal nodes of this equation for the hybrid differencing scheme are as follows:

a_W	a_E	ΔF
$max\left[F_w, \left(D_w + \frac{F_w}{2}\right), 0\right]$	$max\left[-F_e, \left(D_e - \frac{F_e}{2}\right), 0\right]$	$F_e - F_w$

In the above expressions the values of F and D are calculated with the following formulae:

$Face$	w	e
F	$(\rho u)_w A_w$	$(\rho u)_e A_e$
D	$\frac{D_k}{\delta x_{WP}} A_w$	$\frac{D_k}{\delta x_{PE}} A_e$

Note that for a one-dimensional case the cell face area is equal to 1. [2]
Special attention is needed for the boundary nodes. For the first node the coefficient a_W is set to 0 and a_E remains unchanged. The coefficient a_P is computed as:

$$a_P = max\left[F_w, \left(D_w + \frac{F_w}{2}\right), 0\right] + a_E + a^\circ_P + F_e - F_w \tag{3.10}$$

The source term for the first node is:

$$b = \dot{w}_k \Delta x + (max\left[F_w, \left(D_w + \frac{F_w}{2}\right), 0\right] + a_P^\circ) \cdot Y_{k,in} \tag{3.11}$$

For the last node the coefficient a_E is set to 0 and a_W remains unchanged. The coefficient a_P is computed as:

$$a_P = a_W + max\left[-F_e, \left(D_e - \frac{F_e}{2}\right), 0\right] + a_P^\circ + F_e - F_w \tag{3.12}$$

At every time step the boundary condition $\frac{\partial Y_k}{\partial x}|_{x=L} = 0$ is enforced.

3.4 Validation

3.4.1 The solver

The solver and hybrid scheme are tested with a simple example from the book "An Introduction to Computational Fluid Dynamics" written by H.K. Versteeg and W. Malalasekera. [2] The case conditions are as follows:

$N = 5;\ L = 1;\ \delta x = \frac{L}{N} = 0.2;\ u = 2.5\ m/s;\ F = \rho u = 2.5;\ D = \frac{\Gamma}{\delta x} = \frac{0.1}{0.2} = 0.5;\ Pe = \frac{F}{D} = 5$
with boundary conditions: $\phi = 1$ at x = 0 and $\phi = 0$ at x = L.

The obtained matrix form of the equation set is

$$\begin{bmatrix} 3.5 & 0 & 0 & 0 & 0 \\ -2.5 & 2.5 & 0 & 0 & 0 \\ 0 & -2.5 & 2.5 & 0 & 0 \\ 0 & 0 & -2.5 & 2.5 & 0 \\ 0 & 0 & 0 & -2.5 & 2.5 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The solution obtained with the TDMA solver is:

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0.7143 \end{bmatrix}$$

which is exactly the same as the provided solution.

3.4.2 Model

The code is validated by comparing with results from here source of pictures. The pressure on retentate and permeate are 4 and 0.4 bar respectively. The feed gas consists of mole percentages:
$H_2O = 0.97$ $CO_2 = 10.33$ $O_2 = 1.54$ and $AR = 87.16$. The total length is set to 1 m and divided over 200 control volumes such that $\Delta x = 0.005$. The time step is 0.5 and the simulation time is 200. The velocity must be high enough such that the flow is convection controlled rather than diffusion controlled. This also means that the cell Péclet number must be at least larger than 2. The density is set to 1 in equation:

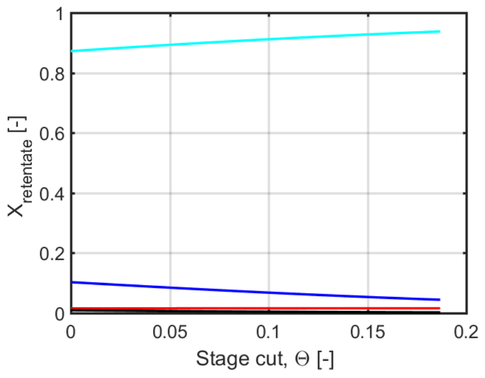
$$\frac{\partial(\rho Y_k)}{\partial t} + \nabla \cdot (\rho u Y_k) = \nabla \cdot (\rho D_k\ (\nabla \cdot Y_k)) + \dot{w}_k \tag{3.13}$$

Equation 3.13 has to be solved twice, once for the retentate and once for the permeate side.

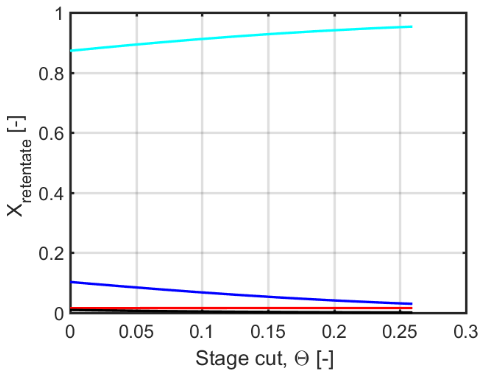
In the original case the mass-flow rate was set to 15.4292 kg/h which is different from the massflow rate used as boundary conditions in this case (which is equal to 5400 kg/h). The reason for this choice is the fact that the equation must be convection controlled to produce reliable results. The formula is convection controlled when $Pe = \frac{\rho u}{(\Gamma/\partial x)} > 2$. However, it is still possible to compare the concentrations of both cases by looking at the same stage cut, which is a dimensionless parameter. The results are shown in Table 3.1, Figure 3.2 and Figure 3.3

