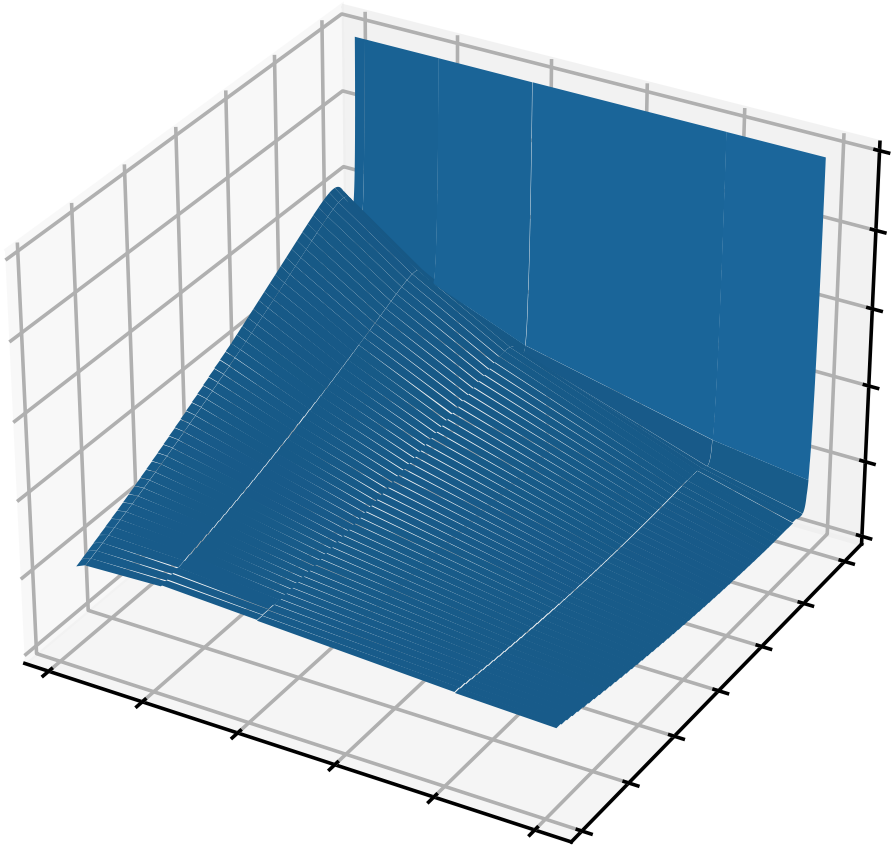
Année académique 2023-2024

Production d'H_2 par reformage catalytique de CH_4 et captage de CO_2 dans un réacteur.



LIÈGE université Sciences Appliquées

7 avril 2024

Moreci Raffaele
S2304531

Odding Luca
S2303933

Lindsey Alexandre
S2302371

Table des matières

Question 1

1.1 Méthode de la sécante

Notre fonction sécante requiert trois arguments nécessaires au bon fonctionnement de la méthode qui sont :

— fun, la fonction dont on recherche les racines,
— x, un tableau contenant les deux valeurs de x de départ,
— tol, la tolérance qui permet de gérer la précision des résultats,

Logiquement nous devons donc compromettre la précision pour limiter les calculs et donc ne pas mettre une tolérance trop faible. Nous avons opté, après une multitude d'essais, pour une tolérance de 5×10^{-4} qui nous permet de garder une bonne précision pour un temps d'exécution faible. Par exemple pour vérifier si une valeur est "égale" à une autre, nous implémentons $|x0 - x1| \leq tol$, les méthodes numériques ne pouvant pas être infiniment précises.

Nous avons implémenté la fonction en s'assurant tout d'abord que toutes les conditions initiales soient bonnes et, pour que la fonction n'itère pas à l'infini, nous avons introduit max_i en arguments $kwargs$ avec une valeur par défaut élevée qui restreint la fonction à ne pas faire plus d'itérations que ce nombre tout en laissant une marge afin de ne pas arrêter trop vite l'exécution. Après avoir pris nos précautions, une boucle va itérer tant que $f(x1) \leq tol$ (nous utilisons de nouveau la tolérance car il est quasiment impossible que la valeur soit exactement égale à 0 par les méthodes numériques).

Nous avons décidé de séparer le calcul du numérateur et du dénominateur afin de s'éviter certaines problématiques : si le numérateur $y1 \times (x1 - x0)$ est égal à 0, $x0$ et $x1$ doivent avoir la même valeur du coup la fonction retourne -1 pour indiquer que la fonction ne converge pas. Si le dénominateur $y1 - y0$ est égal à 0, la fonction ne converge pas vu la méthode de la sécante : la droite passe par deux points qui forment une droite parallèle à l'abscisse. La boucle se finira soit par le non-respect de sa condition ou bien si la boucle fait un nombre anormalement grand d'itérations.

