CFD study of the intra and inter particles transport phenomena in a fixed-bed reactor

by
M. Alexandre Troupel

A Dissertation
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
In Chemical Engineering

_____

May 2009

Approved:

_____

Prof. A.G. Dixon, Advisor

_____

Prof. D. DiBiasio, Departement head

Summary

Actual models for fixed-bed reactor modeling make this assumption that temperature is uniform, or at least symmetric, within the catalytic pellets. However, if this holds true for large beds (tube-to-particle diameter ratio *N* greater than 10), it appears that for small *N* tubes (*N* = 3-10) that wall effects cannot be neglected anymore. A large temperature gradient appears in the near wall region. Hence for a particle at the wall a variation in temperature of up to 50°C was noticed.

This temperature change was investigated, and it has been noticed that the proximity to the wall, but also to a low velocity region could explain a maximum in temperature. Furthermore, species concentration discrepancies were also notice. An adiabatic run was made to show that these were not due to heated wall effects. Instead it appeared that these concentration variations are due to both their proximity to a low flow region and to a confined area. Hence incoming diffusion in these zones appeared to be lower than for the rest of the surface.

We also could notice a strong impact of the flow on the temperature patterns in the near wall regions. Hence in our case, it appeared that the 4 holes geometries allowed a better flow in front the particle at the flow, and therefore better transport phenomena. On the contrary, the full cylinder geometry tend to block the flow, consequently temperature on the wall particles were hotter than what they were with the 4 holes cylinder geometry.

A study of the diffusion within the catalytic particles was also conducted. Hence, the Maxwell-Stefan, the dusty gas and the binary friction models were implemented in Fluent. The goal here is to refine step by step the diffusion model used. First products and reactants molar fluxes were assumed to be proportional. The next step was to compute the actual molar fluxes; however this added one more parameter to converge; that is the diffusion coefficient. Finally the assumption of negligible pressure variation within the pellets was dropped. Unfortunately, the implementation into Fluent was not successful, and few possible reasons were given.

Acknowledgments

If there are two years that I know I will never forget, these would be the two years that I spent at WPI. It has been a very rewarding and exiting experience. And this would not have been possible if it wasn't for some persons.

Therefore, I would like to first thank my advisor, Professor Dixon, who has guided me through my studies. His support, moral but also financial have been of great value. And I particularly want to say how much I appreciated the opportunity he gave me to give a talk at the 2008 AIChE annual conference in Philadelphia. And since I know he can read French, I would like to say "un Grand Merci", for all this and much more.

I also like to take this opportunity to thank Professor Kazantzis who recommended me for a WPI summer scholarship. Thanks also to Mohsen who kept me company in the window-free lab. And finally I like to thank all the rest of WPI's Chemical Engineering Department who granted me a Teaching Assistantship. This is at the same time a very demanding and formative job, and I know I will be of value throughout my career.

Finally I am very grateful to my whole family. To my grandparents, Paul and Lisette, who were kind enough to come and see my several times, and to René and Nadou, who I know would have had also if they had been able.

My greatest gratitude goes to my parents, Robert and Dominique, who always pushed me further and farther. I know how much I owe them, and I am sooooo grateful for this. I am also very thankful to my brother Guillaume, and my little sister Chloé, who has send me so wonderful and lovely letters.

Table of Contents

List of Figures

List of Tables

I) Introduction

1) The Hydrogen production

If one considers the future of the energy economy, one will be likely to think of hydrogen. Indeed, very promising researches are going on in the fuel cells area, using hydrogen as an energy carrier.

However, one must not forget that hydrogen is already widely used in chemical industries. In the first place, we found the ammonia production, consuming about half the annual production. The second biggest hydrogen consumers are the refineries using 37% of the annual production. The remaining 8% is used to produce methanol (Hydrogen Today and Tomorrow).

Nevertheless, hydrogen presents several issues such as its lightness and its explosiveness. And therefore, it is often technically, and economically favorable to produce hydrogen in the same plant, where it will be consumed.

We know a lot of different ways of producing hydrogen, such as water hydrolysis or fermentation. However, the main process used nowadays is the methane to hydrogen conversion reaction, i.e. the Steam Reforming (MSR) reaction. The following table (Table 1) allows a rapid comparison with other means of producing hydrogen. One can see that MSR presents a high energy efficiency of 86%.

Category	Process	Efficiency (%) Energy
Hydrocarbon-based	Methane Steam Reforming (MSR)	86
	Coal gasification	59
Non-hydrocarbon-based	Current-technology water electrolysis	30
	Advanced-technology water electrolysis	49
	Thermochemical water decomposition	21
Integrated	MSR/current-technology water electrolysis	55
	MSR/advanced-technology water electrolysis	70
	MSR/thermochemical water decomposition	45

Table 1 - Hydrogen production processes comparison(Rosen & Scott, 1998)

MSR is a strongly endothermic reaction, and therefore conducted in a multi-tubular fixed bed. The present work focuses on understanding the mechanisms taking place in such fixed bed reactors. In order to do that, computational fluid dynamics (CFD) has been used to study both the influence of the pellet geometry on the surrounding flow field and of the species diffusion model.

2) Heat effects

Temperature is one of the key parameters an engineer wants to control in a reactor. However, some reactions such as MSR require more control over temperature than others. Hence two categories of fixed bed reactor are commonly seen in industry. To be able to compare them, one needs to introduce the dimensionless number *N*:

$$N = \frac{tube\ \ diameter}{particl\ \ diameter} \qquad eq.\ I.2\text{-}1$$

The first type of reactor is the large *N* reactor (*N* > 10, c.f. Figure 2). In this type of reactor, the overall cross-sectional temperature can be assumed to be constant and equal to the bulk temperature.

However, this convenient case supposes that the heat flux needed to entertain the reaction is not too large, i.e. the reaction is not highly endothermic. This type of reactor is used for cases such as ammonia production and methanol synthesis. Due to its large N, one can neglect the wall effects. Hence a common computational approach of such geometry will be to consider a unit cell (Figure 1) that would map the whole tube. One can easily see that following this approach leads to disregard any wall effect. This is often a good assumption.

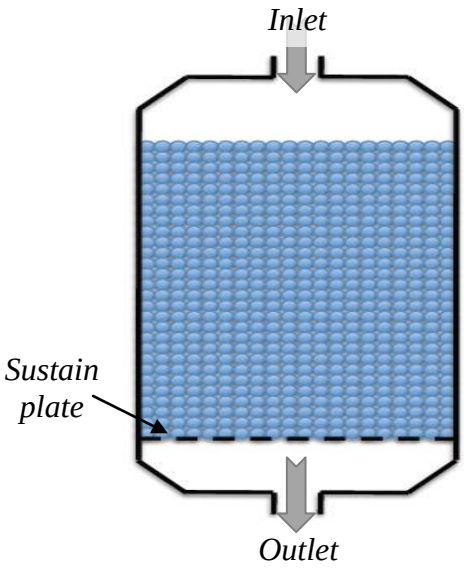

Figure 2 – Large *N* tube (*N* > 10)

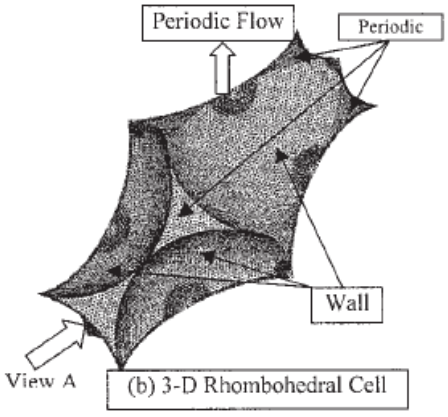

Figure 1 - Unit cell (Gunjal, Ranade, & Chaudhari, 2005)

The second type of reactor is the low *N* (*N* = 3-10) reactor. This type of reactors is used for highly endothermic or exothermic reaction. Indeed, a large heat flux must travel through the wall, which naturally entails a great temperature gradient between the wall and the center of the tube. Hence to lower this gradient, small tube diameter is needed.

However, to ensure a proper flowrate, several of these tubes must be put in parallel (Figure 3). These multi-tubular reactors are used in steam reforming reactions as well as ethylene oxidation and maleic anhydride formation.

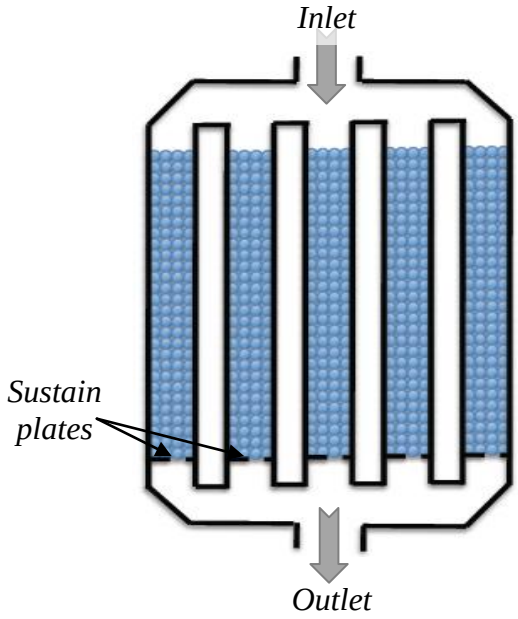

Figure 3 - Small *N* multitubular reactor (*N* = 3-10)

The following picture (Figure 4**Erreur ! Source du renvoi introuvable.**a) shows a steam reformer. It is composed of hundreds of catalytic tubes arranged in rows. Since the MSR reaction is a strongly endothermic reaction, an important heat flux is needed. This is ensured by several burners placed between each row and fueled by natural gas.

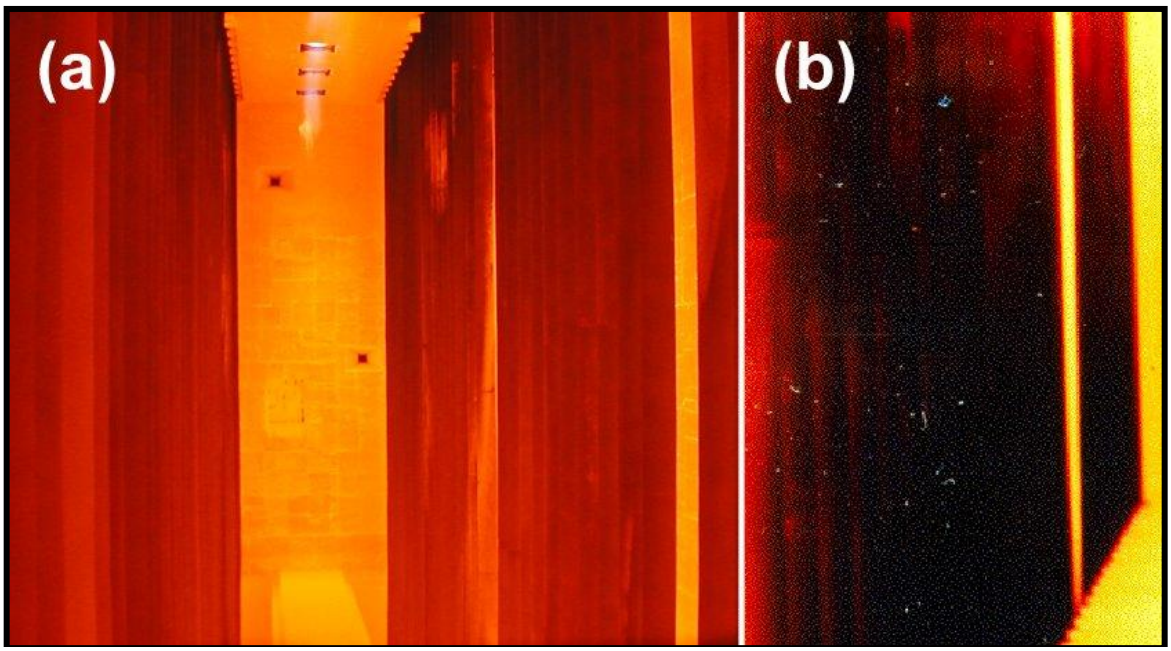

Figure 4 – Top-fired methane Steam reformer (Dixon, Nijemeisland, & Stitt, 2006)

We can also see on that picture that parts of the tubes are lighter than others; this is the result of temperature variations within the tube and even within the particles as it will be discussed in chapter III. This stands out on Figure 4b, where a whole tube has turned out yellow; hence this tube has undergone a high temperature increase that has weaken its wall, and that will eventually break. Indeed an increase of 20°C of the wall temperature will cut the tube lifetime by half (Dixon, Nijemeisland, & Stitt, Packed Tubular Reactor Modeling and Catalyst Design using Computational Fluid Dynamics, 2006).

Temperature variations have several origins, the first being the flow. Indeed as we will see later (chapter III), flow patterns have a strong influence on energy and species transport. Hence if the flow is not well distributed, low convection zones will appear where consequently heat transfers quality will be bad. Besides reactions rates will also not be uniform everywhere creating some species discrepancies. Resulting of all this, particles will undergo thermal and mechanical stresses, and eventually break.

Two reasons explain why flow could not be uniform. First, a poor particles packing creates some favorable path in the reactor, leaving some zone with hardly any flow. Second is dust. Indeed, the mechanical and thermal constraints we spoke of in the previous paragraph can result in some particles crushing and dust creation. This dust will then deposit and eventually block some paths.

The last reason of the temperatures non homogeneity is carbon deposition on the particles' surfaces. This is known as the coke formation, and is due to the methane decomposition. This carbon formation makes it more difficult for reactant to access the surface of the catalyst, and therefore reactions rates decrease and eventually become null. This process is known as the particle deactivation process. And yet, the methane steam reforming reaction is strongly endothermic. Therefore a strong heat fluxes is brought to the reactor to sustain the reaction. This heat is then drained by the reactions. However, if the rates of reaction decrease, part of that flux will no longer be drained and the overall temperature of the near particle region will increase and so will the coke rate formation in the particle neighborhood... This vicious circle will eventually lead to weaken the tube wall and thus to the tube breaking.

Knowing that, it becomes clear that well understanding the mechanisms that bring these hotspots is a key step to improve reformers viability. And this is the goal of this present work.

3) Current models

a. Model development

Nowadays, if one wants to model a fixed-bed reactor, one will very likely have to use a homogenous model. This model assumes that solid and liquid phase are lumped into a same homogenous phase. Hence, each point of the reactor will be associated with two sets of equations: one for the fluid and one for the solid.

To write these two sets of equations, one often starts writing the model for a single particle and in a single direction ξ. The following two equations correspond to a species (equation I.1-1) and an energy (equation I.1-2) balance for the solid:

$$D^e \left(\frac{\partial^2 C_s}{\partial \xi^2} + \frac{2}{\xi}\frac{\partial C_s}{\partial \xi}\right) - \rho_s r_A(C_s, T_s) = 0 \qquad \textit{eq. I.3-1}$$

$$k^e \left(\frac{\partial^2 T_s}{\partial \xi^2} + \frac{2}{\xi}\frac{\partial T_s}{\partial \xi}\right) - \rho_s(-\Delta H)\, r_A(C_s, T_s) = 0 \qquad \textit{eq. I.3-2}$$

with the following set of boundary conditions:

$$k_g\left(C_s^{surface} - C\right) = -D^e \frac{\partial C_s}{\partial \xi} \qquad \textit{eq. I.3-3}$$

$$h_f\left(T_s^{surface} - T\right) = -k^e \frac{\partial T_s}{\partial \xi} \qquad \textit{eq. I.3-4}$$

where T_s and C_s are respectively the surface temperature and concentrations; and T and C respectively the fluid temperature and concentrations.

In order to discuss further this model, we must introduce the effectiveness factor. At a given point of a catalytic particle, the effectiveness factor is equal to the ratio of the reaction rate over the surface reaction rate (equation I.3-5):

$$\eta = \frac{r}{r_{surface}} \qquad \textit{eq. I.3-5}$$

η goes from 0 to 1. For an average effectiveness factor close to 1, the reaction rate is homogenous in the catalyst; therefore reaction is the limiting step. On the contrary for average effectiveness factor close to 0, reaction happens essentially only near the surface; the case is therefore diffusion limited.

One can combine equations I.3-1 and I.3-2 with equations I.3-3 and I.3-4 using the effectiveness factor as following:

$$k_g\, a_v\left(C - C_s^{surface}\right) = \eta \rho_s r_A\left(C_s^{surface}, T_s^{surface}\right) \qquad \textit{eq. I.3-6}$$

$$h_f\, a_v\left(T_s^{surface} - T\right) = \eta(-\Delta H)\rho_s r_A\left(C_s^{surface}, T_s^{surface}\right) \qquad \textit{eq. I.3-7}$$

Here, in order to simplify the equations, two assumptions are made. The first one is to assume that the species transfers outside the particles are much faster than within the particles, and the external concentration gradients are neglected. And the second is to

assume that heat transfers in the particle are much quicker that in the fluid, and therefore the temperature gradient within the solid are neglected.

These equations can be extended to a two dimensional fixed-bed. Hence for the fluid, one obtains

$$D^e\left(\frac{\partial^2 C}{\partial r^2}+\frac{1}{r}\frac{\partial C}{\partial r}\right)-u_s\frac{\partial C}{\partial z}=k_g a_v\left(C-C_s^{surface}\right) \qquad \textit{eq. I.3-8}$$

$$k^e\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}\right)-u_s\rho_s C_p\frac{\partial T}{\partial z}=h_f a_v\left(T-TC_s^{surface}\right) \qquad \textit{eq. I.3-9}$$

And for the solid the equations are:

$$k_g\ a_v\left(C-C_s^{surface}\right)=\eta\rho_s r_A\left(C_s^{surface},T_s^{surface}\right) \qquad \textit{eq. I.3-10}$$

$$h_f\ a_v\left(T_s^{surface}-T\right)=\eta(-\Delta H)\rho_s r_A\left(C_s^{surface},T_s^{surface}\right)+k_s^e\left(\frac{\partial^2 T_s}{\partial r^2}+\frac{1}{r}\frac{\partial T_s}{\partial r}\right) \qquad \textit{eq. I.3-11}$$

And the corresponding boundary conditions for the reactor wall:

$$k_{wf}\left(T_w-T\right)=k_f^e\frac{\partial T}{\partial r} \qquad \textit{eq. I.3-12}$$

$$k_{ws}\left(T_w-T_s\right)=k_s^e\frac{\partial T_s}{\partial r} \qquad \textit{eq. I.3-13}$$

where T_w it the reactor wall temperature, and h_{wf} and h_{ws} the respective heat transfer coefficients for the fluid and the solid.

Note that in equation I.3-13, a temperature radial diffusion has been added, in order to increase the overall accuracy.

b. Discussion about the model

Several variations of this model exist, depending on the degree of complexity one wants to gibe to the model. This is shown by Amundson (1970), where he gradually adds axial then radial transports phenomenon. However, for practical reasons, it is always assumed that temperature is at least symmetric or even constant. This is made to simplify computations. However, authors a fully aware that, in reality, distribution around a particle is not uniform.

Moreover, in this model, the particle interactions with the surrounded fluid are model after the single particle model. However, as it will be shown latter in chapter III, the surrounding particles have significant affect on the flow patterns which then has significant impact on the diffusion process. According to Schwedock et al. (1989), considering the reaction/diffusion process from point of view of a single particle and not in the packed bed context is "an important simplification".

4) The reaction

The reaction considered here is the Methane Steam reforming reaction (MSR). This reaction has been modeled by the following three reactions, among which is the Water Gas Shift reaction:

$$\mathbf{CH_4 + H_2O = CO + 3\,H_2} \qquad -\Delta H_{r1} = -206.10\ \text{kJ.mol}^{-1}$$

$$\mathbf{CO + H_2O = CO_2 + H_2} \qquad -\Delta H_{r2} = 41.15\ \text{kJ.mol}^{-1}$$

$$\mathbf{CH_4 + 2\,H_2O = CO_2 + 4\,H_2} \qquad -\Delta H_{r3} = -165.00\ \text{kJ.mol}^{-1}$$

This reaction is done over a nickel/alumina catalyst (Ni/Al_2O_3) usually promoted with potassium in order to limit the coke formation. The inlet conditions used for this study are a pressure of 21.59 bar and a temperature of 824.15 K.

Notice that this equations neglect coke formation. Although we know that coke formation does take place in actuality, we wanted at that point to keep the kinetics relatively simple, and therefore suppose that no coke was formed.

The following kinetics (Hou & Hughes, 2001) are issued from experiments. In order to limit the effect of deactivation due to coke formation, a flux of hydrogen was added to the inlet stream. Consequently, although a decrease in the reaction rate was observer in the first 200 min, the system eventually reached a quasi steady state. The following expressions were found:

$$r_1 = k_1 \frac{P_{CH_4} {P_{H_2O}}^{0.5}}{{P_{H_2}}^{1.25}} \left(1 - \frac{P_{CO} {P_{H_2}}^3}{K_{p1} P_{CH_4} P_{H_2O}}\right) / DEN^2 \qquad \textit{eq. I.4-1}$$

$$r_2 = k_2 \frac{P_{CO} {P_{H_2O}}^{0.5}}{{P_{H_2}}^{0.5}} \left(1 - \frac{P_{CO_2} P_{H_2}}{K_{p2} P_{CO} P_{H_2O}}\right) / DEN^2 \qquad \textit{eq. I.4-2}$$

$$r_3 = k_1 \frac{P_{CH_4} P_{H_2O}}{{P_{H_2}}^{1.75}} \left(1 - \frac{P_{CO_2} {P_{H_2}}^4}{K_{p3} P_{CH_4} {P_{H_2O}}^2}\right) / DEN^2 \qquad \textit{eq. I.4-3}$$

where

$$DEN = 1 + K_{CO} P_{CO} + K_H {P_H}^{0.5} + K_{H_2O} \frac{P_{H_2O}}{P_{H_2}} \qquad \textit{eq. I.4-4}$$

Interesting to notice is that the expressions found by Hou et al. compare well with the ones found by Xu et al. earlier (Xu & Froment, 1989).

The numerical values for pre-exponential factors and activation energy are listed in appendix F.

II) The CFD approach

Using computers to model the behavior of a fixed bed reactor is the smart way to get "experimental" data without intruding in the system. Moreover, Computational Fluid Dynamics (CFD) is a very powerful tool getting access to virtually any parameters throughout the system.

However, one needs to carefully set the system in order to get good results. The first thing to take care of is the size of the computations. And this mainly depends on the size of the geometry, i.e. the size of the mesh. This latest must be a compromise between a coarse mesh that will enable a small computation time and a fine mesh that will give more accurate results.

1) Mesh generation

a. Symmetries and periodicity

The first step in the meshing process is to reduce the size of the actual geometry one wants to use, by taking advantage of its natural symmetries and periodicities. For our fixed bed, it is thus interesting to only model about two layers of pellets and assume that the flow is periodic.

Figure 5b shows that use of periodicity. The middle layer is comprised of full pellets whereas the top and bottom layers are compromised of partial pellets. Notice that the top and bottom layers are really connected one to the other, meaning that if one would overlay two of that same geometry, the pellets will connect perfectly.

This geometry has been obtained by looking at an actual packing of cylinders and by reproducing the main pattern that could be seen. The top layer has been obtained by direct transposition of the bottom layer to obtain that perfect periodicity. Finally the void fraction of the geometry has been compared to one of an actual packing to ensure a good representation of reality.

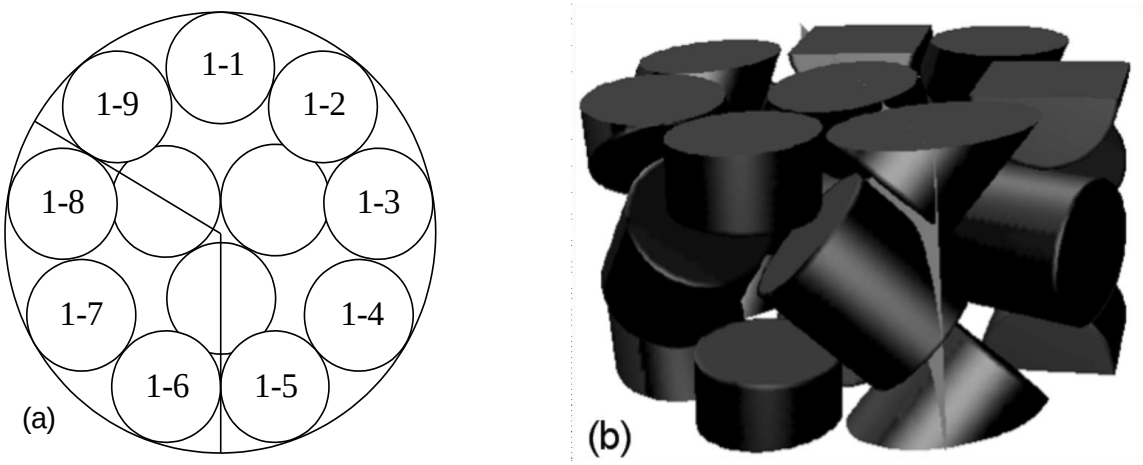

Figure 5 – Geometry shrinking of a fixed bed reactor (Dixon, Nijemeisland, & Stitt, 2006)

The second step in our process is to only take one third of the previous geometry as shown on Figure 5a. We then assume that the two cutting planes are symmetrical planes for the system. We thus obtain what we call the Wall Segment (WS).

Though in fact these two planes are not really symmetrical planes for the system, it has been shown(Taskin, 2007) that this assumption has little quantitative influence on both the axial velocity and temperature. However, it has been noticed that pressure drop was significantly influenced by this assumption, due to the "squeezing constraint" imposed by the symmetrical walls.

Therefore, Taskin divided the geometry in three and focused his study on the central 60° as shown on the following picture (Figure 6). He compared averaged values for flow and temperature between this 60° zone and the full bed (360°). The results he obtained were closer to each other. Therefore, the WS geometry is a good compromise between accuracy and computation time.

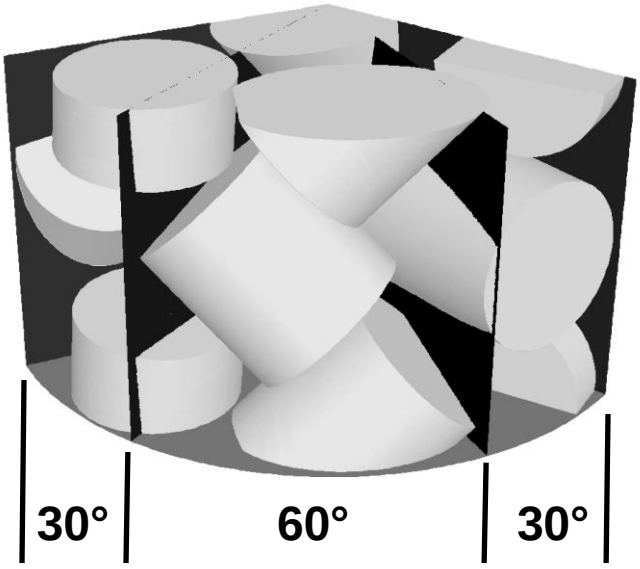

Figure 6 – WS study by zones (Taskin, 2007)

However, the WS geometry brings some meshing issues. Indeed the symmetric planes cut through the particles, leading sometimes to some squeezed zones, where the meshing tends to be problematic. Thus skewed cells appear in these zones. Nevertheless, both the size of the mesh and the boundary layers help improving the mesh quality as shown thereafter.

b. Size mesh and boundary layers

Thanks to the previous step, we have been able to reduce the size of the geometry we want to mesh. This allows us to go to small size in meshing and therefore, improve on the accuracy of the results. The size of the mesh is goes from 0.02 to 0.03 inch (0.0508 – 0.0762 cm) for a total number of about 4.5 million cells (including boundary layers).

We adopted an unstructured mesh, meaning that each node can be connected to any number of other nodes. Each node will therefore contain information letting the solver know to which node it is connected. The reason we chose unstructured mesh is that the meshing is much easier with an unstructured mesh, and though structured mesh are known to give better results, the complexity of our geometry simply doesn't allow us to use structured mesh.

As we will see later, in certain zones the gradients are much higher than for the rest of the geometry. It is thus very important to have a refined meshing in these zones, and boundary layers help to fulfill this very nicely for two reasons.

The first is that boundary layers allow a progressive change in the mesh size in a defined region. This progressing change in meshing helps to keep a good solving stability and refines the mesh in the desired zones. Hence, the gradients will be caught with more accuracy.

Second, it is important to keep in mind that we want to keep computation time reasonable, and thus not have too many cells. Hence boundary layers only refine the meshing in the desired zone, and leave the remaining ones unchanged. Therefore, the zone where small gradients occur will not be refined, and thus not contribute a lot in the overall number of cells.

As one can suspect, and as shown by early results, large gradients appear at the interface between particles and the fluid, and at the interface between the reactor wall and the fluid. Hence boundary layers are added in these zones. The following picture shows boundary layers applied to the fluid and solid zones:

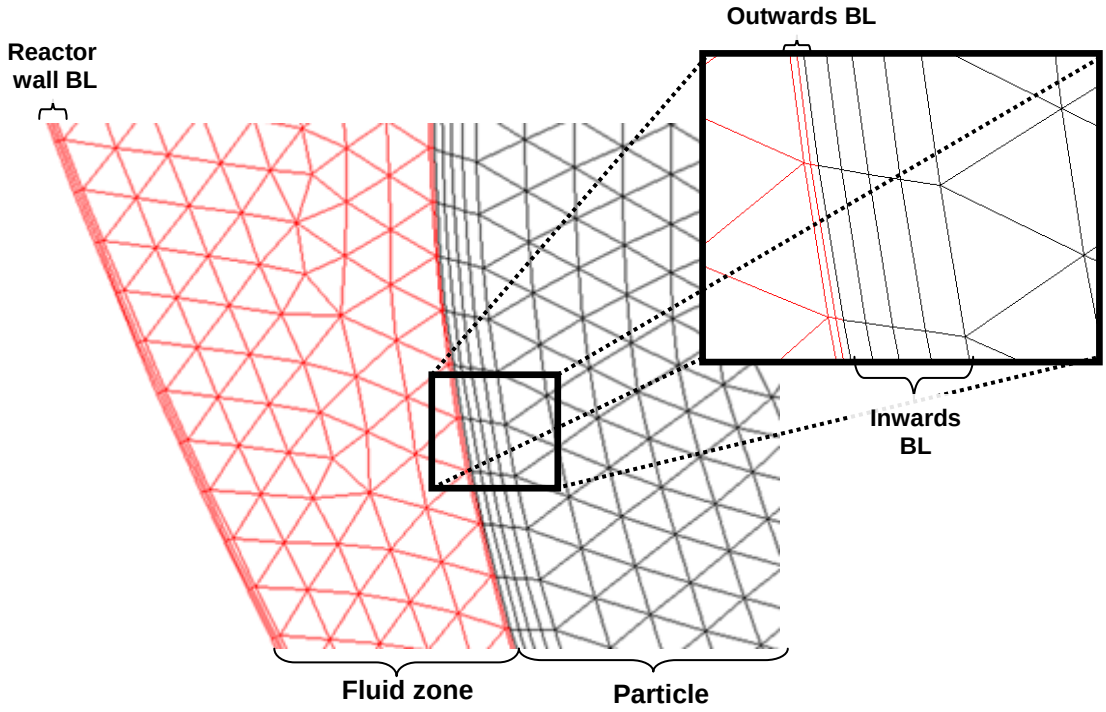

Figure 7 - Meshing pattern

Notice that the solid particles have been mesh also. This is because reaction and diffusion are taking place within these particles also. Therefore, both solid and fluid zones are mesh, and boundary layers are added. However, in more complex geometries, meshing was not possible if boundary layers were applied to all particles, hence some boundary layers have been removed (c.f. appendix A).

c. Contact points

An important issue comes when meshing fixed bed, which is contact points: when two particles touch each other, it creates a narrow zone in the fluid, where the meshing tends to be problematic. To avoid that problem, several options have been studied (Dixon, Nijemeisland, & Stitt, 2006).

The first one consists in increasing slightly the size of the particles, on the order of 1%. This will bring particles to overlap(Guardo, Coussirat, Recasens, Larrayoz, & Escaler, 2006). The second is to reduce the size of the particles. Hence, there will be no more contact between particles. The last option that has been chosen by our group for cylinders is to dispose particles such that they do not touch each others. With this option, the actual size of the pellet is conserved. However the downside of these two last methods is that the voidage of the bed is slightly increased.

2) Turbulence model

a. The k-ε model

Fixed-bed reactors are run under turbulent regime in order to improve transportation mechanisms. There one needs to specify to the solver which turbulent model to use. Our group initially started using the k-ε model.

For turbulent regime, the only "easy" values to access are the averaged value. Hence it is more convenient to express these parameters values vis-à-vis their average values. Following that, the k-ε model is based on the averaged Navier-Stokes equation. That is to say, each variable X is expressed under the following form:

$$X = X_0 + \partial X \qquad \textit{eq. II.2-1}$$

were X_0 is the average value of X, and ∂X the variation of X to its average value.

Hence the Navier-Stokes equation

$$\frac{\partial u}{\partial x} + \nabla uu = \nu \nabla^2 u - \frac{\nabla P}{\rho} \qquad \textit{eq. II.2-2}$$

becomes

$$\frac{\partial u_0}{\partial x} + \nabla u_0 u_0 = \nu \nabla^2 u_0 - \frac{\nabla P_0}{\rho} + \nabla \langle \partial u \, \partial u \rangle \qquad \textit{eq. II.2-3}$$

A new term appears which is $\langle \partial u \, \partial u \rangle$. It is called the Reynold's Stress Tensor and accounts for the momentum transfer between two particles moving at different speeds. This term is problematic since we are left with one more parameter than we have equations to solve them. Therefore one must somehow relate that term to the other parameters in order to reduce the number of parameters to the same number of equations. And this is the purpose of the k-ε model. However, one must keep in mind that this step is purely artificial, and there is no rigorous way to achieve this.

The k-ε model depends, at its name implies, on two parameters which are k and ε. The first parameter k is the turbulent kinetic energy. It hence relates to the energy transported by turbulence, but does not take into account the size of the turbulence. This is done by the second parameter ε. This last parameter is called the turbulence dissipation rate.

The expressions of these two parameters are as following:

$$k = \frac{\langle \partial u_x \, \partial u_x \rangle + \langle \partial u_y \, \partial u_y \rangle + \langle \partial u_z \, \partial u_z \rangle}{2} \qquad \textit{eq. II.2-4}$$

$$\varepsilon = \nu \left\langle \frac{\partial(\partial u_i)\, \partial(\partial u_i)}{\partial x_j \partial x_j} \right\rangle \qquad \textit{eq. II.2-5}$$

This enables us to rewrite equation *II.2-3* as

$$\frac{\partial u_0}{\partial x} + \nabla u_0 u_0 = (\nu + \nu_T)\nabla^2 u_0 - \frac{\nabla P_0}{\rho} \qquad \textit{eq. II.2-6}$$

where the kinematic eddy viscosity ν_T is given by

$$\nu_T = C_\mu \frac{k^2}{\varepsilon} \qquad \textit{eq. II.2-7}$$

It is then possible to write two separate balance equations (one for each parameter). They generally have the following form:

$$\frac{\partial k}{\partial t} + \nabla u_0 k = \nabla D_k \nabla k + production - dissipation \qquad \textit{eq. II.2-8}$$

$$\frac{\partial \varepsilon}{\partial t} + \nabla u_0 \varepsilon = \nabla D_\varepsilon \nabla \varepsilon + production - dissipation \qquad \textit{eq. II.2-9}$$

The k-ε model possesses several variations that differ on the actual expression of the right hand side of equations *II.2-8* and *II.2-9*.

For a no slip condition, one expects the turbulent energy to go to zero as one approaches the wall. However the k-ε model lacks to model this properly. Therefore, wall functions must be added to the k-ε model to account for wall effects on turbulence. These wall functions are semi-empirical based, and model the behavior of the fluid between the wall and the bulk flow.

However, for these wall functions to work properly, the distance y^+ between the wall and the bulk flow on which the wall function is used must be set properly. However, in our model, due to a complex geometry, this distance varies. And since one is able to specify only one y^+, the wall function will lead to some inaccuracies.

b. The k-ω model

The k-ω model is a variation of the k-ε model, in that that both equations *II.2-8* and *II.2-9* are still used. However, the second parameter is changed to ω which is

proportional to the ratio of k and ε. All the results that will be presented here are based on this k-ω model, which is defined by the following equations.

Equations *II.2-8* and *II.2-9* become:

$$\frac{\partial k}{\partial t} + \frac{\partial u_{0i} k}{\partial x_i} = \frac{\partial}{\partial \mathrm{x_j}}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial \mathrm{x_j}}\right] - \overline{u_{0i}u_{0j}}\frac{\partial u_{0j}}{\partial x_i} - Y_k \qquad \textit{eq. II.2-10}$$

$$\frac{\partial \omega}{\partial t} + \frac{\partial u_{0i} \omega}{\partial x_i} = \frac{\partial}{\partial \mathrm{x_j}}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial \omega}{\partial \mathrm{x_j}}\right] - \frac{1/9 + {Re_t}/{6}}{1 + {Re_t}/{6}}\frac{\overline{u_{0i}u_{0j}}}{\nu_t}\frac{\partial u_{0j}}{\partial x_i} - Y_\omega \qquad \textit{eq. II.2-11}$$

where

$$\nu_t = \frac{0.024 + Re_t/6}{1 + Re_t/6}\frac{k}{\omega} \qquad \textit{eq. II.2-12}$$

$$Re_t = \frac{k}{\nu\omega} \qquad \textit{eq. II.2-13}$$

And the two dissipation terms are given by:

$$Y_k = 0.09\ \frac{4/15 + (Re_t/8)^4}{1 + (Re_t/8)^4}\,(1 + 1.5\ F)\ f_*\ k\ \omega \qquad \textit{eq. II.2-14}$$

$$Y_\omega = 0.072\left(1 - 0.09\ \frac{1.5\ F}{0.072}\ \frac{4/15 + (Re_t/8)^4}{1 + (Re_t/8)^4}\right)\ f\ \omega^2 \qquad \textit{eq. II.2-15}$$

The various parameters used are defined as following:

$$F = \begin{cases} 0 & if\ \sqrt{\frac{2k}{\gamma RT}} \le 0.25 \\ \frac{2k}{\gamma RT} - 0.25^2 & if\ \sqrt{\frac{2k}{\gamma RT}} > 0.25 \end{cases} \qquad \textit{eq. II.2-16}$$

$$f_* = \begin{cases} 1 & if\ \chi_k \le 0 \\ \frac{1 + 640\ {\chi_k}^2}{1 + 400\ {\chi_k}^2} & if\ \chi_k > 0 \end{cases} \qquad \textit{eq. II.2-17}$$

$$f = \frac{1 + 70\ \chi_\omega}{1 + 80\ \chi_\omega} \qquad \textit{eq. II.2-18}$$

$$\chi_k = \frac{1}{\omega^3}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \qquad \textit{eq. II.2-19}$$

$$\chi_\omega = \left|\frac{1}{0.72\ \omega^3}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)\left(\frac{\partial u_j}{\partial x_k} - \frac{\partial u_k}{\partial x_j}\right)\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)\right| \qquad \textit{eq. II.2-20}$$

This model is an improvement over the k-ε model in that that it has shown good agreements with flow near a wall. Note that the actual model used is a variation of the k-ε model called the SST k-ε model. This model is a refinement of the basic k-ε model, given that it mimics the k-ε model far from the wall (i.e. for high Reynolds number), but reduces to a low Reynolds k-ε model near the wall. Hence the near wall behavior is well captured, and wall functions are no longer needed. The model parameters are as for the k-ε model based on experiments. However, a refined boundary layers meshing is still required in order to properly capture the near wall behavior.

3) The catalytic zone

Modeling the reaction through a CFD software can be a bit of an issue since these software have often been developed for mechanical engineers rather than chemical engineers. The Fluent software that we use for instance allows combustion, but very little is offered besides it. Therefore, one must use user defined function (UDF) that allows specifying source terms in the balance equations. This source terms can of course be either positive for products or negative for reactants.

At first our group used to model the catalytic particles as porous. A zero velocity was then imposed to these porous particles to account for the fact that there is no flow in these particles. Indeed concentration changes are only due to reaction and species diffusion in these particles, but the overall fluid is stagnant.

However, some inaccuracies were found in the results. This is due to the fact that Fluent computes the velocity at the interface between the porous zone and the fluid as the weighted average of the velocities of the nearest cells, both in the fluid and porous zones. And if the velocities are null on the particle side, there are not on the fluid side. This is causing the velocity at the particle-fluid interface to be different from zero, and therefore wrong.

That is why, our group then moved on using solid particle. Here Fluent does impose a zero velocity value at the interface. However, Fluent does not allow using species in solid. Therefore, the mass fractions are now stored as user defined scalars (UDS). Reaction is once more assured through UDF.

4) Diffusion

As we have seen, Fluent does not allow species in solid. It is then not surprising that species diffusion in not implemented either. However, it does allow UDS diffusion in all zones, and that it how we took care of it.

For a scalar Φ, diffusion in Fluent must be put in a Fickian-like form:

$$F_\phi = -\Gamma \nabla \phi$$

where Γ is the diffusion coefficient and F_Φ the flux of scalar Φ.

The diffusion coefficient Γ can be user defined through a UDF. It also can be set anisotropic. In chapter III we will discuss further the diffusion model used as it is an important part of the present work.

More details on the steps followed and on the parameters used are detailed in appendix B.