Table 3.1: *Caption*

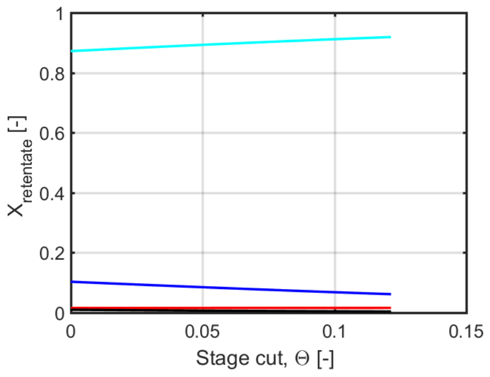
	Farouck	Case 1	Case 2	Case 3	Case 3*
L	1	1	1	1	2
Θ	0.1598	0.1598	0.1598	0.1215	0.1598
P^r	4	4	4	4	4
P^p	0.4	0.4	0.4	0.4	0.4
$\dot{m}$	15.4292	5400	3600	9000	9000
$X_{H_2O}^{r,in}$	0.973272	0.973272	0.973272	0.973272	0.973272
$X_{CO_2}^{r,in}$	10.329175	10.329175	10.329175	10.329175	10.329175
$X_{O_2}^{r,in}$	1.541694	1.541694	1.541694	1.541694	1.541694
$X_{AR}^{r,in}$	87.155861	87.155861	87.155861	87.155861	87.155861
$X_{H_2O}^{r,out}$	0.269631	0.280611	0.280908	0.384081	0.279205
$X_{CO_2}^{r,out}$	5	5.109224	5.107196	6.173649	5.098282
$X_{O_2}^{r,out}$	1,594795	1.594100	1.593979	1.588198	1.594247
$X_{AR}^{r,out}$	93.135571	93.016066	93.017917	91.854072	93.0282656
$\bar{X}_{H_2O}^{p}$	4.711435	4.543060	4.534002	5.192736	4.541872
$\bar{X}_{CO_2}^{p}$	38.640964	38.137105	38.137651	40.739224	38.102186
$\bar{X}_{O_2}^{p}$	1.259588	1.272995	1.272973	1.207574	1.273867
$\bar{X}_{AR}^{p}$	55.388010	56.046840	56.055374	52.860466	56.082075



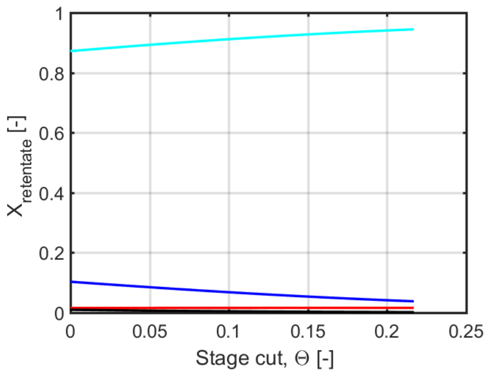
(a)



(b)

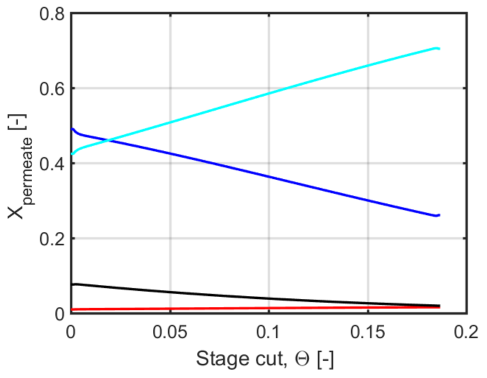


(c)

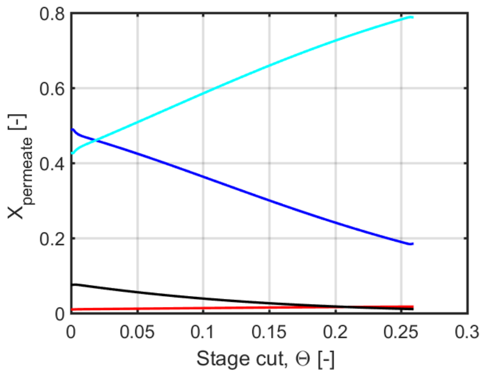


(d)

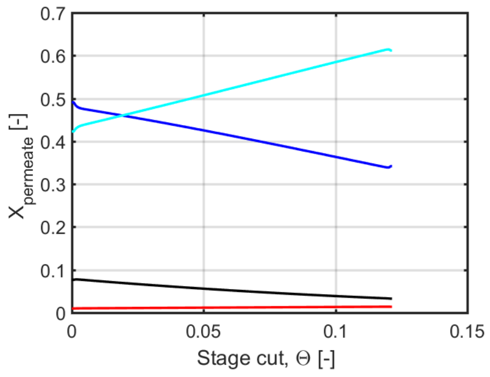
Figure 3.2: *a,b,c,d) X^r for cases 1,2,3 and 3^* respectively*



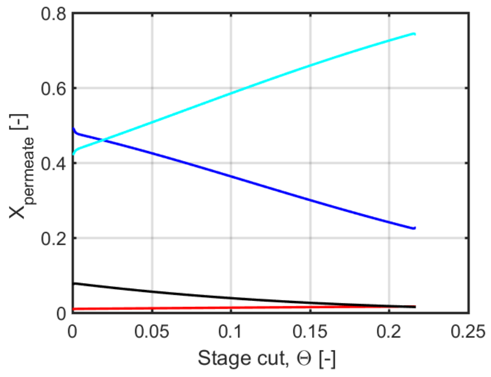
(a)



(b)



(c)



(d)

Figure 3.3: *a,b,c,d) X^p for cases 1,2,3 and 3^* respectively*

From the results it follows that at the same stage cut there is very little difference between the reference case (Faroucks results) and all other cases. Meaning that the velocity/ mass-flow rate has very little influence on the composition at a certain stage cut. However, the velocity does significantly influence the amount of membrane transport. In case 2, where the velocity is only 1 m/s and the mass-flow rate 3600 kg/h, the stage cut reaches a much larger value than for case 1 and 3. In case 3 the stage cut doesn't even reach the same value as the reference case. Hence, a second case 3 (case 3^*) has been simulated with a length of 2 m instead of 1.

