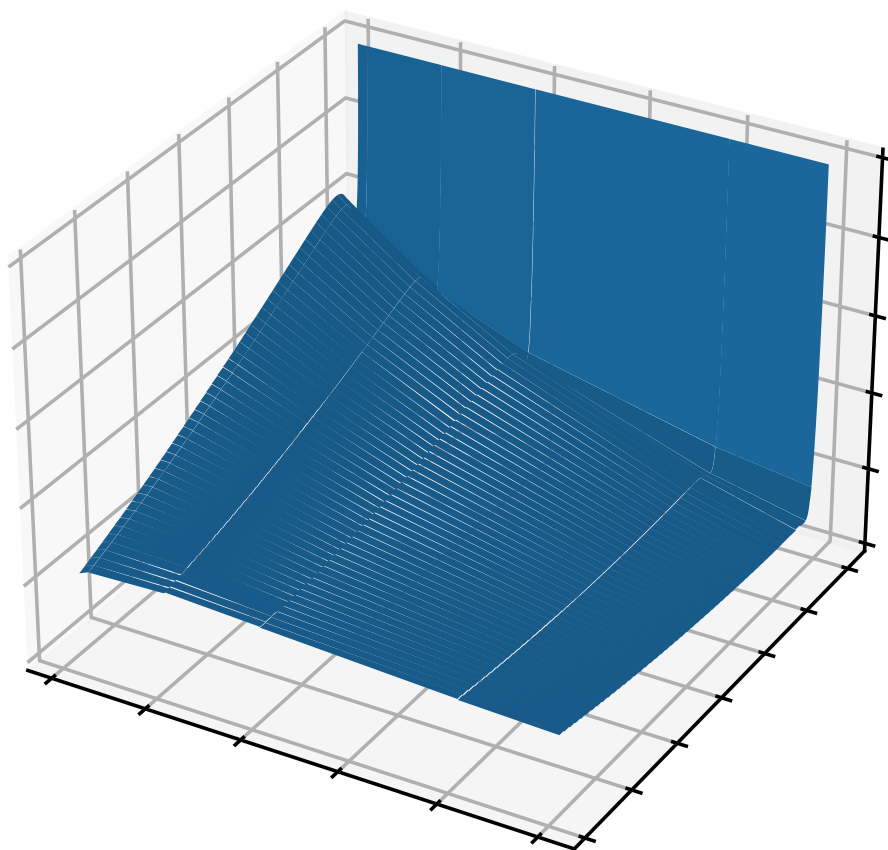


# Production d' $\text{H}_2$ par reformage catalytique de $\text{CH}_4$ et captage de $\text{CO}_2$ dans un réacteur.



# Table des matières

<b>Question 1</b>	<b>2</b>
1.1 Méthode de la sécante . . . . .	2
1.2 Méthode de la bisection . . . . .	2
<b>Question 2</b>	<b>3</b>
2.1 Création d'odefunction . . . . .	3
2.2 Résolution d'équation différentielle . . . . .	3
<b>Question 3</b>	<b>5</b>
3.1 Modèle sans capture de $\text{CO}_2$ . . . . .	5
3.2 Effet de l'augmentation du flux des solides sur le modèle . . . . .	6
3.3 Effets de l'augmentation du flux des gaz sur le modèle . . . . .	6
3.4 Effet de la composition du gaz d'entrer sur le modèle . . . . .	6
<b>Question 4</b>	<b>8</b>
4.1 Taux de $\text{CO}_2$ optimal par rapport à <b>ug</b> . . . . .	8
4.2 Recherche d'un <b>us</b> optimal . . . . .	8
4.3 Effet de la température sur le <b>us</b> optimal . . . . .	9

# 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  $5 \times 10^{-4}$  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 bisection

Les arguments sont les mêmes que dans la méthode de la sécante avec la même tolérance de  $5 \times 10^{-4}$  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
- `CO`, le tableau avec les valeurs initiales des huit variables d'état : les concentrations initiales en  $\text{CH}_4$ ,  $\text{H}_2\text{O}$ ,  $\text{H}_2$ ,  $\text{CO}$ ,  $\text{CO}_2$  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 défaut à  $5 \times 10^{-8}$
- `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 2.1. Dans ce tableau créé 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 étant 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  $5 \times 10^{-7}$  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  $5 \times 10^{-9}$  (en prenant en compte que toutes ces valeurs sont sur un intervalle réduit). Du coup, nous avons choisi  $5 \times 10^{-8}$  qui prend quasiment 2 minutes sur cet intervalle réduit puisque le palier suivant,  $5 \times 10^{-9}$ , prend plus de dix fois le temps. Donc, grâce à toutes ces données, nous avons conclu que la valeur la plus appropriée pour la tolérance est  $5 \times 10^{-8}$ .

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

step	Temps	Taux d'erreur
$5 \times 10^{-1}$	19,7 $\mu$ s	61,41 %
$5 \times 10^{-2}$	20,9 $\mu$ s	61,41 %
$5 \times 10^{-7}$	9,32 s	64,52 %
$5 \times 10^{-8}$	1 min 56 s	0,98 %
$5 \times 10^{-9}$	20 min 21 s	0,01 %
$5 \times 10^{-10}$	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 millisecondes, 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 supérieur. 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 compte 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 à 4, on remplace `us` par `param[0]` et `TW` par `param[1]`.

