Année académique 2023-2024

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

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 `secant` 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 0, nous implémentons `abs(y1) < 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 `fun(x1) < 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*(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 si il y a un non-respect de sa condition ou 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 doit recevoir 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 module `constants`. Certaines constantes sont implémentées sous forme de fonction. 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 par défault à 5×10^{-8}
- `mode` et `param`, qui nous serviront plus tard.

La fonction renvoie une approximation de la fonction inconnue.

Pour trouver le meilleur `step`, nous nous sommes référés aux données de la figure **??**. Dans ce tableau crée expérimentalement, nous avons répertorié les taux d'erreur et temps d'exécution par `step`. Pour cela, nous avons effectué plusieurs simulations sur une distance radiale réduite de 0,01 m (les calculs étants plus courts). Pour déterminer l'erreur, nous avons fait résoudre les équations à `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^{-7} car leur 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^{-9} (en prennant en compte que toutes ces valeurs sont sur un intervalle réduit). Du coup, nous avons choissi 5×10^{-8} qui prend quasiment 2 minutes sur cet intervalle réduit puisque le palier suivant, 5×10^{-9}, prend plus de dix fois le temps. Donc, grâce a toutes ces données, nous avons conclu que la valeur la plus approprié pour la tolérance est 5×10^{-8}.

La fonction `calculConcentrationIVP` prend tous les mêmes arguments que notre fonction `calculConcentrationEuler`. Elle utilise la fonction préexistente de `scipy`, `solve_IVP`. 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 5×10^{-7} puisque c'est une valeur qui apporte la précision requise en un temps d'execution imperceptible.

`step`	Temps	Taux d'erreur
5×10^{-1}	19,7 µs	61,41 %
5×10^{-2}	20,9 µs	61,41 %
5×10^{-7}	9,32 s	64,52 %
5×10^{-8}	1 min 56 s	0,98 %
5×10^{-9}	20 min 21 s	0,01 %
5×10^{-10}	3 h 3 min	0,0003 %

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

Après plusieurs essais, nous pouvons conclure que `solve_ivp` est beaucoup plus efficace que notre fonction qui prend environs 48,67 minutes pour résoudre l'équation selon le profiler intégré de spyder tandis que `calculConcentrationIVP` ne prend que 88,69 milisecondes, soit environ 33000 fois plus rapide. Ceci est probablement dû au fait que `solve_ivp` utilise un pas variable contrairement à notre fonction. La fonction `solve_ivp` est également beaucoup plus précise puisqu'elle utilise une méthode d'ordre suprérieur. Nous concluons que `calculConcentrationIVP` est beaucoup plus efficace, que ce soit pour le temps d'exécution ou pour la précision.

Question 3

Pour la question 3, nous avons introduit à `odefunction` les arguments optionnels `mode` et `param`. `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`, lorsqu'il est à 3, on remplace `ug` par la valeur de `param` et lorsqu'il est a 4, on remplace `us` par `param[0]` et `TW` par `param[1]`.

3.1 Modèle sans capture de CO_2

FIGURE 3.2 – Variation des concentrations avec carbonatation

FIGURE 3.3 – Variation des concentrations sans carbonatation

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.

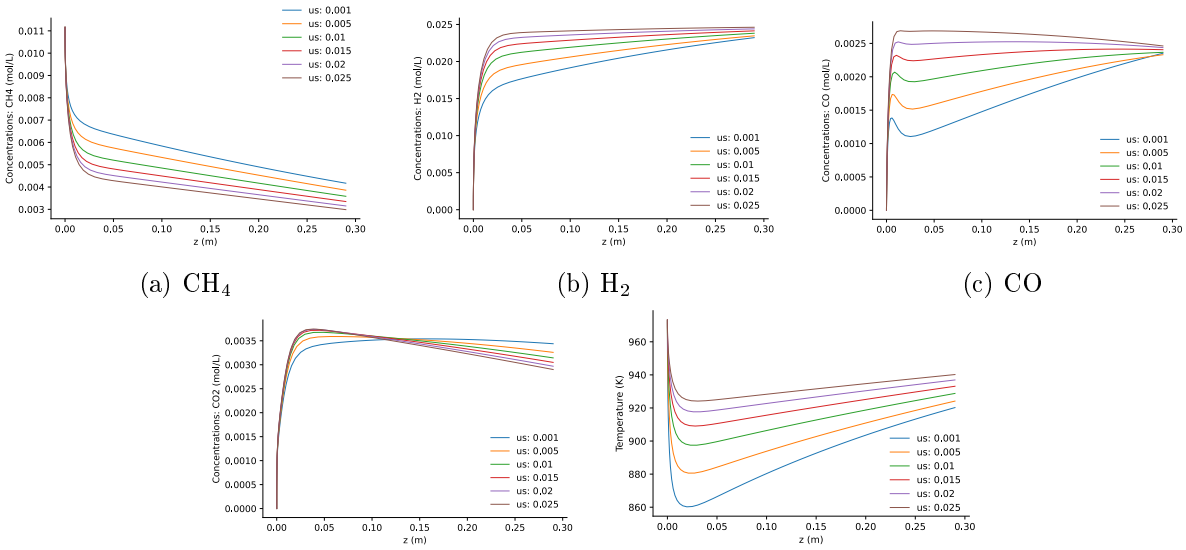
Nom	`mode` 0	`mode` 1	Différence
CH_4	$0{,}00417\,\mathrm{mol\,L^{-1}}$	$0{,}00478\,\mathrm{mol\,L^{-1}}$	+14,7 %
H_2	$0{,}02322\,\mathrm{mol\,L^{-1}}$	$0{,}02302\,\mathrm{mol\,L^{-1}}$	−0,9 %
CO	$0{,}00235\,\mathrm{mol\,L^{-1}}$	$0{,}00250\,\mathrm{mol\,L^{-1}}$	+6,5 %
CO_2	$0{,}00344\,\mathrm{mol\,L^{-1}}$	$0{,}00387\,\mathrm{mol\,L^{-1}}$	+12,6 %
`T°`	920,23 K	915,85 K	−0,5 %