4 Sensitivity

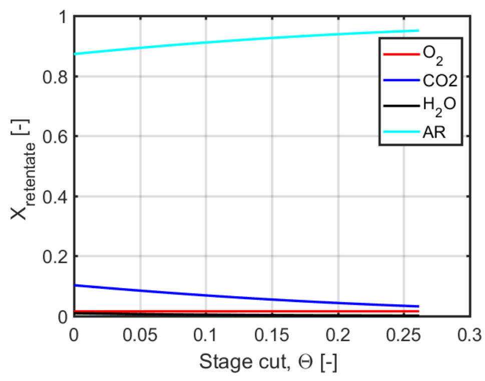
A sensitivity analysis is performed on the pressure ratio and permeability of the species. First the results on the pressure ratio will be discussed. All results will be compared with a reference case, case 0, which has the following geometric and thermodynamic properties:

The length of the domain is 1 meter and the grid is divided in 200 equal sized grid cells such that $\Delta x = 0.005$ meter. The inlet velocity is 1 m/s and remains constant. The temperature is 298 K and assumed to be constant over the entire domain, such that the flow can be regarded as isothermal. The total simulation time is 200 seconds and a time step of 0.05 seconds is used. The pressure at the retentate and permeate side are 400 and 40 kPa respectively, such that the pressure ratio is 0.1. The feed stream contains $X_{O_2} = 1.54\%$, $X_{CO_2} = 10.33\%$, $X_{H_2O} = 0.97\%$ and $X_{AR} = 87.16\%$. The permeability of O_2, CO_2, H_2O and AR are 28, 350, 1750 and 21 $\frac{n\ mol}{m\ s\ Pa}$ respectively

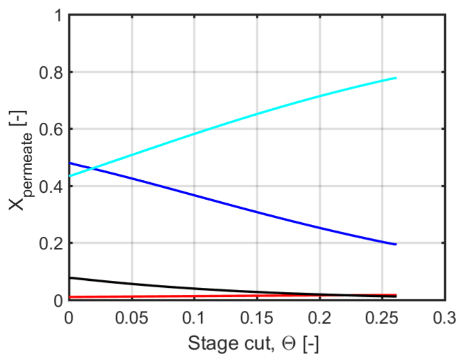
4.1 Pressure ratio

	case 0	case A1	case A2	case A3
Θ [-]	0.261631	0.217092	0.116270	0.283972
P^r [Pa]	400	400	400	400
P^p [Pa]	40	80	200	20
$X_{H_2O}^{r,out}$ [-]	0.135988	0.135988	0.869079	0.021645
$X_{CO_2}^{r,out}$ [-]	3.239114	5.728823	9.343992	1.848629
$X_{O_2}^{r,out}$ [-]	1.593139	1.562543	1.535189	1.610194
$X_{AR}^{r,out}$ [-]	95.031758	92.270670	88.251740	96.519533
$X_{H_2O}^{p}$ [-]	3.332514	3.075152	1.811850	2.942572
$X_{CO_2}^{p}$ [-]	31.952574	28.636889	18.284952	32.553121
$X_{O_2}^{p}$ [-]	1.416297	1.467233	1.594041	1.422606
X_{AR}^{p} [-]	63.298615	66.820727	78.309158	63.081701

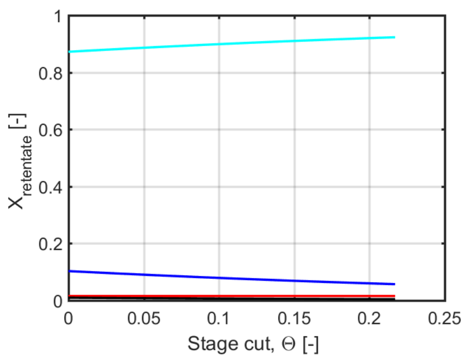
From these results one can conclude that the concentration of species is quite sensitive to a change in pressure ratio. A lower pressure ratio results in more permeation and hence a larger value for the stage-cut and a decrease in concentration at the retenate side of species with a relative large permeability, as these species are most likely to permeate. Also, the mole fraction at the permeate side will increase for those species, due to the increase in permeation.
Comparing the reference case to cases with a larger pressure ratio shows that less species will permeate as the stage-cut decreases. As a consequence, the species with higher permeability will have a higher mole fraction at the retenate side and a lower mole fraction at the permeate side. For the species with a relative low permeability an increase of mole fraction is observed at the permeate side and a decrease at the retenate side. The change in mole fraction for the less permeating species is mainly because of the change in permeation of the species with a high permeability. The absolute value of species that has been permeated decreases for all species, no matter what the permeability is.