1.2 Méthode de la bissection

Les arguments sont les mêmes que dans la méthode de la sécante avec la même tolérance de 5×10^{-4} mais sans introduire max_i car si les conditions initiales sont respectées, la fonction convergera toujours. Nous nous assurons donc que les conditions initiales sont respectées en posant l'hypothèse que la fonctions traitée est continue.

Pour se rapprocher de la racine nous allons itérer pendant k itérations égal à $log_2(\frac{x1-x0}{2\times tol})$ comme vu dans le cours. Dans cette boucle, on calcule la moyenne des abscisses des deux points qui sera celle de notre nouveau point. Si la valeur de la fonction en ce point est positive, on remplace $x1$ (à image positive) par cette moyenne et inversement. Après toutes les itérations, la valeur de xi sera la valeur de notre racine.

Question 2

2.1 Création d'odefunction

Notre fonction *odefunction* est une fonction qui reçoit trois arguments :
- z, la distance axiale du réacteur
- $C0$, le tableau avec les valeurs initiales des huit variables d'état : les concentrations initiales en CH_4, H_2O, H_2, CO, CO_2 la conversion fractionnaire X, la température T et la pression P
- *mode* et *param*, qui nous serviront plus tard

La fonction odefunction calcule la dérivée du tableau $\frac{dC}{dz}$ et retourne ces valeurs. Pour plus de clarté et de flexibilité, les constantes ont été placées dans le fichier *constants.py*. Certaines constantes sont implémentées sous forme de fonction, certes cela n'est pas optimisé pour le temps mais nous avons opté pour cette option par soucis de lisibilité et de flexibilité, ce qui nous a aidé pour la question 3.

2.2 Résolution d'équation différentielle

Notre fonction *calculConcentrationEuler* est une fonction qui suit la méthode d'Euler explicite au premier ordre pour résoudre approximativement l'équation contenue dans odefunction. Cette fonction prend cinq arguments en entrée :
- fun, la fonction à résoudre
- x, un tableau contenant les bornes pour lesquels l'équation doit être résolue
- $y0$, les valeurs initiales de la fonction
- *step*, le pas qui est défini à 5×10^{-8}
- *mode* et *param*, qui nous serviront plus tard.

La fonction renvoie une approximation de la fonction inconnue.

Pour justifier notre choix de pas de 5×10^{-8} nous allons nous servir de la figure 2.1 où, dans ce tableau crée expérimentalement, nous avons répertorié les taux d'erreur et temps d'exécution par pas. Pour cela, nous avons effectué plusieurs simulations sur une distance radiale réduite de 0.01m (les calculs étants plus courts) et comme comparaison pour l'erreur la résolution des équations par *solve_ivp* avec l'accent mis sur une grande précision. Les valeurs de pas qui ont un taux d'erreur acceptable sont toutes les valeurs plus petites que 5×10^{-8} car le taux d'erreur est en dessous de 1% et les valeurs de pas qui prennent un temps acceptable sont celle plus grandes que 5×10^{-8} qui prend quasiment 2 minutes sur cet intervalle réduit alors que le palier suivant a 5×10^{-9} prend plus de dix fois plus de temps. Donc grâce a toutes ces données nous avons conclu que la valeur la plus approprié pour la tolérance est 5×10^{-8}.

step	Temps	Taux d'erreur
$5e-1$	$19.7\ \mu s$	61.41%
$5e-2$	$20.9\ \mu s$	61.41%
$5e-7$	$9.32\ s$	64.52%
$5e-8$	$1\ min\ 56\ s$	0.98%
$5e-9$	$20\ min\ 21\ s$	0.01%
$5e-10$	$3\ h\ 3\ min$	0.0003%

FIGURE 2.1 – Taux d'erreur et temps d'exécution par pas décroissants

La fonction *solve_ivp* prend les mêmes arguments que *calculConcentrationEuler*. Nous avons modifier la valeur de la tolérance relative *rtol* qui a pour but d'augmenter la précision de notre approximation. Nous avons choisi notre *rtol* égal à 10^{-6}, c'est une valeur qui apporte la précision requise en un temps d'execution imperceptible.