III) Results

1) Earlier results

As we saw in chapter I.3., when chemical engineers need to model a fixed-bed, they often make the assumption that temperature is uniform in the bed, and therefore neglect the wall effects. But for highly endothermic reactions such as methane steam reforming, an important heat flux is needed. Hence one can wonder if such an assumption is still worthy. Furthermore such reactions are conducted in small N tubes. Consequently, a non negligible number of particles stand near the wall.

The following picture (Figure 8) shows us a high gradient taking place in the near wall region. A 60-80˚C difference between the wall and the bulk flow can be noticed. And this is true both at inlet and middle tube conditions. Therefore the assumption of a symmetrical or constant temperature is not a valid assumption near the wall.

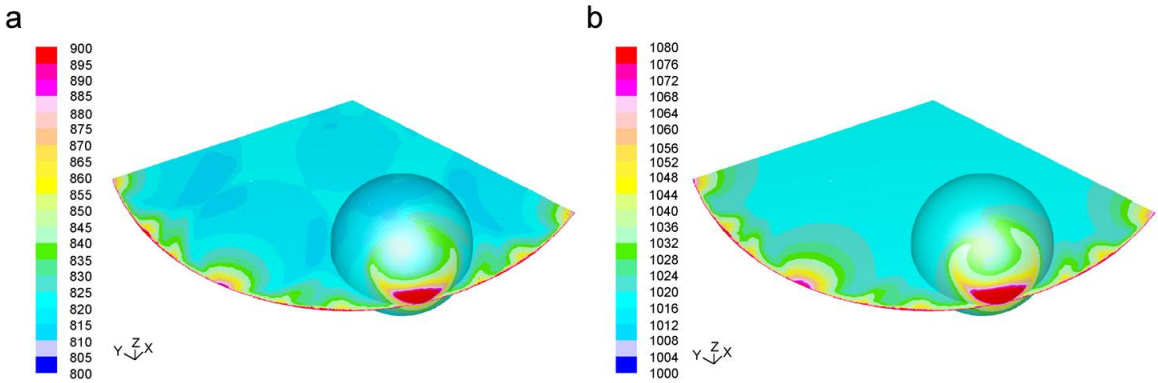

Figure 8 - Temperature profiles at inlet (a) and mid-tube (b) conditions (Dixon, Taskin, Stitt, & Nijemeisland, 2007)

Taskin (2007) then expanded this study to more complex geometries (Full and 4 holes cylinders), and was still able to see a high gradient due to wall heat effects.

The overall goal of this chapter is to compare the effects of the different geometries and draw conclusions on the mechanisms that bring asymmetrical behavior among particles.

2) Geometries used

All geometries used in this work are cylinder based. Their height and diameter are both 1 inch (2.54 cm).

The first geometry used is the full cylinder geometry (Figure 9a). It has initially been run by Taskin (2007), but with particles set as porous. At that time Fluent did not allow UDS in solid zones. Later Dixon (2008) ran that same geometry using the set of parameters we introduced earlier (c.f. chapter II and appendix B). The full geometry results shown here are issued from Dixon's run.

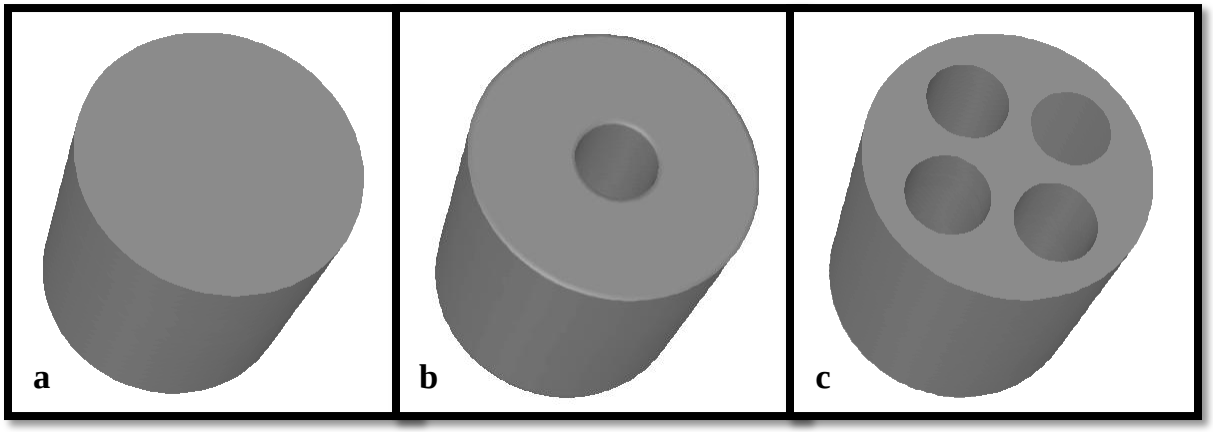

Figure 9 - Pellets geometries: (a) Full; (b) 1 hole and (c) 4 holes

The second geometry used is the one hole cylinder geometry (Figure 9b). Its coaxial center hole has a diameter of 0.2868 inch (0.7285 cm). No results had been published for this geometry yet.

The last geometry used is the four holes cylinder geometry (Figure 9c). It is composed of 4 holes of the same diameter, i.e. 0.2868 inch (0.7285 cm) and at a distance of 0.26 inch (0.6604 cm) of the center. This geometry had also been run previously by Taskin (2007), but once more particles were set as porous. Results presented here are issued from latest runs done with the set of parameters described earlier (c.f. chapter II).

Figure 10 - Real industrial pellets randomly packed form Johnson Mattey (Dixon, Nijemeisland, & Stitt, 2006)

The geometries considered here are simplified versions of the one used in the industry. Indeed, real particles often have domed top and bottom. This enables a more efficient and denser packing. Besides the usual height-to-diameter ratio is not 1 as used in our model but often greater than unity. Typically a value of 1.2 is adopted. Furthermore, commercial pellets also differ. Hence Johnson Matthey's posess grooves as presented in the following pictures (Figure 10). This reduces the overall porosity, and thus the pressure drop of the fixed-bed reactor. As for Haldor Topsoe, they opted for a seven holes cylinder. Finally the BASF group chose the four holes cylinders.

But for the sack of simplicity, and because our main goal is to understand the mechanisms that take place in packed bed reactors, we chose not to consider more complex geometries. Should one be interested in comparing the effectiveness of real particle shapes, the results given here will help predicting their behaviors.

3) Near wall particle surface study

The first element we compared was the surface of the particle highlighted in the following picture (Figure 11). The reason we choose this particle, is that it is within the 60˚ of validity for the WS (c.f. chapter II), and that it is the only particle to not be cut by either a symmetry plane or a periodic plane.

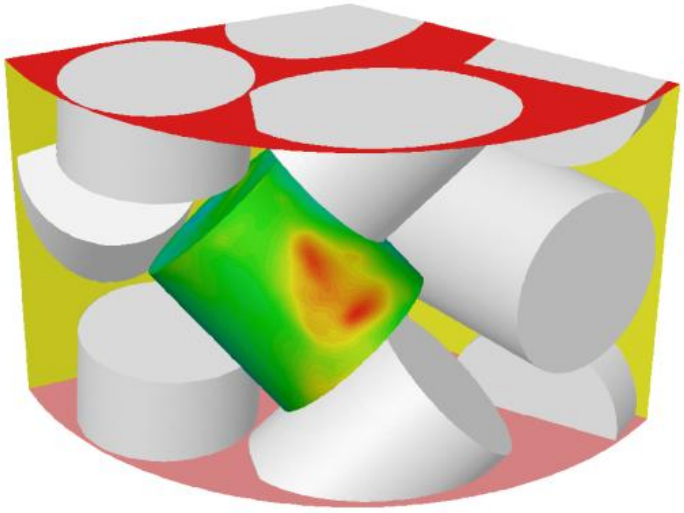

Figure 11 - WS geometry: particle test – surface

a. Temperature comparison

To obtain the following pictures, the lateral surface of the test particle has been unwrapped. To do this, we first performed geometrical transformations on the surface coordinates. As a result the point of origin and the axes direction were adapted to the test particle and any point on the surface could be identified thanks to only two parameters in

the cylindrical coordinates: the angle θ and the height z; the radius r remaining constant. Finally a surface coordinate s was created as the product of the angle and the radius ($s = \theta r$).

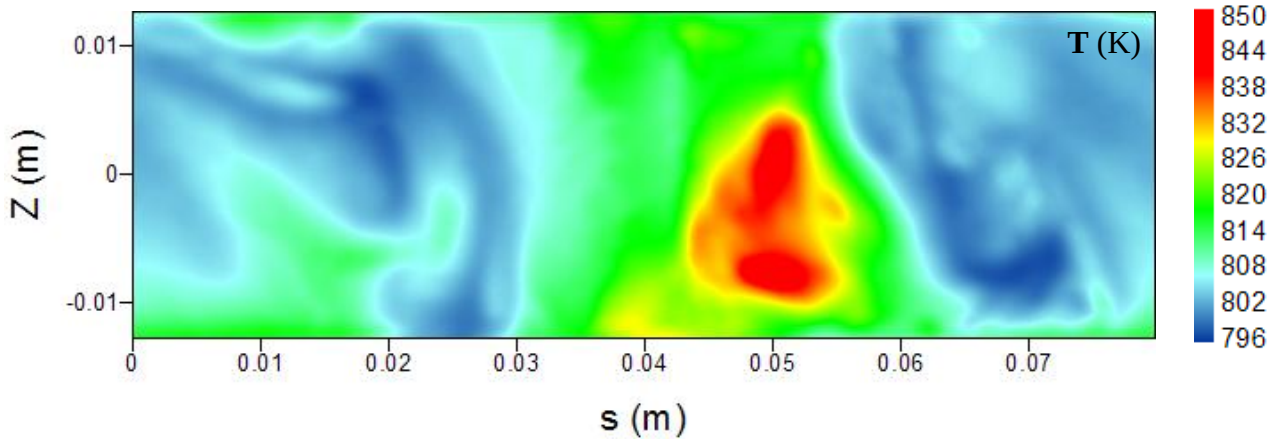

Figure 12 - Surface temperature profile for test particle - Full cylinder

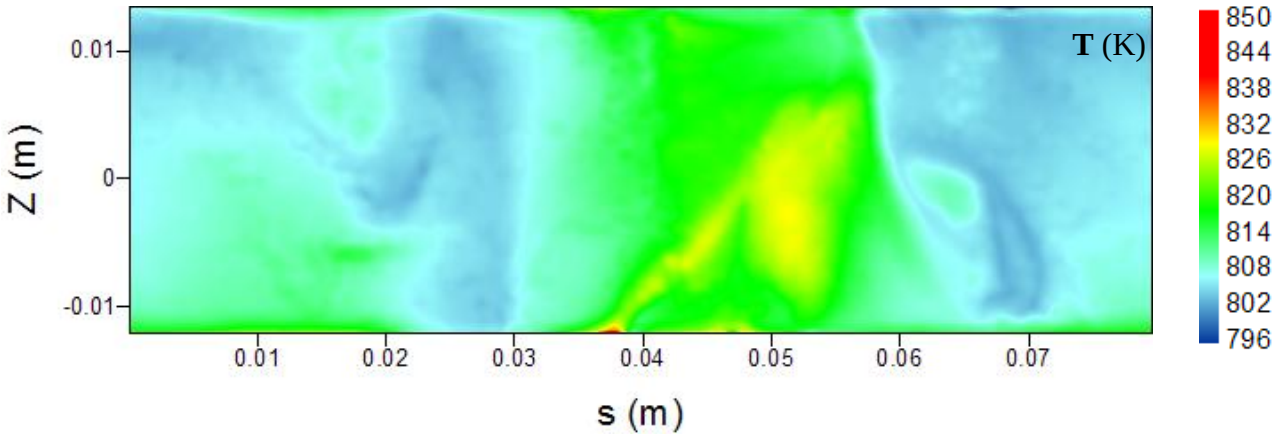

Figure 13 - Surface temperature profile for test particle - 1 hole cylinder

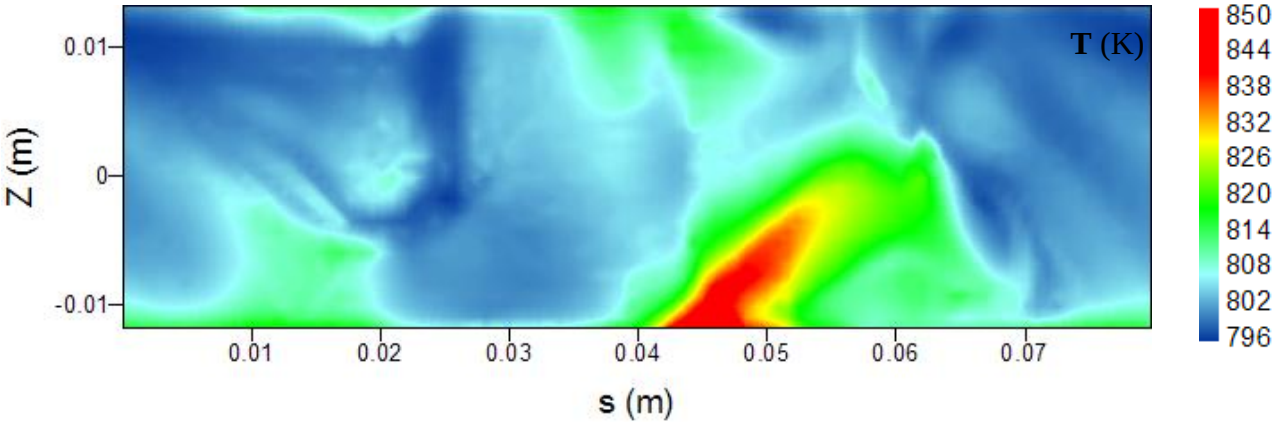

Figure 14 - Surface temperature profile for test particle - 4 holes cylinder

To simplify the comparison between the various geometries, the same color scale has been kept for the three previous figures. Notice that the overall mass flow rate has been kept constant for all geometries, and therefore the overall Reynolds number is also constant in all these simulations.

In agreement with the earlier results, a strong variation in temperature can be seen. Hence for the same particle, one can notice a difference in temperature of 50˚C on the wall.

The hottest zone (S ~ 0.05 m) correspond is the closest one to the reactor wall, whereas the coolest zone is the farthest from that cylinder wall. The can be qualitatively seen on Figure 11. It is as if the temperature increases the more one move towards the reactor wall, and decreases the further one moves from it.

Comparing the geometries, we can first notice a similarity in the temperature patterns. The hottest zone and the coldest are roughly located at the same places, not matter the number of holes in the particle. However, significant variations appear on the values of the extrema. Though the full and 4 holes temperature ranges are similar, it appears that the 1 hole temperature range is significantly reduced. Its temperature range is only of roughly 30˚C (20˚C lower than for the other geometries).

In order to understand this, one must focus on the flow around the particle. Figure 15, Figure 16 and Figure 17 on next page show a comparison between the flows. The left figure (figure a) shows the surface temperature profile, and the right figure (figure b) shows the pathlines of particles coming from the inlet (bottom surface). These pathlines are colored by velocity magnitude. Since these pathlines track particles coming from the inlet, a lack of pathlines on a particular area means no particle issued from the inlet passes through that area. In other word, a lack of pathlines will highlight stagnant fluid, and thus exchanges of both energy and species with the bulk flow is only due to diffusion.

In order to spot the hotspots on the test particle's surface, the same view has been kept for all the figures (Figure 15 through Figure 17). The first thing we can notice is that there is a strong correlation between the localization of these hotspots and the flow. That is to say, the hottest spots can be found where there is no flow. This is perfectly understandable, since in these zones, hardly any fresh flow arrives. And it is well know that heat transfer is highly more efficient when convection is at stake.

The relative coolness of the 1 hole geometry can also be understood thanks to flow. Indeed it appears that the dead zones around the test particles are more confined, and that the overall convection around these zones is better.

In order to understand the differences in the flow patterns, one needs to consider the particle which is just on top of the test particle. Figure 18 still considers the same pathlines, but includes the top particle. One can recognize the test particle at the center of the images.

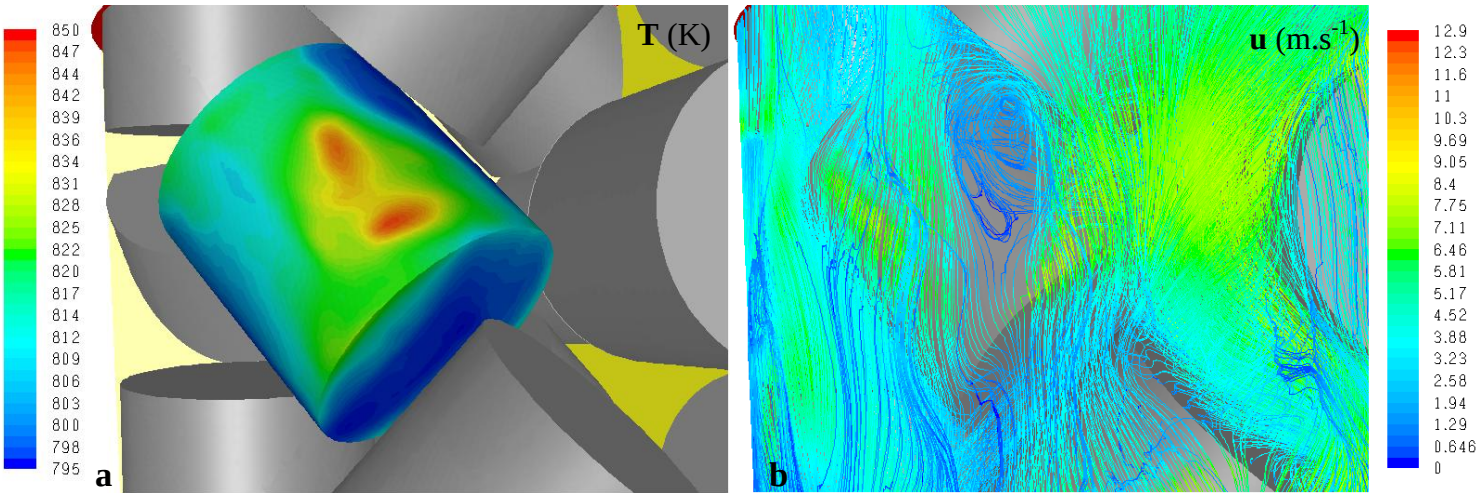

Figure 15 – Full cylinder - Particle test (a) Surface temperature profile; (b) Pathlines colored by velocity magnitude

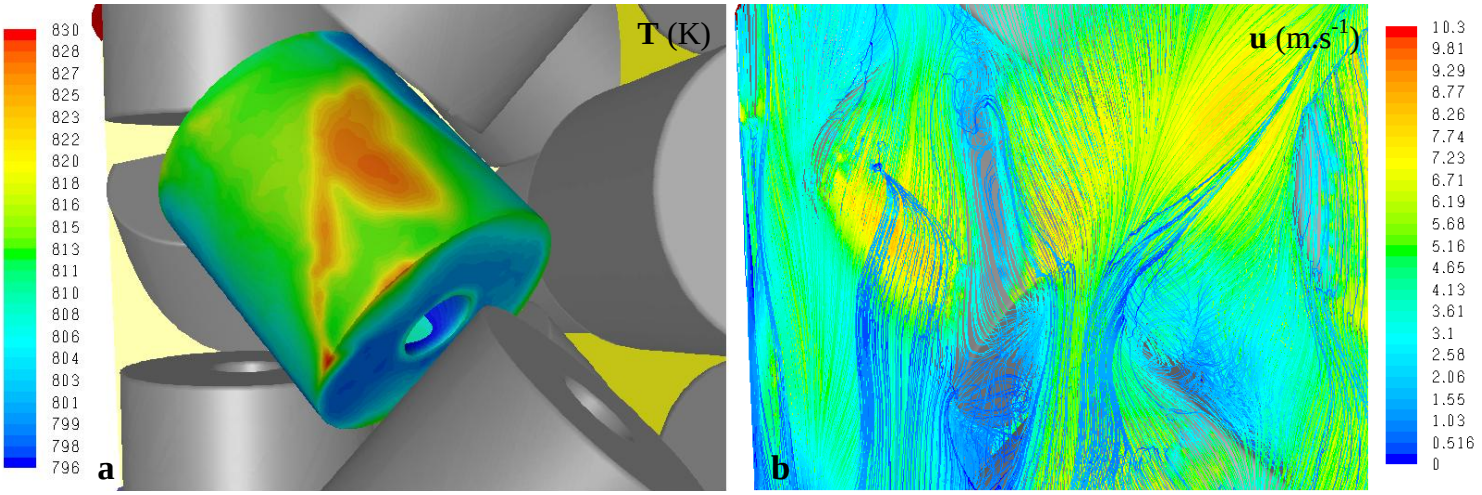

Figure 16 - 1 hole cylinder - Particle test (a) Surface temperature profile; (b) Pathlines colored by velocity magnitude

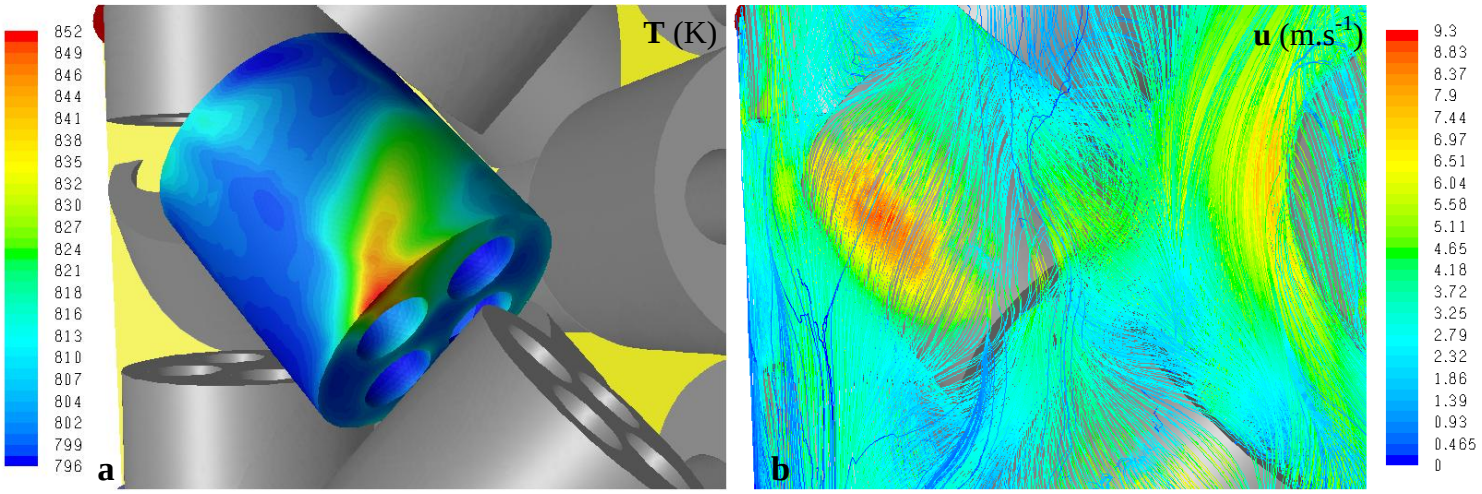

Figure 17 - 4 holes cylinder - Particle test (a) Surface temperature profile; (b) Pathlines colored by velocity magnitude

It appears that the particle on top of the test particle is blocking the flow coming from between the wall and the test particle. Hence for the full cylinders and to a smaller extent for the 1 hole cylinders geometries, swirls are formed, and velocities significantly decrease. Hence the top particle acts like a barrier, and thus the path coming from in front of the test particle becomes a non favorable path. However, if one adds a hole to the top particle, fluid can now flow more easily, and the swirls become less important.

The 4 holes geometry could be consider as the most favorable case in that that flow can now easily pass through one of the holes. It thus clearly appears that the fluid flow more rapidly between the top and the test particle that it used to for the other two geometries.

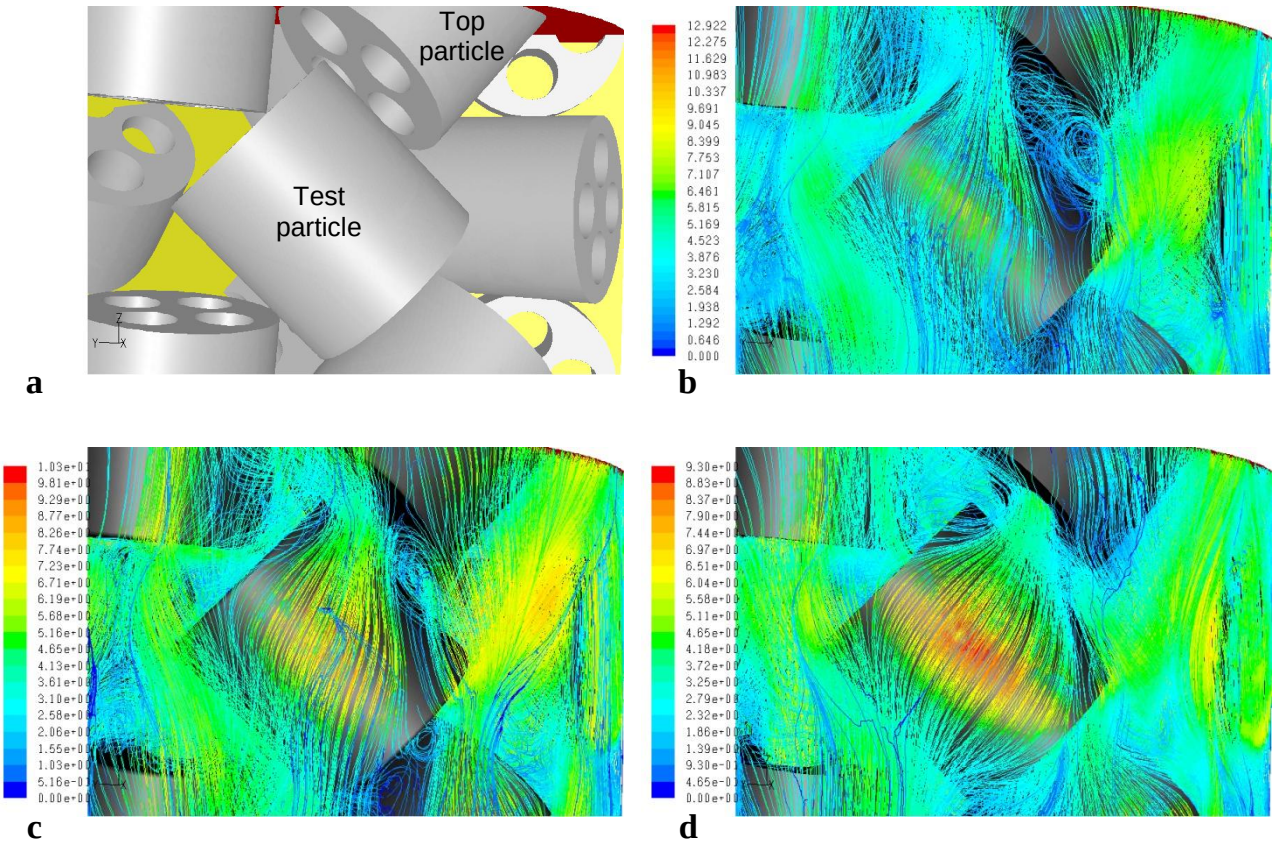

Figure 18 - Flow pattern study -
(a) 4 holes cylinder geometry only; (b) Pathlines for the full cylinder geometry;
(c) Pathlines for the 1 hole cylinder geometry; (d) Pathlines for the 4 holes cylinder geometry

b. Mass fraction comparison

As before, the following plots (Figure 19 to Figure 24) have been made from the test particle's surface. The following three focus on the methane mass fraction.

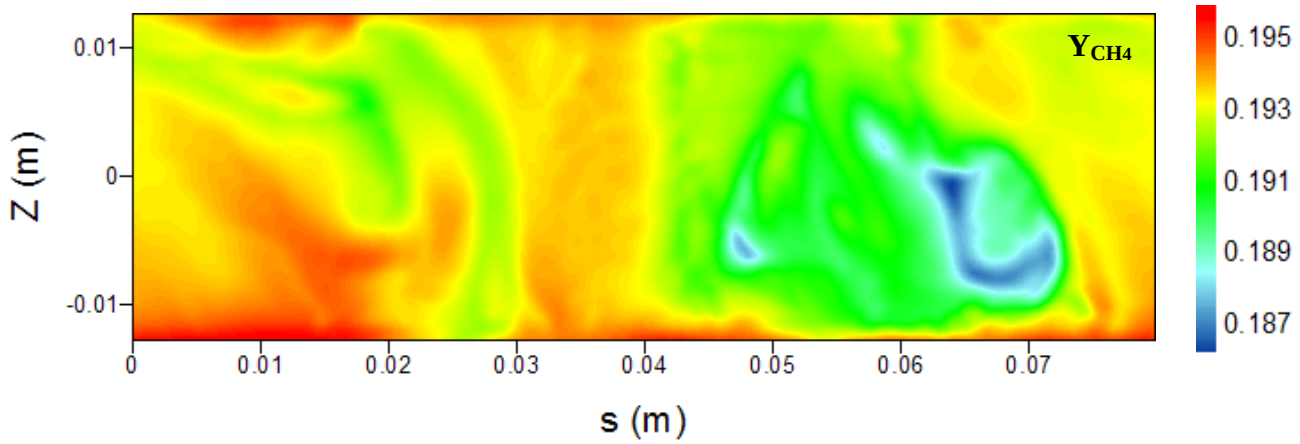

Figure 19 – Surface methane mass fraction profile for test particle - Full cylinder

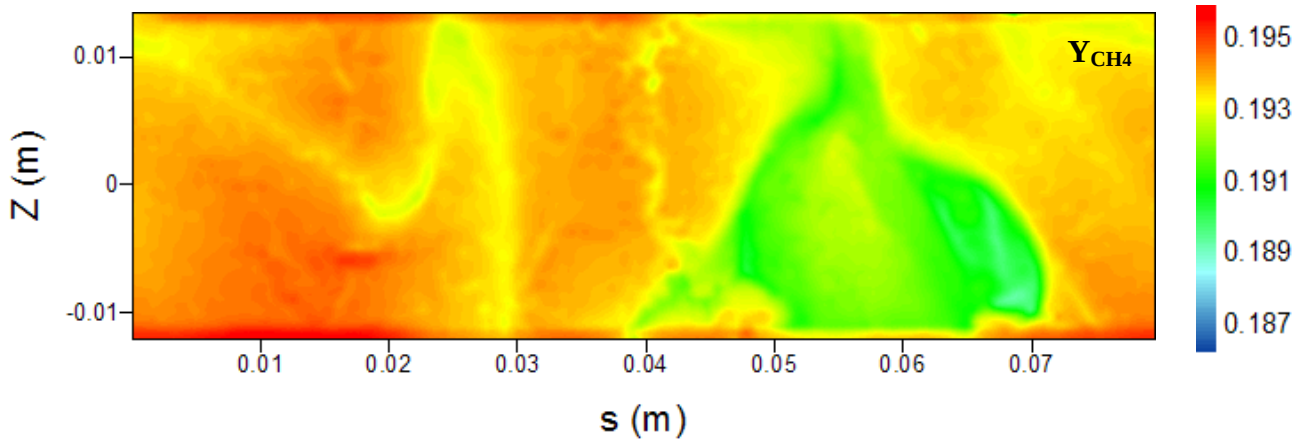

Figure 20 - Surface methane mass fraction profile for test particle - 1 hole cylinder

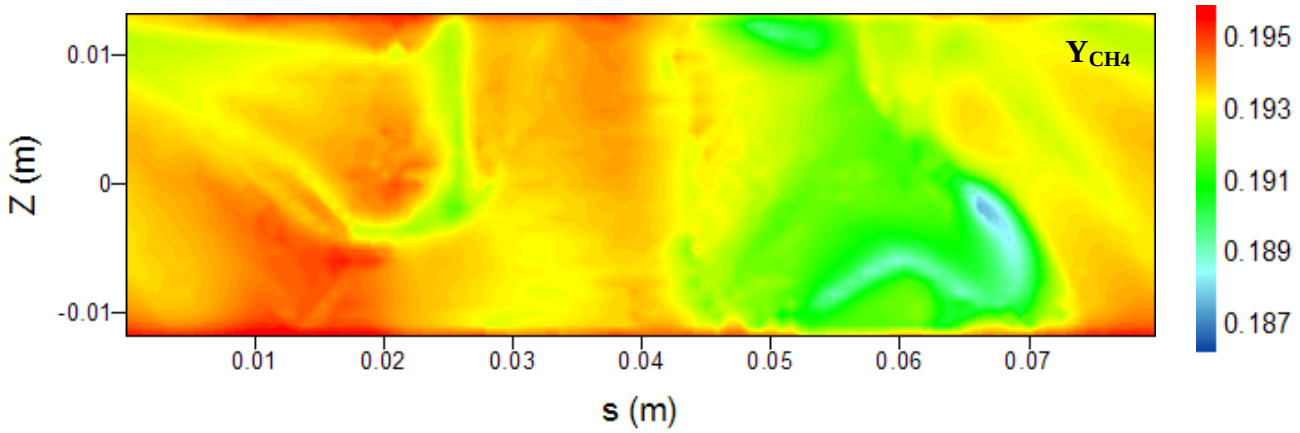

Figure 21 - Surface methane mass fraction profile for test particle - 4 holes cylinder

And the following three on the hydrogen mass fraction for the three different geometries used.

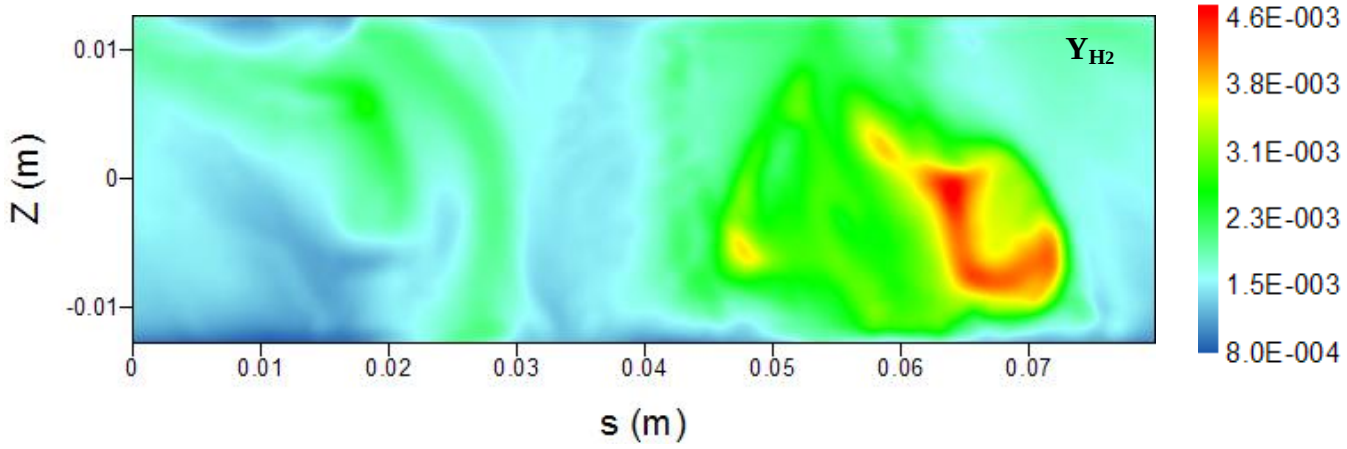

Figure 22 – Surface hydrogen mass fraction profile for test particle - Full cylinder

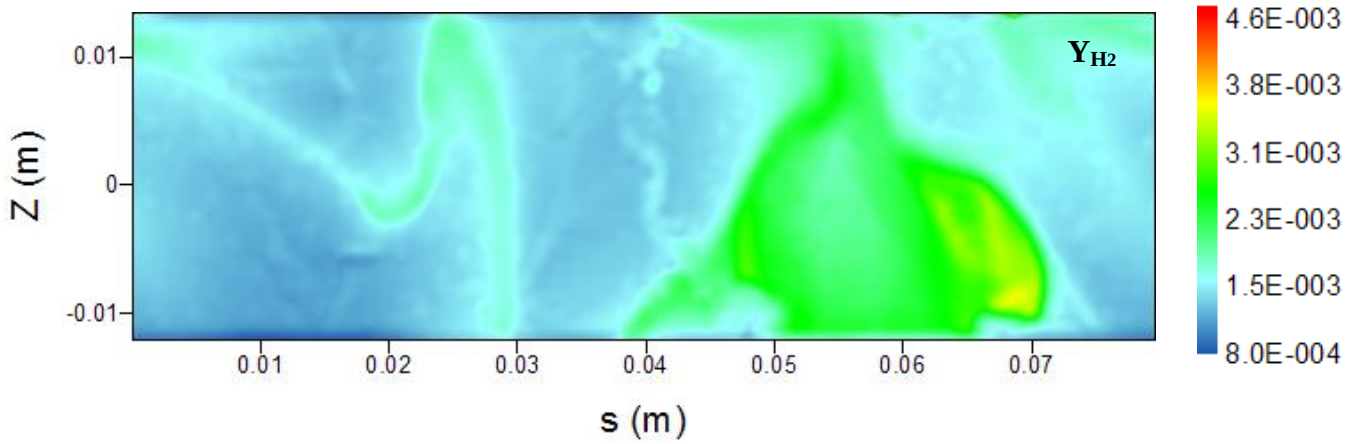

Figure 23 - Surface hydrogen mass fraction profile for test particle - 1 hole cylinder

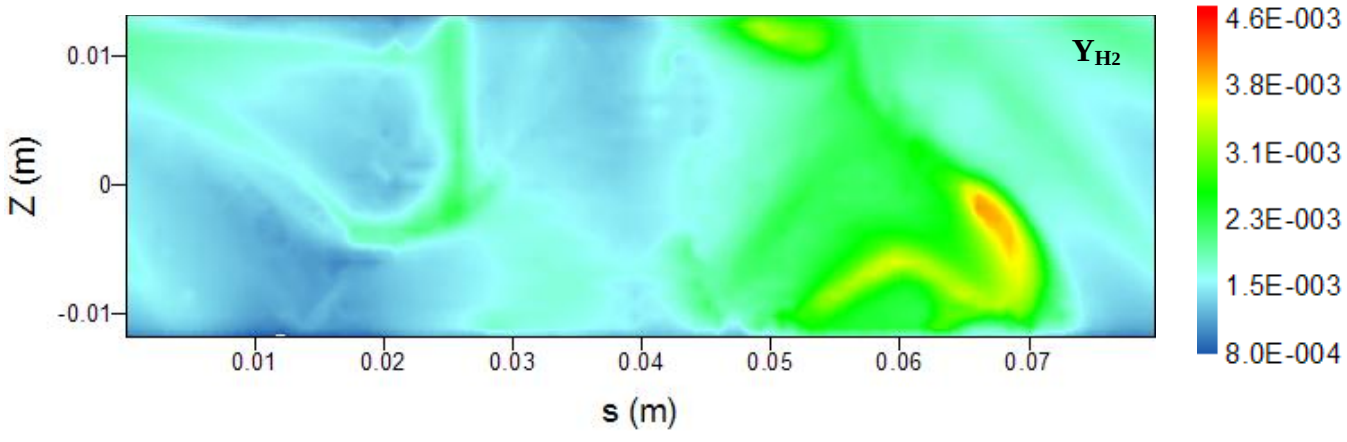

Figure 24 - Surface hydrogen mass fraction profile for test particle - 4 holes cylinder

As for temperatures, high gradients appear. Hence for the methane mass fraction, a relative variation of over 4% can be noticed between the highest and lowest concentration on the particle surface. The same thing is true with hydrogen, whose concentration ratio varies over a factor of 5. We can also clearly identify zones on the surface where the concentration is either high or low.

Moreover, these zones seem to overlap with the ones one could identify for temperature, i.e. hot zones have low methane and high hydrogen concentrations, and cold zones have high methane and low hydrogen concentrations. This is perfectly understandable as that the two main reactions are highly endothermic. Hence an increase in temperature will lead to an increase in reaction rates.

One can also notice that although temperatures and concentrations zones are in good agreement, the maximum in temperature and the respective maximum and minimum in methane and hydrogen do not have exactly the same location. The temperature maximum is roughly located at S ~ 0.05 m, whereas the concentration extrema are located at roughly S ~ 0.065-0.07 m. Hence, another phenomenon besides temperature seems to be needed to explain concentration discrepancy. This will be addressed in a further chapter (chapter III.6.).

If we once more compare the results of the three geometries, we can notice a close resemblance in the concentration patterns between the three. Not surprising is to notice that the full cylinder geometry has the lowest methane mass fraction among the three geometries. This is in perfect agreement with the fact that the full cylinder geometry has the highest temperatures. Hydrogen concentration also confirms that last point since the highest mass fractions are obtained for the full cylinder pellets.