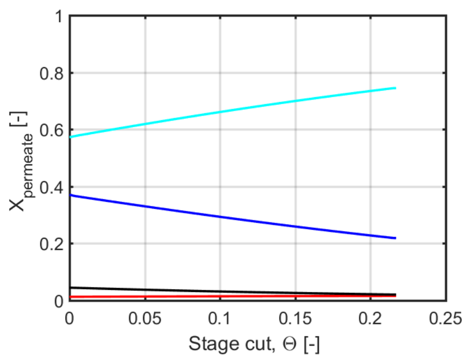
(a)



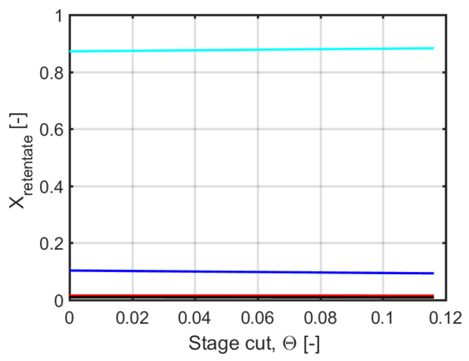
(b)



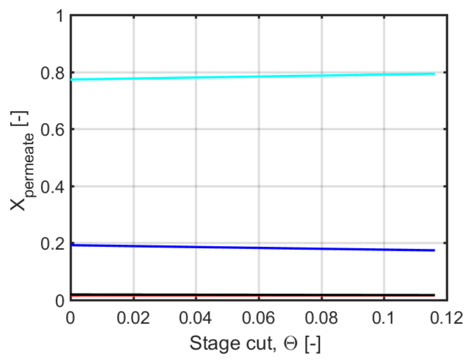
(c)



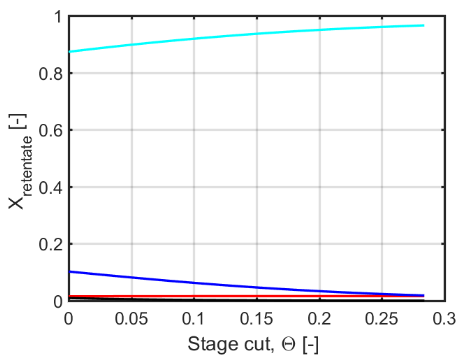
(d)



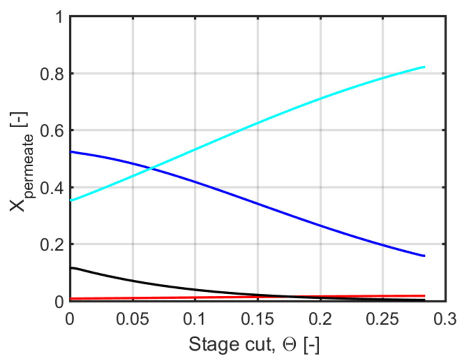
(e)



(f)



(g)



(h)

Figure 4.1: *a,c,e,g) X^r for cases 0, A1, A2 and A3 respectively.*
b,d,f,h) X^p for cases 0, A1, A2 and A3 respectively.

4.2 Permeability

This subsection treats the sensitivity of the species permeability. The permeability of only one species will be changed at the time.
Note that for better readability, the permeability in tables 4.1-4.4 is given in $\frac{n\ mol}{m\ s\ Pa}$.

Table 4.1: *Change in permeability of H_2O*

	case 0	case B1	case B2
Θ	0.261631	0.261339	0.261727
P_{H_2O}	1750	**875**	**2625**
P_{CO_2}	350	350	350
P_{O_2}	28	28	28
P_{AR}	21	21	21
$X^{r,out}_{H_2O}$	0.135988	0.179939	0.121488
$X^{r,out}_{CO_2}$	3.239114	3.241211	3.238580
$X^{r,out}_{O_2}$	1.593139	1.592407	1.593379
$X^{r,out}_{AR}$	95.031758	94.986443	95.046554
$X^{p}_{H_2O}$	3.332514	3.280739	3.341481
$X^{p}_{CO_2}$	31.952574	32.019410	31.929929
$X^{p}_{O_2}$	1.416297	1.416082	1.416550
X^{p}_{AR}	63.298615	63.283768	63.312041

Table 4.2: *Change in permeability of CO_2*

	case 0	case C1	case C2	case C3	case C4
Θ	0.261631	0.257586	0.219532	0.267289	0.272401
P_{H_2O}	1750	1750	1750	1750	1750
P_{CO_2}	350	**300**	**100**	**450**	**600**
P_{O_2}	28	28	28	28	28
P_{AR}	21	21	21	21	21
$X^{r,out}_{H_2O}$	0.135988	0.139131	0.175055	0.131866	0.128497
$X^{r,out}_{CO_2}$	3.239114	3.594803	6.800644	2.734511	2.268858
$X^{r,out}_{O_2}$	1.593139	1.587386	1.535971	1.601333	1.608944
$X^{r,out}_{AR}$	95.031758	94.678680	91.488330	95.532290	95.993701
$X^{p}_{H_2O}$	3.332514	3.401261	3.960111	3.229445	3.126479
$X^{p}_{CO_2}$	31.952574	31.511810	24.227410	32.364962	32.434709
$X^{p}_{O_2}$	1.416297	1.424700	1.571161	1.408974	1.408945
X^{p}_{AR}	63.298615	63.662229	70.241317	62.996620	63.029867

Table 4.3: *Change in permeability of* O_2

	case 0	case D1	case D2	case D3
Θ	0.261631	0.263137	0.259802	0.267235
P_{H_2O}	1750	1750	1750	1750
P_{CO_2}	350	350	350	350
P_{O_2}	28	**42**	**14**	**100**
P_{AR}	21	21	21	21
$X^{r,out}_{H_2O}$	0.135988	0.134462	0.137878	0.130443
$X^{r,out}_{CO_2}$	3.239114	3.227465	3.253670	3.197499
$X^{r,out}_{O_2}$	1.593139	1.443002	1.774387	1.029461
$X^{r,out}_{AR}$	95.031758	95.195071	94.834065	95.642597
$X^{p}_{H_2O}$	3.332514	3.314943	3.353014	3.262660
$X^{p}_{CO_2}$	31.952574	31.806157	32.128574	31.396662
$X^{p}_{O_2}$	1.416297	1.914029	0.793902	3.192724
X^{p}_{AR}	63.298615	62.964871	63.724510	62.147954