Après plusieurs essais, nous pouvons conclure que *solve_ivp* est beaucoup plus efficace que notre fonction. Notre fonction prend environs 48.67 minutes pour résoudre l'équation selon le profiler tandis que *solve_ivp* ne prend que 90 *milisecondes*, soit environ 33000 fois plus vite. Ceci est probablement dû au fait que *solve_ivp* utilise un pas variable mais pas notre fonction. La fonction *solve_ivp* est également beaucoup plus précise puisqu'elle utilise une méthode d'ordre suprérieur. Nous concluons que *solve_ivp* est beaucoup plus efficace, que ce soit pour le temps d'exécution ou pour la précision.

Question 3

Pour cette question 3, nous avons introduit les arguments optionnels *mode* et *param* à odefunction. *mode* nous permet de tester différentes conditions de notre réacteur. Lorsque *mode* est à 0, le système fonctionne avec les paramètres de base, lorsqu'il est à 1, le modèle ne prend pas en compote la carbonatation, lorsqu'il est à 2, on remplace us par la valeur de *param* et lorsqu'il est à 3, on remplace ug par la valeur de *param*.

3.1 Modèle sans capture de CO_2

Afin de comparer un modèle de réacteur avec et sans capture de CO_2, nous avons mis le mode de odefunction à 1, soit un modèle sans capture de CO_2. Pour simuler ceci, nous avons remplacé la conversion fractionnaire (X), le taux de consommation de CO_2 par carbonatation ($rcbn$) et la vitesse d'entré du CaO dans le réacteur (us) par 0 car ils n'interviennent plus dans le modèle sans capture de CO_2.

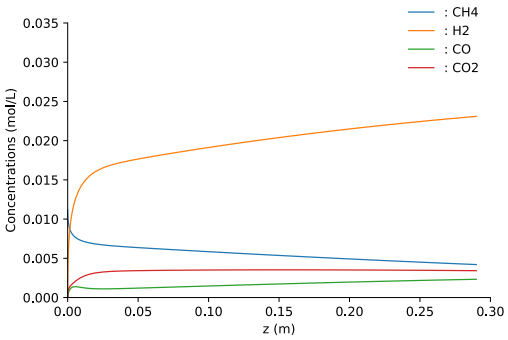


FIGURE 3.2 – Variation des concentrations avec carbonatation

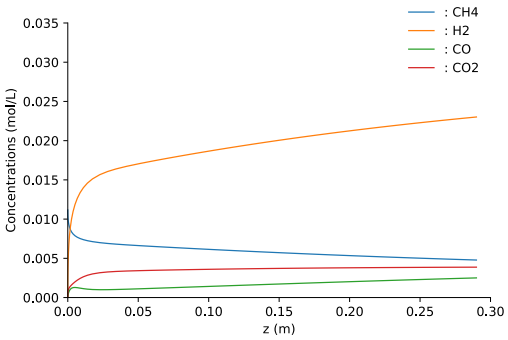


FIGURE 3.3 – Variation des concentrations sans carbonatation

Pour comparer un système avec et sans carbonatation, il est difficile d'utiliser des graphiques car, comme on peut le voir aux figures 3.2 et 3.3 , la tendance générale du graphique ne varie pas énormément avec ou sans carbonatation. C'est pour cela que nous allons nous fier aux valeurs finales des concentrations (voir figure 3.4). Donc quand on bloque la capture de CO_2 : le CH_4, le CO_2 et le CO le augmentent de 14.7%, 12.6% et 6.5% respectivement. Le H_2 et la température diminuent seulement de 0.9% et 0.5% respectivement.

Nom	*mode* 0	*mode* 1	Différence
CH_4	$0,00417\ \frac{mol}{L}$	$0,00478\ \frac{mol}{L}$	$+14,7\%$
H_2	$0,02322\ \frac{mol}{L}$	$0,02302\ \frac{mol}{L}$	-0.9%
CO	$0,00235\ \frac{mol}{L}$	$0,00250\ \frac{mol}{L}$	$+6.5\%$
CO_2	$0,00344\ \frac{mol}{L}$	$0,00387\ \frac{mol}{L}$	$+12.6\%$
$T°$	$920,23\ K$	$915,85\ K$	-0.5%

FIGURE 3.4 – Effet de la carbonatation sur le modèle

3.2 Effet de l'augmentation du flux de CaO sur le modèle

Pour analyser l'effet du flux d'entrée de CaO, nous avons fait varier la valeur de us à l'aide du mode 2. Nous l'avons fait varier de 0.001 à 0.025 $m.s^{-1}$.

(((EXPLIQUER GRAPHIQUES ET METTRE GRAPHIQUES)))

3.3 Effets de l'augmentation du flux des gaz sur le modèle

Pour analyser le flux des gaz, nous avons fait varier ug à l'aide du mode 3 en changeant la valeur de 1 à 3 $m.s^{-1}$ avec un pas de 0.5.