4) Near wall particle inside study

Let's now focus our attention on what is happening inside the particle. Therefore, we will still consider the same test particle and we will consider two perpendicular planes inside that particle:

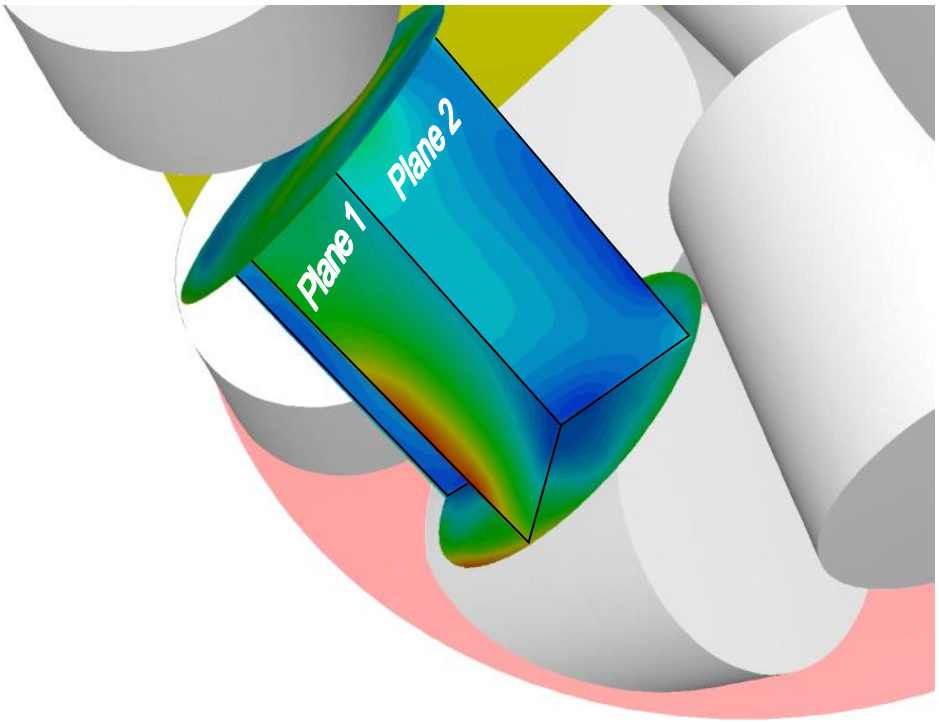

Figure 25 - WS geometry: particle test – inside

These planes have been chosen such as plane 1 is roughly perpendicular to the reactor wall and plane 2 parallel to that same wall.

a. Temperature comparison

Figure 26, Figure 27 and Figure 28 represent the temperature variation along the two planes previously described. Notice that for the 4 holes cylinder geometry's results, it is as if the two holes displayed where not of the same diameter. This is simply due to the way planes 1 and 2 pass through the holes. They do not follow the holes' diameters. Let's first consider only plane 1, the one perpendicular to the reactor wall.

Looking at the temperature variation inside the test particle, we can notice that the gradient is essentially located in the region of the pellets which is the closest to the wall. Past that zone, temperature smoothes out rapidly.

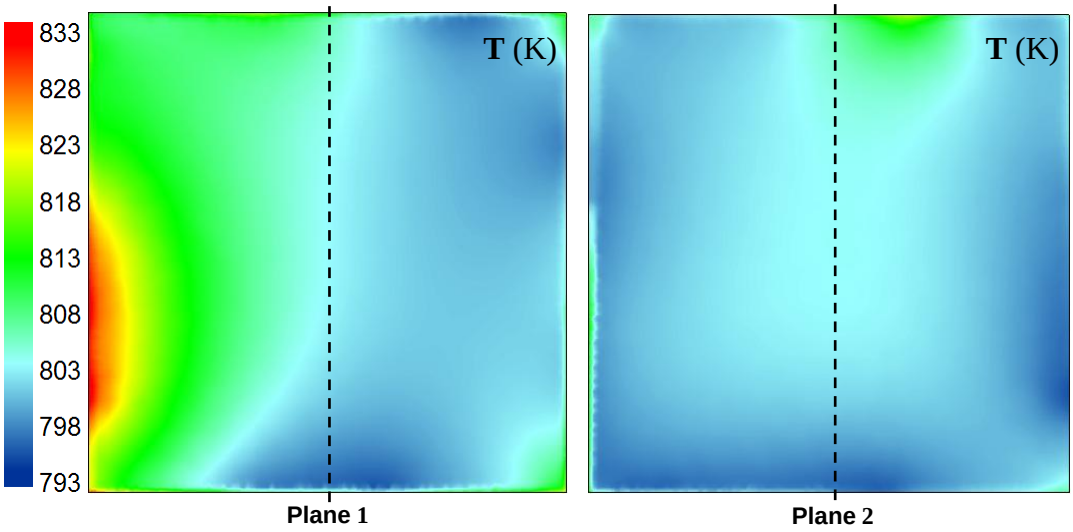

Figure 26 – Temperature profile through test particle - Full cylinder

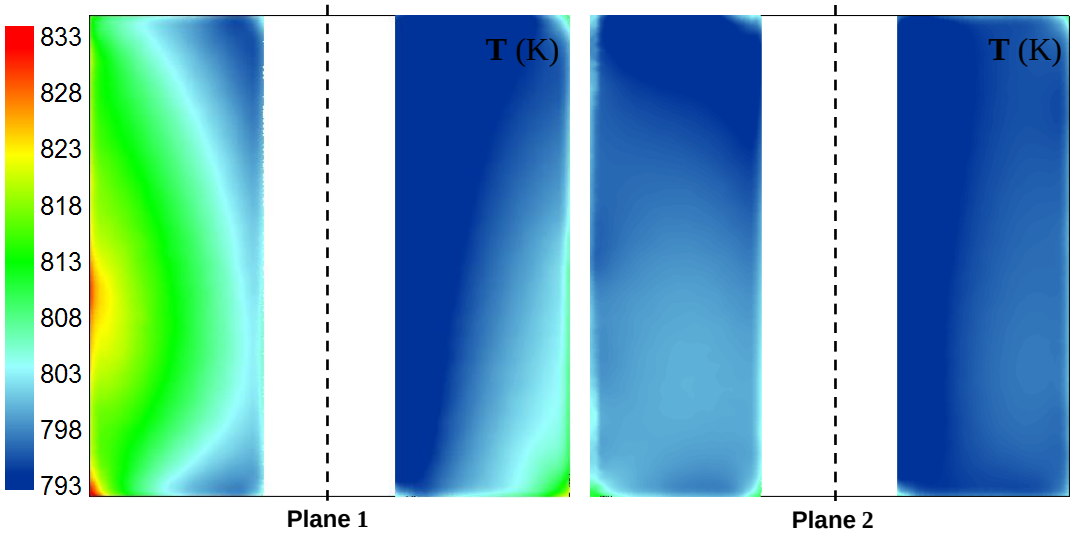

Figure 27 - Temperature profile through test particle – 1 hole cylinder

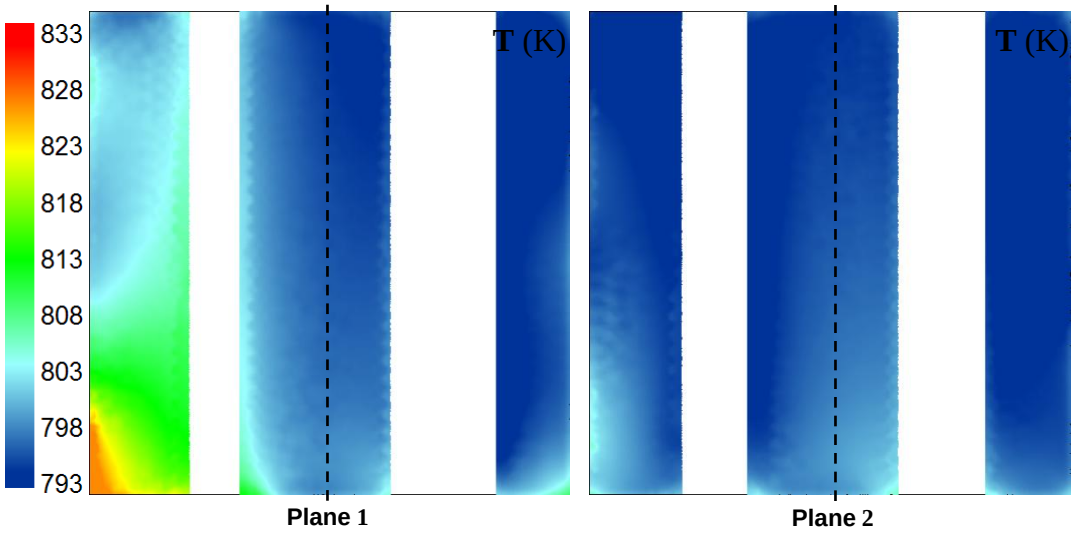

Figure 28 - Temperature profile through test particle – 4 holes cylinder

This becomes even more striking as we consider the other geometries. For instance with the 1 hole geometry, we can clearly see that the hole divides the plane in two. The left side presents a high gradient, whereas the right side has a more homogenous temperature. This is showing that the right side of part1 is too far from the wall to see any of the heat effects.

If we now consider plane 2, we can see that temperature is quite homogenous, no matter what geometry we consider. Besides, the mean temperature decreases going from full to 4 holes cylinder. This can be explained by the fact that the more holes are added the larger the interface surface with the fluid will be. And as we will see in the next chapter, the methane steam reforming reaction is diffusion limited. Therefore, if the interface surface increases, the more fresh reactant will be brought to the pellet and the more products will be converted. Consequently, there will be more heat removal, and a lower mean temperature.

b. Mass fraction comparison

The following pictures show the methane (Figure 29 through Figure 31) and the hydrogen (Figure 32 through Figure 34) mass fraction for plane 1 and 2. The first thing we can notice is a high gradient in concentration near the interface solid/fluid. This is due to diffusion limitations, i.e. the reactant that enters the particle reacts so quickly that it doesn't have time to reach the center of the pellet. Hence we can see that the methane concentration is much higher at the interface than at the core of the particle.

The same thing can be seen for hydrogen. A relatively low concentration can be seen near the interface. The hydrogen that is produced by the reaction at the interface will diffuse where its gradient is most favorable, that is to say towards the fluid zone where the hydrogen concentration is the lowest.

Besides, we can see that the core of the pellets is almost homogenous in concentration. It is as if diffusion is so limited that the system had enough time to reach the reaction equilibrium. Nevertheless, a slightly lower methane concentration and a slightly higher concentration can be noticed in the near wall region. The reaction being endothermic, an increase in temperature will be in favor of more hydrogen production. And the system equilibrium will thus be move towards a lower methane concentration.

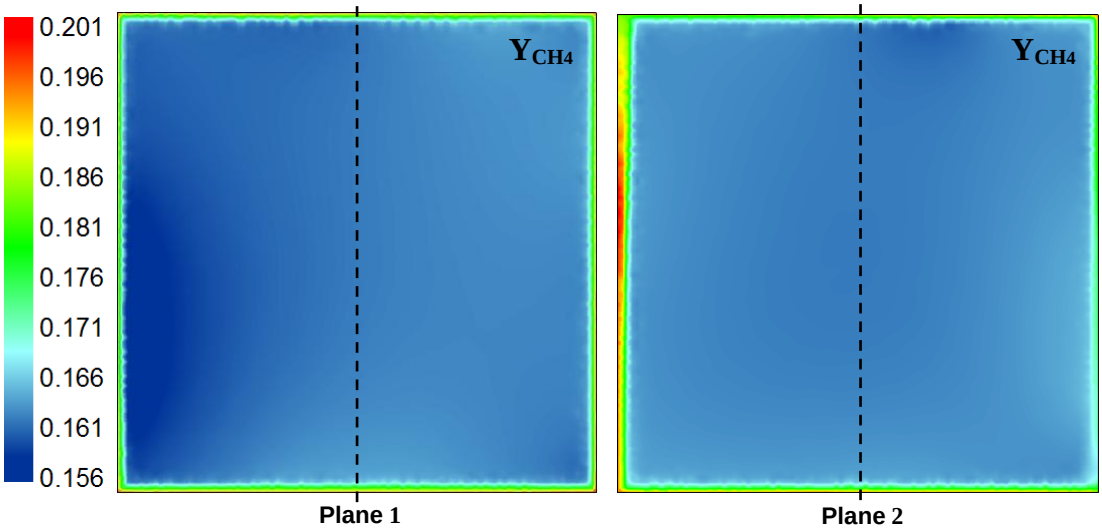

Figure 29 - Methane mass fraction profile through test particle - Full cylinder

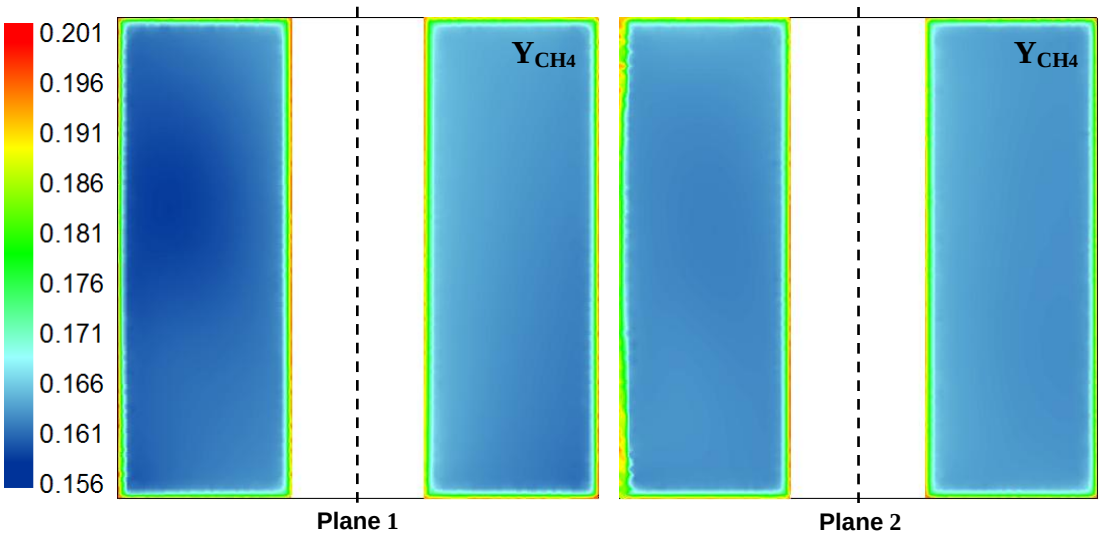

Figure 30 - Methane mass fraction profile through test particle – 1 hole cylinder

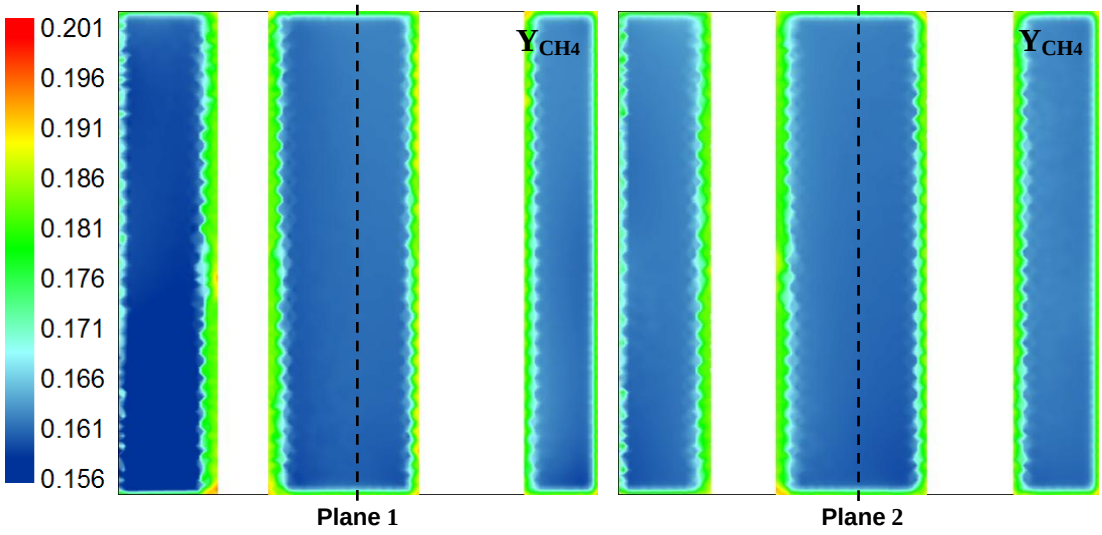

Figure 31 - Methane mass fraction profile through test particle – 4 holes cylinder

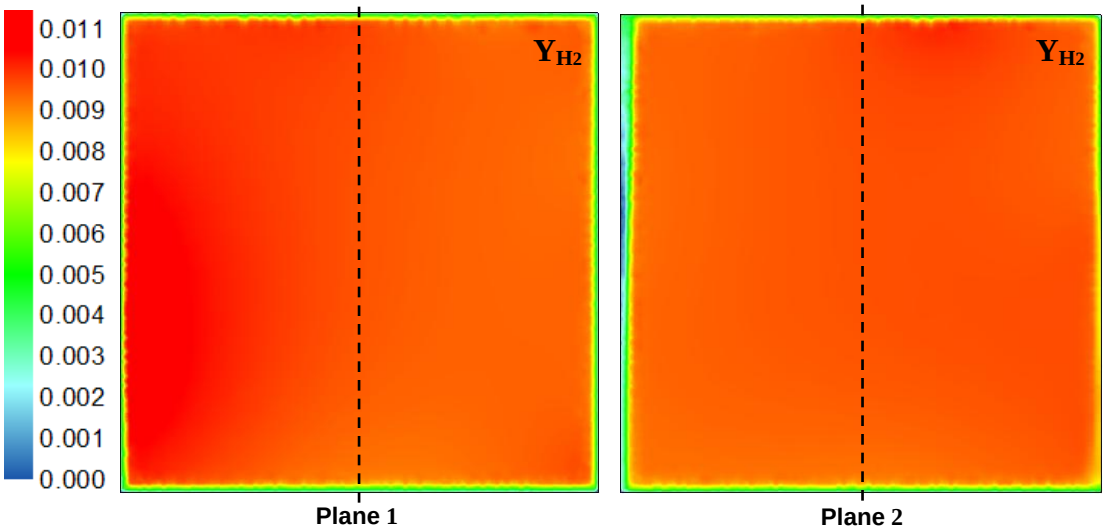

Figure 32 - Hydrogen mass fraction profile through test particle - Full cylinder

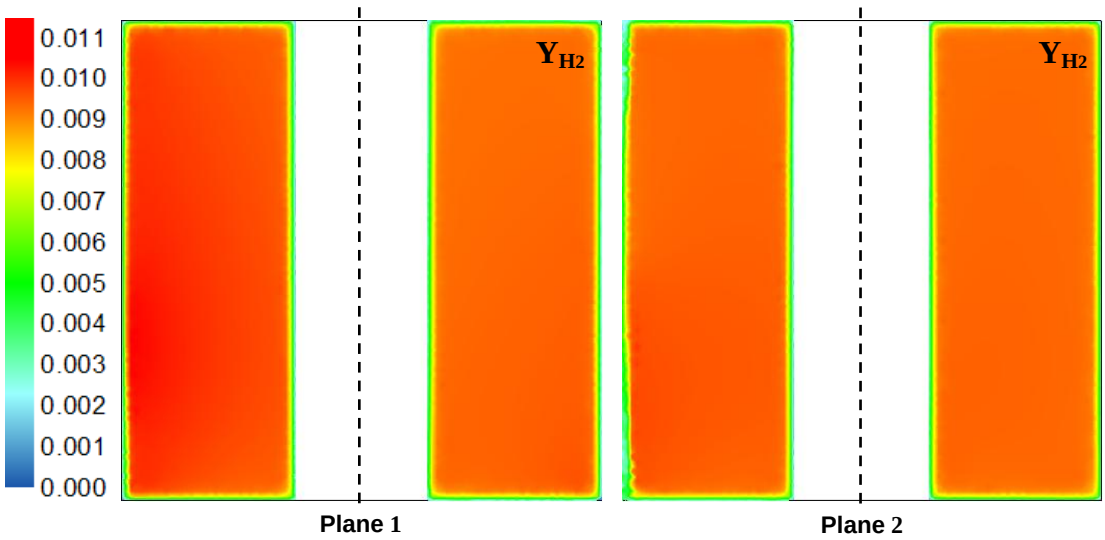

Figure 33 - Hydrogen mass fraction profile through test particle – 1 hole cylinder

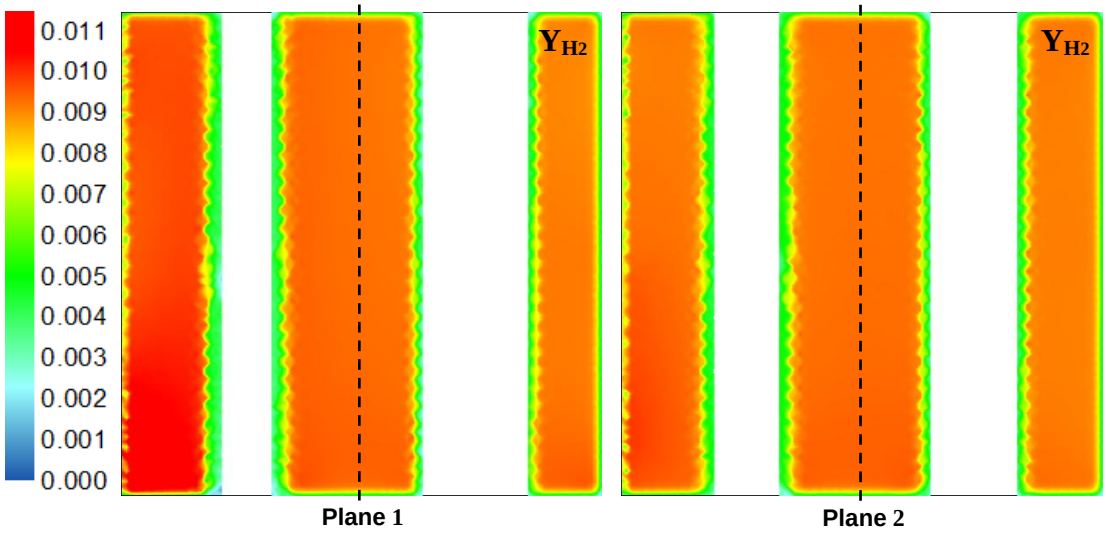

Figure 34 - Hydrogen mass fraction profile through test particle – 4 holes cylinder

5) Zones of low methane concentration

a. Cross sectional plane

The near wall region is not the only zone where some asymmetries in concentration appear. As one could see on Figure 19, surface concentration seems to vary even away from the wall region. The following picture highlights some zones of methane depletion.

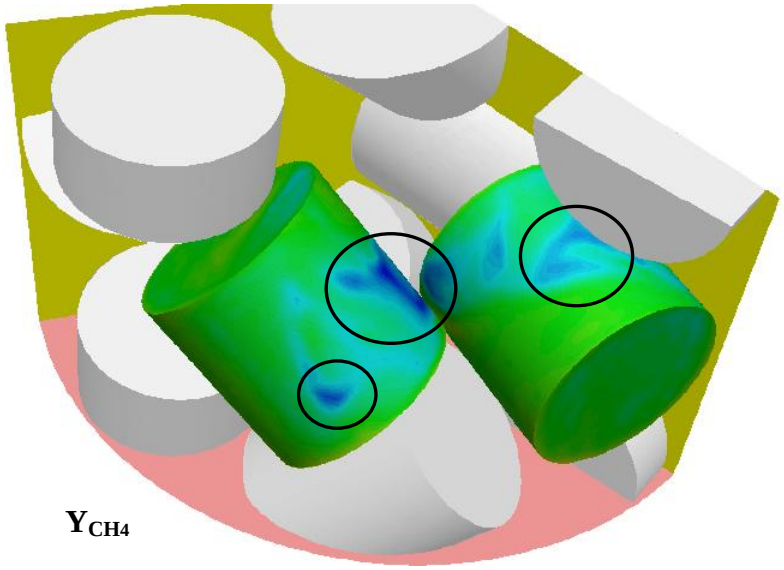

Figure 35 - Examples of zones of methane depletion

In order to study in more detail this phenomenon we created a plane going through some of these zones. Figure 36 - Cross sectional plane for methane depletion study shows this plane:

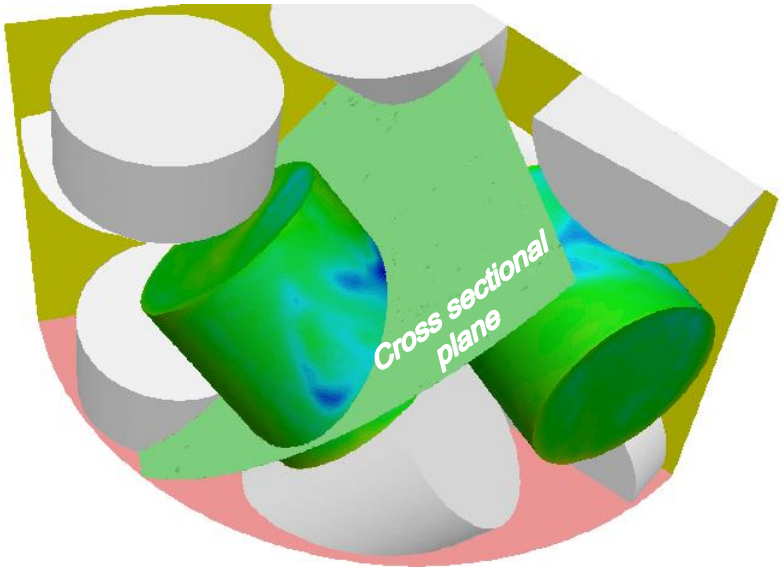

Figure 36 - Cross sectional plane for methane depletion study

b. Results for the full cylinder geometry

The first parameter we will consider on that cross sectional plan is the methane mass fraction:

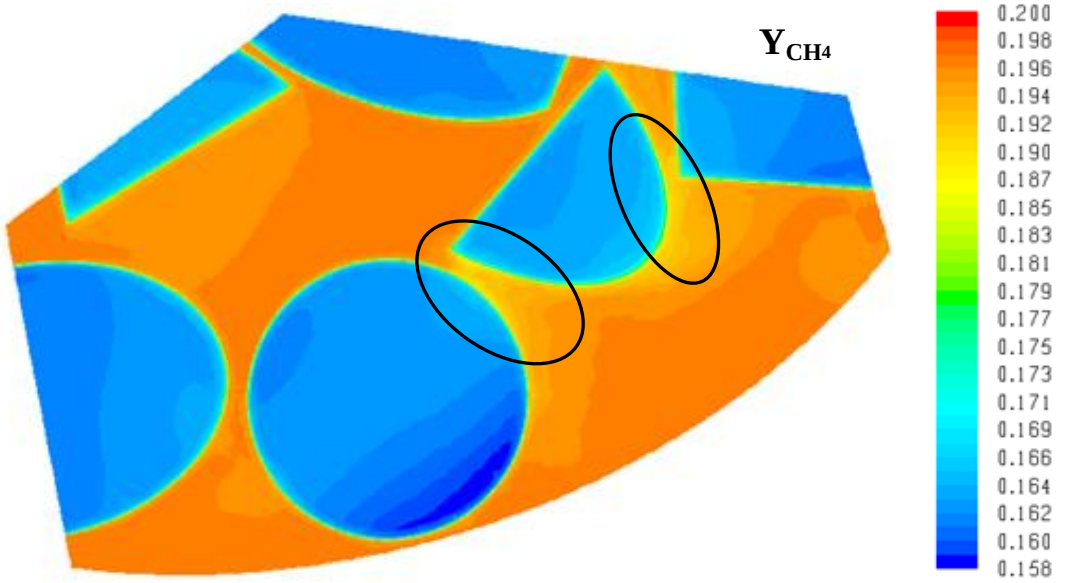

Figure 37 - Cross sectional plane – Methane mass fraction profile

We can notice on Figure 37, two zones of relatively low methane concentration. The following figure (Figure 38) shows the hydrogen mass fraction. We can notice a higher hydrogen concentration in the two zones of methane depletion.

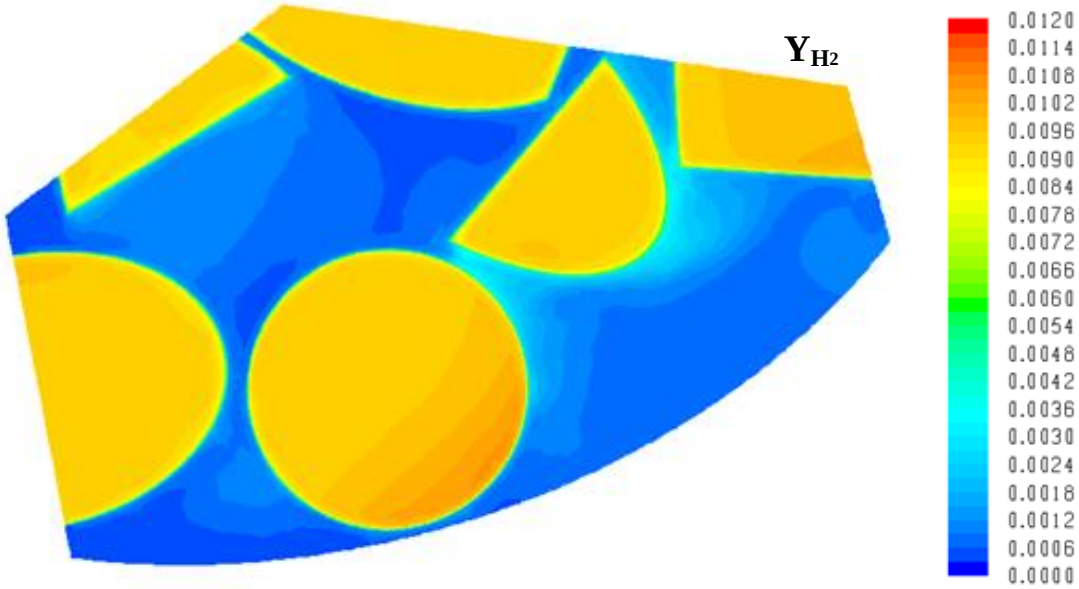

Figure 38 - Cross sectional plane - Hydrogen mass fraction profile

Looking carefully, we can see that the methane depletion zone appears in a narrow zone between particles, where the flow velocity is likely to be low. Therefore, the following figure (Figure 39), we looked at the velocity magnitude profile of that same cross sectional plane.

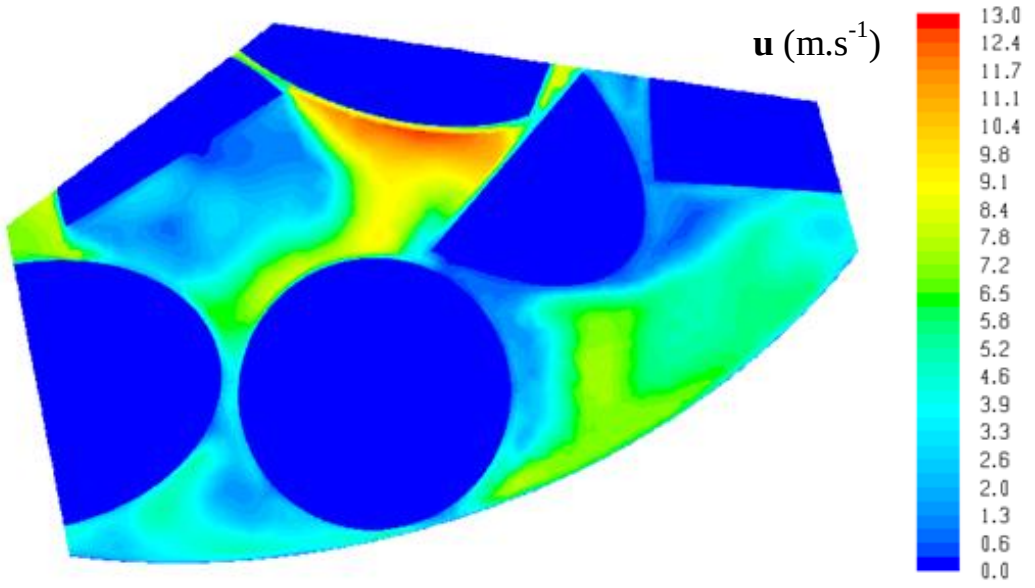

Figure 39 - Cross sectional plane – Velocity magnitude profile

We also had to make sure that temperature was not playing a role in the methane depletion, and that is what that next figure (Figure 40) is here for:

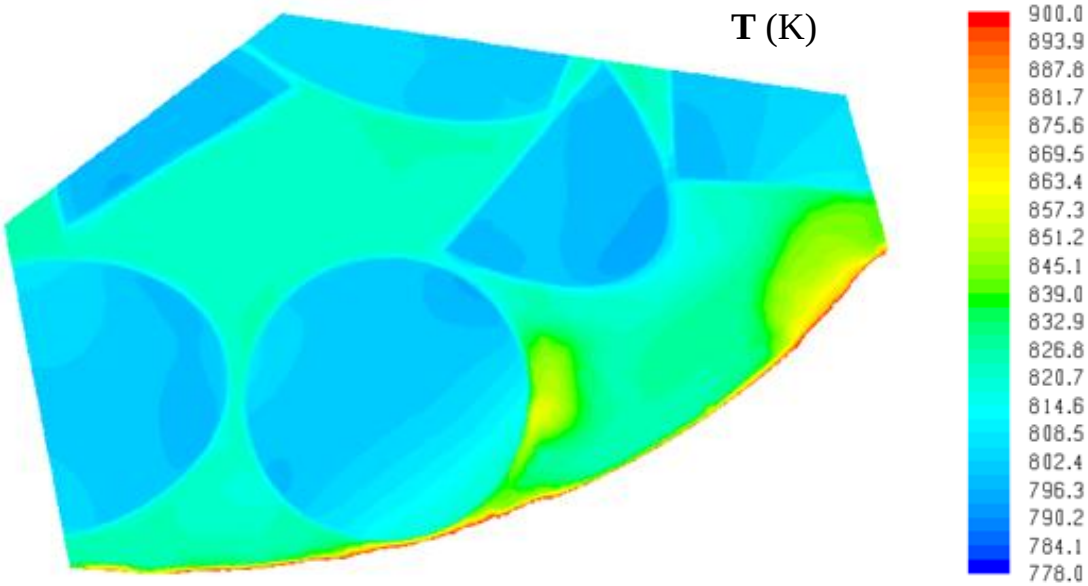

Figure 40 - Cross sectional plane - Temperature profile

We can notice a slight decrease in temperature. However, if we could notice a temperature variation of about 50˚C in the near wall region, the temperature variation in comparison to the bulk temperature, is only of about 10˚C. This change in temperature seems therefore too weak to explain the depletion and it more likely to be a consequence of the low velocity field which would slow down energy transfer.

A quick computation shows that last point. Here let's consider the rate of the third reaction, which is, as we will see later (chapter III.6.b), the dominant reaction. Average values for the species have be taken (Y_{CH_4} = 0.193, Y_{H_2} = 0.00185, Y_{CO} = 0.00133 and Y_{CO_2} = 0.191). We thus obtain the following ratios:

$$\frac{r^{III}_{834.15K}}{r^{III}_{824.15K}} = 1.6 \qquad \frac{r^{III}_{874.15K}}{r^{III}_{824.15K}} = 8.7$$

Hence an increase of 10˚C in temperature increases the reaction rate of a factor of 1.6, and an increase of 50˚C in temperature increases the reaction rate of a factor of 8.7. This clearly shows that the reaction rate variation in the near wall region is more than 5 times higher than what it is away from the wall.

c. Results for the 1 hole cylinder geometry

To make sure that this phenomenon was not just an artifact of the computations, we did the same thing, but using a different geometry. This time, the 1 hole geometry is used. Once more, a cross sectional plane passing through some methane depletion zones was created.

In the same order as previously, the following four figures (Figure 41 through Figure 44) show respectively the methane and hydrogen mass fractions, the velocity profile and the temperature profile.

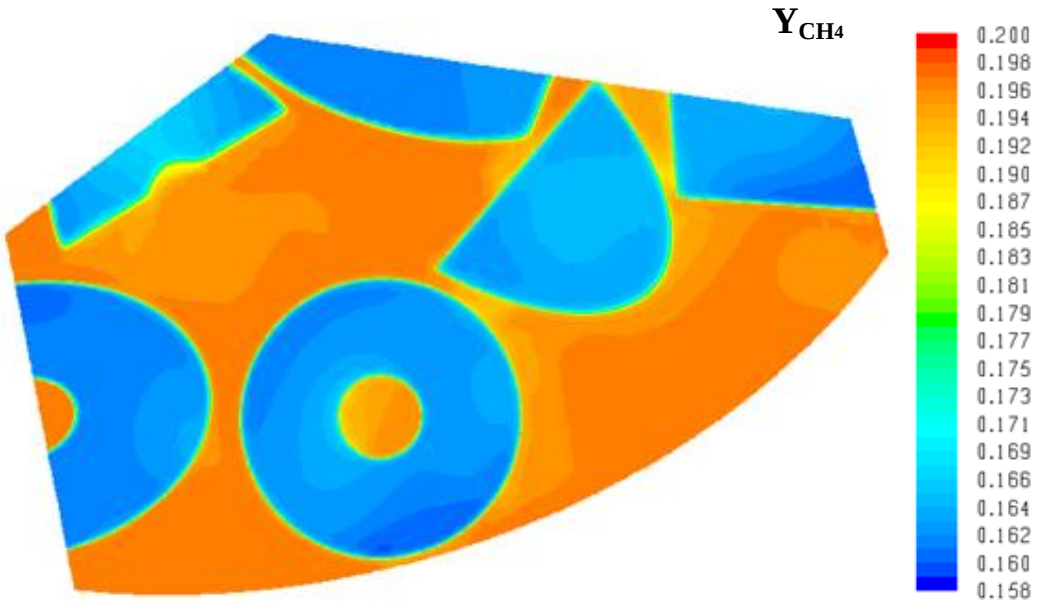

Figure 41 - Cross sectional plane – Methane mass fraction profile

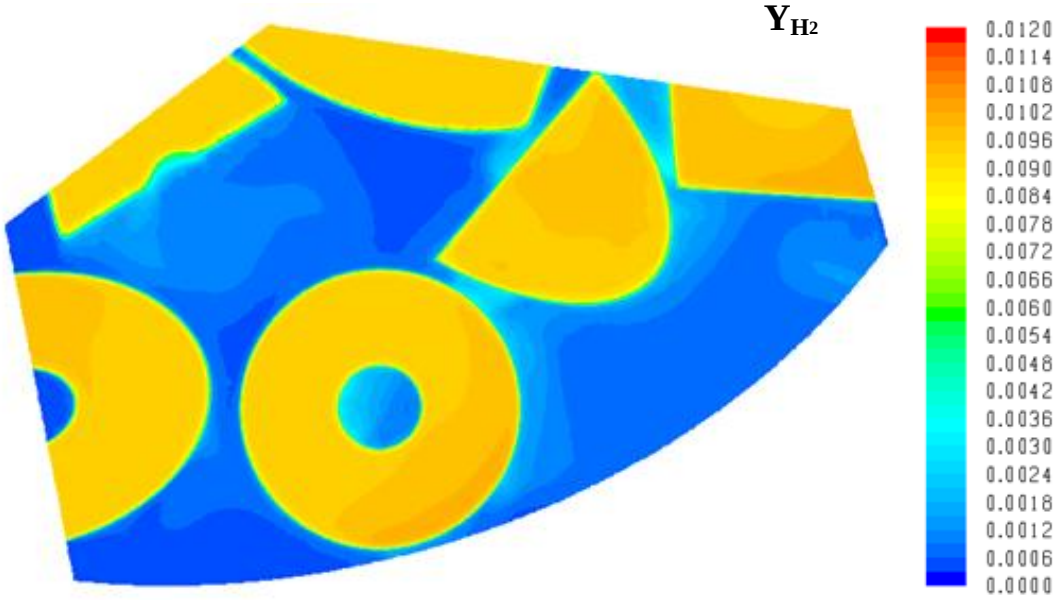

Figure 42 - Cross sectional plane - Hydrogen mass fraction profile

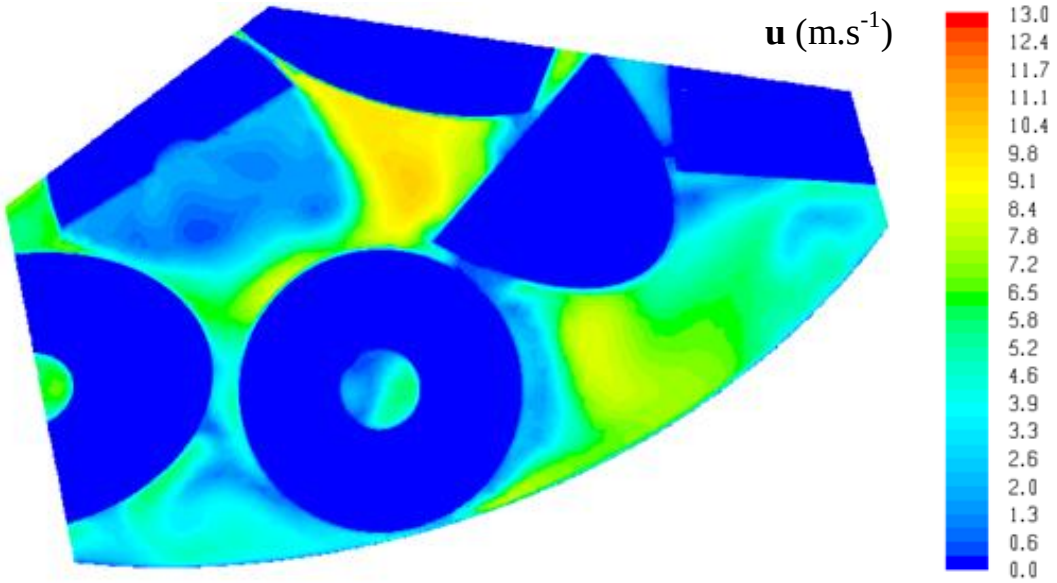

Figure 43 - Cross sectional plane – Velocity magnitude profile

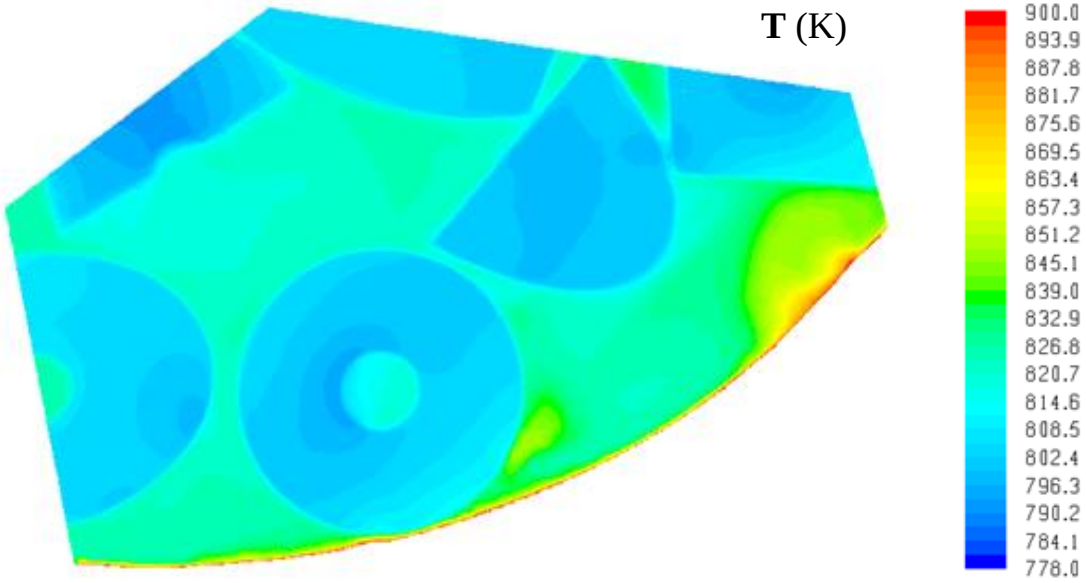

Figure 44 - Cross sectional plane - Temperature profile

The results of the 1 hole geometry are very close to the full cylinder geometry. For instance, both low methane concentration zones that we highlighted for the previous geometry are present here. And they both correspond to a higher hydrogen concentration, and a low velocity zone. And once more, the temperature variation seems too low to really be a key parameter for this phenomenon to appear.