Table 4.4: *Change in permeability of AR*

	case 0	case E1	case E2
Θ	0.261631	0.166233	0.350280
P_{H_2O}	1750	1750	1750
P_{CO_2}	350	350	350
P_{O_2}	28	28	28
P_{AR}	21	**10.5**	**30.5**
$X^{r,out}_{H_2O}$	0.135988	0.259462	0.079392
$X^{r,out}_{CO_2}$	3.239114	4.166193	2.793925
$X^{r,out}_{O_2}$	1.593139	1.463864	1.127407
$X^{r,out}_{AR}$	95.031758	94.110481	95.392418
$X^{p}_{H_2O}$	3.332514	4.523370	2.652273
$X^{p}_{CO_2}$	31.952574	42.510894	26.185366
$X^{p}_{O_2}$	1.416297	2.041440	1.416550
X^{p}_{AR}	63.298615	50.924295	70.034954

From tables 4.1-4.4 one can conclude that the permeability can have a major influence on the permeation of species, but that the impact of the permeability also depends on the concentration of species. For instance, if there is only a small amount of a certain species there is not much to permeate even though the permeability might be relatively high, as is the case for H_2O in Table 4.1. If on the other hand the permeability is relative small but the concentration of species is very high, then only a small difference in permeability can causes a big difference in concentration at the permeate side, see Table 4.4.

5 Numerical Simulations

5.1 Stationary single membrane model

5.2 Stationary 3 membrane model

This case deals with a stationary ideal combustion gas separation process with pure methane as fuel. The fuel is mixed with a 20% oxygen and 5% carbon dioxide mixture and burned stoichiometric. Note that the percentages refer to the mole percentage of the mixture. The combustion takes place at station 1 in Figure 5.1. The mole percentages of the reaction products are easy to compute since stoichiometric combustion is assumed. This leads to the following reaction equation:

$$2CH_4 + 4O_2 + 15AR + CO_2 \rightarrow 4H_2O + 3CO_2 + 15AR \tag{5.1}$$

Directly behind the combustion chamber a condenser is located which will condense all water in the ideal case such that the remaining gas mixture only consists of argon and carbon dioxide. In this case the combustion products are a mixture of 16.667 % CO_2 and 83.333 % AR. This gas mixture is the feed gas stream for station 2 which is a membrane. Stations 3 and 4 represent a membrane as well.

The fluid mixture leaving the membranes at the retenate side is feeded back to the previous stage. It is desired that this mixture has the same composition as the original feed. By changing the length of the membrane it is possible to achieve this preference. A longer membrane has a lower concentration of CO_2 at the retenate side and a shorter membrane a higher CO_2 concentration. The length for each membrane is determined with the single membrane model. Once the length of the membranes are determined the 1 membrane model is extended to the 3 membrane model shown in Figure 5.1.

The gas mixture leaving the membrane at the permeate side is the feed stream for the next membrane and has a relatively low pressure. To increase the pressure a compressor is placed behind each membrane to enhance gas separation.

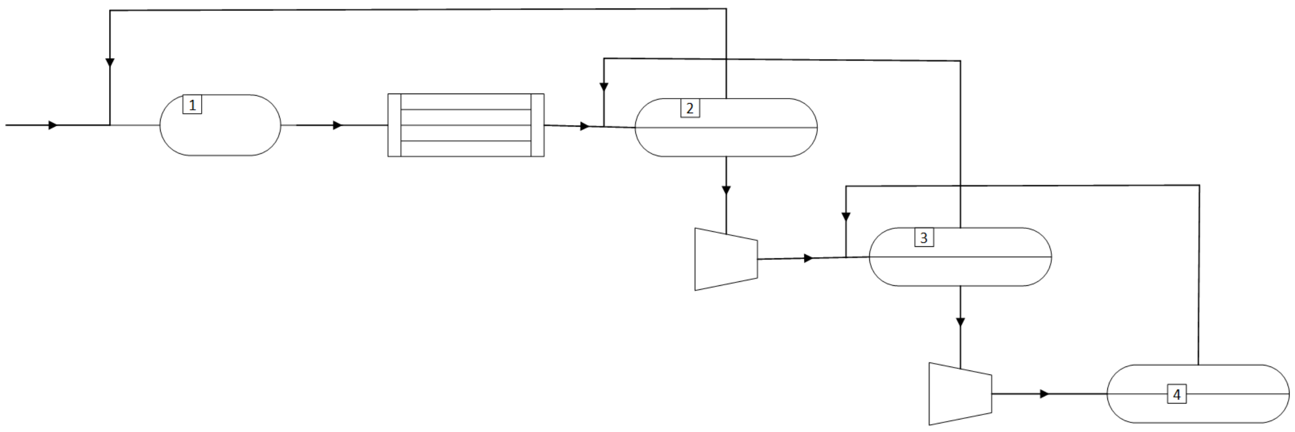


Figure 5.1: *Fluid flow chart 3 membrane model*

Table 5.1: *Fluid flow conditions 3 membrane model*

Stream no.	2	3	4
L	0.88	0.265	0.665
Θ	0.368874	0.176977	0.800115
P^r	4	4	4
P^p	0.4	0.4	0.4
$X^f_{CO_2}$	16.666667	24.660470	60.288259
X^f_{AR}	83.333333	75.339530	39.711741
$X^r_{CO_2}$	5.000000	16.630816	24.801102
X^r_{AR}	95.000000	83.369184	75.198898
$X^p_{CO_2}$	24.660470	60.288259	73.258709
X^p_{AR}	75.339530	39.711741	26.641291

5.3 Transient 3 membrane model

6 Discussion

At the start of this project the SIMPLE algorithm was used due to its ease of implementation. One of the main features of the SIMPLE algorithm is the fact that it solves for conservation of mass. However, it turned out that in case of membrane transport the SIMPLE algorithm might not be the most convenient choice due to this feature, as will be explained below.