FIGURE 3.4 – Effet de la carbonatation sur le modèle

Pour comparer un système avec et sans carbonatation, il est difficile d'utiliser ces graphiques car, comme on peut le voir aux figures **??** et **??**, la tendance générale du graphique ne varie pas énormément avec ou sans carbonatation mais ils peuvent être utiles pour distinguer les différences se créer au fil de l'avancée dans le réacteur. C'est pour cela que nous allons nous fier aux valeurs finales des concentrations (figure **??**). Donc, dans un modèle sans carbonatation, on remarque que le CH_4, le CO_2 et le CO augmentent de 14,7 %, 12,6 % et 6,5 % respectivement. Le H_2 et la température (`T`) diminuent de 0,9 % et 0,5 % respectivement.

3.2 Effet de l'augmentation du flux des solides sur le modèle

Pour analyser l'effet du flux des solides, nous avons fait varier la valeur de `us` à l'aide du `mode` 2. Nous l'avons fait varier de $0{,}001\,\mathrm{m\,s^{-1}}$ à $0{,}025\,\mathrm{m\,s^{-1}}$ et créer ces graphiques pour chaque variable d'état indépendamment avec plusieurs valeurs de `us`. De manière générale, les variations de vont



(a) CH_4 (b) H_2 (c) CO

(d) CO_2 (e) Temperature (`T`)

FIGURE 3.5 – Concentrations et températures (`T`) des gaz

impacter relativement beaucoup les concentrations et la température au début du réacteur mais à la sortie, l'écart avec le us de base ($0{,}001\,\mathrm{m\,s^{-1}}$), est plus faible. C'est pour cela que nous n'avons pas pris une plage de valeur de us plus grande, nous voyons avec ces valeurs une bonne représentation de ce qui se passe avec des valeurs de `us` > 0.001. Nous pouvons séparer toutes les variables d'états en deux comportements différents : celles qui diminuent au fur et à mesure que la valeur de us augmente qui sont le CH_4 et le CO_2 et celle qui augmentent au fur et mesure que us augmente, à savoir le H_2, le CO, et la température. Afin de mieux comprendre quels sont les variations des gaz à la sortie du réacteur, pour mieux voir les quantités que l'on récupère, voici un autre graphique des concentrations des gaz secs en fonction de `us`. (((GRAPHIQUEEE)))

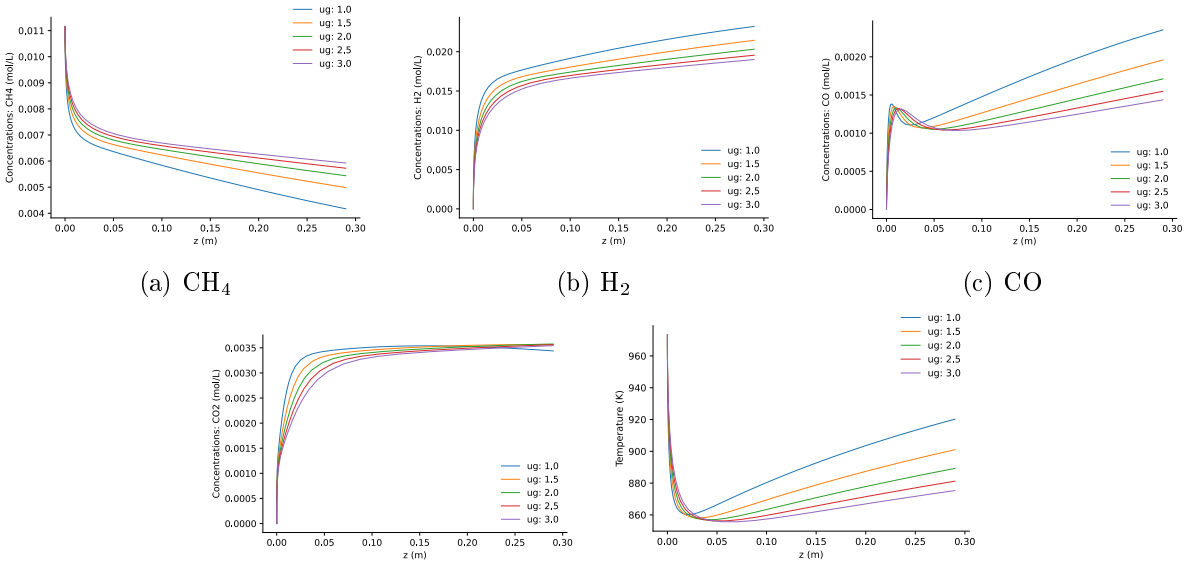
3.3 Effets de l'augmentation du flux des gaz sur le modèle