We can also point out that the 1 hole geometry the overall velocities are lower that what they used to be with the full cylinder pellets. This can be explained by the fact that the mass flowrate imposed to the system remains the same while switching from one geometry to the other. Therefore a higher void fraction will entail lower velocities. Consequently, we can notice that more methane depletion zones appear for the 1 hole geometry, mainly in the pellets' hole. This new low velocities eventually leads to low methane concentrations.

d. The adiabatic run for the 1 hole cylinder geometry

As for now, results have shown that low methane concentration regions seem to be linked to low velocities region, and that temperature seems not to be playing a role. To confirm that last point, one more simulation was run. The idea was to remove all temperature effects and see if the methane depletion zones remained. Therefore, the heat flux through the reactor wall was turned off. We thus considered an adiabatic case. The results obtained were the following (Figure 45 through Figure 48):

As shown by Figure 45, the methane concentration has increased. Indeed, the lack of wall flux has decreased the particle temperature by 10 to 15˚C. Consequently the reactions rates have decreased, and so did methane conversion. However, we can still clearly see the same methane depletion zones.

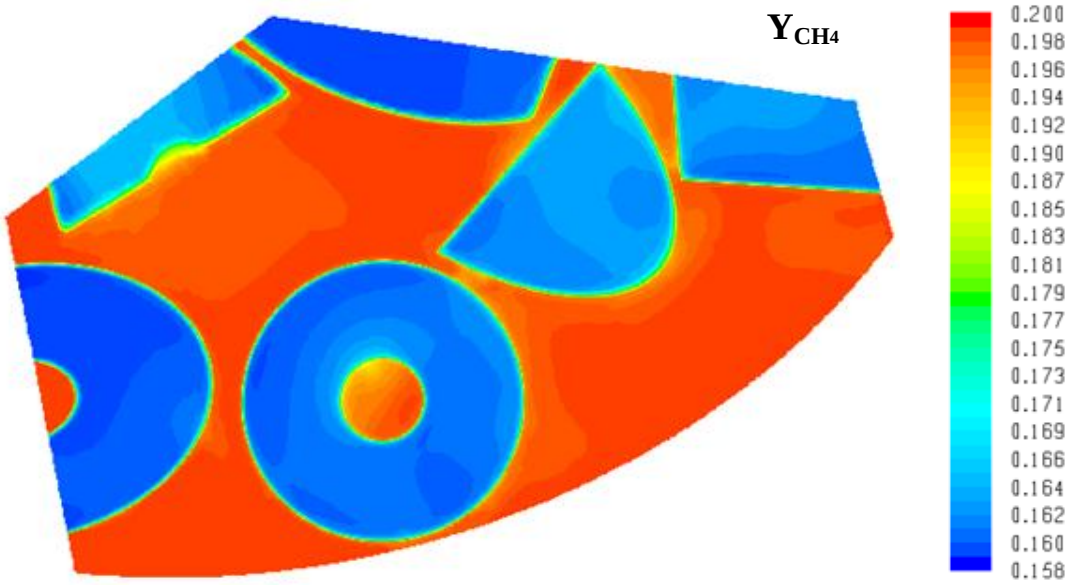

Figure 45 - Cross sectional plane – Methane mass fraction profile – Adiabatic case

Once more, Figure 46 shows that higher hydrogen concentrations appear for low methane concentrations zones.

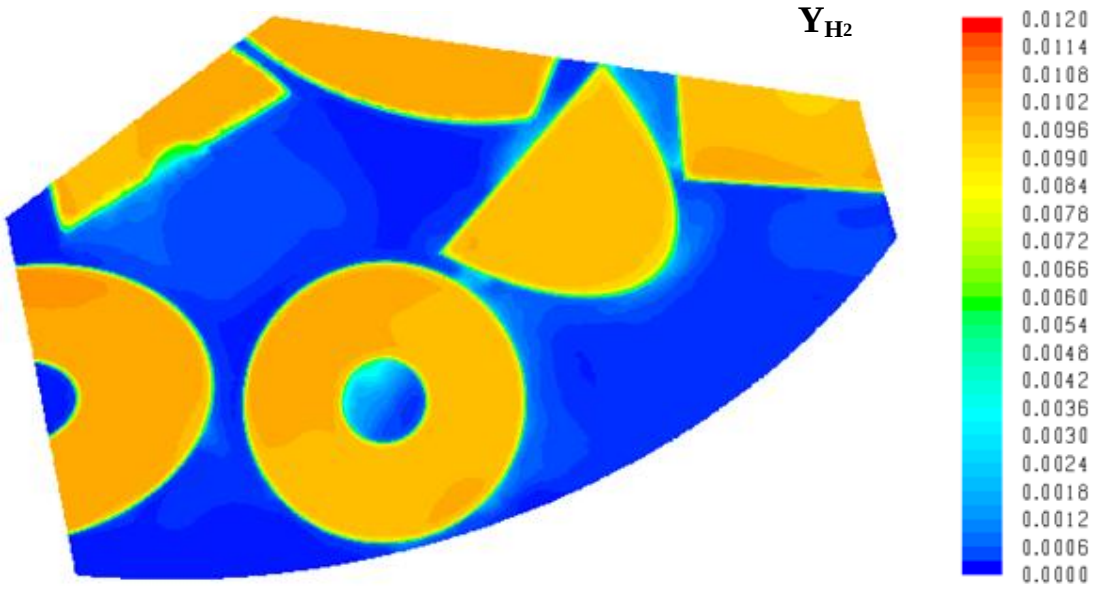

Figure 46 - Cross sectional plane - Hydrogen mass fraction profile – Adiabatic case

Figure 47 shows velocity magnitude. One can notice that the velocity field for the adiabatic case is almost identical to the non adiabatic case (Figure 43). This is a result of our choice of taking non temperature dependent flow parameters.

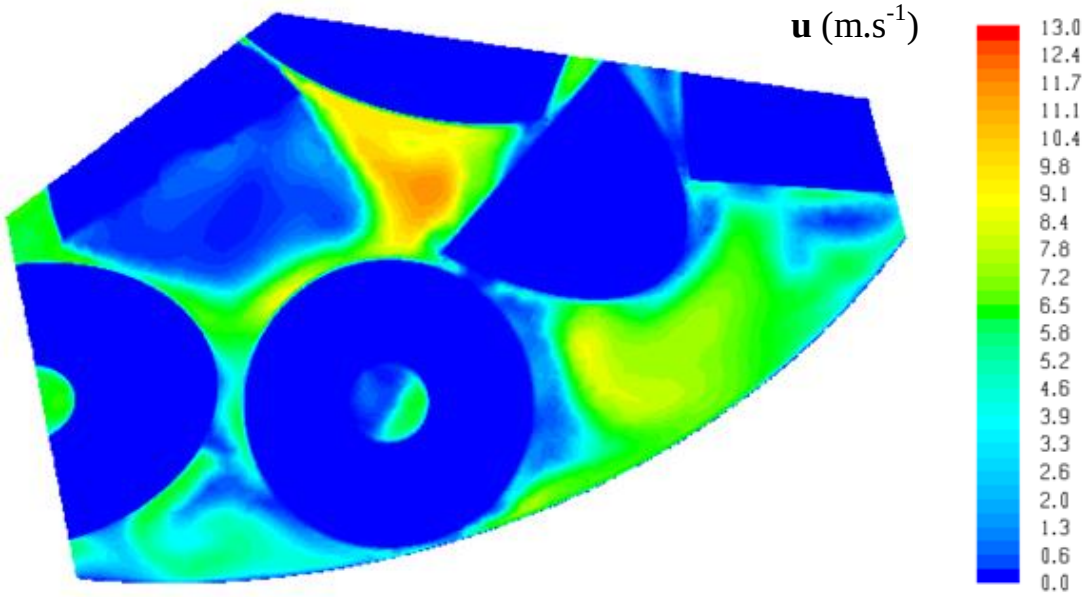

Figure 47 - Cross sectional plane – Velocity magnitude profile – Adiabatic case

This last figure (Figure 48) is very interesting since it shows the temperature profile. Given that we imposed an adiabatic regime, no wall heat effect can influence the results; however we can still see a decrease in temperature in the methane depletion zones.

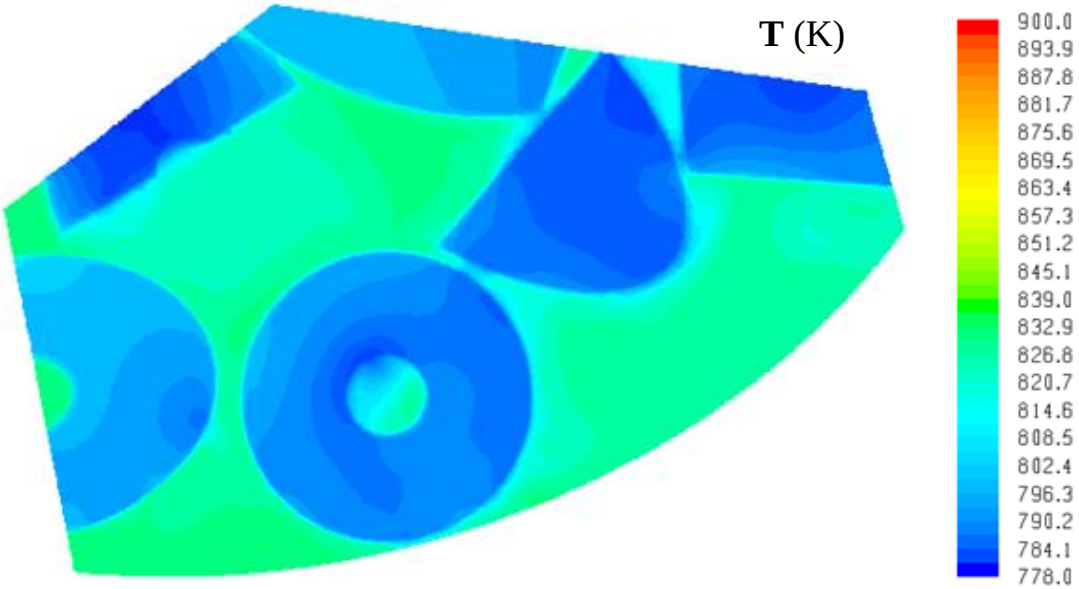

Figure 48 - Cross sectional plane - Temperature profile – Adiabatic case

e. Conclusion

As previously seen in chapter III.2., asymmetries in both temperature and species concentrations appear as we considered a packed bed. If an obvious reason for this asymmetry was due to the wall effect, several methane depletion zones could be seen away from that near wall region. Hence, the wall heat could not be the main reason for these singularities in the methane concentration to appear.

Given the position of these depletion zones (in narrow gaps between particles), it seemed adequate to suspect these zones to be almost stagnant. Therefore a cross sectional plane passing through some of these depletions regions was created. We could then notice that indeed low velocities regions and low methane concentration (as well as high hydrogen concentration) were closely linked. This has been done for both the full and the 1 hole geometry. The results obtained for both showed that same behavior.

To confirm that wall effects were not playing any role in that behavior, an adiabatic case has been run. We then were able to notice that a system still showed the same behavior, though it was denied of any heated wall effects. This is a clear proof that the methane depletion is not only due to heated wall effects, but is also influenced by low velocities region.

In an actual reactor, this variation of species concentration will eventually lead to a different rate of coke formation over the surface. Hence the coke layer will not be of the

same height everywhere on the particle's surface. Consequently this particle will undergo different mechanical stress depending on that coke layer and eventually break. And as seen in chapter I.2, particles' dust is responsible for non homogenous flow and consequently hotspots.

6) Extrema study

a. Phenomenon description

As we saw previously, the temperature and concentrations extrema are not located exactly at the same place. At first glance, this seems a bit contradictory, since a high temperature for an endothermic reaction increases the reaction rate. And thus more methane will be consumed and more hydrogen produced. So following that way of thinking, it is natural to expect the maximum temperature, the minimum methane concentration and the maximum hydrogen concentration to be located at the same place.

The following figure (Figure 49) helps better seeing that. It is composed of previously seen Figure 12 and Figure 19, that is to say temperature and methane mass fraction for the test particle's surface.

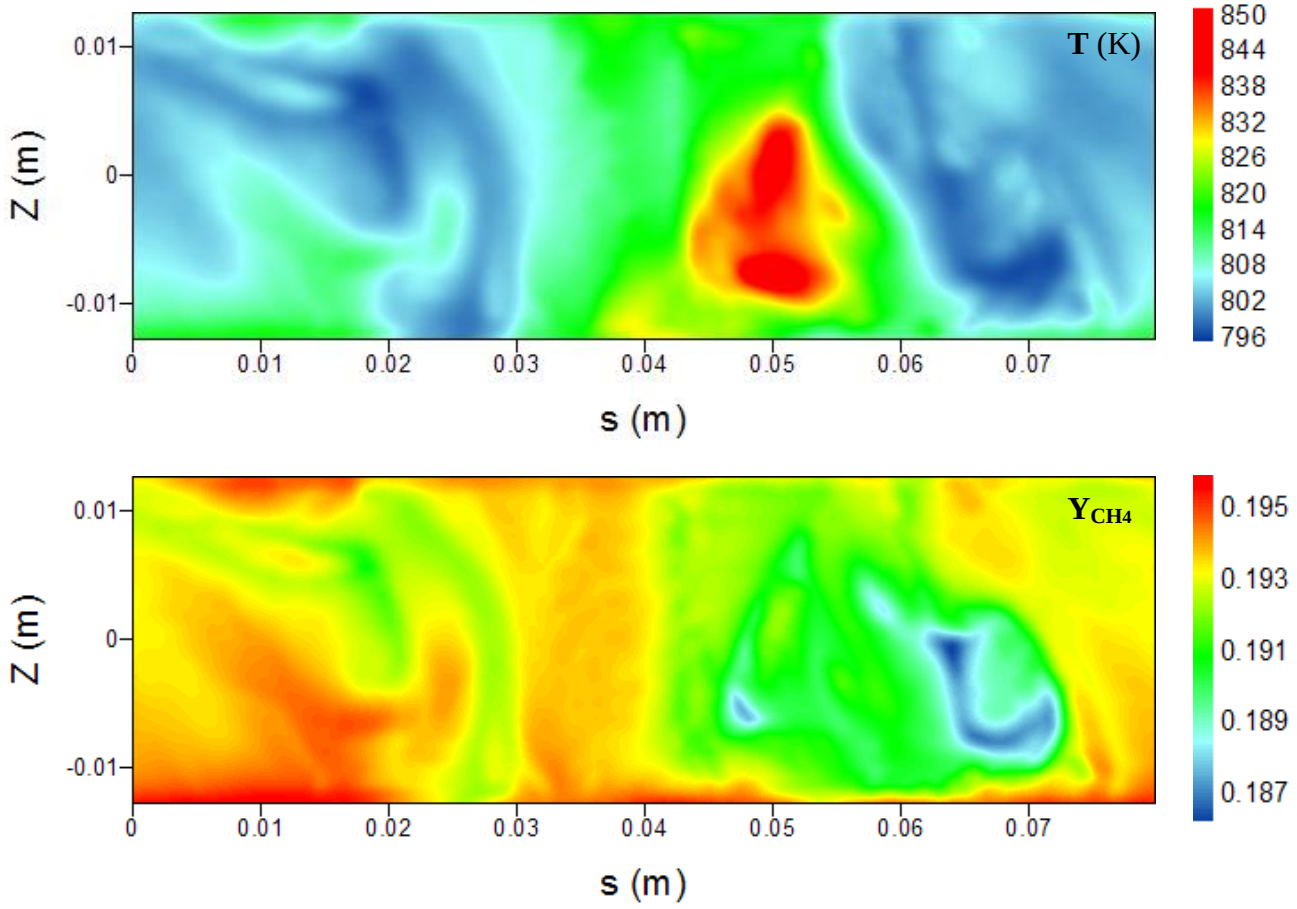

Figure 49 – Extrema comparison for the full cylinder geometry.

This figure only shows the comparison for the full cylinder geometry, but the same thing can be seen for the other geometries, as their results have previously been shown in chapter III.3. The maximum temperature displayed on Figure 49 appears for S ~ 0.05 m, and the minimum methane concentration appears for S ~ 0.065-0.07 m.

b. Reaction rates profiles for the full cylinder geometry

For the full cylinder geometry, the reaction rates are given in Figure 50. We can first notice that for all the reactions, their reaction rate maxima are located in the same area, i.e. around S ~ 0.05 m. This is also the temperature maximum location. Moreover, we can very clearly see a good resemblance between the pattern drawn by the high temperature area and the one drawn by the high reaction rate area of the first and third reaction. We can therefore conclude that temperature variation have more effect than the other species concentrations on the reaction rates at the working conditions.

However, for lower temperatures, i.e. T ~ 800-820 ˚K, temperature variations seems to be more or less counterbalanced by the species concentrations. In other words, for the first reaction, for a temperature difference between the hottest and coldest zones of 55˚C, the reaction rates ratio increases over 5 times; whereas in a cold zone, a temperature variation of 20˚C only give rise to a variation in reaction rates of less than 1 to 2. If we consider the two other reactions, although if the figures are different, we can still see that the variation in reaction rates is very low for cold zones and increases quickly in the hottest zone.

The other interesting thing to notice is that the main reaction to take place at the inlet of the reactor tube is the third reaction. Its rate of reaction is two orders of magnitude higher that the ones of the other reactions.

This however does not explain the difference in the extrema locations, and a closer look to the methane minimum is required. This is done in the coming chapter.

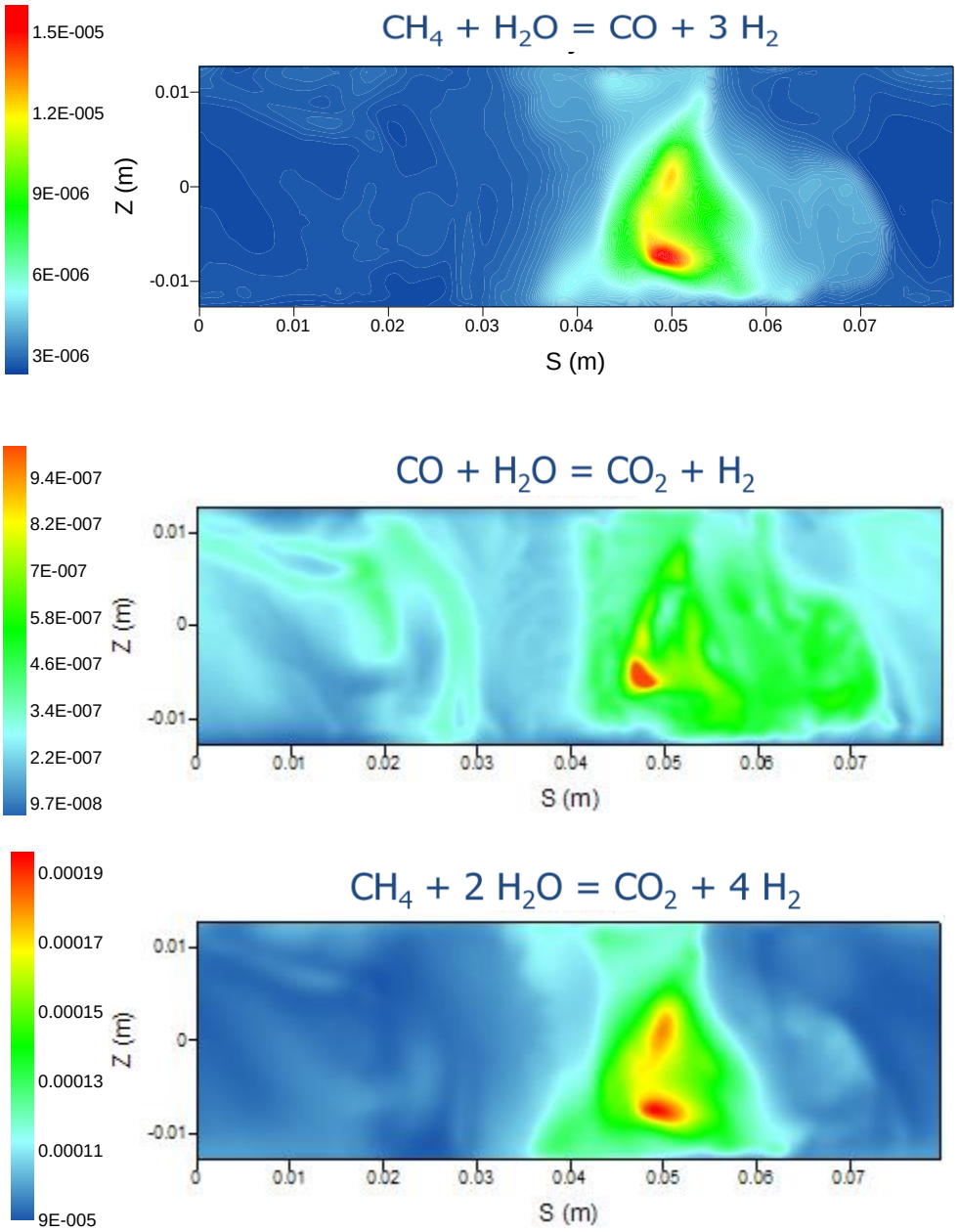

Figure 50 – Reaction rates for the full cylinder geometry

c. Cross sectional plane

In this part, a new cross sectional plane was created, passing through the temperature maximum and the methane mass fraction minimum. The plots for temperature, velocity magnitude, methane and hydrogen mass fraction are given on the next page (Figure 52).

Surprisingly, we can notice that the pellet's highest temperature is not achieved by the section of that surface which is the closest to the wall, but in fact slightly further

on. This gets explained by the velocity profile. There is an important flow passing near that zone, cooling down the section of the pellet. On the contrary, the hottest section is near an almost dead zone (i.e. with almost no flow). Consequently, heat exchanged by convection is very low, and less heat gets removed.

However, one can see that the region where the velocity is even lower is not near that hot zone, but between the two particles. And this is where the methane concentration is the lowest. Hence what is happening is that methane is not diffused by the turbulent flow, and only natural diffusion occurs. And yet, we know that this diffusion is a lot slower. Therefore, the rate at which the reactant gets renewed in the nearby section of the pellet is low.

Moreover, whereas usually, methane can diffuse to the surface from five different directions as shown by the green arrows (Figure 51a), for zone comprised between the two pellets, it can only diffuse from four directions (Figure 51a):

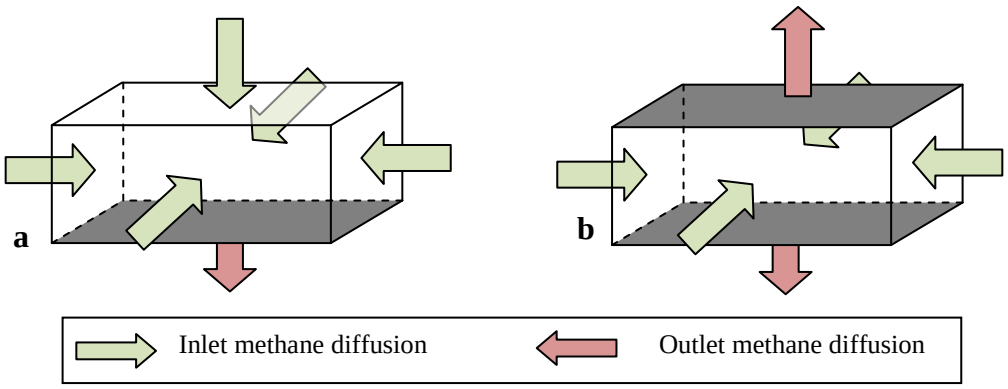

Figure 51 – Schema of the methane diffusion between to pellets (a) for a free surface; (b) between two pellets

In other words, the zone between the two pellets is more confined, and therefore diffusion from the surrounded fluid is limited. Moreover, pellets are a zone of methane consumption and therefore, the methane concentration is lower there that in the fluid. Therefore, methane diffuses from the fluid to the solid. We can see that in the case of a confined area (case b), methane is drained from two surfaces as opposed to only one surface for a free particle's surface (case a). In other words, in a confined area, less methane diffuses in, and more diffuses out than for a free surface. As a result the methane concentration for a confined area is lower than for the rest of the fluid.

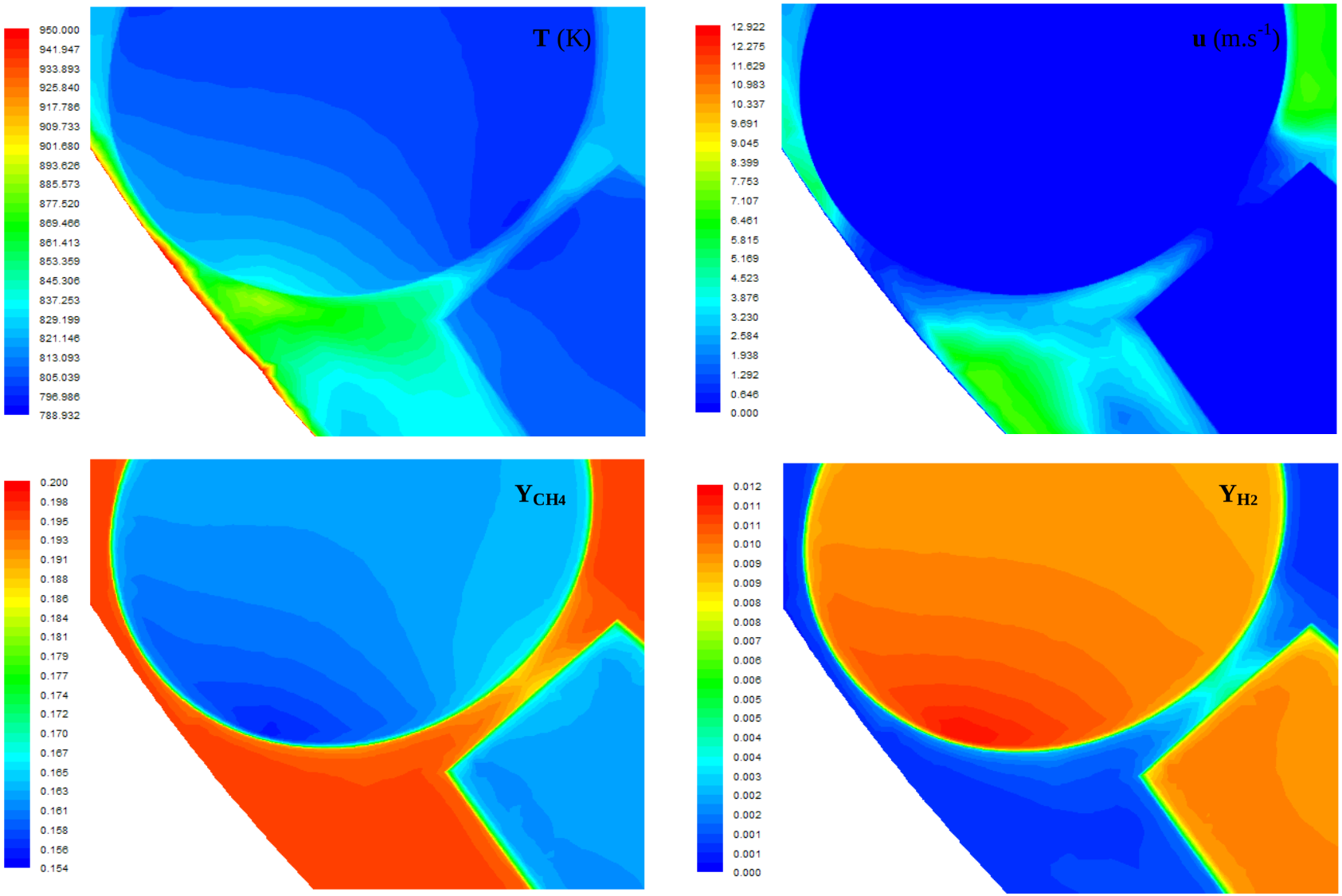

Figure 52 - Results for the full cylinder cross sectional plane

d. Conclusion

If one studies the test particles' surface, it appears that species and temperature extrema do not have the same location. Further investigations have shown that reaction rates and temperature maxima match.

Furthermore, it has been notice that the extrama (i.e. temperature maximum and methane minimum) are always located near a low velocity region. This is especially true for temperature, since the area which gets the hottest is not the one the closest to the wall, but rather located near the wall and near a low velocity region.

Therefore, both temperature and methane concentration minimums can be explained by their proximity to a low velocity zone. In these latest, flow can be considered laminar, and therefore convection transportations for both energy and species are low.

Nevertheless an extra parameter comes into play. Otherwise why would the methane maximum concentration not be located at the same place than the temperature maximum? It appears that the fluid neat the methane minimum is also located is a confined area between two particles. Hence, methane diffusion can only come from limited directions. Moreover, since this confined zone is between two particles, methane has to diffuse toward two different particles' surface. Schematically, less can diffuse in, but more has to diffuse out. The methane concentration's minimum can therefore be explained its closed location to a low flow region and a confined zone.

Note that flow region and a confined zone do not necessarily come together. However, the confinement of an area makes is even more difficult for fluid to access it, and that zone will be more likely to be a dead zone for flow.

IV) Diffusion model - Theory

1) The various diffusion regimes

Diffusion in a porous medium depends on various parameters among which are the pore diameter, the species, temperature and pressure gradients, as well as external forces. The importance of these parameters depends on the case studied, and therefore, three main types of regime can be identified (Mason & Malinauskas, 1983):

- **The Knudsen Flow.** For low density region, the main type of collision will be between the gas molecule and the wall, and thus collisions between gas molecules will be neglected. This mainly happens in small pores, for which

$$l >> d_p \qquad \textit{eq. IV.1-1}$$

  where l is the mean free path and d_p the pore diameter.

- **The Viscous, Convective or Bulk Flow.** Here, the main type of collision is between the gas molecules, and thus the gas molecule/wall collisions are neglected, and we will have

$$l << d_p \qquad \textit{eq. IV.1-2}$$

  In that case, the fluid will be considered as a continuous medium, and the flow will be pressure driven.

- **The Ordinary or Continuum Diffusion.** Molecules migrate because of species gradients, temperature gradients or because of external forces. Once more in this case, the gas molecules to molecules collisions dominate (*eq. IV.1-2*).

A fourth regime also exists, but since in most cases, it is not worth being considered, and that it is independent of the others, it is often neglected. However it can be added easily in the computations since it occurs in parallel to the others. This fourth regime is:

- **The Surface Flow or Diffusion.** This regime addresses the flow on the solid surfaces. The species diffuse along the surface in the adsorbed phase.

We will not consider that last case in our study, since it is assumed to play a minor role here given that our applications are at high temperature.

2) The Maxwell-Stefan based diffusion model

a. Maxwell-Stefan diffusion model

The Stefan-Maxwell governing equation for the diffusion of species r in the z direction is (Martinez, Shimpalee, & Van Zee, 2008)

$$\frac{dy_r}{dz} = \frac{RT}{P}\sum_{s\neq r}\frac{y_r N_{z,s} - y_s N_{z,r}}{\mathfrak{D}^e_{rs}} \qquad \textit{eq. IV.2-1}$$

The binary diffusion coefficient is obtained by the following formula (Fuller, Schettler, & Giddings, 1966):

$$\mathfrak{D}_{rs} = \frac{10^{-3}T^{1.25}\sqrt{(M_r+M_s)/M_r M_s}}{P\left[(\sum v)_r{}^{1/3}+(\sum v)_s{}^{1/3}\right]^2} \qquad \textit{eq. IV.2-2}$$

with $\mathfrak{D}^e_{rs}$ in $cm^2.s^{-1}$, T in K and P in atm.

However, CFD limitations impose that we work with a Fickian form of the diffusivity. Therefore, we will consider a mathematically equivalent form of *eq II.2-1*, were the second term of the right hand side of the equation is equivalent to Fick's first law.

$$N_{z,r} = y_r \sum_s N_{z,s} - \frac{P}{RT}\frac{\varepsilon_s}{\tau} D_{r,m}\frac{dy_r}{dz} \qquad \textit{eq. IV.2-3}$$

The mixture diffusion component $D_{r,m}$ is defined here by

$$D_{r,m} = \frac{N_{z,r} - y_r \sum_s N_{z,s}}{\sum_{s\neq r}\frac{y_s N_r - y_r N_s}{\mathfrak{D}_{rs}}} \qquad \textit{eq. IV.2-4}$$

b. The Maxwell-Stefan based diffusion model

The simplest approach when considering the diffusion in a porous medium is to consider diffusion in a fluid and to add a term that addresses the diffusion in the pores. Therefore, one can consider that two diffusions are taking place simultaneously: the Knudsen Diffusion and the Stefan-Maxwell Diffusion. And thus the effective diffusivity for species *r*, D_r^e, is obtained by:

$$D_r^e = \frac{\varepsilon_s}{\tau} D_r \qquad \textit{eq. IV.2-5}$$

where the straight-pore diffusivity D_r is given by:

$$\frac{1}{D_r} = \frac{1}{D_{r,m}} + \frac{1}{D_{Kr}} \qquad \textit{eq. IV.2-6}$$

$D_{r,m}$ is the mixture diffusivity, and D_{Kr} the Knudsen Diffusivity of species r.

The Knudsen Diffusivity deals with the small pores of the catalyst, where mainly only molecule to wall collisions occur. And as shown by the following formula, it is not pressure dependent:

$$D_{Kr} = 9.70.10^{3}.\bar{r}_p.\sqrt{\frac{T}{M_r}} \qquad \textit{eq. IV.2-7}$$

The mixture diffusivity $D_{r,m}$ is based on the Maxwell-Stefan model (ch IV.2.a). Therefore, we will use equation *IV.2.4* to compute it.

This model has been used in our simulations. Details on calculations and the values of the different parameters are reported in appendix C.

c. The constant fluxes ratio approximation

The aim of this approximation is to simplify the computations. As for now, we want to get an approximation that will enable us to compute the diffusion coefficient, without knowing the values of the molar fluxes. Therefore, we assume that at steady state fluxes of products and reactants are proportionnal.

Let's consider that one can identify a dominating reaction. For instance, let's take the following reaction, where **A**, **B**, **C** and **D** are species, and α, β, γ and δ their respective stoichiometric coefficients.

$$\alpha\ \mathbf{A} + \beta\ \mathbf{B} = \gamma\ \mathbf{C} + \delta\ \mathbf{D}$$

Since this reaction is dominating, one can assume, with a good approximation that the ratios of the fluxes are equal to the ratio of their corresponding stoichiometric coefficients:

$$\frac{N_{z,A}}{N_{z,B}} = \frac{\alpha}{\beta};\quad \frac{N_{z,A}}{N_{z,C}} = \frac{\alpha}{\gamma};\quad \frac{N_{z,A}}{N_{z,D}} = \frac{\alpha}{\delta};\quad \frac{N_{z,B}}{N_{z,C}} = \frac{\beta}{\delta}\quad \ldots \qquad \textit{eq. IV.2-8}$$

If one wants to use to use this approximation with the Maxwell-Stefan's expression of the mixture diffusivity, one should consider the following slightly modified equation of *eq IV.2-4*.

$$D_{r,m} = \frac{1-y_r \sum_s \left(N_{z,s}/N_{z,r}\right)}{\sum_{s\neq r} \frac{y_s - y_r (N_s/N_r)}{\mathfrak{D}_{rs}}} \qquad \textit{eq. IV.2-9}$$

For latter model, this assumption will be dropped, and molar fluxes will be computed. However, this entails a much heavier program, and needs one more convergence on the diffusion coefficients as it will be discussed in chapter V.

d. The Approximated Multi-Component model

i. Theory

Some CFD applications, such as STAR CD developed an alternative formula for the mixture diffusivity $D_{r,m}$ (Martinez, Shimpalee, & Van Zee, 2008).

$$D_{r,m} = \frac{\sum_{s \neq r} y_s M_s}{M \sum_{s \neq r} \frac{y_s}{\mathfrak{D}_{rs}}} \qquad \textit{eq. IV.2-10}$$

However, when applied to Maxwell-Stefan equation (*eq IV.2.3*), this definition of the mixture diffusivity does not satisfy mass conservation. Therefore, a correction term is added to the mass conservation equation and derived as follows.

The general mass balance equation can be written as:

$$\frac{\partial C_r}{\partial t} + \nabla N_r = \frac{r_r}{M_r} \qquad \textit{eq. IV.2-11}$$

Using the Fickian form of Maxwell-Stefan (*eq II.2.5*) and combining it to *eq II.2.13* leads to:

$$\frac{\partial C_r}{\partial t} + \nabla \left[y_r \sum_s N_s - \frac{P}{RT} D_{r,m} \nabla y_r \right] = \frac{r_r}{M_r} \qquad \textit{eq. IV.2-12}$$

Since this equation is not satisfied with *eq II.2.10* definition of the mixture diffusivity, a correction molecular flux, N_c is added to *eq II.2.12*:

$$\frac{\partial C_r}{\partial t} + \nabla \left[y_r \sum_s (N_s + \boldsymbol{N_c}) - \frac{P}{RT} D_{r,m} \nabla y_r \right] = \frac{r_r}{M_r} \qquad \textit{eq. IV.2-13}$$

Hence summing of all species, leads to:

$$\frac{\partial C}{\partial t} + \nabla \left[\sum_s (N_s + N_c) - \frac{P}{RT} \sum_r \left(D_{r,m} \nabla y_r \right) \right] = \sum_r \frac{r_r}{M_r} \qquad \textit{eq. IV.2-14}$$

And since the global mass balance gives us:

$$\frac{\partial C}{\partial t} + \nabla \sum_s N_s = \sum_r \frac{r_r}{M_r} \qquad \textit{eq. IV.2-15}$$

Then

$$N_c = \frac{P}{RT} \sum_r \left(D_{r,m} \nabla y_r \right) \qquad \textit{eq. IV.2-16}$$

ii. Comparison with the Maxwell-Stefan model

Martinez et al. compared this model to the Maxwell-Stefan one. For that, they considered diffusion through the anode and cathode backing layers of a Proton Exchange Membrane Fuel Cell (PEMFC). In that study, they disregarded the heat transfer and all the physical state changes, since none of these are needed to compare both models. Figure 53 shows a schematic of the PEMFC.

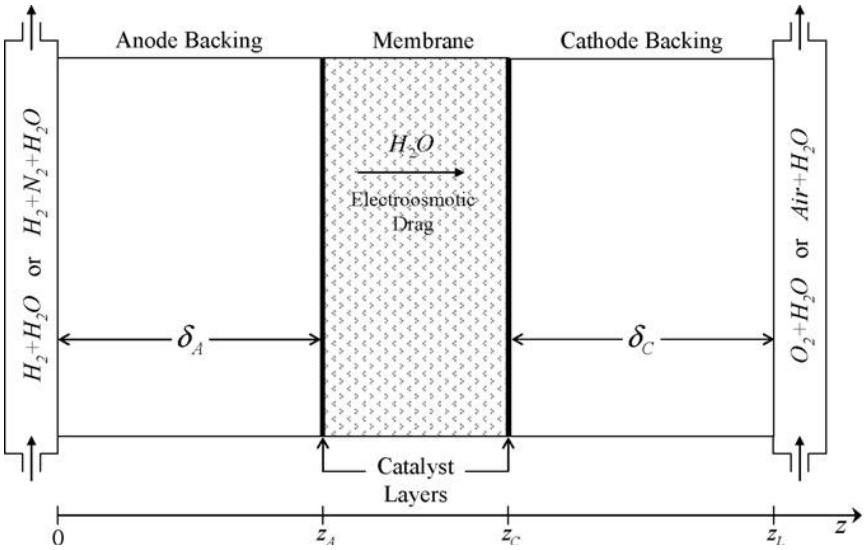

Figure 53- Schematic of a PEMFC (Martinez, Shimpalee, & Van Zee, 2008)

They applied usual running conditions (P = 101 kPa, T = 355 K and i = 0.7 A.cm^{-2}), and plotted the mole fractions according to the different models. The following figure shows the results at the cathode backing layer. Notice that the same results can be seen when considering the anode backing layer.