Consider a pipe with a membrane such that a species A is going from on side to the other side. Once the mass-flow in this pipe is determined at the first node, the SIMPLE algorithm forces the momentum and pressure equation to behave such that the mass flow rate stays constant. Meaning, if there is a species leaking from side A to side B then the density at both sides changes. As a consequence of the SIMPLE algorithm, the velocity also changes because of the change in density. Physically this is not what would happen. The velocity should remain (almost) constant and the pressure should change due to the change in density.

Since the SIMPLE algorithm was not giving stable, reliable results for the problem of interest, it was decided to simplify the problem to constant velocity and constant pressure. For future work it is worthwhile to develop a code that overcomes the shortcommings described above.

The accuracy of hybrid and upwind schemes is only first-order in terms of Taylor series truncation error. The use of upwind quantities ensures that the schemes are very stable and obey the transportiveness requirement, but the first-order accuracy makes them prone to numerical diffusion errors. Such errors can be minimised by employing higher-order discretisation. Higher-order schemes involve more neighbore points and reduce the discretisation errors by bringing in a wider influence. The central differencing scheme, which has second-order accuracy, proved to be unstable and does not possess the transportiveness property. Formulations that do not take into account the flow direction are unstable and, therefore, more accurate higher order schemes, which preserve upwinding for stability and sensitivity to the flow direction, are recommended.

7 Conclusion

8 Code manual

In this section the code will be explained into detail, such that an user who is familiar with MATLAB can operate the code.

8.1 Data storage

In the beginning of the main code 2 path's, directing to a file in the results folder, is created to store the results. One file contains the data for the retenate and the other for the permeate. The piece of code that executes this task is shown below:

```
%% set path to folder where values must be stored, clean the old file
path_Results1 = ...
    'C:\Users\s137280\Documents\Master_tue\Internship\internship_github\ ...
    Toos_simplified_species_v02_02_2019\results\retenate.txt';
path_Results2 = ...
    'C:\Users\s137280\Documents\Master_tue\Internship\internship_github\ ...
    Toos_simplified_species_v02_02_2019\results\permeate.txt';

% add name taggs to created file and close file again.
test = fopen(path_Results1,'w');
fprintf(test,'%-12s %-12s %-12s %-12s %-12s %-12s %-12s \n', 'Time','Position', ...
    'u_Position', 'X1', 'X2', 'X3','X4');
fclose(test);

% add name taggs to created file and close file again.
test = fopen(path_Results2,'w');
fprintf(test,'%-12s %-12s %-12s %-12s %-12s %-12s %-12s\n', 'X2_1', 'X2_2', ...
    'X2_3', 'X2_4', 'stagecut');
fclose(test);
```

At specific times, declared in the array *store_times*, the data is stored in the files 'retenate.txt' and 'permeate.txt'. In this case the time, position, velocity-position and mole fraction of the retenate are stored in 'retenate.txt'. The stagecut and mole fraction of the permeate are stored in 'permeate.txt'. More variables could be stored as long as they have the same dimensions. Note, the variable must be added in the part above and below.

```
%     store data at different time steps
if time == store_times(ii)
time_x = time*ones(1,length(u));
file1 = fopen(path_Results1,'a');
fprintf(file1,'%-12.6f %-12.6f %-12.6f %-12.6f %-12.6f %-12.6f %-12.6f ...
    \n',[time_x; x; x_u; X_k(1,:); X_k(2,:); X_k(3,:); X_k(4,:)]);
fprintf(file1,'\n');
fclose(file1);

file2 = fopen(path_Results2,'a');
fprintf(file2,'%-12.12f %-12.12f %-12.12f %-12.12f %-12.12f %-12.12f ...
    %-12.12f\n',[X2_k(1,:); X2_k(2,:); X2_k(3,:); X2_k(4,:); theta]);
fprintf(file2,'\n');
fclose(file2);

ii = ii +1;
end
```

8.2 Initializing variables

Part of the variables are initialized in the main code. Other variables, especially variables that can have a different value for every grid cell, are initialized upon use of a function. First, the initialization in the main code will be explained, followed by an explanation of the functions *species_init()* and *param_init()*.