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

3.4 Effet de la composition du gaz d'entrée 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.

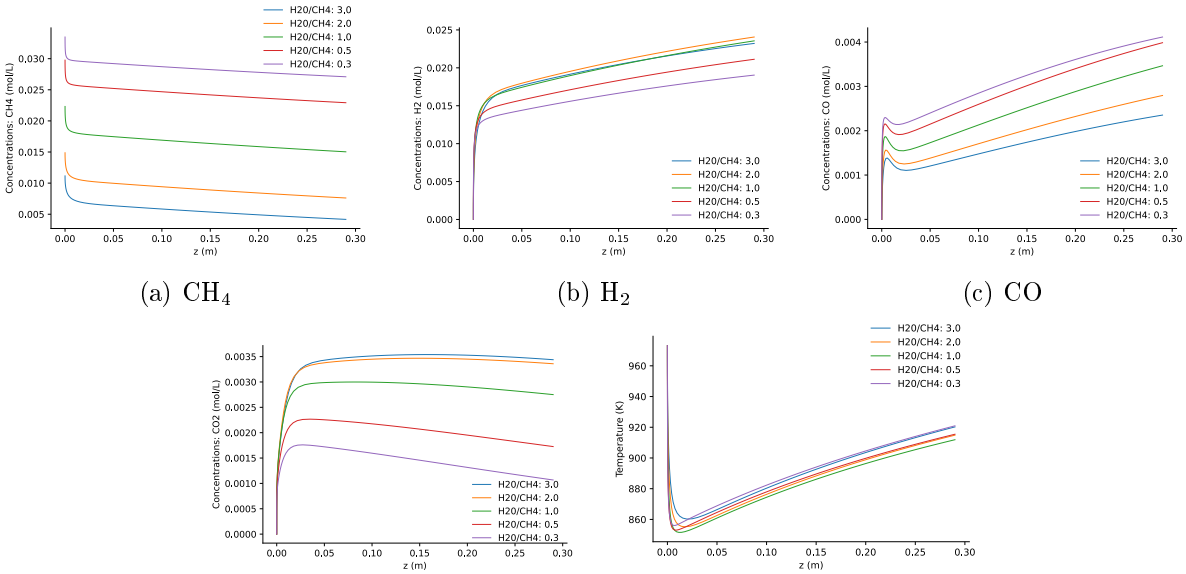
Comme illustré sur les 5 graphiques ci-dessous, l'augmentation de ug produit exactement l'effet inverse à l'augmentation de `us`. Au début, les données sont très sensibles aux variation de `ug` et très peu sensible a ces variation à la fin du réacteur. Nous pouvons aussi séparer les données de sorties en deux, mais cette fois-ci ce sont le H_2, le CO et la température qui diminuent tandis que le CH_4 et le CO_2 augmentent quand la valeur de `ug` augmente



(a) CH_4 (b) H_2 (c) CO

(d) CO_2 (e) Temperature (T)

FIGURE 3.6 – Concentrations et températures (T) des gaz



(a) CH_4 (b) H_2 (c) CO

(d) CO_2 (e) Temperature (T)

FIGURE 3.7 – Concentrations et températures (T) des gaz

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` optimal

FIGURE 4.8 – Valeur optimale 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 optimale sur la figure **??** qui représente `us` en fonction du pourcentage de CO_2 parmis les gaz secs en sortie. Notre fonction `secant` en mode `hybrid` nous renvoit une racine située en 0.103.

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 `secant`. 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 `fun(x0)` et `fun(x1)` de signes contraires. C'est pour ces raisons que `secant` nous a paru être la bonne fonction a utiliser.

4.3 Effet de la température sur le `us` optimal

La figure **??** illustre la variation du `us` optimal en fonction de la température (`T`). 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 4.9 – L'effet de la température (`T`) sur le `us` optimal