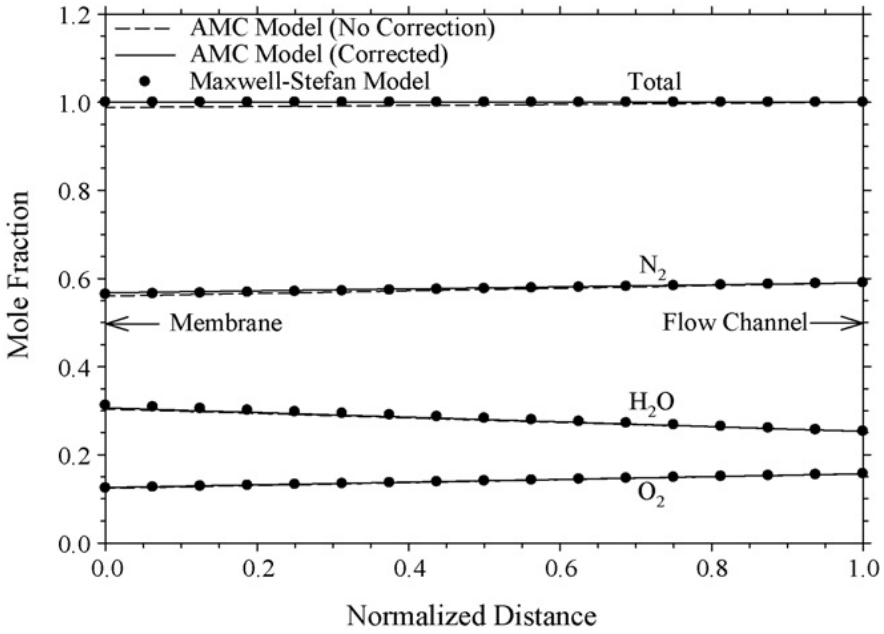

Figure 54 - AMC and Maxwell-Stefan model comparison for low concentration gradients (Martinez, Shimpalee, & Van Zee, 2008)

As we see on Figure 54, the difference between the models is hardly noticeable (less than 2%). However, the difference does get more important towards the membrane, where the concentration gradients are supposed to be the higher.

The following figure considers now an unrealistic case, where all the oxygen would be consumed. This entails a higher concentration difference throughout the backing layers, and thus higher concentration gradients. The running conditions here are 413 K, 304 kPa and i = 22.4 $A.cm^{-2}$.

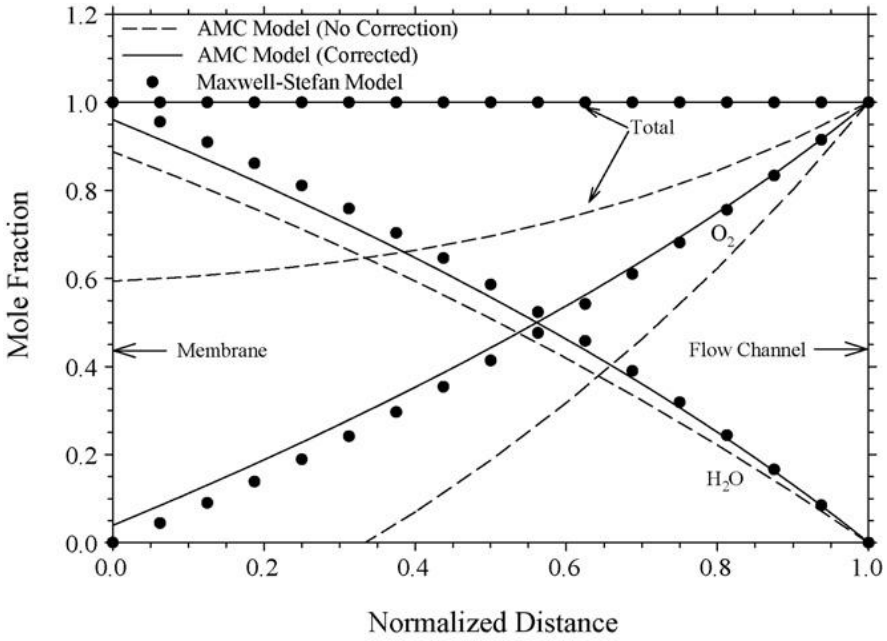

Figure 55 - AMC and Maxwell-Stefan model comparison for high concentration gradients (Martinez, Shimpalee, & Van Zee, 2008)

Figure 55 shows the cathode behavior for these unrealistic conditions. Although the uncorrected AMC model follows the same pattern as the Maxwell-Stefan model, it fails to accurately predict the concentrations. On the other side, the corrected AMC model differs for the water mole fraction at most for only 4% to the Maxwell-Stefan model.

The authors conclude by saying that the "insignificant inaccuracies" of the corrected AMC model are compensated by the gain in simplicity of this model. Thus the computation speed and storage needs will be decrease. This makes this model suitable for CFD computations of a PEMFC.

3) The dusty gas model

The second model we will study here is based on the Dusty gas model. The main idea of such a model is to consider the porous media as one species but without any motion.

The usual form of the Dusty Gas model consist of the three main diffusion regimes discussed previously (*ch. IV.1*): the Knudsen regime, the Ordinary Diffusion and the pressure-driven Viscous Flow. Thus for species *s*:

$$\frac{N_r}{D_{Kr}^e} + \sum_{s\neq r} \frac{y_s N_r - y_r N_s}{\mathfrak{D}_{rs}^e} = -\frac{P}{RT}\nabla y_r - \frac{y_r}{RT}\left(1 + \frac{B_0 P}{\mu D_{Kr}^e}\right)\nabla P \qquad \textit{eq. IV.3-1}$$

If we sum over all species, we obtain the following equation:

$$\sum_r \frac{N_r}{D_{Kr}^e} = -\frac{1}{RT}\left(1 + \frac{B_0 P}{\mu}\sum_r \frac{y_r}{D_{Kr}^e}\right)\nabla P \qquad \textit{eq. IV.3-2}$$

Combining *eq IV.3-1* and *eq IV.3-2* to eliminate the $\frac{B_0 P}{\mu D_{Kr}^e}$ term, gives us:

$$\sum_{s\neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}} = -\frac{P}{RT}\nabla y_r - \frac{y_r}{RT}\left(1 - \frac{1/D_{Kr}^e}{\sum_s y_s / D_{Ks}^e}\right)\nabla P \qquad \textit{eq. IV.3-3}$$

where

$$\frac{1}{\Delta_{rs}} = \frac{1}{\mathfrak{D}_{rs}^e} + \frac{1}{D_{Kr}^e D_{Ks}^e \sum_t y_t / D_{Kt}^e} \qquad \textit{eq. IV.3-4}$$

As previously, in order to be able to apply the Dusty gas model to Fluent, we have to put it in the Fickian form. Therefore we will follow the same approach as Hite & Jackson (1977). Their idea is to get rid of the pressure variable by saying that the pressure variation within the particle is negligible in regards to the overall operating pressure P_0. Hence, the pressure is set as constant, and the pressure gradient term becomes negligible when compared to the mass fraction term in equation IV.2-21. Hence we obtain:

$$\sum_{s\neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}} = -\frac{P_0}{RT}\nabla y_r \qquad \textit{eq. IV.3-5}$$

Using eq. III.2-3 to eliminate the mass fraction gradient in eq. IV.3-5 leads to

$$\sum_{s\neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}} \frac{RT}{P_0} = \frac{N_r - y_s \sum_s N_s}{\frac{P_0}{RT} D_r^e} \qquad \textit{eq. IV.3-6}$$

i.e.

$$\frac{1}{D_r^e} = \frac{\sum_{s\neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}}}{N_r - y_s \sum_s N_s} \qquad \textit{eq. IV.3-7}$$

Here the constant fluxes ratio approximation can be used by putting eq. IV.3-7 under the following form:

$$\frac{1}{D_r^e} = \frac{\sum_{s\neq r} \frac{y_s - y_r {}^{N_s}/_{N_r}}{\Delta_{rs}}}{1 - y_s \sum_s {}^{N_s}/_{N_r}} \qquad \textit{eq. IV.3-7}$$

As shown in chapter IV.2-c, the N_s / N_r term can be approximated as a constant.

4) Limitations of such models

As we have seen so far, all the diffusion models we used assumed that the pressure variation in the pellet is negligible, and as a result that the pressure in the particles is constant. Graham's relation says that for an isobaric diffusion

$$\sum_r N_r \sqrt{M_r} = 0 \qquad \textit{eq. IV.4-1}$$

or

$$\sum_r \operatorname{div} N_r \sqrt{M_r} = 0 \qquad \textit{eq. IV.4-1}$$

Besides a material balance leads to

$$\operatorname{div} N_r = \upsilon_r r \qquad \textit{eq. IV.4-1}$$

Thus

$$\sum_r \upsilon_r \sqrt{M_r} = 0 \qquad \textit{eq. IV.4-1}$$

Hence, according to Hite and Jackson (1977), a pressure gradient will appear if there is a change in the $\upsilon_r \sqrt{M_r}$ quantity for a given reaction. They thus highlighted that a change in moles for a given reaction is not a valid parameter to tell if pressure varies or not within a particle.

Runstedtler (2006) argues that the Graham's relation stops being valid for diffusion in the bulk regime, though no pressure drop appears. To show that he studied gases passing through a cylinder as shown on the following figure:

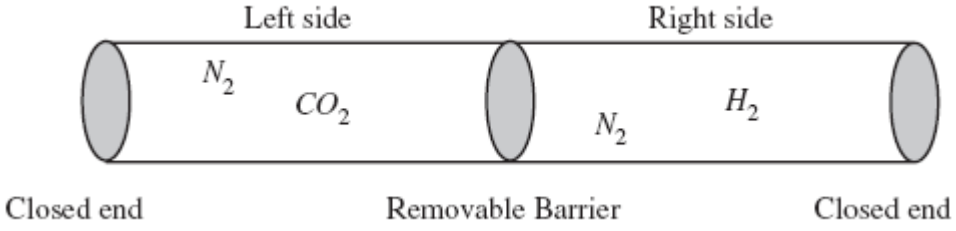

Figure 56 – Multicomponent diffusion in a cylinder (Runstedtler, 2006)

The cylinder is composed of two sections which concentrations are different and set as follows:

Left side	Right side
$x_{H2} = 0$ $x_{N2} = 0.5$ $x_{CO2} = 0.5$	$x_{H2} = 0.5$ $x_{N2} = 0.5$ $x_{CO2} = 0$

Table 2 - Species concentrations for Rundstedtler's experiment

The experiment starts at t = 0, when the barrier is removed. And as one could expect carbon dioxide will diffuse towards the right side and dihydrogen in the other direction. To simulate the bulk or the Knudsen regime, the tube's diameter varies from 10^{-7} to 10^{-4} m.

The following figure (Figure 57) shows the results for the Knudsen regime (10^{-7} m) and for the bulk regime (10^{-4} m). Intermediate tube diameters are also studied by Runstedtler, and are presented in the referred paper.

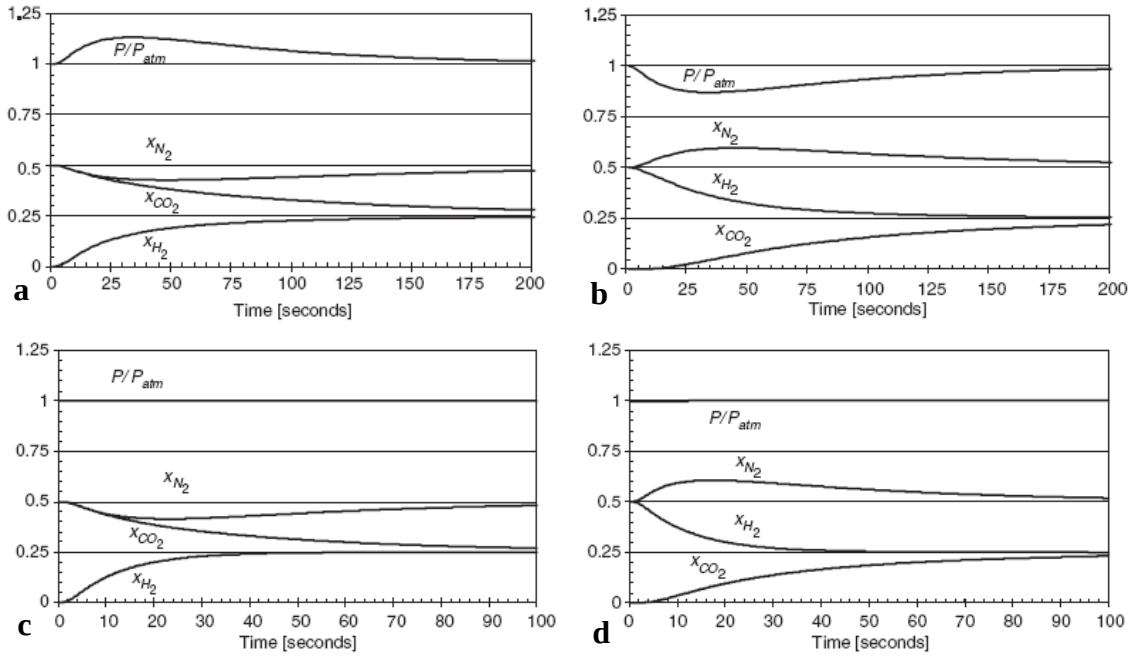

Figure 57 – Concentrations and pressure variation through time(Runstedtler, 2006)
Knudsen regime (10^{-7}m): (a) Left side; (b) Right side
Bulk regime (10^{-4}m): (c) Left side; (d) Right side

One can notice that in the bulk regime, hardly any pressure variation occurs, though diffusion does take place. This means that diffusion taking place in the bulk regime is equimolar, and this contradicts Graham's relation (eq. IV.4-1). Hence, Runstedtler shows that Graham's relation stops being valid for the bulk diffusion limit.

However, one can also notice that for the Knudsen regime, a pressure gradiant does occur, as expected by Graham's relation.

The question raised here is whether or not a pressure gradient occurs in the particle due to reaction and if that gradient is or not negligible. It is indeed tempting to think that new species created through a reaction are likely to be of different size and number than the reactants they are issued from, and as a result form a pressure gradient.

That is why another model has been studied, namely the binary friction model. This model does not assume that pressure variations are negligible. This model is presented in the next chapter.

5) The Binary friction model (BFM)

The binary diffusion model for gases relies on the following equation:

$$\nabla P_r = RT \sum_s \phi_{rs} \frac{P_r N_s - P_s N_r}{P \mathfrak{D}_{rs}} - \frac{RT N_r}{D_{Kr} + \frac{B_0}{K_r}} \qquad \textit{eq. IV.5-1}$$

Notice that the first term on the right side of equation IV.5-1 is really the Stefan-Maxwell diffusion term multiplied by the ϕ_{ij} coefficient. A good agreement with experiments was found for $\phi_{ij} = 1$ (Kerkhof & Geboers, 2005).

The second term on the right side of equation IV.5-1 accounts for wall frictions. The Darcy permeability B_0 is given by:

$$B_0 = \frac{{r_p}^2}{8} \qquad \textit{eq. IV.5-2}$$

and

$$K_r = \frac{\mu_r^0}{\sum_s P_s \varepsilon_{rs}} \qquad \textit{eq. IV.5-3}$$

Finally, the Wilke parameter ε_{ij} is given by (Kerkhof, Geboers, & Ptasinski, 2001):

$$\varepsilon_{rs} = \frac{\left[1 + \sqrt{\mu_r^0/\mu_s^0} + (M_r/M_s)^{1/4}\right]^2}{\sqrt{8(1 + M_r/M_s)}} \qquad \textit{eq. IV.5-4}$$

Besides for straight pores

$$N_r = -\frac{D_r}{RT} \nabla P_r \qquad \textit{eq. IV.5-5}$$

So by eliminating the partial pressure gradient term between equations IV.5-1 and IV.5-5, we obtain

$$\frac{1}{D_r} = \frac{1}{D_{Kr} + \frac{B_0}{K_r}} - \sum_s \phi_{rs} \frac{P_r N_s - P_s N_r}{P N_r \mathfrak{D}_{rs}} \qquad \textit{eq. IV.5-6}$$

These equations (eq. IV.5-1 through eq. IV.5-6) are valid for straight pores. Similarly, and to account for the pore tortuosity, effective diffusion coefficients are considered, and analogous equations can be written:

$$N_r = -\frac{D_r^e}{RT} \nabla P_r \qquad \textit{eq. IV.5-7}$$

$$\frac{1}{D_r^e} = \frac{1}{D_{Kr}^e + \frac{\varepsilon B_0}{\tau K_r}} - \sum_s \phi_{rs} \frac{P_r N_s - P_s N_r}{P N_r \mathfrak{D}_{rs}^e} \qquad \textit{eq. IV.5-8}$$

Note now that all equations are written for the partial pressures of the species, and no longer for their mass fraction. Hence, the UDS in fluent will now be changed to partial pressure. Details on the changes that this implies and how to implement this model in a UDF in Fluent are given in chapter V.2.

V) Diffusion model – Implementation into Fluent

As seen in the previous chapter, diffusion if computed with accuracy requires several parameters, such as temperature and species concentrations. This parameters vary throughout the geometry, and therefore the diffusion coefficient also vary. To a first approximation, one can take average values and calculate the mean diffusion coefficient. This has been done by our group and all the results presented in chapter III are obtained with these mean diffusion coefficients values. The details of the computations are given in appendix C.

In this section of the research, we wanted to increase the accuracy by computing the values of the diffusion coefficient at each point of the geometry. Therefore, a code needed to be created in Fluent that would for each iteration and for each cell compute a new value of the diffusion coefficient.

This has been done for two diffusion models: the dusty gas model (ch. IV.3) and for the binary friction model (ch. IV.5).

1) The dusty gas model

As seen before, here the species mass fractions are stored in some UDS. And since all mass fractions sum to one, the last species (i.e. water) mass fraction can easily be computed from the others, and therefore was not modeled directly in Fluent. Hence, we therefore used four user defined scalars assigned as follows:

UDS 0 ⇔ y_{CH_4}
UDS 1 ⇔ y_{H_2}
UDS 2 ⇔ y_{CO}
UDS 3 ⇔ y_{CO_2}

The diffusion coefficient is given by equation III.3-7, which has been rewritten here for practicle reasons:

$$\frac{1}{D_r^e} = \frac{\sum_{s \neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}}}{N_r - y_s \sum_s N_s} \qquad \textit{eq. IV.3-7}$$

The fluxes N_r and N_s can be accessed through the formula:

$$N_r = -\frac{\rho}{M_r} D_r^e \nabla Y_r \qquad \textit{eq. V.1-1}$$

Here we can notice two things. First that in order to compute the fluxes, a value of the diffusion coefficient is needed. Yet, that is the value we want to compute. Therefore, an initial value is given. A new value is then obtained using that initial value using equation III.3-7. Therefore, we must iterate on the diffusion coefficient in order to get an accurate value.

Note that for the initial value, the values found from the Stefan Maxwell equation at constant fluxes ratio was used. This ensures that the initial values are not too far from the converged values, and thus converge more quickly.

Second, computing the molar fluxes requires the species mass fraction gradient. One can access this variable in Fluent. However, what Fluent really does, is giving the UDS gradient for each of the three space component. Therefore, the diffusion coefficient will be anisotropic, that is to say, the molar flux will take three different values, one for each space direction. We will therefore be left with N_{rx}, N_{ry} and N_{rz}, and compute D_{rx}, D_{ry} and D_{rz},

The last point we need to be aware is that at each iteration, Fluent needs to know the value of the diffusion coefficient from the last iteration in order to compute the new molar flux, and therefore store them. There are two ways to store scalars in fluent. The first one is the user defined scalar (UDS). This is usually used if one wants to apply some diffusion to the scalar. The second is the user defined memory (UDM). On the contrary to UDSs, UDMs do not diffuse. Therefore, the diffusion coefficients are stored in 15 UDMs (three for each species, including water).

In order to simplify the writing of the code and make it more easily readable, some operation have been decomposed and the following three intermediate sums have been added.

$$sum1_i = \frac{\sum_{s \neq r} \frac{y_s N_r - y_r N_s}{\Delta_{rs}}}{N_r - y_s \sum_s N_s} \qquad \textit{eq. V.1-2}$$

$$sum2 = \sum_s N_s \qquad \textit{eq. V.1-3}$$

$$sum3 = \sum_t \frac{y_t}{D_{Kt}^e} \qquad \textit{eq. V.1-4}$$

The following figure describes the overall scheme of the program (Figure 58). The whole program can be found in appendix D. However, I couldn't get this program to work correctly, and a few plausible reasons for this will be detailed in chapter V.3.

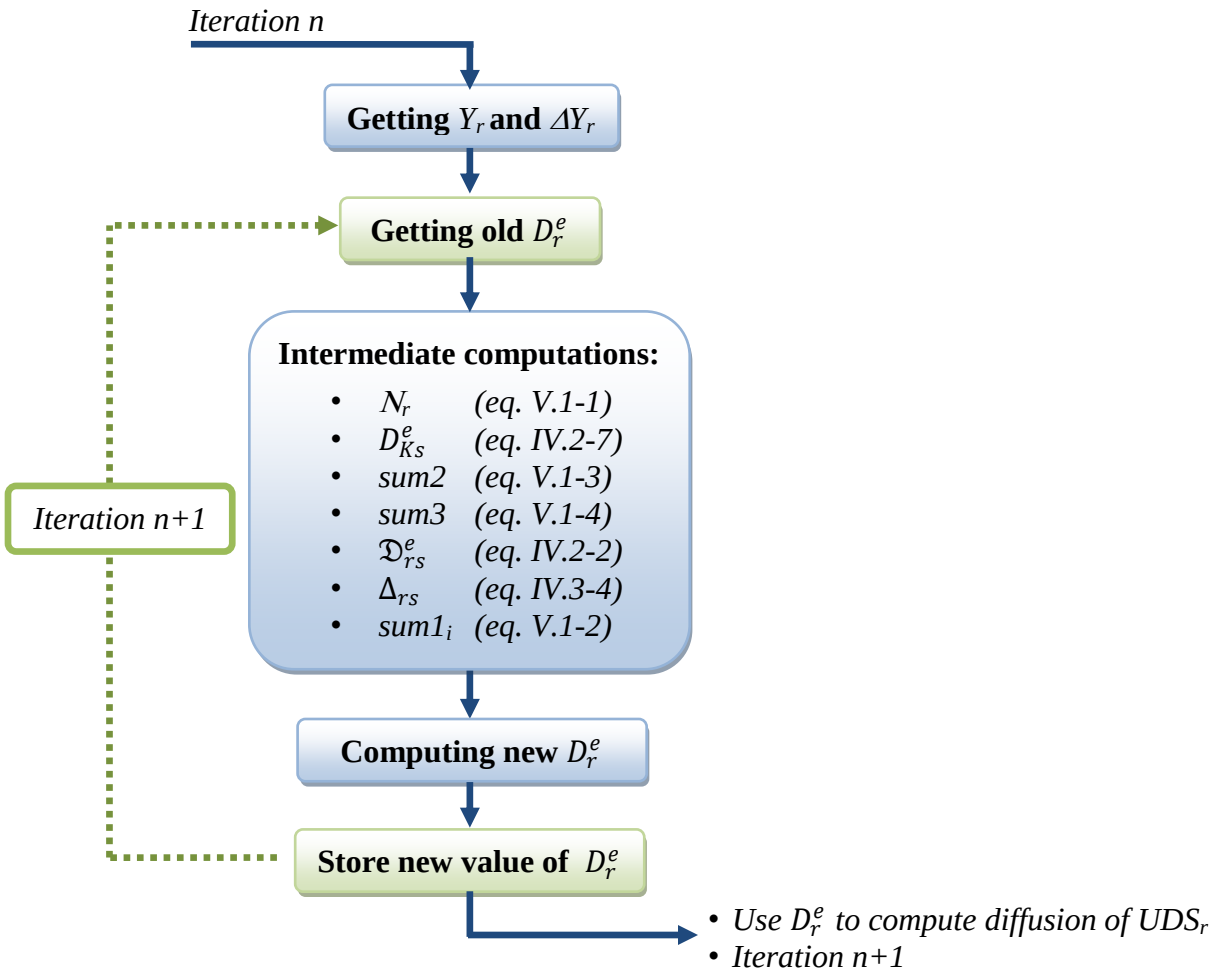

Figure 58 – Scheme of the dusty gas model implemented into Fluent

Note that all the main calculations of this program are done in a `DEFINE_EXECUTE_AT_END` UDF. This means that the change in the value of the diffusion coefficients is the last thing Fluent does for a given iteration. Therefore, the new value of the diffusion coefficient will only be used in the following iteration. This does not change the values of the final converged case, but has been done for practical reasons. Indeed, when Fluent loads the case and data files, it somehow tries to run diffusion before knowing the values of the UDMs. It therefore tries to run the program showed in Figure 58 without being able to access the UDMs values and therefore crashes. By forcing Fluent to run the program at the end, UDMs are accessed before that program runs.

2) The Binary friction model (BFM)

The binary friction model differs from the dusty gas model in that that the UDS now stand for the partial pressure, and since the total pressure in the particle can be

known only by summing all the partial pressure, there is no simple relation that enable Fluent to compute the last species partial pressure from the others. Hence, the BFM model now requires working with five UDSs:

UDS 0 $\Leftrightarrow$ P_{CH_4}
UDS 1 $\Leftrightarrow$ P_{H_2}
UDS 2 $\Leftrightarrow$ P_{CO}
UDS 3 $\Leftrightarrow$ P_{CO_2}
UDS 4 $\Leftrightarrow$ P_{H_2O}

Note that since the definition of the UDSs has changed, the other UDFs have been adapted, and so some slight differences can be noticed in the program shown in appendix E.

$$\frac{1}{D_r^e} = \frac{1}{D_{Kr}^e + \frac{\varepsilon B_0}{\tau K_r}} - \sum_s \phi_{rs} \frac{P_r N_s - P_s N_r}{P N_r \mathfrak{D}_{rs}^e} \qquad \textit{eq. IV.5-8}$$

And the molar flux N_r is now given by

$$N_r = -\frac{D_r^e}{RT} \nabla P_r \qquad \textit{eq. IV.5-7}$$

where value of ϕ_{rs} used is 1.

Here we can once more notice that the value of the UDS gradient is needed, and therefore an anisotropic diffusion is once more considered here (cf. chapter. V.1). Moreover, as for the previous diffusion model, an earlier value of the diffusion coefficient is needed in order to compute the molar fluxes. Therefore, these diffusion coefficient values are stored in 15 UDMs, and initialized with the values obtained from the Stefan Maxwell model.

One intermediate sum is introduced here as follows:

$$sum_r = \sum_s \phi_{rs} \frac{P_r N_s - P_s N_r}{P N_r \mathfrak{D}_{rs}^e} \qquad \textit{eq. V..2-1}$$

Note that since the molar fluxes are vectors (i.e. have three components), sum_r will also have three components.

Figure 59 shows the scheme of the program. Note that this program is very similar to the dusty gas one in the steps followed, but of course equations are different.

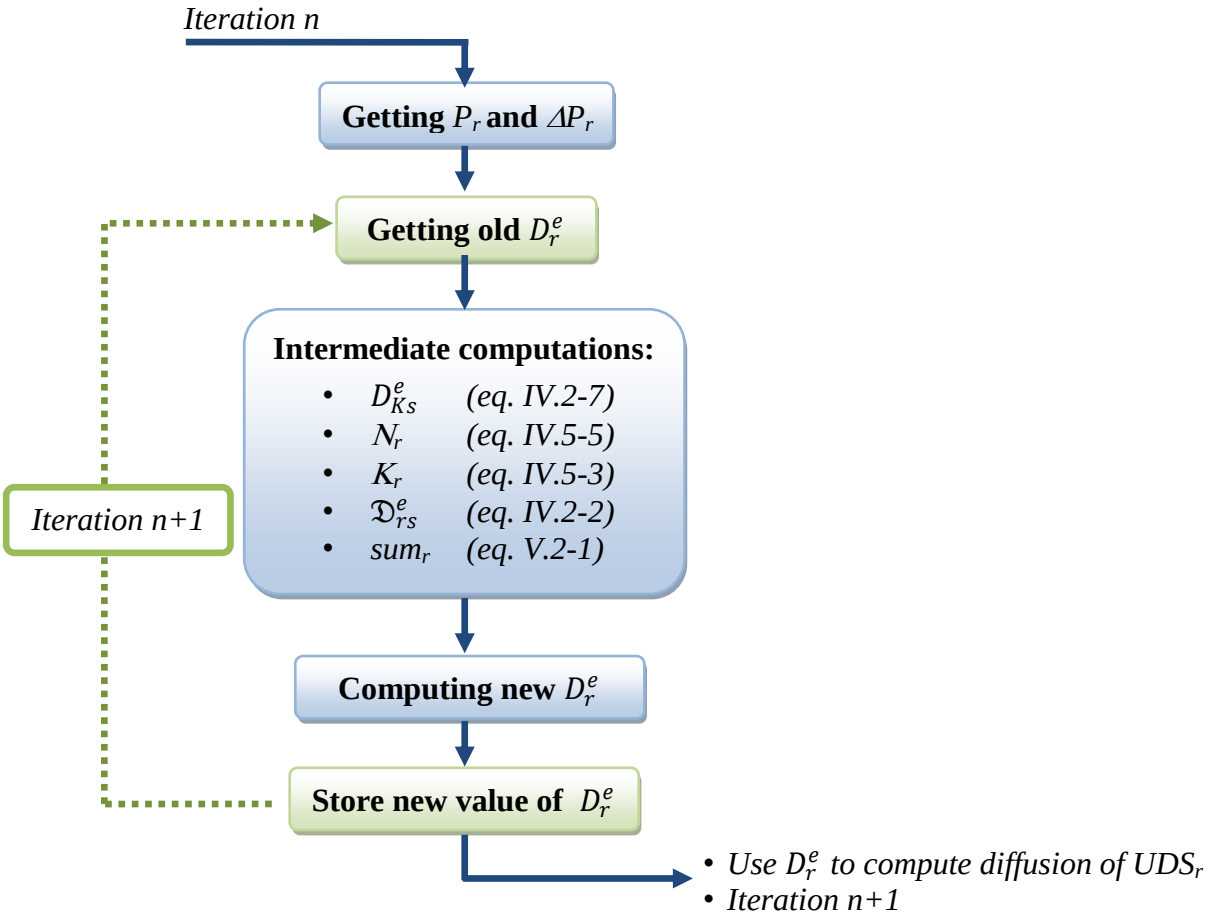

Figure 59 – Scheme of the binary diffusion model implemented into Fluent

Like with the previous program, this one does run but fail to converge. Hence after a couple of iterations, the case eventually starts to diverge and crash.

3) Program issues

The two programs presented here both fail to converge. Although the precise reason for this is unknown, here are a few clues.

First, it turned out that the implementation of the C code in fluent is not perfect, and from time to time, random problem have shown up. For instance, one can notice that, in the diffusion program neither for loops nor matrixes have been used. This is because it clearly appeared that Fluent has sometimes issues when trying to access certain components of a matrix or the pointer of the for loop starts jumping some values with no reason. This eventually made the program crash.

The second issue was the zero values. Indeed one wants to take great care in preventing any forbidden operation such as dividing by zero. This is especially true here for the UDS gradients. Indeed it is very likely that at the symmetry planes, one gradient goes to zero. To prevent this, a verification step has been added to the program that tells Fluent to not change the value of the diffusion coefficient if it sees a zero value in the UDS gradients.

The last issue, which is likely to be the one remaining and causing the programs to crash, is the convergence issue. Indeed we added one more variable that needed to converge, and this can have made the program unstable. A few tricks have therefore been used to ease the convergence.

Before seeing the first one, it is worth noticing that with both models used, the diffusion coefficient is able to become negative. Physically, this means that species are able to diffuse against their own gradients. However, this is a numerical issue since this will entail high changes in the diffusion coefficient. Indeed these latest can jump from one positive value to one negative value, and creates oscillations. These oscillations will likely diffuse through the system, creating some instability. Therefore in the program all diffusion coefficients have been restricted to only positive values. This is a rather drastic solution, which is likely to lower the accuracy of the solution. However, the idea of it is to obtain a converged case and eventually allow the diffusion coefficient to become negative. Another possible solution to take care of that issue is to use artificial viscosity. This is a very common way in numerical methods to smooth out oscillations. However, this requires the use of the UDM gradients, which are not available in Fluent.

Another trick that has proven useful is to add a under-relaxation factor (udf) in order to reduce the change in the UDM values. We thus write:

$$\mathrm{UMD}_{\mathrm{stored}}^{\mathrm{i+1}} := \mathrm{UMD}_{\mathrm{stored}}^{\mathrm{i}} + \left(\mathrm{UMD}_{\mathrm{computed}}^{\mathrm{i+1}} - \mathrm{UMD}_{\mathrm{stored}}^{\mathrm{i}}\right).\mathrm{udf} \qquad \textit{eq. V.3-1}$$

Hence the change in value of the UDM will be slightly decreased. A typical value for the under-relaxation factor is 0.1 to 1.

This helps having a more progressive change in the values, and thus increases the chances of convergence. However, it does also slow down the convergence process, and therefore one does not want to give to that under-relaxation factor a too small value.

VI) Conclusion

This study mainly focused on two aspects of the CFD approach of small N packed bed. The first one was to understand more deeply the mechanisms of diffusion and reaction within the particles and the effect of the flow on these mechanisms. The common approach to model this type of reactor is to assume that temperature is either symmetrical or uniform in the particle. Results have shown that this is clearly not the case, and asymmetries could very clearly be seen in the near wall region.

Moreover, some species depletion could also be seen away from the near wall region. It appeared these zones were all characterized also by a low velocity field. An adiabatic case has been run in order to make sure that wall heat was not responsible for it and since that phenomenon was still observable, we concluded that species depletion was caused by low velocities. Furthermore, if that low velocity zone happens to be in a confined area, the depletion will be even more noticeable. This appeared when we tried to understand why the methane minimum on the surface of the test particle was not located in the near wall region. And it appeared that this minimum was located in a region where flow is squeezed between two particles, and velocities are very low.

We also looked at what was taking place within the particle. And as one could suspect, all the reaction in taking place in the section of the particle near the interface. This is in perfect agreement with the fact that methane steam reforming is a fast reaction, and therefore it is diffusion limited. Therefore, the cores of the particles really act as if they were a closed system, close to the equilibrium state. That is why even if a temperature gradient could be seen within the particles in the near wall region, it the impact on the species concentration was very little.

Finally, we decided to work on a way to improve the diffusion model used. Initially a Stefan-Maxwell based model was used. For this model, we assumed the ratios of the molar fluxes inside the particle to be constant, and therefore the same diffusion coefficient was applied in the entire solid zone. To improve that, the dusty gas model has been considered, and molar fluxes were now on computed. However, this model does assume that the pressure variation within the particles is negligible, a point that has been argued several times in the literature. Therefore, the binary friction model proposed by Kerkhof et al. has been used. This model allows pressure variations. Unfortunately, no results have been obtained from these two models as for now, since some convergence issues appeared.

VII) Recommendations

1) Coke formation

One of the biggest challenges in catalytic reactions is the coke formation. Indeed as we saw previously, layers of carbon often deposit on the surface of the catalyst. This makes it harder for the reactant to access the catalyst, and therefore the rate of reaction decreases. If the reaction is suppose to remove heat (i.e. in the case of an endothermic reaction), temperature will increase. And as we have seen, the lifetime of a tube is cut in half if temperature increases of 20°C.

Moreover, as we saw, low velocities region creates some species concentration variations, and thus coke formation in these areas will happen at a different rate than elsewhere. Hence the size of the coke layer can vary over a same particle. This creates some mechanical stress. And if we had stress caused by the temperature variations, the particle will eventually break, creating dust. This dust will then eventually block some paths and flow will become even less symmetrical.

Further work should therefore add the coke formation reaction as well as the effect of coke on the other reaction rates.

2) Low N tubes modeling

Our results have clearly shown that for low N tube, a great temperature change appears in the near wall region, and some slight methane depletion appears in the reactor. This is not well described by current fixed-bed reactor models (cf chapter I.3).

In the future, improvements over the actual model need to be done. Actual models have two sets of equations, one for the fluid and one for the solid. And these equations are solved one after the other. It does therefore not take into account the interdependence of the flow and diffusion within the particles. Therefore, a possible way improve the results could be to solve the solid and the fluid equations simultaneously.

3) Diffusion model

Two diffusion models have been studied here, namely the dusty gas model and the binary friction model. Unfortunately, I could not get both of these programs to work properly with Fluent. Future work should therefore expand this study.

Once a model works properly, it would be interesting to rerun the cases presented here. One will then be able to compare the results. If the results obtained are not

significantly different, one could then say that the assumptions made to obtain the results presented in chapter III, are good assumptions, and therefore maybe worth applying in order to reduce the computation time as well as to increase the ease of convergence.

It could also be interesting to run a reaction for which the limiting step is reaction and not diffusion. This is the case of the propane dehydrogenation reaction that has also been studied by our group (Taskin, 2007).

Nomenclature

B_0	Darcy permeability (m^2)
d_p	Pore diameter (m)
C	Total concentration of all the species ($mol.m^{-3}$)
C_r	Species *r* concentration ($mol.m^{-3}$)
D_r	Overall diffusivity of species *r* ($m^2.s^{-1}$)
D_r^e	Effective overall diffusivity of species *r* ($m^2.s^{-1}$)
D_{rs}	Binary diffusivity of species *r* ($m^2.s^{-1}$)
D_{rs}^e	Effective binary diffusivity of species *r* ($m^2.s^{-1}$)
D_{Kr}	Knudsen diffusivity of species *r* ($m^2.s^{-1}$)
D_{Kr}^e	Effective Knudsen diffusivity of species *r* ($m^2.s^{-1}$)
i	Current density ($A.m^{-2}$)
k	Turbulent kinetic energy ($m^2.s^{-2}$)
l	Mean free path (m)
M	Mean molecular weight ($g.mol^{-1}$)
M_r	Molecular weight of species *r* ($g.mol^{-1}$)
N	Tube to particle diameter ratio (-)
N_c	Correction molar flux ($mol.m^{-2}.s^{-1}$)
N_r	Molar flux of species *r* ($mol.m^{-2}.s^{-1}$)
P	Pressure (Pa)
P_0	Operating pressure (Pa)
R	Gas constant ($J.K^{-1}.mol^{-1}$)
r	Reaction rate ($mol.m^{-3}.s^{-1}$)
r_r	Reaction rate of species r ($kg.m^{-3}.s^{-1}$)
	Mean pore size (m^{-3})
T	Temperature (K)
u	Velocity ($m.s^{-1}$)
u_x	Velocity in the x direction ($m.s^{-1}$)
u_y	Velocity in the y direction ($m.s^{-1}$)
u_z	Velocity in the z direction ($m.s^{-1}$)
y_s	*r* mass fraction (-)
ε	Turbulent dissipation rate ($m^2.s^{-3}$)
ε_s	Porosity of catalyst particle (m_{void}^3/m_{cat}^3)
ξ	Dimensionless coordinate
ϖ	Specific dissipation rate (s^{-1})
μ	Viscosity (Pa.s)
μ^0	Pure viscosity (Pa.s)
η	Effectiveness factor
ν	Stoechiometric coefficient
η	Effectivness factor (-)
τ	Tortuosity factor (-)
$\langle\ \rangle$	Average value

Bibliography

Amundson, N. R. (1970). Mathematical Models of Fixed Bed Reactors. *Berichte der Bunsen-Gesellschaft Vol. 74 (2)* , 90-98.

Dixon, A. G. (2008). Private communication.

Dixon, A. G., Nijemeisland, M., & Stitt, E. H. (2006). Packed Tubular Reactor Modeling and Catalyst Design using Computational Fluid Dynamics. *Advances in Chemical Engineering, vol. 31* , 307-389.

Dixon, A. G., Taskin, M. E., Stitt, E. H., & Nijemeisland, M. (2007). 3D CFD simulations of steam reforming with resolved intraparticle reaction and gradients. *Chemical Engineering Science 62* , pp. 4963 – 4966.

Fuller, E. N., Schettler, P. D., & Giddings, J. C. (1966). A new method for prediction of binary gas-phase diffusion coefficients. *Industrial and Engineering Chemistry Vol. 58* , 18-27.

Guardo, A., Coussirat, M., Recasens, F., Larrayoz, M., & Escaler, X. (2006). CFD study on particle-to-fluid heat transfer in fixed bed reactors: Convective heat transfer at low and high pressure . *Chemical Engineering Science Vol. 61* , 4341-4353.

Gunjal, P. R., Ranade, V. V., & Chaudhari, R. V. (2005, February). Computational Study of a Single-Phase Flow in Packed Beds of Spheres. *AIChE Journal, Vol. 51, No. 2* , pp. 365-378.

Hite, R., & Jackson, R. (1977). Pressure gradients in porous catalyst pellets in the intermediate diffusion regime. *Chemical Engineering Science Vol. 32* , 703-709.