8.2.1 Variables in main code

Next step is to initialize a set of variables. The following set of variables are declared in the main code. The meaning of each variable is written behind it as a comment, with it's corresponding unit.

```
%% initializing
global Patm Runiv
Patm         = 101325;       % athmosphesric pressure              [Pa]
Runiv         = 8.314;       % universal gas constant              [J/mol.K]

NPI          = 200;          % number of grid cells in x-direction  [-]
XMAX         = 1;            % length of the domain                [m]
u_in         = 1;            % inflow velocity                     [m/s]
T            = 298;          % temperature                         [K]
A            = 1;            % area of one cell                    [m^2]
Total_time   = 200;          % total simulation time               [s]
Pr           = 400*10^3;     % pressure retenate                   [Pa]
Pp           = 40*10^3;      % pressure permeate                   [Pa]
w            = 1;            % width of cell domain                [m]
```

8.2.2 Species_init()

The species parameters are initialized with a separate function called '*species_init()*'. This function requires the number of grid cells, NPI, as an input and it returns:

- *rho_s*, the density of the individual species
- *rho*, an artificial density required for computation
- *Gamma* and *Gamma_k*, thermal diffusion coefficient of the individual species and thermal diffusion coefficient at the retenate side.
- *MW_mix* and *MW2_mix*, the molar weight of the mixture at the retenate and permeate side.
- *Y_k* and *X_k*, the mass and mole fraction at the retenate side.
- *Y_in* and *X_in*, the inlet composition at the retenate side in terms of mass fractions and mole fractions.
- *Y2_k* and *X2_k*, the mass and mole fraction at the permeate side.
- *Y2_in* and *X2_in*, the inlet composition at the permeate side in terms of mass fractions and mole fractions.
- *iAll*, a matrix containing the numbers of species of interest to get data from a MechanismFile, which is included in the same folder as the code.
- *MW*, the molar weight of the individual species.
- *rho_real* and *rho2_real*, the real density at the retenate and permeate side.
- *rho_old* and *rho2_old*, the old real density at the retenate and permeate side.
- *D*, *D_k* and *D2_k*, the diffusion coefficient of the individual species into another species and the average diffusion coefficient at each cell for the retenate and permeate side.
- *P_k*, the permeability of the individual species.
- *f_old* and *f2_old*, the old mass fraction at the retenate and permeate side.
- *sink*, a vector which indicates which species will leak and which not. A zero represent no leakage and a one represents leakage.
- *n*, the number of species used in the computation.

```
% species properties some values have to be set manually!!
[rho_s, rho,Gamma, Gamma_k, MW1, MW2, Y_k, X_k, Y_in, X_in, Y2_k, X2_k, Y2_in, ...
    X2_in, iAll, MW, rho_real, rho2_real, rho_old, rho2_old, D, D_k, D2_k, P_k, ...
    f_old, f2_old, sink, n] = species_init(NPI);
```

Note that the permeate side has not a real inlet condition, *Y2_in*, this variable is only used to make sure that this side is initialized with a certain mixture. The composition at the permeate side can only change due to permeation which is modelled as a source term and will be explained later. If no permeation would occur, there is no fluid flow at the permeate side and no change in composition.

The next code snippet contains the script of the function *species_init()*.

```matlab
function [rho_s, rho,Gamma, Gamma_k, MW_mix, MW2_mix, Y_k, X_k, Y_in, X_in, Y2_k, ...
    X2_k, Y_in2, X_in2, iAll, MW, rho_real, rho2_real, rho_old, rho2_old, D, D_k, ...
    D2_k, P_k, f_old, f2_old, sink, n] = species_init(NPI)

    %% Read species data
    % Make these variable global
    global El Sp Patm Runiv

    MechanismFile = 'fuels.trot';

    % Read the reaction mechanism
    [El, Sp] = ReadTrotDat(MechanismFile);

    %% Define some species
    iO2    = find(strcmp({Sp.Name},'O2'));
    iCO2   = find(strcmp({Sp.Name},'CO2'));
    iH2O   = find(strcmp({Sp.Name},'H2O'));
    iAr    = find(strcmp({Sp.Name},'AR'));

    iAll   = [iO2 iCO2 iH2O iAr];

    MW    = [Sp(iAll).Mass];                    % Molar weight of species [gr/mol]
    n = length(iAll);

    %% mole flow rate in system and sink term
    moles = [1.541694; 10.329175; 0.973272;  87.155861];
    moles2 =[0  ;  0 ; 0 ; 1];    % code will crash if everything is set to 0

    X_in   = moles/sum(moles);
    X_in2  = moles2/sum(moles);

    sink   = [1  1  1  1];          % this array can prevent species from leakage
    P_n    = [28; 350; 1750; 21]*10^(-9);  % Permeability of species

    Gamma = [ThermProp(300,Sp(iO2),'Gamma','Mass') ...
        ThermProp(300,Sp(iCO2),'Gamma','Mass') ...
        ThermProp(350,Sp(iH2O),'Gamma','Mass') ThermProp(350,Sp(iAr),'Gamma','Mass')];
    rho_s = [1.429 1.98 1.2504 1.784];    % 'Real' density of species [kg/m^3]

    for i = 1:n
        Y(i,:)  = X_in(i)*MW(i)/(MW*X_in);
        Y2(i,:)  = X_in2(i)*MW(i)/(MW*X_in2);

    end

    Y_in   = Y(:,1);
    Y_in2  = [0;0;0;0];

    MW1 = MW*X_in;                          % molar weight of mixture
    MW2 = MW*X_in2;                         % molar weight of mixture

    D        = species_diff(NPI, 300, iAll, iAll, 'Diffusivity', n); % Diffusivity ...
        of species [m^2/s]

    for j = 1:n

        for I = 1:NPI+2
            p_k  (j,I)    = Patm;           % pressure of species k
            Y_k(j,I)      = Y(j);           % mass of species k
            Y2_k(j,I)     = Y2(j);          % mass of species k
```

```
                MW_k(j,I)    = MW(j);            % molar weight of species k
                P_k(j,I)     = P_n(j);           % Permeability
                X_k(j,I)     = X_in(j);
                X2_k(j,I)    = X_in2(j);
                MW_mix(1,I)  = MW1;
                MW2_mix(1,I)= MW2;
            end

        end

        for j = 1:n
            for i = 1:NPI+2
                D_k(j,i) = D(j,:)*Y_k(:,i);     % Diffusion is in terms of mass
                                                 % so use massfraction!!
                D2_k(j,i) = D(j,:)*Y2_k(:,i);   % Diffusion is in terms of mass
                                                 % so use massfraction!!
                Gamma_k(1,i) = Gamma*Y_k(:,i);
                rho(1,i)     = 1;
                rho_real(1,i) = rho_s*Y_k(:,i);
                rho2_real(1,i) = rho_s*Y2_k(:,i);
            end
        end

        f_old = Y_k;
        f2_old = Y2_k;
        rho_old = rho_real;
        rho2_old = rho2_real;

    end
```