(((EXPLIQUER GRAPHIQUES ET METTRE GRAPHIQUES)))

3.4 Effet de la composition du gaz d'entrer sur le modèle

Dans le but de modifier les compositions initiales de gaz, nous avons fait varier le rapport des concentrations initiales en H_2O et CH_4 de 3 à 0.3.

(((EXPLIQUER GRAPHIQUES ET METTRE GRAPHIQUES)))

Question 4

4.1 Taux de CO_2 optimal par rapport à ug

optimise_us crée la fonction du pourcentage de CO_2 en sortie par rapport à us, ce qui nous permet de déterminer un us optimal pour une concentration donnée. Cette fonction prend 5 arguments en entrée :

— Y, le pourcentage de CO_2 souhaité à la sortie du réacteur
— us, la valeur de us
— $C0$, le tableau des conditions initiales pour $odefunction$
— $mode\_$ et $param1$, équivalents du $mode$ et $param$ de $odefunction$

Cette fonction résout odefunction grâce à *solve_ivp* et retourne la concentration finale de CO_2 divisé par la somme des concentration finale de CH_4, H_2, CO et de CO_2. Le tout est ensuite soustrait par Y pour corriger le décalage vertical.

4.2 Recherche d'un us optimale

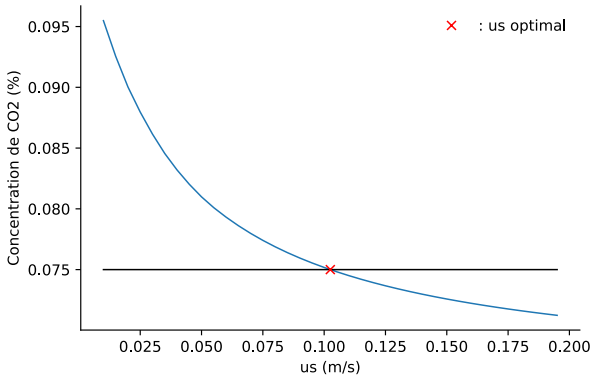


FIGURE 4.5 – Valeur optimal de us

Dans ce cas, nous devons trouvez la valeur optimale de us pour une concentration en CO_2 de 7.5%. On peut voir la fonction *optimise_us* ainsi que la valeur optimal sur la figure 4.5 qui représente us en fonction du pourcentage de CO_2 parmis les gaz secs en sortie. Notre fonction *hybrid* sécante nous renvoit une racine située en $x = 0.101$.

La méthode utilisée pour trouver la racine est une méthode hybride qui est simplement la méthode de la sécante légèrement modifiée : si la sécante ne converge pas dû à des images de $x0$ et $x1$ égales, alors on s'inspire de la méthode de la bissection pour débloquer la situation. Pour ce faire, on effectue la moyenne des deux points et on utilise ce nouveau point pour relancer la fonction sécante. Ce faisant, on bénéficie de la précision accrue de la méthode de la sécante ainsi que sa vitesse bien supérieure a la bissection. De plus, la méthode de la sécante ne demande pas d'avoir $f(x0)$ et $f(x1)$ de signes contraires. C'est pour ces raisons que la sécante nous a paru être la bonne méthode a utiliser.

4.3 Effet de la température sur le us optimale

La figure 4.6 illustre la variation du *us* optimal en fonction de la température. Il est clair que *us* dépend de la température. Nous le voyons décroitre très rapidement, presque exponentiellement à mesure que la température augmente. Donc, plus la température est élevée, plus la vitesse d'entrée optimale des gaz pour un pourcentage de CO_2 donné est petite.

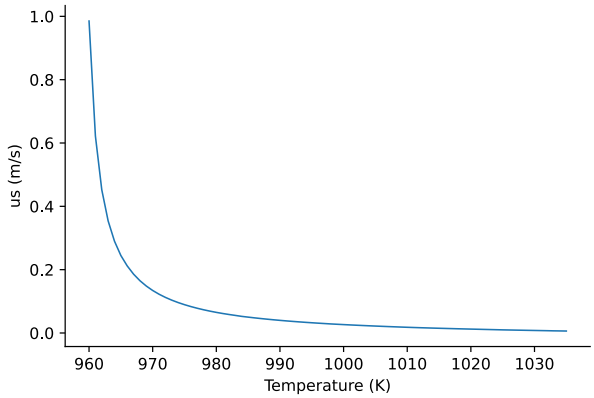


FIGURE 4.6 – L'effet de la température T sur le us optimal