Hou, K., & Hughes, R. (2001). The kinetics of methane steam reforming ov a Ni/a-Al2O catalyst. *Chemical Engineering Journal 82* , 311-328.

Hydrogen Today and Tomorrow. (n.d.). Retrieved May 2009, from IEA Greenhouse Gas R&D Programme: http://www.ieagreen.org.uk/hydrogen.pdf

Kerkhof, P. J., & Geboers, M. A. (2005). Analysis and ectension of the theory of multicomponent fluid diffusion. *Chemical Engineering Science 60* , 3129-3167.

Kerkhof, P. J., Geboers, M. A., & Ptasinski, K. J. (2001). On the isothermal binary mass transport in a single pore. *Chemical Engineering Journal 83* , 107-121.

Martinez, M. J., Shimpalee, S., & Van Zee, J. (2008). Comparing predictions of PEM fuel cell behavior using Maxwell-Stefan and CFD approximation equations. *Computers and Chemical Engineering* , 2958-2965.

Mason, E., & Malinauskas, A. (1983). *Gas transport in porous media : the dusty-gas model.* Amesterdam, New York: Elsevier.

Rosen, M. A., & Scott, D. S. (1998). Comparative efficiency assessments for a range of hydrogen production processes. *Int. J. Hydrogen Energy, Vol. 23, No. 8* , 653-659.

Runstedtler, A. (2006). On the modified Stefan–Maxwell equation for isothermal multicomponent gaseous diffusion. *Chemical Engineering Science 61* , 5021-5029.

Schwedock, M. J., Windes, L. C., & Ray, W. H. (1989). Steady state and dynamic modelling of a packed bed reactor for the partial oxidation of methanol to formaldehyde II. Eperimental results compared with model predictions. *Chemical Engineering Communications Vol. 78 Issue 1* , 45-71.

Taskin, M. E. (2007). *CFD simulation of transport and reaction in cylindrical catalys particles.* Worcester, MA: Worcester Polytechnic Intsitute.

Xu, J., & Froment, G. F. (1989). Methane Steam Reforming, Methanation and Water-Gas Shift: I. Intrinsic Kinetics. *AIChE Journal Vol.35, No. 1* , 88-96.

Xu, J., & Froment, G. F. (1989). Methane Steam Reforming: II. Diffusional Limitations and Reactor Simulation. *AIChE Journal Vol.35, No. 1* , 97-103.

Appendices A – Boundary layers

The software used to obtain our mesh is GAMBIT. In this appendix, we will detail the boundary layers parameters for each particle and for the reactor wall. But before this, we must agree on the appellation we will give to each particle. The following figure (Figure 60) shows the label we opted for. Note that the same labeling has been kept for the full, 1 hole and 4 holes geometries.

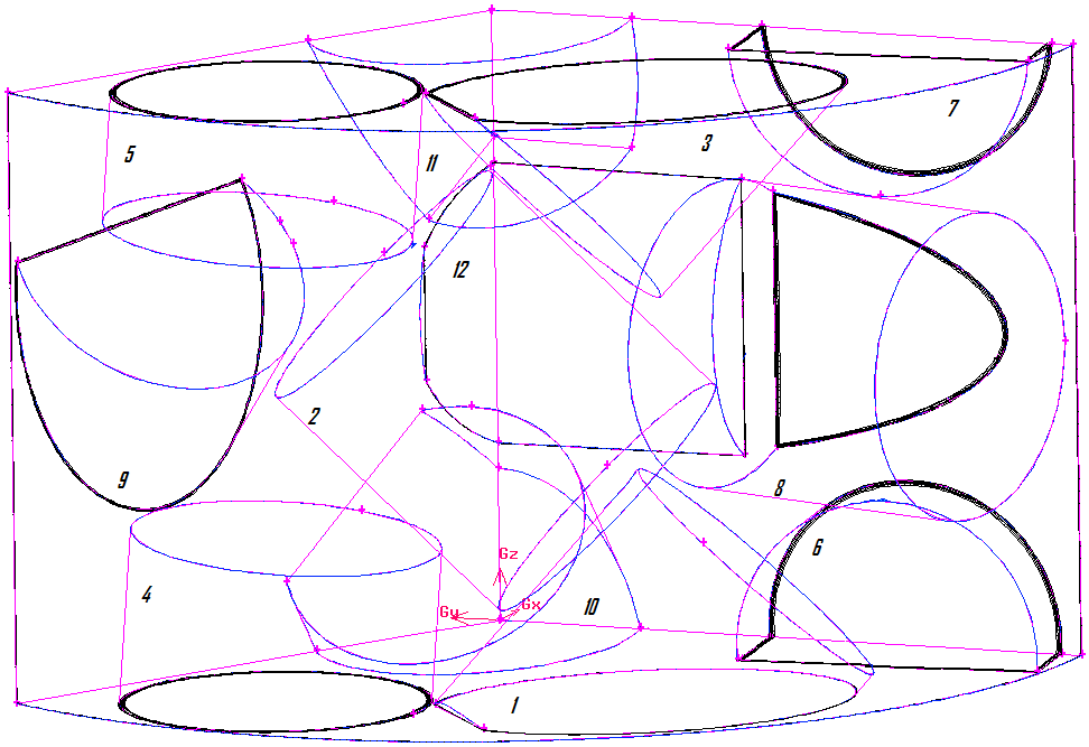

Figure 60 - Particles numbering

When one wants to set boundary layers, one has to consider three parameters:

- The **height of the first layer** (a)
- The **growth factor**, i.e. the ratio of the height of the second layer over the height of the first layer (b/a)
- The **number of layers**

These set of three parameters then give access to the total height of the boundary layers, called **depth** in GAMBIT (D).

Not that the values varies from one particle to the other. The reason is that these numbers have been adjusted in order to get the more boundary layers as possible. However, adding layers and layers sometimes ends getting an overall poor mesh quality.

One can also notice that some boundary layers are missing. The reason is that some particles are too close to each other, and in some narrows zone, adding boundary layers makes it impossible for Gambit to mesh.

The inwards boundary layers refer to the boundary layers inside the solid particle and the outwards ones to the boundary layers attached to the particles but in the fluid zone.

1) Full cylinder

Particles ID	a (inch)	b/a (inch)	Number of layers	Depth (inch)
		Inwards		
1	0.001	1.2	3	0.00364
2	0.001	1.2	3	0.00364
3	0.001	1.2	3	0.00364
4	0.001	1.2	3	0.00364
5	0.001	1.2	3	0.00364
6	0.001	1.0	3	0.00364
7	0.001	1.0	3	0.00364
8	0.001	1.2	2	0.0022
9	0.001	1.2	3	0.00364
10	0.001	1.0	3	0.00364
11	0.001	1.0	3	0.00364
12	0.001	1.0	3	0.00364
		Outwards		
1	0.003	1.2	4	0.016104
2	0.003	1.2	4	0.016104
3	0.003	1.2	4	0.016104
4	0.003	1.2	4	0.016104
5	0.003	1.2	4	0.016104
6	0.003	1.2	4	0.016104
7	0.003	1.2	4	0.016104
8	0.003	1.2	4	0.016104
9	0.003	1.2	4	0.016104
10	0.003	1.2	4	0.016104
11	0.003	1.2	4	0.016104
12	0.003	1.2	4	0.016104
Wall	0.001	1.2	4	0.005368

2) 1 hole cylinder

Particles ID	a (inch)	b/a (inch)	Number of layers	Depth (inch)
		Inwards		
1	-	-	-	-
2	0.003	1.2	4	0.016104
3	0.003	1.2	4	0.016104
4	0.003	1.2	5	0.0223248
5	0.003	1.2	5	0.0223248
6	0.003	1.2	4	0.016104
7	0.003	1.2	4	0.016104
8*	0.003	1.2	4	0.016104
9	0.003	-	1	0.003
10	-	-	-	-
11	-	-	-	-
12	-	-	-	-
		Outwards		
1	0.001	1.0	2	0.002
2	0.001	-	1	0.001
3	0.001	1.0	2	0.002
4	0.001	1.2	3	0.00364
5	0.001	1.2	3	0.00364
6	0.001	1.2	2	0.0022
7	0.001	1.2	2	0.0022
8	0.001	-	1	0.001
9	0.001	1.2	4	0.005368
10	-	-	-	-
11	-	-	-	-
12	0.001	1.0	2	0.002
Wall	0.001	1.1	3	0.00331

** Usually boundary layers are only attached on particles' "natural faces", that is to say they are not attached on faces created by the particle and the periodic or symmetry planes. However, in that particular case, some convergence issue appeared. Therefore a mesh refinement was required and the boundary layer was extended to the intersection between the symmetry plane and particle 8.*

3) 4 holes cylinder

Particles ID	a (inch)	b/a (inch)	Number of layers	Depth (inch)
		Inwards		
1	-	-	-	-
2	0.005	1.0	6	0.03
3	-	-	-	-
4	-	-	-	-
5	-	-	-	-
6	-	-	-	-
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	-
11	-	-	-	-
12	-	-	-	-
		Outwards		
1	0.001	1.0	2	0.002
2	0.001	1.0	3	0.003
3	0.001	1.0	2	0.002
4	0.001	1.0	2	0.002
5	0.001	1.0	2	0.002
6	0.001	-	1	0.001
7	0.001	-	1	0.001
8	0.001	-	1	0.001
9	0.001	1.0	3	0.003
10	-	-	-	-
11	-	-	-	-
12	0.001	1.0	2	0.002
Wall	0.001	1.2	4	0.005368

Appendices B – Running procedure

As we will see later, as set of two runs is needed to obtain the final converge case. However, most of the parameters remain the same from one run to the other. These parameters are the following:

Operating pressure: 2,159,000 Pa
Turbulent model: k-ω STT model
Species diffusion: see appendix B

And for the particles:

Density: 1,947 $kg.m^{-3}$
Specific heat C_p: 1,000 $J.kg^{-}1.K^{-1}$
Thermal conductivity: 1.0 $kg.m^{-3}$

The inlet conditions are

Species	Inlet mass fraction	Partial pressure (Pa)
CH_4	0.1966	424,459.4
H_2	0.0005	1,079.5
CO	0.0007	1,511.3
CO_2	0.1753	378,472.7
H_2O	0.6269	1,353,477

Table 3 - Inlet conditions for UDSs

Notice that the inlet conditions for water is only used with the binary friction model (c.f. chapter V.2). Indeed with that model, partial pressures are required, and since the total pressure is computed as the sum of the partial pressure, there is no easy way to access the fifth partial pressure knowing the four others. Nevertheless, for the other diffusion models, the simple relation $\sum Y_i = 1$ enables us to very easily deduce the fifth mass fraction from the four others, and hence water is not implemented in the model. Notice also that in order to minimize the error due to the numerical methods used, the species that is not implemented in Fluent has to be the one which has the largest mass fraction; hence water has been chosen as the fifth species.

As previously stated, we typically follow two steps. The First one is a flow only run, where the inlet and outlet are linked together. In other words, what flows out at the outlet is what flows in at the inlet. The second case it the real case, where we solve for both flow and reaction (including energy). This case is no longer periodic, and the inlet flow is taken equal to the inlet of the first case. The idea behind such a process is not to have a homogenous velocity at the inlet, which would be unrealistic.

1st case – Periodic flow

The goal of this first run is to get a heterogeneous velocity profile for the inlet. Therefore we impose a periodic flow with a fix overall mass flowrate of0.02677 $kg.s^{-1}$.

The geometry is made periodic by linking the bottom and top faces. For this run only the flow and turbulence equations are solved. The reaction and energy equations are not solve since here we are only interested in taking out the inlet velocity profile, and this profile does not depend on the temperature since all the parameters have been taken independent of temperature.

Once the case is converged, the inlet profile is exported and used in the second case.

2nd case – Non periodic run

For this second case, the same mesh is used as in the first case, but the flow is not made periodic. The following conditions are used:

Inlet velocities:	from profile file of the 1st case
Species sinks & sources:	defined by UDF (c.f. appendices D & E)
Heat sinks:	defined by UDF (c.f. appendices D & E)
Heat flux:	113,300 $W.m^{-2}$

To ease convergence, the first few thousand iterations are done on flow and convergence only. Then reaction and heat transfer are added till the whole case has converged.

Appendices C – Applying the Maxwell-Stefan based model

The following values were taken in order to apply the Maxwell-Stefan based model describes in ch. IV.2.b.

Turtuosity:	$\tau = 3.54$	(Xu & Froment, 1989, p. 99)
Void fraction:	$\varepsilon_s = 0.44$	(Hou & Hughes, 2001)
Mean pore size:	$r_p = 10^3$ Å $= 10^{-5}$ cm	(Hou & Hughes, 2001)
Pressure:	P = 21.3 atm	
Temperature:	T = 834.15 K	

The moles fractions taken are the inlet moles fractions:

$y_{CH_4} = 0.2392$ $\quad y_{CO_2} = 0.0776$
$y_{H_2} = 0.0005$ $\quad y_{H_2O} = 0.6777$
$y_{CO} = 0.0005$

Finally, the following parameters for the binary diffusivity are used:

Species	$(\sum v)_r$	M_r (g.mol^{-1})
CH_4	24.42	16
H_2	7.07	2
CO	18.9	28
CO_2	26.9	44
H_2O	12.7	18

Table 4 - Parameters for the binary diffusivity

In order to use the fluxes approximation (chapter IV.2.c), we need to identify the dominant reaction. For the inlet conditions, the following reaction is the dominant reaction:

$$\mathbf{CH_4 + 2\,H_2O = CO_2 + 4\,H_2}$$

Therefore

$$\frac{N_{H_2}}{N_{CH_4}} = -4; \quad \frac{N_{CO}}{N_{CH_4}} = 0; \quad \frac{N_{CO_2}}{N_{CH_4}} = -1; \quad \frac{N_{H_2O}}{N_{CH_4}} = 2; \quad \ldots$$

After few calculations we obtain the following results:

Species	$D_{r,m}$ (cm^2.s^{-1})	D_r (cm^2.s^{-1})	D_r^e (m^2.s^{-1})
CH_4	0.123	0.104	$1.3.10^{-6}$
H_2	0.225	0.202	$2.5.10^{-6}$
CO	0.072	0.063	$8.0.10^{-7}$
CO_2	0.049	0.044	$5.0.10^{-7}$
H_2O	0.209	0.158	$2.0.10^{-6}$