8.2.3 Param_init()

With the function *param_init()* another set of parameters is initialized. This function requires the number of grid cells and inlet velocity, *u_in*, as input and returns:

- *u*, the initial velocity at the retenate side.
- *u2*, the initial velocity at the permeate side.
- *relax_f*, relaxation factor for the computation of the new mass fraction of the species.
- *Dt*, time step size

```
% make a vector with initial values for all non-specie dependent parameters
[u, u2, relax_f, Dt] = param_init(NPI, u_in);
```

Below the script is given of the function *param_init*.

```
function [u, u2, relax_f, Dt] = param_init(NPI, u_in);
    Dt = 0.5;
    for I=1:NPI+2
        i = I;
        u(i)    = u_in;          % Velocity in x-direction
        u2(i)   = 0;
    end

    relax_f = 1;               % relaxation for temperature
end
```

9 Grid generation

A 1D staggered grid is created with the function *grid_gen()*. A staggered grid is a setting in which the discretised variables are not defined at the same position. Typically, the velocity is defined at cell faces and all other variables are defined at cell centres. The main benefit of the staggered grid is that it does not result in the decoupling of pressure and velocity, leading to the checkerboard problem. In Figure 9.1 a picture of a staggered grid is shown. Note, two different notations are used in this figure to describe the cell faces and centres. The variables I, referring to cell centres, and i, referring to cell faces, with i = 0,1,2,...,N+2 and I = 0,1,2,...,N+2 is more generic. The use of W, P, E and w, e, p, is more convenient to describe the discretised equations.

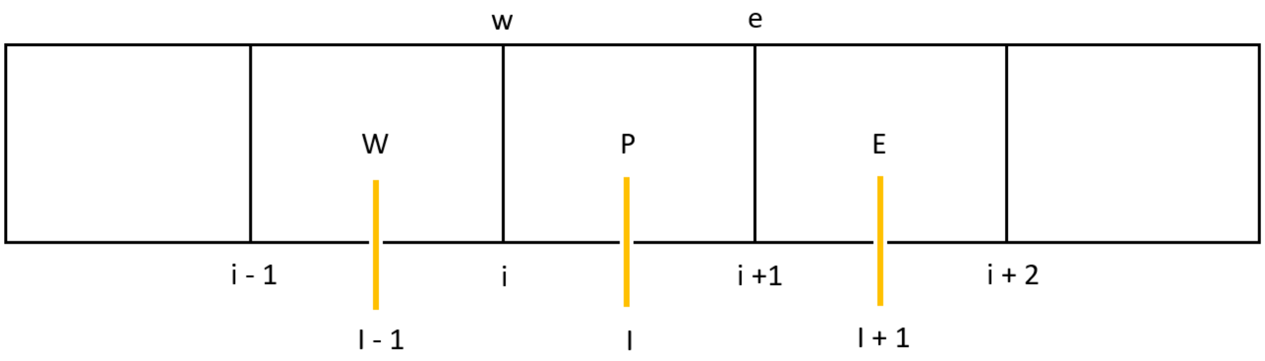


Figure 9.1: *Discretised staggered grid.*

The use of a staggered grid is not necessary for this particular case since pressure and velocity don't change. However, the staggered grid was implemented at an early stage of the project and for future improvements it is more convenient to use the staggered grid. At this stage the use of a staggered grid has no influence on the results.

The function *grid_gen()* requires the number of grid cells and the length of the domain, *XMAX*, as input. It returns:

- *Dx*, the distance between two grid cells/ faces.
- *x* the positions where all variables except velocity are defined.
- *x_u*, the positions where velocity is defined.

```
    %% grid generation
    [Dx, x, x_u] = grid_gen(NPI,XMAX);      % create staggered grid
```

The script of the function *grid_gen()* is shown below. If the user would like to decrease the grid spacing, *Dx*, this should be done by increasing the parameter *NPI*, which is declared at the beginning of the main code. Also, if the grid should be longer, this should be modified by increasing the value for *XMAX* which is also declared at the beginning of the main code.

```
function [Dx, x, x_u] = grid_gen(NPI, XMAX)
    % Length of the element
    Dx = XMAX/NPI;
    x(1) = 0;
    x(2) = 0.5*Dx;

    % Length variable for the scalar points in the x direction
    for I=3:NPI+1
        x(I) = x(I-1)+Dx;
    end
    x(NPI+2) = x(NPI+1) + 0.5*Dx;

    % Length variable for the velocity components u[i] in the x direction
    % Used to make staggered grid
    x_u(1) = 0;
    x_u(2) = 0;
    for i=3:NPI+2
        x_u(i) = x_u(i-1)+Dx;
    end
end
```

References

[1] F. Chourou, "Optimization of combustion strategies for the Argon Power Cycle internal combustion engine," no. December, 2017.

[2] H. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics*, vol. M. 2007.