Appendices D – Dusty gas model code

```
#include "udf.h"
#include "mem.h"

/* Gas constant in kJ/mol.K or m3.kPa/mol.K */
#define rgas 0.0083144
/* Solid density in kg/m3 */
#define rhos 1947.0
/* Adsorption enthalpies and activation energies in kJ/mol */
#define delhco -140.0
#define delhh -93.4
#define delhh2o 15.9
#define E1 209.2
#define E2 15.4
#define E3 109.4
/* Pre-exponential factors for ki (kmol/kg(cat.).s) */
#define A1 5.922e8
#define A2 6.028e-4
#define A3 1.093e3
/* Pre-exponential factors for Ki */
#define AKco 5.127e-13
#define AKh 5.68e-10
#define AKh2o 9.251
/* Heats of reaction in J/kgmol */
#define delHr1 -206100000.0
#define delHr2 41150000.0
#define delHr3 -165000000.0
/* Molecular weights in g.mol-1 */
#define Mco 28.01055
#define Mh2 2.01594
#define Mh2o 18.01534
#define Mch4 16.04303
#define Mco2 44.00995
#define epsilon 0.44
#define tau 3.54

#define alpha 1.
#define urf 0.01
#define diff_limit_upper 1.e-3
#define diff_limit_lower 1.e-15

#define flux_limit_upper 1e-4
#define flux_limit_lower -1e-4

FILE *fout;

/*****************************************************************************/
/*                                    DIFFUSION                              */
/*****************************************************************************/

DEFINE_EXECUTE_AT_END(UDMI_computation)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real Dk_i_0, Dk_i_1, Dk_i_2, Dk_i_3, Dk_i_4;
  real sum1_0x, sum1_1x, sum1_2x, sum1_3x, sum1_4x;
  real sum1_0y, sum1_1y, sum1_2y, sum1_3y, sum1_4y;
  real sum1_0z, sum1_1z, sum1_2z, sum1_3z, sum1_4z;
  real sum2_x, sum2_y, sum2_z;
  real sum3, diff;
```

```
real sum_v_0, sum_v_1, sum_v_2, sum_v_3, sum_v_4;
real Delta_Yi_0x, Delta_Yi_1x, Delta_Yi_2x, Delta_Yi_3x, Delta_Yi_4x;
real Delta_Yi_0y, Delta_Yi_1y, Delta_Yi_2y, Delta_Yi_3y, Delta_Yi_4y;
real Delta_Yi_0z, Delta_Yi_1z, Delta_Yi_2z, Delta_Yi_3z, Delta_Yi_4z;
real Ni_0x, Ni_1x, Ni_2x, Ni_3x, Ni_4x;
real Ni_0y, Ni_1y, Ni_2y, Ni_3y, Ni_4y;
real Ni_0z, Ni_1z, Ni_2z, Ni_3z, Ni_4z;
real Dij_01, Dij_02, Dij_03, Dij_04, Dij_12, Dij_13, Dij_14, Dij_23, Dij_24, Dij_34;
real inv_Delta_01, inv_Delta_02, inv_Delta_03, inv_Delta_04;
real inv_Delta_12, inv_Delta_13, inv_Delta_14;
real inv_Delta_23, inv_Delta_24;
real inv_Delta_34;
real dens;
int zone_ID;
real diff_matrix_0x, diff_matrix_1x, diff_matrix_2x, diff_matrix_3x, diff_matrix_4x;
real diff_matrix_0y, diff_matrix_1y, diff_matrix_2y, diff_matrix_3y, diff_matrix_4y;
real diff_matrix_0z, diff_matrix_1z, diff_matrix_2z, diff_matrix_3z, diff_matrix_4z;
int pb;

Domain *d;
cell_t c;
Thread *t;
int ID;

d = Get_Domain(1);

for (ID = 3;  ID <= 14;  ++ID)
  {
    t = Lookup_Thread(d, ID);
    begin_c_loop(c,t)
      {
        pb = 0;

        p_operating = RP_Get_Real ("operating-pressure");
        cell_temp = C_T(c, t);
        rp = 1e-5;     /* cm */

        Yi_0 = C_UDSI(c, t, 0);
        Yi_1 = C_UDSI(c, t, 1);
        Yi_2 = C_UDSI(c, t, 2);
        Yi_3 = C_UDSI(c, t, 3);
        Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

        sum_v_0 = 24.42;
        sum_v_1 = 7.07;
        sum_v_2 = 18.9;
        sum_v_3 = 26.9;
        sum_v_4 = 12.7;

     /* Density calculation (kg.m-3). */
        MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
        dens = p_operating/rgas/cell_temp*MWav;
        dens = dens*1.0e-06; /* adjust density for wrong value of R (kg.m-3) */

     /* Getting Di's (m2.s-1) values from the previous iteration.  */
        diff_matrix_0x = C_UDMI(c,t,0);
        diff_matrix_0y = C_UDMI(c,t,1);
        diff_matrix_0z = C_UDMI(c,t,2);
        diff_matrix_1x = C_UDMI(c,t,3);
        diff_matrix_1y = C_UDMI(c,t,4);
        diff_matrix_1z = C_UDMI(c,t,5);
        diff_matrix_2x = C_UDMI(c,t,6);
        diff_matrix_2y = C_UDMI(c,t,7);
```

```
         diff_matrix_2z = C_UDMI(c,t,8);
         diff_matrix_3x = C_UDMI(c,t,9);
         diff_matrix_3y = C_UDMI(c,t,10);
         diff_matrix_3z = C_UDMI(c,t,11);
         diff_matrix_4x = C_UDMI(c,t,12);
         diff_matrix_4y = C_UDMI(c,t,13);
         diff_matrix_4z = C_UDMI(c,t,14);

      /* Species gradient (m-1) calculation (Water mass fraction gradient is computed using
the other gradients since the Water mass fraction is unknown by Fluent). */
         Delta_Yi_0x = C_UDSI_G(c, t, 0)[0]; Delta_Yi_0y = C_UDSI_G(c, t, 0)[1]; Delta_Yi_0z =
C_UDSI_G(c, t, 0)[2];
         Delta_Yi_1x = C_UDSI_G(c, t, 1)[0]; Delta_Yi_1y = C_UDSI_G(c, t, 1)[1]; Delta_Yi_1z =
C_UDSI_G(c, t, 1)[2];
         Delta_Yi_2x = C_UDSI_G(c, t, 2)[0]; Delta_Yi_2y = C_UDSI_G(c, t, 2)[1]; Delta_Yi_2z =
C_UDSI_G(c, t, 2)[2];
         Delta_Yi_3x = C_UDSI_G(c, t, 3)[0]; Delta_Yi_3y = C_UDSI_G(c, t, 3)[1]; Delta_Yi_3z =
C_UDSI_G(c, t, 3)[2];
         Delta_Yi_4x = - Delta_Yi_0x - Delta_Yi_1x - Delta_Yi_2x - Delta_Yi_3x;
         Delta_Yi_4y = - Delta_Yi_0y - Delta_Yi_1y - Delta_Yi_2y - Delta_Yi_3y;
         Delta_Yi_4z = - Delta_Yi_0z - Delta_Yi_1z - Delta_Yi_2z - Delta_Yi_3z;

      /* Computation of the effective Knudsen diffusivity (m2.s-1).  */
         Dk_i_0 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mch4,0.5);
         Dk_i_1 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mh2,0.5);
         Dk_i_2 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mco,0.5);
         Dk_i_3 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mco2,0.5);
         Dk_i_4 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mh2o,0.5);

      /* Computation of the species molecular fluxes (mol.m-2.s-1)(with Delta_Yi updated to
the new values). */
         Ni_0x = - dens * diff_matrix_0x * Delta_Yi_0x / Mch4 * 1e3;   Ni_0y = - dens *
diff_matrix_0y * Delta_Yi_0y / Mch4 * 1e3;   Ni_0z = - dens * diff_matrix_0z * Delta_Yi_0z /
Mch4 * 1e3;
         Ni_1x = - dens * diff_matrix_1x * Delta_Yi_1x / Mh2  * 1e3;   Ni_1y = - dens *
diff_matrix_1y * Delta_Yi_1y / Mh2  * 1e3;   Ni_1z = - dens * diff_matrix_1z * Delta_Yi_1z /
Mh2  * 1e3;
         Ni_2x = - dens * diff_matrix_2x * Delta_Yi_2x / Mco  * 1e3;   Ni_2y = - dens *
diff_matrix_2y * Delta_Yi_2y / Mco  * 1e3;   Ni_2z = - dens * diff_matrix_2z * Delta_Yi_2z /
Mco  * 1e3;
         Ni_3x = - dens * diff_matrix_3x * Delta_Yi_3x / Mco2 * 1e3;   Ni_3y = - dens *
diff_matrix_3y * Delta_Yi_3y / Mco2 * 1e3;   Ni_3z = - dens * diff_matrix_3z * Delta_Yi_3z /
Mco2 * 1e3;
         Ni_4x = - dens * diff_matrix_4x * Delta_Yi_4x / Mh2o * 1e3;   Ni_4y = - dens *
diff_matrix_4y * Delta_Yi_4y / Mh2o * 1e3;   Ni_4z = - dens * diff_matrix_4z * Delta_Yi_4z /
Mh2o * 1e3;

      /* Checking for 0 values. */
         if (Ni_0x == 0 || Ni_0x > flux_limit_upper || Ni_0x < flux_limit_lower)
           pb = 1;
         if (Ni_0y == 0 || Ni_0y > flux_limit_upper || Ni_0y < flux_limit_lower)
           pb = 1;
         if (Ni_0z == 0 || Ni_0z > flux_limit_upper || Ni_0z < flux_limit_lower)
           pb = 1;
         if (Ni_1x == 0 || Ni_1x > flux_limit_upper || Ni_1x < flux_limit_lower)
           pb = 1;
         if (Ni_1y == 0 || Ni_1y > flux_limit_upper || Ni_1y < flux_limit_lower)
           pb = 1;
         if (Ni_1z == 0 || Ni_1z > flux_limit_upper || Ni_1z < flux_limit_lower)
           pb = 1;
         if (Ni_2x == 0 || Ni_2x > flux_limit_upper || Ni_2x < flux_limit_lower)
           pb = 1;
         if (Ni_2y == 0 || Ni_2y > flux_limit_upper || Ni_2y < flux_limit_lower)
```

```
            pb = 1;
          if (Ni_2z == 0 || Ni_2z > flux_limit_upper || Ni_2z < flux_limit_lower)
            pb = 1;
          if (Ni_3x == 0 || Ni_3x > flux_limit_upper || Ni_3x < flux_limit_lower)
            pb = 1;
          if (Ni_3y == 0 || Ni_3y > flux_limit_upper || Ni_3y < flux_limit_lower)
            pb = 1;
          if (Ni_3z == 0 || Ni_3z > flux_limit_upper || Ni_3z < flux_limit_lower)
            pb = 1;
          if (Ni_4x == 0 || Ni_4x > flux_limit_upper || Ni_4x < flux_limit_lower)
            pb = 1;
          if (Ni_4y == 0 || Ni_4y > flux_limit_upper || Ni_4y < flux_limit_lower)
            pb = 1;
          if (Ni_4z == 0 || Ni_4z > flux_limit_upper || Ni_4z < flux_limit_lower)
            pb = 1;

          if (pb == 0)
            {
           /* Intermediate sums. */
              sum2_x = Ni_0x + Ni_1x + Ni_2x + Ni_3x + Ni_4x;   /* (mol.m-2.s-1) */
              sum2_y = Ni_0y + Ni_1y + Ni_2y + Ni_3y + Ni_4y;   /* (mol.m-2.s-1) */
              sum2_z = Ni_0z + Ni_1z + Ni_2z + Ni_3z + Ni_4z;   /* (mol.m-2.s-1) */

              sum3 = Yi_0/Dk_i_0 + Yi_1/Dk_i_1 + Yi_2/Dk_i_2 + Yi_3/Dk_i_3 + Yi_4/Dk_i_4;   /*
(s.m-2) */

           /* Dij (m2.s-1) effective computation */
              Dij_01 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mch4+Mh2)
/(Mch4*Mh2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_0,1/3) + pow(sum_v_1,1/3),2));
              Dij_02 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mch4+Mco)
/(Mch4*Mco),0.5)  / ((p_operating/101325) * pow(pow(sum_v_0,1/3) + pow(sum_v_2,1/3),2));
              Dij_03 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) *
pow((Mch4+Mco2)/(Mch4*Mco2),0.5) / ((p_operating/101325) * pow(pow(sum_v_0,1/3) +
pow(sum_v_3,1/3),2));
              Dij_04 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) *
pow((Mch4+Mh2o)/(Mch4*Mh2o),0.5) / ((p_operating/101325) * pow(pow(sum_v_0,1/3) +
pow(sum_v_4,1/3),2));
              Dij_12 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mh2+Mco)
/(Mh2*Mco),0.5)   / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_2,1/3),2));
              Dij_13 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mh2+Mco2)
/(Mh2*Mco2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_3,1/3),2));
              Dij_14 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mh2+Mh2o)
/(Mh2*Mh2o),0.5)  / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_4,1/3),2));
              Dij_23 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mco+Mco2)
/(Mco*Mco2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_2,1/3) + pow(sum_v_3,1/3),2));
              Dij_24 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) * pow((Mco+Mh2o)
/(Mco*Mh2o),0.5)  / ((p_operating/101325) * pow(pow(sum_v_2,1/3) + pow(sum_v_4,1/3),2));
              Dij_34 = epsilon / tau * 1.e-7  * pow(cell_temp,1.75) *
pow((Mco2+Mh2o)/(Mco2*Mh2o),0.5) / ((p_operating/101325) * pow(pow(sum_v_3,1/3) +
pow(sum_v_4,1/3),2));

           /* Computation of the inverse of Delta_rs (s.m-2) */
              inv_Delta_01 = 1./Dij_01 + 1./(Dk_i_0*Dk_i_1*sum3);
              inv_Delta_02 = 1./Dij_02 + 1./(Dk_i_0*Dk_i_2*sum3);
              inv_Delta_03 = 1./Dij_03 + 1./(Dk_i_0*Dk_i_3*sum3);
              inv_Delta_04 = 1./Dij_04 + 1./(Dk_i_0*Dk_i_4*sum3);
              inv_Delta_12 = 1./Dij_12 + 1./(Dk_i_1*Dk_i_2*sum3);
              inv_Delta_13 = 1./Dij_13 + 1./(Dk_i_1*Dk_i_3*sum3);
              inv_Delta_14 = 1./Dij_14 + 1./(Dk_i_1*Dk_i_4*sum3);
              inv_Delta_23 = 1./Dij_23 + 1./(Dk_i_2*Dk_i_3*sum3);
              inv_Delta_24 = 1./Dij_24 + 1./(Dk_i_2*Dk_i_4*sum3);
              inv_Delta_34 = 1./Dij_34 + 1./(Dk_i_3*Dk_i_4*sum3);
```

```
          /* One last intermediate sum (mol.m-4). */
             sum1_0x = inv_Delta_01*(Yi_1*Ni_0x - Yi_0*Ni_1x) + inv_Delta_02*(Yi_2*Ni_0x -
Yi_0*Ni_2x) + inv_Delta_03*(Yi_3*Ni_0x - Yi_0*Ni_3x) + inv_Delta_04*(Yi_4*Ni_0x - Yi_0*Ni_4x);
             sum1_0y = inv_Delta_01*(Yi_1*Ni_0y - Yi_0*Ni_1y) + inv_Delta_02*(Yi_2*Ni_0y -
Yi_0*Ni_2y) + inv_Delta_03*(Yi_3*Ni_0y - Yi_0*Ni_3y) + inv_Delta_04*(Yi_4*Ni_0y - Yi_0*Ni_4y);
             sum1_0z = inv_Delta_01*(Yi_1*Ni_0z - Yi_0*Ni_1z) + inv_Delta_02*(Yi_2*Ni_0z -
Yi_0*Ni_2z) + inv_Delta_03*(Yi_3*Ni_0z - Yi_0*Ni_3z) + inv_Delta_04*(Yi_4*Ni_0z - Yi_0*Ni_4z);

             sum1_1x = inv_Delta_01*(Yi_0*Ni_1x - Yi_1*Ni_0x) + inv_Delta_12*(Yi_2*Ni_1x -
Yi_1*Ni_2x) + inv_Delta_13*(Yi_3*Ni_1x - Yi_1*Ni_3x) + inv_Delta_14*(Yi_4*Ni_1x - Yi_1*Ni_4x);
             sum1_1y = inv_Delta_01*(Yi_0*Ni_1y - Yi_1*Ni_0y) + inv_Delta_12*(Yi_2*Ni_1y -
Yi_1*Ni_2y) + inv_Delta_13*(Yi_3*Ni_1y - Yi_1*Ni_3y) + inv_Delta_14*(Yi_4*Ni_1y - Yi_1*Ni_4y);
             sum1_1z = inv_Delta_01*(Yi_0*Ni_1z - Yi_1*Ni_0z) + inv_Delta_12*(Yi_2*Ni_1z -
Yi_1*Ni_2z) + inv_Delta_13*(Yi_3*Ni_1z - Yi_1*Ni_3z) + inv_Delta_14*(Yi_4*Ni_1z - Yi_1*Ni_4z);

             sum1_2x = inv_Delta_02*(Yi_0*Ni_2x - Yi_2*Ni_0x) + inv_Delta_12*(Yi_1*Ni_2x -
Yi_2*Ni_1x) + inv_Delta_23*(Yi_3*Ni_2x - Yi_2*Ni_3x) + inv_Delta_24*(Yi_4*Ni_2x - Yi_2*Ni_4x);
             sum1_2y = inv_Delta_02*(Yi_0*Ni_2y - Yi_2*Ni_0y) + inv_Delta_12*(Yi_1*Ni_2y -
Yi_2*Ni_1y) + inv_Delta_23*(Yi_3*Ni_2y - Yi_2*Ni_3y) + inv_Delta_24*(Yi_4*Ni_2y - Yi_2*Ni_4y);
             sum1_2z = inv_Delta_02*(Yi_0*Ni_2z - Yi_2*Ni_0z) + inv_Delta_12*(Yi_1*Ni_2z -
Yi_2*Ni_1z) + inv_Delta_23*(Yi_3*Ni_2z - Yi_2*Ni_3z) + inv_Delta_24*(Yi_4*Ni_2z - Yi_2*Ni_4z);

             sum1_3x = inv_Delta_03*(Yi_0*Ni_3x - Yi_3*Ni_0x) + inv_Delta_13*(Yi_1*Ni_3x -
Yi_3*Ni_1x) + inv_Delta_23*(Yi_2*Ni_3x - Yi_3*Ni_2x) + inv_Delta_34*(Yi_4*Ni_3x - Yi_3*Ni_4x);
             sum1_3y = inv_Delta_03*(Yi_0*Ni_3y - Yi_3*Ni_0y) + inv_Delta_13*(Yi_1*Ni_3y -
Yi_3*Ni_1y) + inv_Delta_23*(Yi_2*Ni_3y - Yi_3*Ni_2y) + inv_Delta_34*(Yi_4*Ni_3y - Yi_3*Ni_4y);
             sum1_3z = inv_Delta_03*(Yi_0*Ni_3z - Yi_3*Ni_0z) + inv_Delta_13*(Yi_1*Ni_3z -
Yi_3*Ni_1z) + inv_Delta_23*(Yi_2*Ni_3z - Yi_3*Ni_2z) + inv_Delta_34*(Yi_4*Ni_3z - Yi_3*Ni_4z);

             sum1_4x = inv_Delta_04*(Yi_0*Ni_4x - Yi_4*Ni_0x) + inv_Delta_14*(Yi_1*Ni_4x -
Yi_4*Ni_1x) + inv_Delta_24*(Yi_2*Ni_4x - Yi_4*Ni_2x) + inv_Delta_34*(Yi_3*Ni_4x - Yi_4*Ni_3x);
             sum1_4y = inv_Delta_04*(Yi_0*Ni_4y - Yi_4*Ni_0y) + inv_Delta_14*(Yi_1*Ni_4y -
Yi_4*Ni_1y) + inv_Delta_24*(Yi_2*Ni_4y - Yi_4*Ni_2y) + inv_Delta_34*(Yi_3*Ni_4y - Yi_4*Ni_3y);
             sum1_4z = inv_Delta_04*(Yi_0*Ni_4z - Yi_4*Ni_0z) + inv_Delta_14*(Yi_1*Ni_4z -
Yi_4*Ni_1z) + inv_Delta_24*(Yi_2*Ni_4z - Yi_4*Ni_2z) + inv_Delta_34*(Yi_3*Ni_4z - Yi_4*Ni_3z);

          /* Final calculation of diff_matrix (m2.s-1). */
             diff_matrix_0x = (Ni_0x - Yi_0 * sum2_x) / sum1_0x;
             diff_matrix_0y = (Ni_0y - Yi_0 * sum2_y) / sum1_0y;
             diff_matrix_0z = (Ni_0z - Yi_0 * sum2_z) / sum1_0z;

             diff_matrix_1x = (Ni_1x - Yi_1 * sum2_x) / sum1_1x;
             diff_matrix_1y = (Ni_1y - Yi_1 * sum2_y) / sum1_1y;
             diff_matrix_1z = (Ni_1z - Yi_1 * sum2_z) / sum1_1z;

             diff_matrix_2x = (Ni_2x - Yi_2 * sum2_x) / sum1_2x;
             diff_matrix_2y = (Ni_2y - Yi_2 * sum2_y) / sum1_2y;
             diff_matrix_2z = (Ni_2z - Yi_2 * sum2_z) / sum1_2z;

             diff_matrix_3x = (Ni_3x - Yi_3 * sum2_x) / sum1_3x;
             diff_matrix_3y = (Ni_3y - Yi_3 * sum2_y) / sum1_3y;
             diff_matrix_3z = (Ni_3z - Yi_3 * sum2_z) / sum1_3z;

             diff_matrix_4x = (Ni_4x - Yi_4 * sum2_x) / sum1_4x;
             diff_matrix_4y = (Ni_4y - Yi_4 * sum2_y) / sum1_4y;
             diff_matrix_4z = (Ni_4z - Yi_4 * sum2_z) / sum1_4z;

             C_UDMI(c,t,0) = C_UDMI(c,t,0) + (diff_matrix_0x - C_UDMI(c,t,0)) * urf;
             C_UDMI(c,t,1) = C_UDMI(c,t,1) + (diff_matrix_0y - C_UDMI(c,t,1)) * urf;
             C_UDMI(c,t,2) = C_UDMI(c,t,2) + (diff_matrix_0z - C_UDMI(c,t,2)) * urf;
             C_UDMI(c,t,3) = C_UDMI(c,t,3) + (diff_matrix_1x - C_UDMI(c,t,3)) * urf;
             C_UDMI(c,t,4) = C_UDMI(c,t,4) + (diff_matrix_1y - C_UDMI(c,t,4)) * urf;
             C_UDMI(c,t,5) = C_UDMI(c,t,5) + (diff_matrix_1z - C_UDMI(c,t,5)) * urf;
```

```
        C_UDMI(c,t,6) = C_UDMI(c,t,6) + (diff_matrix_2x - C_UDMI(c,t,6)) * urf;
        C_UDMI(c,t,7) = C_UDMI(c,t,7) + (diff_matrix_2y - C_UDMI(c,t,7)) * urf;
        C_UDMI(c,t,8) = C_UDMI(c,t,8) + (diff_matrix_2z - C_UDMI(c,t,8)) * urf;
        C_UDMI(c,t,9) = C_UDMI(c,t,9) + (diff_matrix_3x - C_UDMI(c,t,9)) * urf;
        C_UDMI(c,t,10) = C_UDMI(c,t,10) + (diff_matrix_3y - C_UDMI(c,t,10)) * urf;
        C_UDMI(c,t,11) = C_UDMI(c,t,11) + (diff_matrix_3z - C_UDMI(c,t,11)) * urf;
        C_UDMI(c,t,12) = C_UDMI(c,t,12) + (diff_matrix_4x - C_UDMI(c,t,12)) * urf;
        C_UDMI(c,t,13) = C_UDMI(c,t,13) + (diff_matrix_4y - C_UDMI(c,t,13)) * urf;
        C_UDMI(c,t,14) = C_UDMI(c,t,14) + (diff_matrix_4z - C_UDMI(c,t,14)) * urf;

if (C_UDMI(c,t,0) > diff_limit_upper)
  C_UDMI(c, t, 0) = diff_limit_upper;
if (C_UDMI(c,t,0) < diff_limit_lower)
  C_UDMI(c, t, 0) = 1.e-10;

if (C_UDMI(c,t,1) > diff_limit_upper)
  C_UDMI(c, t, 1) = diff_limit_upper;
if (C_UDMI(c,t,1) < diff_limit_lower)
  C_UDMI(c, t, 1) = 1.e-10;

if (C_UDMI(c,t,2) > diff_limit_upper)
  C_UDMI(c, t, 2) = diff_limit_upper;
if (C_UDMI(c,t,2) < diff_limit_lower)
  C_UDMI(c, t, 2) = 1.e-10;

if (C_UDMI(c,t,3) > diff_limit_upper)
  C_UDMI(c, t, 3) = diff_limit_upper;
if (C_UDMI(c,t,3) < diff_limit_lower)
  C_UDMI(c, t, 3) = 1.e-10;

if (C_UDMI(c,t,4) > diff_limit_upper)
  C_UDMI(c, t, 4) = diff_limit_upper;
if (C_UDMI(c,t,4) < diff_limit_lower)
  C_UDMI(c, t, 4) = 1.e-10;

if (C_UDMI(c,t,5) > diff_limit_upper)
  C_UDMI(c, t, 5) = diff_limit_upper;
if (C_UDMI(c,t,5) < diff_limit_lower)
  C_UDMI(c, t, 5) = 1.e-10;

if (C_UDMI(c,t,6) > diff_limit_upper)
  C_UDMI(c, t, 6) = diff_limit_upper;
if (C_UDMI(c,t,6) < diff_limit_lower)
  C_UDMI(c, t, 6) = 1.e-10;

if (C_UDMI(c,t,7) > diff_limit_upper)
  C_UDMI(c, t, 7) = diff_limit_upper;
if (C_UDMI(c,t,7) < diff_limit_lower)
  C_UDMI(c, t, 7) = 1.e-10;

if (C_UDMI(c,t,8) > diff_limit_upper)
  C_UDMI(c, t, 8) = diff_limit_upper;
if (C_UDMI(c,t,8) < diff_limit_lower)
  C_UDMI(c, t, 8) = 1.e-10;

if (C_UDMI(c,t,9) > diff_limit_upper)
  C_UDMI(c, t, 9) = diff_limit_upper;
if (C_UDMI(c,t,9) < diff_limit_lower)
  C_UDMI(c, t, 9) = 1.e-10;

if (C_UDMI(c,t,10) > diff_limit_upper)
  C_UDMI(c, t, 10) = diff_limit_upper;
if (C_UDMI(c,t,10) < diff_limit_lower)
```

```
        C_UDMI(c, t, 10) = 1.e-10;

      if (C_UDMI(c,t,11) > diff_limit_upper)
        C_UDMI(c, t, 11) = diff_limit_upper;
      if (C_UDMI(c,t,11) < diff_limit_lower)
        C_UDMI(c, t, 11) = 1.e-10;

      if (C_UDMI(c,t,12) > diff_limit_upper)
        C_UDMI(c, t, 12) = diff_limit_upper;
      if (C_UDMI(c,t,12) < diff_limit_lower)
        C_UDMI(c, t, 12) = 1.e-10;

      if (C_UDMI(c,t,13) > diff_limit_upper)
        C_UDMI(c, t, 13) = diff_limit_upper;
      if (C_UDMI(c,t,13) < diff_limit_lower)
        C_UDMI(c, t, 13) = 1.e-10;

      if (C_UDMI(c,t,14) > diff_limit_upper)
        C_UDMI(c, t, 14) = diff_limit_upper;
      if (C_UDMI(c,t,14) < diff_limit_lower)
        C_UDMI(c, t, 14) = 1.e-10;
          }
       }
     end_c_loop(c,t)
   }
}

DEFINE_DIFFUSIVITY(uds0_diff_x,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 0 in the x direction. */
  diff = C_UDMI(c,t,0) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds0_diff_y,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
```

```
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 0 in the y direction. */
  diff = C_UDMI(c,t,1) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds0_diff_z,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 0 in the z direction. */

  diff = C_UDMI(c,t,2) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds1_diff_x,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
```

```
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 1 in the x direction. */
  diff = C_UDMI(c,t,3) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds1_diff_y,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 1 in the y direction. */
  diff = C_UDMI(c,t,4) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds1_diff_z,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */
```

```
/* Calculation of diff for specy 1 in the z direction. */
  diff = C_UDMI(c,t,4) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds2_diff_x,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 2 in the x direction. */
  diff = C_UDMI(c,t,6) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds2_diff_y,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 2 in the y direction. */
  diff = C_UDMI(c,t,7) * dens * alpha;

  return diff;
}
```

```
DEFINE_DIFFUSIVITY(uds2_diff_z,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 2 in the z direction. */
  diff = C_UDMI(c,t,8) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_x,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 3 in the x direction. */
  diff = C_UDMI(c,t,9) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_y,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;
```

```
  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 3 in the y direction. */
  diff = C_UDMI(c,t,10) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_z,c,t,i)
{
  real rp, MWav, cell_temp, p_operating;
  real Yi_0, Yi_1, Yi_2, Yi_3, Yi_4;
  real diff;
  real dens;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(c, t);
  rp = 1e-5;     /* cm */

  Yi_0 = C_UDSI(c, t, 0);
  Yi_1 = C_UDSI(c, t, 1);
  Yi_2 = C_UDSI(c, t, 2);
  Yi_3 = C_UDSI(c, t, 3);
  Yi_4 = 1. - Yi_0 - Yi_1 - Yi_2 - Yi_3;

/* Density calculation. */
  MWav = 1.0/(Yi_0/Mch4 + Yi_1/Mh2 + Yi_2/Mco + Yi_3/Mco2 + Yi_4/Mh2o);
  dens = p_operating/rgas/cell_temp*MWav;
  dens = dens*1.0e-06; /* adjust density for wrong value of R */

/* Calculation of diff for specy 3 in the z direction. */
  diff = C_UDMI(c,t,11) * dens * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds0_fluid_diff,c,t,i)
{
     return C_R(c,t) * 1.23e-05 + C_MU_T(c,t)/0.7;
}

DEFINE_DIFFUSIVITY(uds1_fluid_diff,c,t,i)
{
     return C_R(c,t) * 2.25e-05 + C_MU_T(c,t)/0.7;
}

DEFINE_DIFFUSIVITY(uds2_fluid_diff,c,t,i)
{
     return C_R(c,t) * 7.2e-06 + C_MU_T(c,t)/0.7;
}
```

```
DEFINE_DIFFUSIVITY(uds3_fluid_diff,c,t,i)
{
     return C_R(c,t) * 4.9e-06 + C_MU_T(c,t)/0.7;
}

/*****************************************************************************************/
/*                                        REACTION                                       */
/*****************************************************************************************/

DEFINE_SOURCE(spe_uds0, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dPch4dYch4, dr1dPch4, dr2dPch4, dr3dPch4;
  real cell_temp, cell_press;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;
  real p_operating;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(cell, thread);
  cell_press = p_operating/1000.0;

  Ych4 = C_UDSI(cell, thread, 0);
  Yh2 = C_UDSI(cell, thread, 1);
  Yco = C_UDSI(cell, thread, 2);
  Yco2 = C_UDSI(cell, thread, 3);
  Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
  MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
  Pch4 = cell_press*Ych4*MWav/Mch4;
  Ph2 = cell_press*Yh2*MWav/Mh2;
  Pco = cell_press*Yco*MWav/Mco;
  Pco2 = cell_press*Yco2*MWav/Mco2;
  Ph2o = cell_press*Yh2o*MWav/Mh2o;

  alph1 = -1.0;
  alph2 = 0.0;
  alph3 = -1.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
```

```
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mch4;

      dPch4dYch4 = cell_press*MWav/Mch4*(1.0-Ych4*MWav/Mch4);

      dr1dPch4 = k1*pow(Ph2o,0.5)/pow(Ph2,1.25)*(1-Prev1/Kp1)/pow(DEN,2.)
             +k1*Pkin1*(Pco*pow(Ph2,3.)/Kp1/pow(Pch4,2.)/Ph2o)/pow(DEN,2.);

      dr2dPch4 = 0;

      dr3dPch4 = k3*Ph2o/pow(Ph2,1.75)*(1-Prev3/Kp3)/pow(DEN,2.)
             +k3*Pkin3*(Pco2*pow(Ph2,4.)/Kp3/pow(Pch4,2.)/pow(Ph2o,2.))/pow(DEN,2.);

      dS[eqn] = rhos*Mch4*(alph1*dr1dPch4+alph2*dr2dPch4+alph3*dr3dPch4)*dPch4dYch4;
    }

  return source;
}

DEFINE_SOURCE(spe_uds1, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dPh2dYh2, dDENdPh2, dr1dPh2, dr2dPh2, dr3dPh2;
  real cell_temp, cell_press;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;
  real p_operating;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(cell, thread);
  cell_press = p_operating/1000.0;

  Ych4 = C_UDSI(cell, thread, 0);
  Yh2 = C_UDSI(cell, thread, 1);
  Yco = C_UDSI(cell, thread, 2);
  Yco2 = C_UDSI(cell, thread, 3);
  Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
  MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
  Pch4 = cell_press*Ych4*MWav/Mch4;
  Ph2 = cell_press*Yh2*MWav/Mh2;
  Pco = cell_press*Yco*MWav/Mco;
  Pco2 = cell_press*Yco2*MWav/Mco2;
  Ph2o = cell_press*Yh2o*MWav/Mh2o;

  alph1 = 3.0;
  alph2 = 1.0;
  alph3 = 4.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
```

```
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mh2;

      dPh2dYh2 = cell_press*MWav/Mh2*(1.0-Yh2*MWav/Mh2);

      dDENdPh2 = 0.5*kh/pow(Ph2,0.5)-kh2o*Ph2o/pow(Ph2,2.);

      dr1dPh2 = k1*(-1.25*Pch4*pow(Ph2o,0.5)/pow(Ph2,2.25))*(1-Prev1/Kp1)/pow(DEN,2.)
             +k1*Pkin1*(-3.0*Pco*pow(Ph2,3.)/Kp1/Pch4/Ph2o)/pow(DEN,2.)
             -2.0*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdPh2;

      dr2dPh2 = k2*(-0.25*Pco*pow(Ph2o,0.5)/pow(Ph2,1.5))*(1-Prev2/Kp2)/pow(DEN,2.)
             +k2*Pkin2*(-1.0*Pco2)/Kp2/Pco/Ph2o/pow(DEN,2.)
             -2.0*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdPh2;

      dr3dPh2 = k3*(-1.75*Pch4*Ph2o)/pow(Ph2,2.75)*(1-Prev3/Kp3)/pow(DEN,2.)
             +k3*Pkin3*(-4.0*Pco2*pow(Ph2,3.))/Kp3/Pch4/pow(Ph2o,2.)/pow(DEN,2.)
             -2.0*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdPh2;

      dS[eqn] = rhos*Mh2*(alph1*dr1dPh2+alph2*dr2dPh2+alph3*dr3dPh2)*dPh2dYh2;
    }

  return source;
}

DEFINE_SOURCE(spe_uds2, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dPcodYco, dDENdPco, dr1dPco, dr2dPco, dr3dPco;
  real cell_temp, cell_press;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;
  real p_operating;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(cell, thread);
  cell_press = p_operating/1000.0;
```

```
Ych4 = C_UDSI(cell, thread, 0);
Yh2 = C_UDSI(cell, thread, 1);
Yco = C_UDSI(cell, thread, 2);
Yco2 = C_UDSI(cell, thread, 3);
Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
Pch4 = cell_press*Ych4*MWav/Mch4;
Ph2 = cell_press*Yh2*MWav/Mh2;
Pco = cell_press*Yco*MWav/Mco;
Pco2 = cell_press*Yco2*MWav/Mco2;
Ph2o = cell_press*Yh2o*MWav/Mh2o;

alph1 = 1.0;
alph2 = -1.0;
alph3 = 0.0;

if (cell_temp <= 550)
    source = dS[eqn] = 0.0;
else
  {
    Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
    Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
    Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
    Prev2 = Pco2*Ph2/Pco/Ph2o;
    Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
    Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

    kco = AKco*exp(-delhco/(rgas*cell_temp));
    kh = AKh*exp(-delhh/(rgas*cell_temp));
    kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

    DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

    Kp1 = 1.198e17*exp(-26830/(cell_temp));
    Kp2 = 1.767e-2*exp(4400/(cell_temp));
    Kp3 = 2.117e15*exp(-22430/(cell_temp));


    k1 = A1*exp(-E1/(rgas*cell_temp));
    r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

    k2 = A2*exp(-E2/(rgas*cell_temp));
    r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

    k3 = A3*exp(-E3/(rgas*cell_temp));
    r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

    source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mco;

    dPcodYco = cell_press*MWav/Mco*(1.0-Yco*MWav/Mco);

    dDENdPco = kco;

    dr1dPco = k1*Pkin1*(-1.0*pow(Ph2,3.)/Kp1/Pch4/Ph2o)/pow(DEN,2.)
           -2.0*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdPco;

    dr2dPco = k2*pow(Ph2o,0.5)/pow(Ph2,0.5)*(1-Prev2/Kp2)/pow(DEN,2.)
           +k2*Pkin2*(Ph2*Pco2)/Kp2/pow(Pco,2.)/Ph2o/pow(DEN,2.)
           -2.0*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdPco;

    dr3dPco = -2.0*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdPco;

    dS[eqn] = rhos*Mco*(alph1*dr1dPco+alph2*dr2dPco+alph3*dr3dPco)*dPcodYco;
```

```
    }

  return source;
}

DEFINE_SOURCE(spe_uds3, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dPco2dYco2, dr1dPco2, dr2dPco2, dr3dPco2;
  real cell_temp, cell_press;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;
  real p_operating;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(cell, thread);
  cell_press = p_operating/1000.0;

  Ych4 = C_UDSI(cell, thread, 0);
  Yh2 = C_UDSI(cell, thread, 1);
  Yco = C_UDSI(cell, thread, 2);
  Yco2 = C_UDSI(cell, thread, 3);
  Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
  MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
  Pch4 = cell_press*Ych4*MWav/Mch4;
  Ph2 = cell_press*Yh2*MWav/Mh2;
  Pco = cell_press*Yco*MWav/Mco;
  Pco2 = cell_press*Yco2*MWav/Mco2;
  Ph2o = cell_press*Yh2o*MWav/Mh2o;

  alph1 = 0.0;
  alph2 = 1.0;
  alph3 = 1.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);
```

```
      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mco2;

      dPco2dYco2 = cell_press*MWav/Mco2*(1.0-Yco2*MWav/Mco2);

      dr1dPco2 = 0;

      dr2dPco2 = k2*Pkin2*(-1.0*Ph2)/Kp2/Pco/Ph2o/pow(DEN,2.);

      dr3dPco2 = k3*Pkin3*(-1.0*pow(Ph2,4.)/Kp3/Pch4/pow(Ph2o,2.))/pow(DEN,2.);

      dS[eqn] = rhos*Mco2*(alph1*dr1dPco2+alph2*dr2dPco2+alph3*dr3dPco2)*dPco2dYco2;
    }

  return source;
}

DEFINE_SOURCE(q_tdep, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dk1dt, dKp1dt, dk2dt, dKp2dt, dk3dt, dKp3dt, dDENdt, dr1dt, dr2dt, dr3dt;
  real cell_temp, cell_press;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real p_operating;

  p_operating = RP_Get_Real ("operating-pressure");
  cell_temp = C_T(cell, thread);
  cell_press = p_operating/1000.0;

  Ych4 = C_UDSI(cell, thread, 0);
  Yh2 = C_UDSI(cell, thread, 1);
  Yco = C_UDSI(cell, thread, 2);
  Yco2 = C_UDSI(cell, thread, 3);
  Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
  MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
  Pch4 = cell_press*Ych4*MWav/Mch4;
  Ph2 = cell_press*Yh2*MWav/Mh2;
  Pco = cell_press*Yco*MWav/Mco;
  Pco2 = cell_press*Yco2*MWav/Mco2;
  Ph2o = cell_press*Yh2o*MWav/Mh2o;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
```

```
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(delHr1*r1+delHr2*r2+delHr3*r3);

      dDENdt = Pco*kco*delhco/rgas/cell_temp/cell_temp
             +pow(Ph2,0.5)*kh*delhh/rgas/cell_temp/cell_temp
             +Ph2o/Ph2*kh2o*delhh2o/rgas/cell_temp/cell_temp;

      dk1dt = k1*E1/rgas/cell_temp/cell_temp;
      dk2dt = k2*E2/rgas/cell_temp/cell_temp;
      dk3dt = k3*E3/rgas/cell_temp/cell_temp;

      dKp1dt = Kp1*26830/cell_temp/cell_temp;
      dKp2dt = Kp2*(-4400)/cell_temp/cell_temp;
      dKp3dt = Kp3*22430/cell_temp/cell_temp;

      dr1dt = dk1dt*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.)
            +k1*Pkin1*(Prev1/Kp1/Kp1)*dKp1dt/pow(DEN,2.)
            -2*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdt;

      dr2dt = dk2dt*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.)
            +k2*Pkin2*(Prev2/Kp2/Kp2)*dKp2dt/pow(DEN,2.)
            -2*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdt;

      dr3dt = dk3dt*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.)
            +k3*Pkin3*(Prev3/Kp3/Kp3)*dKp3dt/pow(DEN,2.)
            -2*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdt;

      dS[eqn] = rhos*(delHr1*dr1dt+delHr2*dr2dt+delHr3*dr3dt);
    }

  return source;
}


DEFINE_ADJUST(Yi_adjust,d)
{
  Thread *t;
  cell_t c;

  thread_loop_c(t,d)
    {
    if(NNULLP(T_STORAGE_R(t,SV_P)))
    /* Test if it is fluid by seeing if pressure is available */
      {
       begin_c_loop(c,t)
         {
          C_YI(c, t, 0) = C_UDSI(c, t, 0);
          C_YI(c, t, 1) = C_UDSI(c, t, 1);
          C_YI(c, t, 2) = C_UDSI(c, t, 2);
          C_YI(c, t, 3) = C_UDSI(c, t, 3);
         }
       end_c_loop(c,t)
```

```
      }
    }
}


/*****************************************************************************************/
/*                                        INTERFACE                                      */
/*****************************************************************************************/

DEFINE_PROFILE(coupled_uds_0, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,0);
    diff1 = C_UDSI_DIFF(c1,tc1,0);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,0) + h1*C_UDSI(c1,tc1,0))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_1, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
```

```
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,1);
    diff1 = C_UDSI_DIFF(c1,tc1,1);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,1) + h1*C_UDSI(c1,tc1,1))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_2, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,2);
    diff1 = C_UDSI_DIFF(c1,tc1,2);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,2) + h1*C_UDSI(c1,tc1,2))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
```

```
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_3, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,3);
    diff1 = C_UDSI_DIFF(c1,tc1,3);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,3) + h1*C_UDSI(c1,tc1,3))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}


/* START OF DEFINE-ON-DEMAND SUBROUTINES
############################################################

##########################################################################################################*
/


DEFINE_ON_DEMAND(heat_sinks_particles)
{
  Domain *d;
  Thread *thread;
  cell_t cell;

  int ID;
  real cell_vol, Totsink, psink, csink;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real cell_temp, cell_press, p_operating;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
```

```
d = Get_Domain(1);

Totsink = 0;

for (ID = 2;  ID <= 2;  ++ID)
{
  thread = Lookup_Thread(d, ID);
  fout = fopen("part_sinks", "w");
  psink = 0;

  begin_c_loop(cell,thread)
  {
    cell_vol = C_VOLUME(cell, thread);
    cell_temp = C_T(cell, thread);
    p_operating = RP_Get_Real ("operating-pressure");
    cell_press = p_operating/1000.0;

    Ych4 = C_UDSI(cell, thread, 0);
    Yh2 = C_UDSI(cell, thread, 1);
    Yco = C_UDSI(cell, thread, 2);
    Yco2 = C_UDSI(cell, thread, 3);
    Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
    MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
    Pch4 = cell_press*Ych4*MWav/Mch4;
    Ph2 = cell_press*Yh2*MWav/Mh2;
    Pco = cell_press*Yco*MWav/Mco;
    Pco2 = cell_press*Yco2*MWav/Mco2;
    Ph2o = cell_press*Yh2o*MWav/Mh2o;

    if (cell_temp <= 550)
      csink = 0.0;
    else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      csink = cell_vol*rhos*(delHr1*r1+delHr2*r2+delHr3*r3);

      psink =  csink + psink;
```

```
        fprintf(fout, "%g %g\n", cell_vol, csink);
      }

    }
    end_c_loop(f,t)

    fprintf(fout, "\n");
    printf("\n");
    printf("Particle %d: Heat sink = %f W \n", ID, psink);
    Totsink = Totsink + psink;
  }

  printf("\n");
  printf("Total heat sink = %f W \n", Totsink);

  fclose(fout);

}


DEFINE_ON_DEMAND(reaction_rates_particles)
{
  Domain *d;
  Thread *thread;
  cell_t cell;

  int ID, uds_i;
  real R1sink, R2sink, R3sink, psink1, psink2, psink3, csink1, csink2, csink3;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real cell_vol, cell_temp, cell_press, p_operating;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;
  real uds_tot[4], uds_part[4], uds_cell[4];

  d = Get_Domain(1);

  R1sink = 0;
  R2sink = 0;
  R3sink = 0;
  for (uds_i = 0;  uds_i <= 3;  ++uds_i)
  {
    uds_tot[uds_i] = 0;
  }

  fout = fopen("part_rxns", "w");

  for (ID = 2;  ID <= 2;  ++ID)
  {
    thread = Lookup_Thread(d, ID);
    psink1 = 0;
    psink2 = 0;
    psink3 = 0;
    for (uds_i = 0;  uds_i <= 3;  ++uds_i)
    {
      uds_part[uds_i] = 0;
    }

    begin_c_loop(cell,thread)
    {
      cell_vol = C_VOLUME(cell, thread);
      cell_temp = C_T(cell, thread);
      p_operating = RP_Get_Real ("operating-pressure");
```

```
cell_press = p_operating/1000.0;

Ych4 = C_UDSI(cell, thread, 0);
Yh2 = C_UDSI(cell, thread, 1);
Yco = C_UDSI(cell, thread, 2);
Yco2 = C_UDSI(cell, thread, 3);
Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
Pch4 = cell_press*Ych4*MWav/Mch4;
Ph2 = cell_press*Yh2*MWav/Mh2;
Pco = cell_press*Yco*MWav/Mco;
Pco2 = cell_press*Yco2*MWav/Mco2;
Ph2o = cell_press*Yh2o*MWav/Mh2o;

if (cell_temp <= 550)
{ csink1 = 0.0;
  csink2 = 0.0;
  csink3 = 0.0;
  for (uds_i = 0;  uds_i <= 3;  ++uds_i)
  {
    uds_cell[uds_i] = 0;
  }
}
else
{
  Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
  Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
  Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
  Prev2 = Pco2*Ph2/Pco/Ph2o;
  Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
  Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

  kco = AKco*exp(-delhco/(rgas*cell_temp));
  kh = AKh*exp(-delhh/(rgas*cell_temp));
  kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

  DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

  Kp1 = 1.198e17*exp(-26830/(cell_temp));
  Kp2 = 1.767e-2*exp(4400/(cell_temp));
  Kp3 = 2.117e15*exp(-22430/(cell_temp));

  k1 = A1*exp(-E1/(rgas*cell_temp));
  r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

  k2 = A2*exp(-E2/(rgas*cell_temp));
  r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

  k3 = A3*exp(-E3/(rgas*cell_temp));
  r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

  csink1 = cell_vol*rhos*r1;
  csink2 = cell_vol*rhos*r2;
  csink3 = cell_vol*rhos*r3;
  uds_cell[0] = cell_vol*rhos*(-r1-r3);
  uds_cell[1] = cell_vol*rhos*(3*r1+r2+4*r3);
  uds_cell[2] = cell_vol*rhos*(r1-r2);
  uds_cell[3] = cell_vol*rhos*(r2+r3);

  psink1 =  csink1 + psink1;
  psink2 =  csink2 + psink2;
  psink3 =  csink3 + psink3;
  for (uds_i = 0;  uds_i <= 3;  ++uds_i)
```

```
        {
          uds_part[uds_i] = uds_part[uds_i] + uds_cell[uds_i];
        }
        fprintf(fout, "%d %g %g %g %g %g %g %g %g %g\n", ID, cell_temp, cell_press, csink1,
csink2, csink3,
                uds_cell[0], uds_cell[1], uds_cell[2], uds_cell[3]);
      }
    }
    end_c_loop(f,t)

    fprintf(fout, "\n");

    printf("\n");
    printf("Particle %d   Reaction 1 (kmol/s): %g\n", ID, psink1);
    printf("Particle %d   Reaction 2 (kmol/s): %g\n", ID, psink2);
    printf("Particle %d   Reaction 3 (kmol/s): %g\n", ID, psink3);
    printf("Particle %d   CH4 consumption (kmol/s): %g\n", ID, uds_part[0]);
    printf("Particle %d   H2 production (kmol/s): %g\n", ID, uds_part[1]);
    printf("Particle %d   CO production (kmol/s): %g\n", ID, uds_part[2]);
    printf("Particle %d   CO2 production (kmol/s): %g\n", ID, uds_part[3]);
    R1sink = R1sink + psink1;
    R2sink = R2sink + psink2;
    R3sink = R3sink + psink3;
    for (uds_i = 0;  uds_i <= 3;  ++uds_i)
    {
      uds_tot[uds_i] = uds_tot[uds_i] + uds_part[uds_i];
    }
  }
  printf("\n");
  printf("Total reaction 1 (kmol/s): %g\n", R1sink);
  printf("Total reaction 2 (kmol/s): %g\n", R2sink);
  printf("Total reaction 3 (kmol/s): %g\n", R3sink);
  printf("Total CH4 consumption (kmol/s): %g\n", uds_tot[0]);
  printf("Total H2 production (kmol/s): %g\n", uds_tot[1]);
  printf("Total CO production (kmol/s): %g\n", uds_tot[2]);
  printf("Total CO2 production (kmol/s): %g\n", uds_tot[3]);

  fclose(fout);

}


DEFINE_ON_DEMAND(solid_species_surface_flow)
{
  Domain *d;
  Thread *t, *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], es0[ND_ND], es1[ND_ND], xf[ND_ND];
  real grad_0[ND_ND], grad_1[ND_ND];
  real uds_0, uds_1, diff0, diff1;
  real uds_face_flow[4], uds_flow[4], uds_flow_tot[4], uds_flux[4], MW[4];
  real pgrad, sgrad, sgrad0, sgrad1, h0, h1;
  real mag, area, ds0, ds1, A_by_es0, A_by_es1;
  int wall_id, uds_i;

  d = Get_Domain(1); /*Get the domain ID*/

  MW[0] = Mch4;
  MW[1] = Mh2;
  MW[2] = Mco;
  MW[3] = Mco2;
```

```
  for (uds_i = 0;  uds_i <= 3;  ++uds_i)
  {
    uds_flow_tot[uds_i] = 0;
  }

  fout = fopen("surface_flows", "w");

  for (wall_id = 16;  wall_id <= 19;  ++wall_id)
  {
    /* Get the thread id of that surface*/
    t = Lookup_Thread(d,wall_id);

    for (uds_i = 0;  uds_i <= 3;  ++uds_i)
    {
      uds_flow[uds_i] = 0;
    }
    area = 0;

    /* Loop over all surface faces*/
    begin_f_loop(f,t)
    {
      F_AREA(A,f,t);      /*Get the area vector*/
      mag = NV_MAG(A);
      c0 = F_C0(f,t);   /*Get the adjacent C0 cell*/
      c1 = F_C1(f,t);   /*Get the adjacent C1 cell*/
      tc0 = THREAD_T0(t);
      tc1 = THREAD_T1(t);
      C_CENTROID(x0,c0,tc0);
      C_CENTROID(x1,c1,tc1);
      F_CENTROID(xf, f,t);
      NV_VV(es0, =, xf,-,x0);
      NV_VV(es1, =, xf,-,x1);
      ds0 = NV_MAG(es0);
      ds1 = NV_MAG(es1);
      NV_S(es0,/=,ds0);
      NV_S(es1,/=,ds1);
      A_by_es0 = NV_DOT(A,A)/NV_DOT(es0,A);
      A_by_es1 = NV_DOT(A,A)/NV_DOT(es1,A);
      for (uds_i = 0;  uds_i <= 3;  ++uds_i)
      {
        /*uds gradients*/
        NV_V(grad_0, =, C_UDSI_G(c0,tc0,uds_i));
        NV_V(grad_1, =, C_UDSI_G(c1,tc1,uds_i));
        /*harmonic mean to allow for discontinuity*/
        diff0 = C_UDSI_DIFF(c0,tc0,uds_i);
        diff1 = C_UDSI_DIFF(c1,tc1,uds_i);
        h0 = diff0/ds0*A_by_es0;
        h1 = -diff1/ds1*A_by_es1;
        uds_0 = C_UDSI(c0,tc0,uds_i);
        uds_1 = C_UDSI(c1,tc1,uds_i);
        /*uds flow through face from primary and secondary gradients*/
        pgrad = (uds_1-uds_0)*h0*h1/(h0+h1);
        sgrad0 = diff0*(NV_DOT(A,grad_0)-A_by_es0*NV_DOT(grad_0,es0));
        sgrad1 = diff1*(NV_DOT(A,grad_1)-A_by_es1*NV_DOT(grad_1,es1));
        sgrad = (sgrad0*h1-sgrad1*h0)/(h0+h1);
        uds_face_flow[uds_i] = (pgrad + sgrad)/MW[uds_i];
        uds_flow[uds_i] = uds_flow[uds_i] + uds_face_flow[uds_i];
      }
      area = area + mag;
      fprintf(fout, "%d %g %g %g %g %g %g %g %g\n", wall_id, xf[0], xf[1], xf[2], mag,
uds_face_flow[0],
              uds_face_flow[1], uds_face_flow[2], uds_face_flow[3]);
```

```
    }
    end_f_loop(f,t)

    for (uds_i = 0;  uds_i <= 3;  ++uds_i)
    {
      uds_flux[uds_i] = uds_flow[uds_i]/area;
      uds_flow_tot[uds_i] = uds_flow_tot[uds_i] + uds_flow[uds_i];
    }
    printf("\n");
    printf("Wall ID: %d    CH4 flux (kmol/m2-s): %g    CH4 flow (kmol/s): %g\n", wall_id,
uds_flux[0], uds_flow[0]);
    printf("Wall ID: %d    H2 flux (kmol/m2-s): %g    H2 flow (kmol/s): %g\n", wall_id,
uds_flux[1], uds_flow[1]);
    printf("Wall ID: %d    CO flux (kmol/m2-s): %g    CO flow (kmol/s): %g\n", wall_id,
uds_flux[2], uds_flow[2]);
    printf("Wall ID: %d    CO2 flux (kmol/m2-s): %g    CO2 flow (kmol/s): %g\n", wall_id,
uds_flux[3], uds_flow[3]);
    }
  printf("\n");
  printf("Total particle CH4 flow (kmol/s): %g\n", uds_flow_tot[0]);
  printf("Total particle H2 flow (kmol/s): %g\n", uds_flow_tot[1]);
  printf("Total particle CO flow (kmol/s): %g\n", uds_flow_tot[2]);
  printf("Total particle CO2 flow (kmol/s): %g\n", uds_flow_tot[3]);
  fclose(fout);
}


DEFINE_ADJUST(reaction_rates,d)
{
  Thread *t;
  cell_t c;

  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real cell_vol, cell_temp, cell_press, p_operating;
  real Ych4, Yh2, Yco, Yco2, Yh2o, MWav, Pch4, Ph2, Pco, Pco2, Ph2o;

  thread_loop_c(t,d)
    {
    if(NULLP(T_STORAGE_R(t,SV_P)))
    /* Test if it is solid by seeing if pressure is not allocated*/
      {
       begin_c_loop(c,t)
         {
          cell_vol = C_VOLUME(c, t);
          cell_temp = C_T(c, t);
          p_operating = RP_Get_Real ("operating-pressure");
          cell_press = p_operating/1000.0;

          Ych4 = C_UDSI(c, t, 0);
           Yh2 = C_UDSI(c, t, 1);
          Yco = C_UDSI(c, t, 2);
           Yco2 = C_UDSI(c, t, 3);
          Yh2o = 1.0-Ych4-Yh2-Yco-Yco2;
          MWav = 1.0/(Ych4/Mch4+Yco/Mco+Yco2/Mco2+Yh2/Mh2+Yh2o/Mh2o);
          Pch4 = cell_press*Ych4*MWav/Mch4;
           Ph2 = cell_press*Yh2*MWav/Mh2;
          Pco = cell_press*Yco*MWav/Mco;
          Pco2 = cell_press*Yco2*MWav/Mco2;
           Ph2o = cell_press*Yh2o*MWav/Mh2o;

          if (cell_temp <= 550)
            {
```

```
            C_UDSI(c, t, 4) = 0.0;
            C_UDSI(c, t, 5) = 0.0;
            C_UDSI(c, t, 6) = 0.0;
          }
        else
          {
            Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
            Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
            Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
            Prev2 = Pco2*Ph2/Pco/Ph2o;
            Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
            Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

             kco = AKco*exp(-delhco/(rgas*cell_temp));
            kh = AKh*exp(-delhh/(rgas*cell_temp));
            kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

             DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

             Kp1 = 1.198e17*exp(-26830/(cell_temp));
            Kp2 = 1.767e-2*exp(4400/(cell_temp));
            Kp3 = 2.117e15*exp(-22430/(cell_temp));

             k1 = A1*exp(-E1/(rgas*cell_temp));
            r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

             k2 = A2*exp(-E2/(rgas*cell_temp));
            r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

             k3 = A3*exp(-E3/(rgas*cell_temp));
            r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

             C_UDSI(c, t, 4) = cell_vol*rhos*r1;
            C_UDSI(c, t, 5) = cell_vol*rhos*r2;
            C_UDSI(c, t, 6) = cell_vol*rhos*r3;
          }
        }
       end_c_loop(c,t)
      }
    }
}

/*****************************************************************************************/
/*                            UDM INITIAL VALUES (Init and On Demand)                    */
/*****************************************************************************************/

DEFINE_ON_DEMAND(On_demand_UDMI_Initialization)
{
  Domain *d;
  cell_t c;
  Thread *t;
  int ID;

  d = Get_Domain(1);

  for (ID = 2;  ID <= 14;  ++ID)
    {
      t = Lookup_Thread(d, ID);
      begin_c_loop(c,t)
        {
          C_UDMI(c, t, 0) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 1) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 2) = 1.1907 * pow(10.,-6.);
```

```
          C_UDMI(c, t, 3) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 4) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 5) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 6) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 7) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 8) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 9) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 10) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 11) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 12) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 13) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 14) = 1.633 * pow(10.,-6.);
        }
      end_c_loop(c,t)
    }
}

DEFINE_INIT(Init_UDMI_Initialization,d)
{
  cell_t c;
  Thread *t;

  /* loop over all cell threads in the domain  */
  thread_loop_c(t,d)
    {

      /* loop over all cells  */
      begin_c_loop_all(c,t)
        {
          C_UDMI(c, t, 0) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 1) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 2) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 3) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 4) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 5) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 6) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 7) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 8) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 9) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 10) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 11) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 12) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 13) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 14) = 1.633 * pow(10.,-6.);
        }
      end_c_loop_all(c,t)
    }
}
```

Appendices E – Binary friction model code

```
#include "udf.h"
#include "mem.h"

/* Gas constant in kJ/mol.K or m3.kPa/mol.K */
#define rgas 0.0083144
/* Solid density in kg/m3 */
#define rhos 19.470
/* Adsorption enthalpies and activation energies in kJ/mol */
#define delhco -140.0
#define delhh -93.4
#define delhh2o 15.9
#define E1 209.2
#define E2 15.4
#define E3 109.4
/* Pre-exponential factors for ki (kmol/kg(cat.).s) */
#define A1 5.922e8
#define A2 6.028e-4
#define A3 1.093e3
/* Pre-exponential factors for Ki */
#define AKco 5.127e-13
#define AKh 5.68e-10
#define AKh2o 9.251
/* Heats of reaction in J/kgmol */
#define delHr1 -206100000.0
#define delHr2 41150000.0
#define delHr3 -165000000.0
/* Molecular weights */
#define Mco 28.01055
#define Mh2 2.01594
#define Mh2o 18.01534
#define Mch4 16.04303
#define Mco2 44.00995
#define epsilon 0.44
#define tau 3.54

#define mu_0 2.41e-5   /* Pa.s */
#define mu_1 1.793e-5  /* Pa.s */
#define mu_2 3.658e-5  /* Pa.s */
#define mu_3 3.587e-5  /* Pa.s */
#define mu_4 3.077e-5  /* Pa.s */

#define epsilon_01 2.9445
#define epsilon_02 0.3490
#define epsilon_03 0.3568
#define epsilon_04 0.3495
#define epsilon_10 0.3507
#define epsilon_12 0.3445
#define epsilon_13 0.3438
#define epsilon_14 0.3504
#define epsilon_20 0.4269
#define epsilon_21 0.4922
#define epsilon_23 0.4015
#define epsilon_24 0.3747
#define epsilon_30 0.4149
#define epsilon_31 2.9309
#define epsilon_32 0.3817
#define epsilon_34 0.3704
#define epsilon_40 0.4972
#define epsilon_41 7.9552
#define epsilon_42 0.4366
```

```
#define epsilon_43 0.4610

#define alpha 1.
#define urf 0.01
#define diff_limit_upper 1.e-5
#define diff_limit_lower 1.e-15

FILE *fout;

/*****************************************************************************************/
/*                                        DIFFUSION                                      */
/*****************************************************************************************/

DEFINE_EXECUTE_AT_END(UDMI_computation)
{
  real rp, MWav, cell_temp, p_operating;
  real Pi_0, Pi_1, Pi_2, Pi_3, Pi_4;
  real Dk_i_0, Dk_i_1, Dk_i_2, Dk_i_3, Dk_i_4;
  real sum0x, sum1x, sum2x, sum3x, sum4x;
  real sum0y, sum1y, sum2y, sum3y, sum4y;
  real sum0z, sum1z, sum2z, sum3z, sum4z;
  real sum2_x, sum2_y, sum2_z;
  real sum3, diff;
  real sum_v_0, sum_v_1, sum_v_2, sum_v_3, sum_v_4;
  real Delta_Pi_0x, Delta_Pi_1x, Delta_Pi_2x, Delta_Pi_3x, Delta_Pi_4x;
  real Delta_Pi_0y, Delta_Pi_1y, Delta_Pi_2y, Delta_Pi_3y, Delta_Pi_4y;
  real Delta_Pi_0z, Delta_Pi_1z, Delta_Pi_2z, Delta_Pi_3z, Delta_Pi_4z;
  real Ni_0x, Ni_1x, Ni_2x, Ni_3x, Ni_4x;
  real Ni_0y, Ni_1y, Ni_2y, Ni_3y, Ni_4y;
  real Ni_0z, Ni_1z, Ni_2z, Ni_3z, Ni_4z;
  real Dij_01, Dij_02, Dij_03, Dij_04, Dij_12, Dij_13, Dij_14, Dij_23, Dij_24, Dij_34;
  real Ki_0, Ki_1, Ki_2, Ki_3, Ki_4;
  real dens;
  int zone_ID;
  real diff_matrix_0x, diff_matrix_1x, diff_matrix_2x, diff_matrix_3x, diff_matrix_4x;
  real diff_matrix_0y, diff_matrix_1y, diff_matrix_2y, diff_matrix_3y, diff_matrix_4y;
  real diff_matrix_0z, diff_matrix_1z, diff_matrix_2z, diff_matrix_3z, diff_matrix_4z;
  int pb;

  Domain *d;
  cell_t c;
  Thread *t;
  int ID;

  d = Get_Domain(1);

  for (ID = 3;  ID <= 14;  ++ID)
    {
      t = Lookup_Thread(d, ID);
      begin_c_loop(c,t)
        {
          pb = 0;

          Pi_0 = C_UDSI(c, t, 0);
          Pi_1 = C_UDSI(c, t, 1);
          Pi_2 = C_UDSI(c, t, 2);
          Pi_3 = C_UDSI(c, t, 3);
          Pi_4 = C_UDSI(c, t, 4);

          p_operating = Pi_0 + Pi_1 + Pi_2 + Pi_3 + Pi_4;     /* Pa */
          cell_temp = C_T(c, t);     /* K */
          rp = 1e-5;     /* cm */
```

```
         sum_v_0 = 24.42;
         sum_v_1 = 7.07;
         sum_v_2 = 18.9;
         sum_v_3 = 26.9;
         sum_v_4 = 12.7;

      /* Getting Di's (m2.s-1) values from the previous iteration.  */
         diff_matrix_0x = C_UDMI(c,t,0);
         diff_matrix_0y = C_UDMI(c,t,1);
         diff_matrix_0z = C_UDMI(c,t,2);
         diff_matrix_1x = C_UDMI(c,t,3);
         diff_matrix_1y = C_UDMI(c,t,4);
         diff_matrix_1z = C_UDMI(c,t,5);
         diff_matrix_2x = C_UDMI(c,t,6);
         diff_matrix_2y = C_UDMI(c,t,7);
         diff_matrix_2z = C_UDMI(c,t,8);
         diff_matrix_3x = C_UDMI(c,t,9);
         diff_matrix_3y = C_UDMI(c,t,10);
         diff_matrix_3z = C_UDMI(c,t,11);
         diff_matrix_4x = C_UDMI(c,t,12);
         diff_matrix_4y = C_UDMI(c,t,13);
         diff_matrix_4z = C_UDMI(c,t,14);

      /* Partial pressure gradient (Pa.m-1) calculation. */
         Delta_Pi_0x = C_UDSI_G(c, t, 0)[0]; Delta_Pi_0y = C_UDSI_G(c, t, 0)[1]; Delta_Pi_0z =
C_UDSI_G(c, t, 0)[2];
         Delta_Pi_1x = C_UDSI_G(c, t, 1)[0]; Delta_Pi_1y = C_UDSI_G(c, t, 1)[1]; Delta_Pi_1z =
C_UDSI_G(c, t, 1)[2];
         Delta_Pi_2x = C_UDSI_G(c, t, 2)[0]; Delta_Pi_2y = C_UDSI_G(c, t, 2)[1]; Delta_Pi_2z =
C_UDSI_G(c, t, 2)[2];
         Delta_Pi_3x = C_UDSI_G(c, t, 3)[0]; Delta_Pi_3y = C_UDSI_G(c, t, 3)[1]; Delta_Pi_3z =
C_UDSI_G(c, t, 3)[2];
         Delta_Pi_4x = C_UDSI_G(c, t, 4)[0]; Delta_Pi_4y = C_UDSI_G(c, t, 4)[1]; Delta_Pi_4z =
C_UDSI_G(c, t, 4)[2];

      /* Computation of the effective Knudsen diffusivity (m2.s-1). */
         Dk_i_0 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mch4,0.5);
         Dk_i_1 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mh2,0.5);
         Dk_i_2 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mco,0.5);
         Dk_i_3 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mco2,0.5);
         Dk_i_4 = 1e-4 * epsilon / tau * 9.70e3 * rp * pow(cell_temp/Mh2o,0.5);

      /* Computation of the species molecular fluxes (mol.m-2.s-1) (with Delta_Pi updated to
the new values, and rgas adjusted to the proper units). */
         Ni_0x = - diff_matrix_0x * Delta_Pi_0x / (1.e3 * rgas * cell_temp);   Ni_0y = -
diff_matrix_0y * Delta_Pi_0y / (1.e3 * rgas * cell_temp);   Ni_0z = - diff_matrix_0z *
Delta_Pi_0z / (1.e3 * rgas * cell_temp);
         Ni_1x = - diff_matrix_1x * Delta_Pi_1x / (1.e3 * rgas * cell_temp);   Ni_1y = -
diff_matrix_1y * Delta_Pi_1y / (1.e3 * rgas * cell_temp);   Ni_1z = - diff_matrix_1z *
Delta_Pi_1z / (1.e3 * rgas * cell_temp);
         Ni_2x = - diff_matrix_2x * Delta_Pi_2x / (1.e3 * rgas * cell_temp);   Ni_2y = -
diff_matrix_2y * Delta_Pi_2y / (1.e3 * rgas * cell_temp);   Ni_2z = - diff_matrix_2z *
Delta_Pi_2z / (1.e3 * rgas * cell_temp);
         Ni_3x = - diff_matrix_3x * Delta_Pi_3x / (1.e3 * rgas * cell_temp);   Ni_3y = -
diff_matrix_3y * Delta_Pi_3y / (1.e3 * rgas * cell_temp);   Ni_3z = - diff_matrix_3z *
Delta_Pi_3z / (1.e3 * rgas * cell_temp);
         Ni_4x = - diff_matrix_4x * Delta_Pi_4x / (1.e3 * rgas * cell_temp);   Ni_4y = -
diff_matrix_4y * Delta_Pi_4y / (1.e3 * rgas * cell_temp);   Ni_4z = - diff_matrix_4z *
Delta_Pi_4z / (1.e3 * rgas * cell_temp);

      /* Checking for 0 values. */
         if (Ni_0x == 0 || Ni_0y == 0 || Ni_0z == 0)
           pb = 1;
```

```
          if (Ni_1x == 0 || Ni_1y == 0 || Ni_1z == 0)
            pb = 1;
          if (Ni_2x == 0 || Ni_2y == 0 || Ni_2z == 0)
            pb = 1;
          if (Ni_3x == 0 || Ni_3y == 0 || Ni_3z == 0)
            pb = 1;
          if (Ni_4x == 0 || Ni_4y == 0 || Ni_4z == 0)
            pb = 1;

          if (pb == 0)
            {
           /* Ki (s) computation */
              Ki_0 = mu_0 / (Pi_0 + Pi_1 * epsilon_01 + Pi_2 * epsilon_02 + Pi_3 * epsilon_03 +
Pi_4 * epsilon_04);
              Ki_1 = mu_1 / (Pi_0 * epsilon_10 + Pi_1 + Pi_2 * epsilon_12 + Pi_3 * epsilon_13 +
Pi_4 * epsilon_14);
              Ki_2 = mu_2 / (Pi_0 * epsilon_20 + Pi_1 * epsilon_21 + Pi_2 + Pi_3 * epsilon_23 +
Pi_4 * epsilon_24);
              Ki_3 = mu_3 / (Pi_0 * epsilon_30 + Pi_1 * epsilon_31 + Pi_2 * epsilon_32 + Pi_3 +
Pi_4 * epsilon_34);
              Ki_4 = mu_4 / (Pi_0 * epsilon_40 + Pi_1 * epsilon_41 + Pi_2 * epsilon_42 + Pi_3 *
epsilon_43 + Pi_4);

           /* Dij (m2.s-1) effective computation */
              Dij_01 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mch4+Mh2)
/(Mch4*Mh2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_0,1/3) + pow(sum_v_1,1/3),2));
              Dij_02 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mch4+Mco)
/(Mch4*Mco),0.5)  / ((p_operating/101325) * pow(pow(sum_v_0,1/3) + pow(sum_v_2,1/3),2));
              Dij_03 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) *
pow((Mch4+Mco2)/(Mch4*Mco2),0.5) / ((p_operating/101325) * pow(pow(sum_v_0,1/3) +
pow(sum_v_3,1/3),2));
              Dij_04 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) *
pow((Mch4+Mh2o)/(Mch4*Mh2o),0.5) / ((p_operating/101325) * pow(pow(sum_v_0,1/3) +
pow(sum_v_4,1/3),2));
              Dij_12 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mh2+Mco)
/(Mh2*Mco),0.5)   / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_2,1/3),2));
              Dij_13 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mh2+Mco2)
/(Mh2*Mco2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_3,1/3),2));
              Dij_14 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mh2+Mh2o)
/(Mh2*Mh2o),0.5)  / ((p_operating/101325) * pow(pow(sum_v_1,1/3) + pow(sum_v_4,1/3),2));
              Dij_23 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mco+Mco2)
/(Mco*Mco2),0.5)  / ((p_operating/101325) * pow(pow(sum_v_2,1/3) + pow(sum_v_3,1/3),2));
              Dij_24 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) * pow((Mco+Mh2o)
/(Mco*Mh2o),0.5)  / ((p_operating/101325) * pow(pow(sum_v_2,1/3) + pow(sum_v_4,1/3),2));
              Dij_34 = epsilon / tau * 1.e-7 * pow(cell_temp,1.75) *
pow((Mco2+Mh2o)/(Mco2*Mh2o),0.5) / ((p_operating/101325) * pow(pow(sum_v_3,1/3) +
pow(sum_v_4,1/3),2));

           /* Intermediate sum calculation (s.m-2). */
                sum0x = 1/(Ni_0x * p_operating) * ((Pi_1 * Ni_0x - Pi_0 * Ni_1x) / Dij_01 +
(Pi_2 * Ni_0x - Pi_0 * Ni_2x) / Dij_02 + (Pi_3 * Ni_0x - Pi_0 * Ni_3x) / Dij_03 + (Pi_4 *
Ni_0x - Pi_0 * Ni_4x) / Dij_04);
              sum0y = 1/(Ni_0y * p_operating) * ((Pi_1 * Ni_0y - Pi_0 * Ni_1y) / Dij_01 +
(Pi_2 * Ni_0y - Pi_0 * Ni_2y) / Dij_02 + (Pi_3 * Ni_0y - Pi_0 * Ni_3y) / Dij_03 + (Pi_4 *
Ni_0y - Pi_0 * Ni_4y) / Dij_04);
              sum0z = 1/(Ni_0z * p_operating) * ((Pi_1 * Ni_0z - Pi_0 * Ni_1z) / Dij_01 +
(Pi_2 * Ni_0z - Pi_0 * Ni_2z) / Dij_02 + (Pi_3 * Ni_0z - Pi_0 * Ni_3z) / Dij_03 + (Pi_4 *
Ni_0z - Pi_0 * Ni_4z) / Dij_04);

              sum1x = 1/(Ni_1x * p_operating) * ((Pi_0 * Ni_1x - Pi_1 * Ni_0x) / Dij_01 +
(Pi_2 * Ni_1x - Pi_1 * Ni_2x) / Dij_12 + (Pi_3 * Ni_1x - Pi_1 * Ni_3x) / Dij_13 + (Pi_4 *
Ni_1x - Pi_1 * Ni_4x) / Dij_14);
```

```
                sum1y = 1/(Ni_1y * p_operating) * ((Pi_0 * Ni_1y - Pi_1 * Ni_0y) / Dij_01 +
(Pi_2 * Ni_1y - Pi_1 * Ni_2y) / Dij_12 + (Pi_3 * Ni_1y - Pi_1 * Ni_3y) / Dij_13 + (Pi_4 *
Ni_1y - Pi_1 * Ni_4y) / Dij_14);
                sum1z = 1/(Ni_1z * p_operating) * ((Pi_0 * Ni_1z - Pi_1 * Ni_0z) / Dij_01 +
(Pi_2 * Ni_1z - Pi_1 * Ni_2z) / Dij_12 + (Pi_3 * Ni_1z - Pi_1 * Ni_3z) / Dij_13 + (Pi_4 *
Ni_1z - Pi_1 * Ni_4z) / Dij_14);

                sum2x = 1/(Ni_2x * p_operating) * ((Pi_0 * Ni_2x - Pi_2 * Ni_0x) / Dij_02 +
(Pi_1 * Ni_2x - Pi_2 * Ni_1x) / Dij_12 + (Pi_3 * Ni_2x - Pi_2 * Ni_3x) / Dij_23 + (Pi_4 *
Ni_2x - Pi_2 * Ni_4x) / Dij_24);
                sum2y = 1/(Ni_2y * p_operating) * ((Pi_0 * Ni_2y - Pi_2 * Ni_0y) / Dij_02 +
(Pi_1 * Ni_2y - Pi_2 * Ni_1y) / Dij_12 + (Pi_3 * Ni_2y - Pi_2 * Ni_3y) / Dij_23 + (Pi_4 *
Ni_2y - Pi_2 * Ni_4y) / Dij_24);
                sum2z = 1/(Ni_2z * p_operating) * ((Pi_0 * Ni_2z - Pi_2 * Ni_0z) / Dij_02 +
(Pi_1 * Ni_2z - Pi_2 * Ni_1z) / Dij_12 + (Pi_3 * Ni_2z - Pi_2 * Ni_3z) / Dij_23 + (Pi_4 *
Ni_2z - Pi_2 * Ni_4z) / Dij_24);

                sum3x = 1/(Ni_3x * p_operating) * ((Pi_0 * Ni_3x - Pi_3 * Ni_0x) / Dij_03 +
(Pi_1 * Ni_3x - Pi_3 * Ni_1x) / Dij_13 + (Pi_2 * Ni_3x - Pi_3 * Ni_2x) / Dij_23 + (Pi_4 *
Ni_3x - Pi_3 * Ni_4x) / Dij_34);
                sum3y = 1/(Ni_3y * p_operating) * ((Pi_0 * Ni_3y - Pi_3 * Ni_0y) / Dij_03 +
(Pi_1 * Ni_3y - Pi_3 * Ni_1y) / Dij_13 + (Pi_2 * Ni_3y - Pi_3 * Ni_2y) / Dij_23 + (Pi_4 *
Ni_3y - Pi_3 * Ni_4y) / Dij_34);
                sum3z = 1/(Ni_3z * p_operating) * ((Pi_0 * Ni_3z - Pi_3 * Ni_0z) / Dij_03 +
(Pi_1 * Ni_3z - Pi_3 * Ni_1z) / Dij_13 + (Pi_2 * Ni_3z - Pi_3 * Ni_2z) / Dij_23 + (Pi_4 *
Ni_3z - Pi_3 * Ni_4z) / Dij_34);

                sum4x = 1/(Ni_4x * p_operating) * ((Pi_0 * Ni_4x - Pi_4 * Ni_0x) / Dij_04 +
(Pi_1 * Ni_4x - Pi_4 * Ni_1x) / Dij_14 + (Pi_2 * Ni_4x - Pi_4 * Ni_2x) / Dij_24 + (Pi_3 *
Ni_4x - Pi_4 * Ni_3x) / Dij_34);
                sum4y = 1/(Ni_4y * p_operating) * ((Pi_0 * Ni_4y - Pi_4 * Ni_0y) / Dij_04 +
(Pi_1 * Ni_4y - Pi_4 * Ni_1y) / Dij_14 + (Pi_2 * Ni_4y - Pi_4 * Ni_2y) / Dij_24 + (Pi_3 *
Ni_4y - Pi_4 * Ni_3y) / Dij_34);
                sum4z = 1/(Ni_4z * p_operating) * ((Pi_0 * Ni_4z - Pi_4 * Ni_0z) / Dij_04 +
(Pi_1 * Ni_4z - Pi_4 * Ni_1z) / Dij_14 + (Pi_2 * Ni_4z - Pi_4 * Ni_2z) / Dij_24 + (Pi_3 *
Ni_4z - Pi_4 * Ni_3z) / Dij_34);

          /* Final calculation of diff_matrix (m2.s-1). */
               diff_matrix_0x = 1 / ( 1/(Dk_i_0 + rp*rp*epsilon*1.e-4/(8*Ki_0*tau)) + sum0x);
               diff_matrix_0y = 1 / ( 1/(Dk_i_0 + rp*rp*epsilon*1.e-4/(8*Ki_0*tau)) + sum0y);
               diff_matrix_0z = 1 / ( 1/(Dk_i_0 + rp*rp*epsilon*1.e-4/(8*Ki_0*tau)) + sum0z);

               diff_matrix_1x = 1 / ( 1/(Dk_i_1 + rp*rp*epsilon*1.e-4/(8*Ki_1*tau)) + sum1x);
               diff_matrix_1y = 1 / ( 1/(Dk_i_1 + rp*rp*epsilon*1.e-4/(8*Ki_1*tau)) + sum1y);
               diff_matrix_1z = 1 / ( 1/(Dk_i_1 + rp*rp*epsilon*1.e-4/(8*Ki_1*tau)) + sum1z);

               diff_matrix_2x = 1 / ( 1/(Dk_i_2 + rp*rp*epsilon*1.e-4/(8*Ki_2*tau)) + sum2x);
               diff_matrix_2y = 1 / ( 1/(Dk_i_2 + rp*rp*epsilon*1.e-4/(8*Ki_2*tau)) + sum2y);
               diff_matrix_2z = 1 / ( 1/(Dk_i_2 + rp*rp*epsilon*1.e-4/(8*Ki_2*tau)) + sum2z);

               diff_matrix_3x = 1 / ( 1/(Dk_i_3 + rp*rp*epsilon*1.e-4/(8*Ki_3*tau)) + sum3x);
               diff_matrix_3y = 1 / ( 1/(Dk_i_3 + rp*rp*epsilon*1.e-4/(8*Ki_3*tau)) + sum3y);
               diff_matrix_3z = 1 / ( 1/(Dk_i_3 + rp*rp*epsilon*1.e-4/(8*Ki_3*tau)) + sum3z);

               diff_matrix_4x = 1 / ( 1/(Dk_i_4 + rp*rp*epsilon*1.e-4/(8*Ki_4*tau)) + sum4x);
               diff_matrix_4y = 1 / ( 1/(Dk_i_4 + rp*rp*epsilon*1.e-4/(8*Ki_4*tau)) + sum4y);
               diff_matrix_4z = 1 / ( 1/(Dk_i_4 + rp*rp*epsilon*1.e-4/(8*Ki_4*tau)) + sum4z);


          C_UDMI(c,t,0) = C_UDMI(c,t,0) + (diff_matrix_0x - C_UDMI(c,t,0)) * urf;
          C_UDMI(c,t,1) = C_UDMI(c,t,1) + (diff_matrix_0y - C_UDMI(c,t,1)) * urf;
          C_UDMI(c,t,2) = C_UDMI(c,t,2) + (diff_matrix_0z - C_UDMI(c,t,2)) * urf;
          C_UDMI(c,t,3) = C_UDMI(c,t,3) + (diff_matrix_1x - C_UDMI(c,t,3)) * urf;
```

```
 C_UDMI(c,t,4) = C_UDMI(c,t,4) + (diff_matrix_1y - C_UDMI(c,t,4)) * urf;
 C_UDMI(c,t,5) = C_UDMI(c,t,5) + (diff_matrix_1z - C_UDMI(c,t,5)) * urf;
 C_UDMI(c,t,6) = C_UDMI(c,t,6) + (diff_matrix_2x - C_UDMI(c,t,6)) * urf;
 C_UDMI(c,t,7) = C_UDMI(c,t,7) + (diff_matrix_2y - C_UDMI(c,t,7)) * urf;
 C_UDMI(c,t,8) = C_UDMI(c,t,8) + (diff_matrix_2z - C_UDMI(c,t,8)) * urf;
 C_UDMI(c,t,9) = C_UDMI(c,t,9) + (diff_matrix_3x - C_UDMI(c,t,9)) * urf;
 C_UDMI(c,t,10) = C_UDMI(c,t,10) + (diff_matrix_3y - C_UDMI(c,t,10)) * urf;
 C_UDMI(c,t,11) = C_UDMI(c,t,11) + (diff_matrix_3z - C_UDMI(c,t,11)) * urf;
 C_UDMI(c,t,12) = C_UDMI(c,t,12) + (diff_matrix_4x - C_UDMI(c,t,12)) * urf;
 C_UDMI(c,t,13) = C_UDMI(c,t,13) + (diff_matrix_4y - C_UDMI(c,t,13)) * urf;
 C_UDMI(c,t,14) = C_UDMI(c,t,14) + (diff_matrix_4z - C_UDMI(c,t,14)) * urf;

if (C_UDMI(c,t,0) > diff_limit_upper)
  C_UDMI(c, t, 0) = diff_limit_upper;
if (C_UDMI(c,t,0) < diff_limit_lower)
  C_UDMI(c, t, 0) = 1.e-10;

if (C_UDMI(c,t,1) > diff_limit_upper)
  C_UDMI(c, t, 1) = diff_limit_upper;
if (C_UDMI(c,t,1) < diff_limit_lower)
  C_UDMI(c, t, 1) = 1.e-10;

if (C_UDMI(c,t,2) > diff_limit_upper)
  C_UDMI(c, t, 2) = diff_limit_upper;
if (C_UDMI(c,t,2) < diff_limit_lower)
  C_UDMI(c, t, 2) = 1.e-10;

if (C_UDMI(c,t,3) > diff_limit_upper)
  C_UDMI(c, t, 3) = diff_limit_upper;
if (C_UDMI(c,t,3) < diff_limit_lower)
  C_UDMI(c, t, 3) = 1.e-10;

if (C_UDMI(c,t,4) > diff_limit_upper)
  C_UDMI(c, t, 4) = diff_limit_upper;
if (C_UDMI(c,t,4) < diff_limit_lower)
  C_UDMI(c, t, 4) = 1.e-10;

if (C_UDMI(c,t,5) > diff_limit_upper)
  C_UDMI(c, t, 5) = diff_limit_upper;
if (C_UDMI(c,t,5) < diff_limit_lower)
  C_UDMI(c, t, 5) = 1.e-10;

if (C_UDMI(c,t,6) > diff_limit_upper)
  C_UDMI(c, t, 6) = diff_limit_upper;
if (C_UDMI(c,t,6) < diff_limit_lower)
  C_UDMI(c, t, 6) = 1.e-10;

if (C_UDMI(c,t,7) > diff_limit_upper)
  C_UDMI(c, t, 7) = diff_limit_upper;
if (C_UDMI(c,t,7) < diff_limit_lower)
  C_UDMI(c, t, 7) = 1.e-10;

if (C_UDMI(c,t,8) > diff_limit_upper)
  C_UDMI(c, t, 8) = diff_limit_upper;
if (C_UDMI(c,t,8) < diff_limit_lower)
  C_UDMI(c, t, 8) = 1.e-10;

if (C_UDMI(c,t,9) > diff_limit_upper)
  C_UDMI(c, t, 9) = diff_limit_upper;
if (C_UDMI(c,t,9) < diff_limit_lower)
  C_UDMI(c, t, 9) = 1.e-10;

if (C_UDMI(c,t,10) > diff_limit_upper)
```

```
              C_UDMI(c, t, 10) = diff_limit_upper;
            if (C_UDMI(c,t,10) < diff_limit_lower)
              C_UDMI(c, t, 10) = 1.e-10;

            if (C_UDMI(c,t,11) > diff_limit_upper)
              C_UDMI(c, t, 11) = diff_limit_upper;
            if (C_UDMI(c,t,11) < diff_limit_lower)
              C_UDMI(c, t, 11) = 1.e-10;

            if (C_UDMI(c,t,12) > diff_limit_upper)
              C_UDMI(c, t, 12) = diff_limit_upper;
            if (C_UDMI(c,t,12) < diff_limit_lower)
              C_UDMI(c, t, 12) = 1.e-10;

            if (C_UDMI(c,t,13) > diff_limit_upper)
              C_UDMI(c, t, 13) = diff_limit_upper;
            if (C_UDMI(c,t,13) < diff_limit_lower)
              C_UDMI(c, t, 13) = 1.e-10;

            if (C_UDMI(c,t,14) > diff_limit_upper)
              C_UDMI(c, t, 14) = diff_limit_upper;
            if (C_UDMI(c,t,14) < diff_limit_lower)
              C_UDMI(c, t, 14) = 1.e-10;

          }
        }
      end_c_loop(c,t)
    }
}

DEFINE_DIFFUSIVITY(uds0_diff_x,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 0 in the x direction. */
  diff = C_UDMI(c,t,0) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds0_diff_y,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 0 in the y direction. */
```

```
  diff = C_UDMI(c,t,1) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds0_diff_z,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 0 in the z direction. */
  diff = C_UDMI(c,t,2) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds1_diff_x,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 1 in the x direction. */
  diff = C_UDMI(c,t,3) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds1_diff_y,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 1 in the y direction. */
  diff = C_UDMI(c,t,4) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}
```

```
DEFINE_DIFFUSIVITY(uds1_diff_z,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 1 in the z direction. */
  diff = C_UDMI(c,t,5) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds2_diff_x,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 2 in the x direction. */
  diff = C_UDMI(c,t,6) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds2_diff_y,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;               /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 2 in the y direction. */
  diff = C_UDMI(c,t,7) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds2_diff_z,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;
```

```
  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 2 in the z direction. */
  diff = C_UDMI(c,t,8) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_x,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 3 in the x direction. */
  diff = C_UDMI(c,t,9) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_y,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 3 in the y direction. */
  diff = C_UDMI(c,t,10) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds3_diff_z,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);
```

```
/* Calculation of diff for specy 3 in the z direction. */
  diff = C_UDMI(c,t,11) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds4_diff_x,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 4 in the x direction. */
  diff = C_UDMI(c,t,12) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}

DEFINE_DIFFUSIVITY(uds4_diff_y,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 4 in the y direction. */
  diff = C_UDMI(c,t,13) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}
DEFINE_DIFFUSIVITY(uds4_diff_z,c,t,i)
{
  real cell_temp;
  real diff;
  real Ptot, Mav;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;
  Mav = 1.e-3 * Mav;                /* putting Mav in kg.mol-1. */
  cell_temp = C_T(c, t);

/* Calculation of diff for specy 4 in the z direction. */
  diff = C_UDMI(c,t,14) / (rgas * cell_temp) * Mav * alpha;

  return diff;
}
```

```
DEFINE_DIFFUSIVITY(uds0_fluid_diff,c,t,i)
{
  real cell_temp;
  real Mav, Ptot;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

  cell_temp = C_T(c, t);
  return (C_R(c,t) * 1.23e-05 + C_MU_T(c,t)/0.7) * Mav/(C_R(c,t) * rgas * cell_temp);
}

DEFINE_DIFFUSIVITY(uds1_fluid_diff,c,t,i)
{
  real cell_temp;
  real Mav, Ptot;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

  cell_temp = C_T(c, t);
  return (C_R(c,t) * 2.25e-05 + C_MU_T(c,t)/0.7) * Mav/(C_R(c,t) * rgas * cell_temp);
}

DEFINE_DIFFUSIVITY(uds2_fluid_diff,c,t,i)
{
  real cell_temp;
  real Mav, Ptot;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

  cell_temp = C_T(c, t);
  return (C_R(c,t) * 7.2e-06 + C_MU_T(c,t)/0.7) * Mav/(C_R(c,t) * rgas * cell_temp);
}

DEFINE_DIFFUSIVITY(uds3_fluid_diff,c,t,i)
{
  real cell_temp;
  real Mav, Ptot;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

  cell_temp = C_T(c, t);
  return (C_R(c,t) * 4.9e-06 + C_MU_T(c,t)/0.7) * Mav/(C_R(c,t) * rgas * cell_temp);
}

DEFINE_DIFFUSIVITY(uds4_fluid_diff,c,t,i)
{
  real cell_temp;
  real Mav, Ptot;

  Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) + C_UDSI(c, t,
4);
```

```
  Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c, t,
3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

  cell_temp = C_T(c, t);
  return (C_R(c,t) * 2.09e-05 + C_MU_T(c,t)/0.7) * Mav/(C_R(c,t) * rgas * cell_temp);
}

/*****************************************************************************************/
/*                                        REACTION                                       */
/*****************************************************************************************/

DEFINE_SOURCE(spe_uds0, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dr1dPch4, dr2dPch4, dr3dPch4;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
  Pco = C_UDSI(cell, thread, 2);
  Pco2 = C_UDSI(cell, thread, 3);
  Ph2o = C_UDSI(cell, thread, 4);

  alph1 = -1.0;
  alph2 = 0.0;
  alph3 = -1.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);
```

```
      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mch4;

      dr1dPch4 = k1*pow(Ph2o,0.5)/pow(Ph2,1.25)*(1-Prev1/Kp1)/pow(DEN,2.)
             +k1*Pkin1*(Pco*pow(Ph2,3.)/Kp1/pow(Pch4,2.)/Ph2o)/pow(DEN,2.);

      dr2dPch4 = 0;

      dr3dPch4 = k3*Ph2o/pow(Ph2,1.75)*(1-Prev3/Kp3)/pow(DEN,2.)
             +k3*Pkin3*(Pco2*pow(Ph2,4.)/Kp3/pow(Pch4,2.)/pow(Ph2o,2.))/pow(DEN,2.);

      dS[eqn] = rhos*Mch4*(alph1*dr1dPch4+alph2*dr2dPch4+alph3*dr3dPch4);
    }

  return source;
}

DEFINE_SOURCE(spe_uds1, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dDENdPh2, dr1dPh2, dr2dPh2, dr3dPh2;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
  Pco = C_UDSI(cell, thread, 2);
  Pco2 = C_UDSI(cell, thread, 3);
  Ph2o = C_UDSI(cell, thread, 4);

  alph1 = 3.0;
  alph2 = 1.0;
  alph3 = 4.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
```

```
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mh2;

      dDENdPh2 = 0.5*kh/pow(Ph2,0.5)-kh2o*Ph2o/pow(Ph2,2.);

      dr1dPh2 = k1*(-1.25*Pch4*pow(Ph2o,0.5)/pow(Ph2,2.25))*(1-Prev1/Kp1)/pow(DEN,2.)
             +k1*Pkin1*(-3.0*Pco*pow(Ph2,3.)/Kp1/Pch4/Ph2o)/pow(DEN,2.)
             -2.0*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdPh2;

      dr2dPh2 = k2*(-0.25*Pco*pow(Ph2o,0.5)/pow(Ph2,1.5))*(1-Prev2/Kp2)/pow(DEN,2.)
             +k2*Pkin2*(-1.0*Pco2)/Kp2/Pco/Ph2o/pow(DEN,2.)
             -2.0*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdPh2;

      dr3dPh2 = k3*(-1.75*Pch4*Ph2o)/pow(Ph2,2.75)*(1-Prev3/Kp3)/pow(DEN,2.)
             +k3*Pkin3*(-4.0*Pco2*pow(Ph2,3.))/Kp3/Pch4/pow(Ph2o,2.)/pow(DEN,2.)
             -2.0*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdPh2;

      dS[eqn] = rhos*Mh2*(alph1*dr1dPh2+alph2*dr2dPh2+alph3*dr3dPh2);
    }

  return source;
}

DEFINE_SOURCE(spe_uds2, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dDENdPco, dr1dPco, dr2dPco, dr3dPco;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
  Pco = C_UDSI(cell, thread, 2);
  Pco2 = C_UDSI(cell, thread, 3);
  Ph2o = C_UDSI(cell, thread, 4);

  alph1 = 1.0;
  alph2 = -1.0;
  alph3 = 0.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));
```

```
      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));


      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mco;

      dDENdPco = kco;

      dr1dPco = k1*Pkin1*(-1.0*pow(Ph2,3.)/Kp1/Pch4/Ph2o)/pow(DEN,2.)
             -2.0*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdPco;

      dr2dPco = k2*pow(Ph2o,0.5)/pow(Ph2,0.5)*(1-Prev2/Kp2)/pow(DEN,2.)
             +k2*Pkin2*(Ph2*Pco2)/Kp2/pow(Pco,2.)/Ph2o/pow(DEN,2.)
             -2.0*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdPco;

      dr3dPco = -2.0*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdPco;

      dS[eqn] = rhos*Mco*(alph1*dr1dPco+alph2*dr2dPco+alph3*dr3dPco);
    }

  return source;
}

DEFINE_SOURCE(spe_uds3, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dr1dPco2, dr2dPco2, dr3dPco2;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
  Pco = C_UDSI(cell, thread, 2);
  Pco2 = C_UDSI(cell, thread, 3);
  Ph2o = C_UDSI(cell, thread, 4);

  alph1 = 0.0;
  alph2 = 1.0;
  alph3 = 1.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
```

```
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mco2;

      dr1dPco2 = 0;

      dr2dPco2 = k2*Pkin2*(-1.0*Ph2)/Kp2/Pco/Ph2o/pow(DEN,2.);

      dr3dPco2 = k3*Pkin3*(-1.0*pow(Ph2,4.)/Kp3/Pch4/pow(Ph2o,2.))/pow(DEN,2.);

      dS[eqn] = rhos*Mco2*(alph1*dr1dPco2+alph2*dr2dPco2+alph3*dr3dPco2);
    }

  return source;
}

DEFINE_SOURCE(spe_uds4, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dDENdPh2o, dr1dPh2o, dr2dPh2o, dr3dPh2o;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;
  real alph1, alph2, alph3;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
  Pco = C_UDSI(cell, thread, 2);
  Pco2 = C_UDSI(cell, thread, 3);
  Ph2o = C_UDSI(cell, thread, 4);

  alph1 = -1.0;
  alph2 = -1.0;
  alph3 = -2.0;

  if (cell_temp <= 550)
      source = dS[eqn] = 0.0;
```

```
  else
    {
      Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
      Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
      Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
      Prev2 = Pco2*Ph2/Pco/Ph2o;
      Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
      Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

      kco = AKco*exp(-delhco/(rgas*cell_temp));
      kh = AKh*exp(-delhh/(rgas*cell_temp));
      kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

      DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

      Kp1 = 1.198e17*exp(-26830/(cell_temp));
      Kp2 = 1.767e-2*exp(4400/(cell_temp));
      Kp3 = 2.117e15*exp(-22430/(cell_temp));

      k1 = A1*exp(-E1/(rgas*cell_temp));
      r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

      k2 = A2*exp(-E2/(rgas*cell_temp));
      r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

      k3 = A3*exp(-E3/(rgas*cell_temp));
      r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

      source = rhos*(alph1*r1+alph2*r2+alph3*r3)*Mh2o;

       dDENdPh2o = kh2o/Ph2;

      dr1dPh2o = 0.5*k1*Pch4/pow(Ph2o,0.5)/pow(Ph2,1.25)*(1-Prev1/Kp1)/pow(DEN,2.)
                    + k1*Pkin1*Pco*pow(Ph2,3.)/Kp1/Pch4/pow(Ph2o,2.)/pow(DEN,2.)
                        - 2*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdPh2o;

      dr2dPh2o = 0.5*k2*Pco/pow(Ph2o*Ph2,0.5)*(1-Prev2/Kp2)/pow(DEN,2.)
                    + k2*Pkin2*Pco2*Ph2/Kp2/Pco/pow(Ph2o,2.)/pow(DEN,2.)
                        - 2*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdPh2o;

      dr3dPh2o = k3*Pch4/pow(Ph2,1.75)*(1-Prev3/Kp3)/pow(DEN,2.)
                    + k3*Pkin3*2*Pco2*pow(Ph2,4.)/Kp3/Pch4/pow(Ph2o,3.)/pow(DEN,2.)
                        - 2*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdPh2o;

      dS[eqn] = rhos*Mh2o*(alph1*dr1dPh2o+alph2*dr2dPh2o+alph3*dr3dPh2o);
    }

  return source;
}

DEFINE_SOURCE(q_tdep, cell, thread, dS, eqn)
{
  real source;
  real kco, kh, kh2o, DEN, k1, Kp1, r1, k2, Kp2, r2, k3, Kp3, r3;
  real Pkin1, Prev1, Pkin2, Prev2, Pkin3, Prev3;
  real dk1dt, dKp1dt, dk2dt, dKp2dt, dk3dt, dKp3dt, dDENdt, dr1dt, dr2dt, dr3dt;
  real cell_temp;
  real Pch4, Ph2, Pco, Pco2, Ph2o;

  cell_temp = C_T(cell, thread);

  Pch4 = C_UDSI(cell, thread, 0);
  Ph2 = C_UDSI(cell, thread, 1);
```

```
Pco = C_UDSI(cell, thread, 2);
Pco2 = C_UDSI(cell, thread, 3);
Ph2o = C_UDSI(cell, thread, 4);

if (cell_temp <= 550)
    source = dS[eqn] = 0.0;
else
  {
    Pkin1 = Pch4*pow(Ph2o,0.5)/pow(Ph2,1.25);
    Prev1 = Pco*pow(Ph2,3.)/Pch4/Ph2o;
    Pkin2 = Pco*pow(Ph2o,0.5)/pow(Ph2,0.5);
    Prev2 = Pco2*Ph2/Pco/Ph2o;
    Pkin3 = Pch4*Ph2o/pow(Ph2,1.75);
    Prev3 = Pco2*pow(Ph2,4.)/Pch4/pow(Ph2o,2.);

    kco = AKco*exp(-delhco/(rgas*cell_temp));
    kh = AKh*exp(-delhh/(rgas*cell_temp));
    kh2o = AKh2o*exp(-delhh2o/(rgas*cell_temp));

    DEN = 1+Pco*kco+pow(Ph2,0.5)*kh+Ph2o/Ph2*kh2o;

    Kp1 = 1.198e17*exp(-26830/(cell_temp));
    Kp2 = 1.767e-2*exp(4400/(cell_temp));
    Kp3 = 2.117e15*exp(-22430/(cell_temp));

    k1 = A1*exp(-E1/(rgas*cell_temp));
    r1 = k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.);

    k2 = A2*exp(-E2/(rgas*cell_temp));
    r2 = k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.);

    k3 = A3*exp(-E3/(rgas*cell_temp));
    r3 = k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.);

    source = rhos*(delHr1*r1+delHr2*r2+delHr3*r3);

    dDENdt = Pco*kco*delhco/rgas/cell_temp/cell_temp
            +pow(Ph2,0.5)*kh*delhh/rgas/cell_temp/cell_temp
            +Ph2o/Ph2*kh2o*delhh2o/rgas/cell_temp/cell_temp;

    dk1dt = k1*E1/rgas/cell_temp/cell_temp;
    dk2dt = k2*E2/rgas/cell_temp/cell_temp;
    dk3dt = k3*E3/rgas/cell_temp/cell_temp;

    dKp1dt = Kp1*26830/cell_temp/cell_temp;
    dKp2dt = Kp2*(-4400)/cell_temp/cell_temp;
    dKp3dt = Kp3*22430/cell_temp/cell_temp;

    dr1dt = dk1dt*Pkin1*(1-Prev1/Kp1)/pow(DEN,2.)
           +k1*Pkin1*(Prev1/Kp1/Kp1)*dKp1dt/pow(DEN,2.)
           -2*k1*Pkin1*(1-Prev1/Kp1)/pow(DEN,3.)*dDENdt;

    dr2dt = dk2dt*Pkin2*(1-Prev2/Kp2)/pow(DEN,2.)
           +k2*Pkin2*(Prev2/Kp2/Kp2)*dKp2dt/pow(DEN,2.)
           -2*k2*Pkin2*(1-Prev2/Kp2)/pow(DEN,3.)*dDENdt;

    dr3dt = dk3dt*Pkin3*(1-Prev3/Kp3)/pow(DEN,2.)
           +k3*Pkin3*(Prev3/Kp3/Kp3)*dKp3dt/pow(DEN,2.)
           -2*k3*Pkin3*(1-Prev3/Kp3)/pow(DEN,3.)*dDENdt;

    dS[eqn] = rhos*(delHr1*dr1dt+delHr2*dr2dt+delHr3*dr3dt);
  }
```

```
  return source;
}


DEFINE_ADJUST(Yi_adjust,d)
{
  Thread *t;
  cell_t c;

  real Mav, Ptot;

  thread_loop_c(t,d)
    {
    if(NNULLP(T_STORAGE_R(t,SV_P)))
    /* Test if it is fluid by seeing if pressure is available */
      {
       begin_c_loop(c,t)
         {
          Ptot = C_UDSI(c, t, 0) + C_UDSI(c, t, 1) + C_UDSI(c, t, 2) + C_UDSI(c, t, 3) +
C_UDSI(c, t, 4);
          Mav = (C_UDSI(c, t, 0)*Mch4 + C_UDSI(c, t, 1)*Mh2 + C_UDSI(c, t, 2)*Mco + C_UDSI(c,
t, 3)*Mco2 + C_UDSI(c, t, 4)*Mh2o) / Ptot;

          C_YI(c, t, 0) = C_UDSI(c, t, 0) * Mch4 / (Ptot * Mav);
          C_YI(c, t, 1) = C_UDSI(c, t, 1) * Mh2  / (Ptot * Mav);
          C_YI(c, t, 2) = C_UDSI(c, t, 2) * Mco  / (Ptot * Mav);
          C_YI(c, t, 3) = C_UDSI(c, t, 3) * Mco2 / (Ptot * Mav);
          /* Yh2o is not adjusted since fluent wants to compute it from "sum Yi = 1". */
         }
       end_c_loop(c,t)
      }
    }
}

/*****************************************************************************************/
/*                                        INTERFACE                                      */
/*****************************************************************************************/

DEFINE_PROFILE(coupled_uds_0, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
```

```
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,0);
    diff1 = C_UDSI_DIFF(c1,tc1,0);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,0) + h1*C_UDSI(c1,tc1,0))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_1, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,1);
    diff1 = C_UDSI_DIFF(c1,tc1,1);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,1) + h1*C_UDSI(c1,tc1,1))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_2, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
```

```
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,2);
    diff1 = C_UDSI_DIFF(c1,tc1,2);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,2) + h1*C_UDSI(c1,tc1,2))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_3, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,3);
```

```
    diff1 = C_UDSI_DIFF(c1,tc1,3);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,3) + h1*C_UDSI(c1,tc1,3))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

DEFINE_PROFILE(coupled_uds_4, t, i)
{
  Thread *tc0, *tc1;
  cell_t c0,c1;
  face_t f;

  real A[ND_ND], x0[ND_ND], x1[ND_ND], C1_COORD[ND_ND], C0_COORD[ND_ND], F_COORD[ND_ND];
  real e_x0[ND_ND], e_x1[ND_ND];
  real uds_b, diff0, diff1;
  real h0, h1, A_by_ex0, A_by_ex1;
  real dx0, dx1;

  begin_f_loop(f, t)
  {
    F_AREA(A,f,t);
    c0 = F_C0(f, t);
    c1 = F_C1(f, t);
    tc0 = THREAD_T0(t);
    tc1 = THREAD_T1(t);
    C_CENTROID(C0_COORD, c0, tc0);
    C_CENTROID(C1_COORD, c1, tc1);
    F_CENTROID(F_COORD, f, t);
    NV_VV(x0, =, F_COORD, -, C0_COORD);
    dx0 = NV_MAG(x0);
    NV_VV(x1, =, F_COORD, -, C1_COORD);
    dx1 = NV_MAG(x1);
    NV_VS(e_x0, =, x0, /, dx0);
    NV_VS(e_x1, =, x1, /, dx1);
    A_by_ex0 = NV_DOT(A,A)/NV_DOT(e_x0,A);
    A_by_ex1 = NV_DOT(A,A)/NV_DOT(e_x1,A);
    diff0 = C_UDSI_DIFF(c0,tc0,4);
    diff1 = C_UDSI_DIFF(c1,tc1,4);
    h0 = diff0/dx0*A_by_ex0;
    h1 = -diff1/dx1*A_by_ex1;
    uds_b = (h0*C_UDSI(c0,tc0,4) + h1*C_UDSI(c1,tc1,4))/(h0+h1);
    F_PROFILE(f,t,i) = uds_b;
  }
  end_f_loop(f, t)
}

/*****************************************************************************************/
/*                            UDM INITIAL VALUES (Init and On Demand)                    */
/*****************************************************************************************/

DEFINE_ON_DEMAND(On_demand_UDMI_Initialization)
{
  Domain *d;
  cell_t c;
  Thread *t;

  int ID;

  d = Get_Domain(1);
```

```
  for (ID = 2;  ID <= 14;  ++ID)
    {
      t = Lookup_Thread(d, ID);
      begin_c_loop(c,t)
        {
          C_UDMI(c, t, 0) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 1) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 2) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 3) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 4) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 5) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 6) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 7) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 8) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 9) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 10) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 11) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 12) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 13) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 14) = 1.633 * pow(10.,-6.);
        }
      end_c_loop(c,t)
    }
}

DEFINE_INIT(Init_UDMI_Initialization,d)
{
  cell_t c;
  Thread *t;

  /* loop over all cell threads in the domain  */
  thread_loop_c(t,d)
    {

      /* loop over all cells  */
      begin_c_loop_all(c,t)
        {
          C_UDMI(c, t, 0) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 1) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 2) = 1.1907 * pow(10.,-6.);
          C_UDMI(c, t, 3) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 4) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 5) = 2.501 * pow(10.,-6.);
          C_UDMI(c, t, 6) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 7) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 8) = 8.0 * pow(10.,-7.);
          C_UDMI(c, t, 9) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 10) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 11) = 4.968 * pow(10.,-7.);
          C_UDMI(c, t, 12) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 13) = 1.633 * pow(10.,-6.);
          C_UDMI(c, t, 14) = 1.633 * pow(10.,-6.);
        }
      end_c_loop_all(c,t)
    }
}
```

Appendices F – Reaction rates parameters

The reaction parameters for Hou & Hughes' (2001) metane steam reforming kinetic model are the following:

Reaction	Pre-exponential factor ($kmol.kg_{cat}^{-1}.s^{-1}$)	Activation energy ($kJ.mol^{-1}$)
1	$5.922.10^{8}$	209.2
2	$6.028.10^{-4}$	15.4
3	$1.093.10^{3}$	109.4

Table 5 – Reaction rates constants

Besides

$$K_{CO} = 5.127\,.\,10^{-13}.\,e^{\frac{140}{RT}} \qquad \textit{eq. F-1}$$

$$K_{H} = 5.68\,.\,10^{-10}.\,e^{\frac{93.4}{RT}} \qquad \textit{eq. F-2}$$

$$K_{H_2O} = 9.251.\,e^{\frac{-15.9}{RT}} \qquad \textit{eq. F-3}$$

$$K_{P1} = 1.198\,.\,10^{17}.\,e^{\frac{-26{,}830}{T}} \qquad \textit{eq. F-4}$$

$$K_{P2} = 1.767\,.\,10^{-2}.\,e^{\frac{4{,}400}{T}} \qquad \textit{eq. F-3}$$

$$K_{P3} = 2.117\,.\,10^{15}.\,e^{\frac{-22{,}430}{T}} \qquad \textit{eq. F-4}$$

where RT is in $kJ.mol^{-1}$.