## 3.1 Modèle sans capture de CO<sub>2</sub>

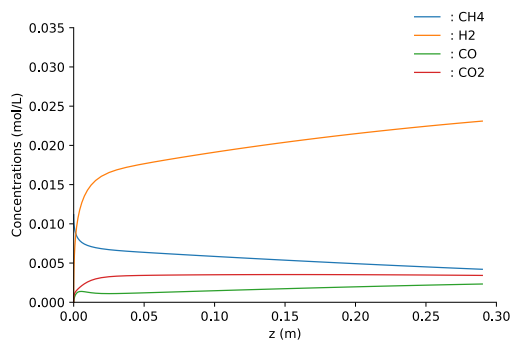


FIGURE 3.2 – Variation des concentrations avec carbonatation

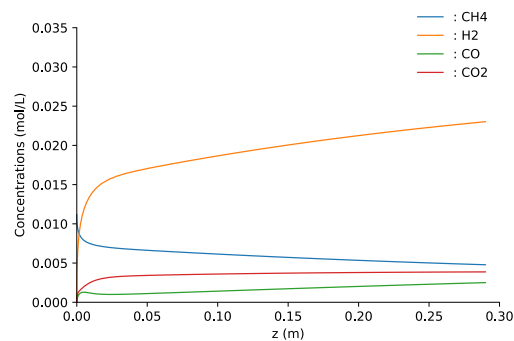


FIGURE 3.3 – Variation des concentrations sans carbonatation

Afin de comparer un modèle de réacteur avec et sans capture de CO<sub>2</sub>, nous avons mis le mode de `odefunction` à 1, soit un modèle sans capture de CO<sub>2</sub>. Pour simuler ceci, nous avons remplacé la conversion fractionnaire (`X`), le taux de consommation de CO<sub>2</sub> par carbonatation (`rcbn`) et la vitesse d'entrée du CaO dans le réacteur (`us`) par 0 car ils n'interviennent plus dans le modèle sans capture de CO<sub>2</sub>.

Nom	mode 0	mode 1	Différence
CH <sub>4</sub>	0,00417 mol L <sup>-1</sup>	0,00478 mol L <sup>-1</sup>	+14,7 %
H <sub>2</sub>	0,02322 mol L <sup>-1</sup>	0,02302 mol L <sup>-1</sup>	-0,9 %
CO	0,00235 mol L <sup>-1</sup>	0,00250 mol L <sup>-1</sup>	+6,5 %
CO <sub>2</sub>	0,00344 mol L <sup>-1</sup>	0,00387 mol L <sup>-1</sup>	+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 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, dans un modèle sans carbonatation, on

remarque que le  $\text{CH}_4$ , le  $\text{CO}_2$  et le  $\text{CO}$  augmentent de 14,7 %, 12,6 % et 6,5 % respectivement. Le  $\text{H}_2$  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  $u_s$  à l'aide du mode 2. Nous l'avons fait varier de  $0,001 \text{ m s}^{-1}$  à  $0,025 \text{ m s}^{-1}$ . (((EXPLIQUER GRAPHIQUES)))

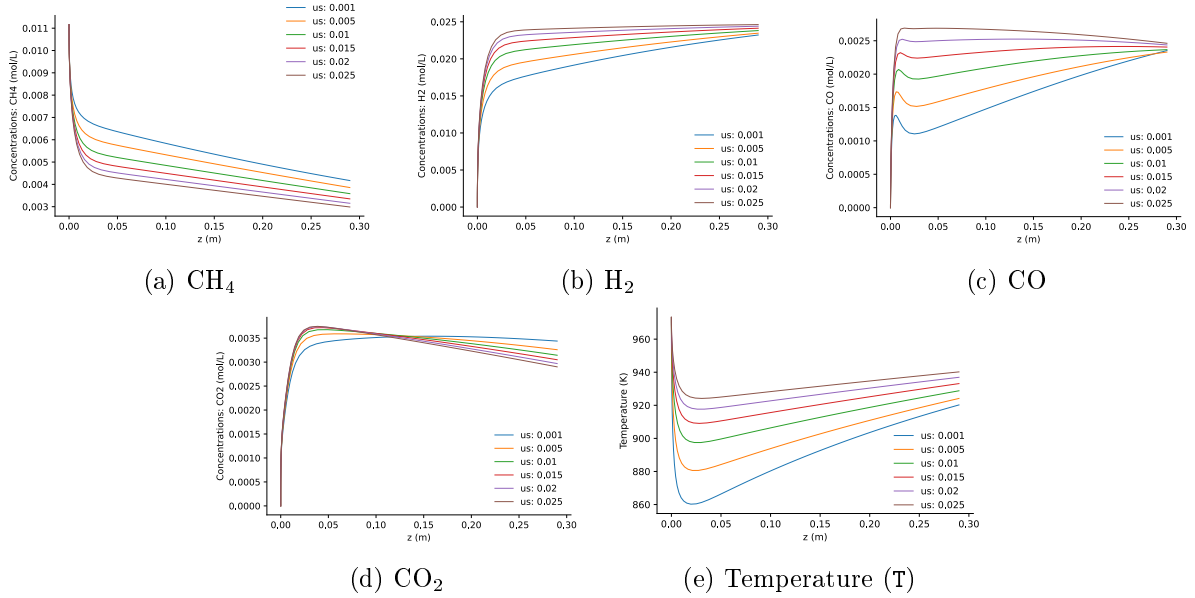


FIGURE 3.5 – Concentrations et températures ( $T$ ) des gaz

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

Pour analyser le flux des gaz, nous avons fait varier  $u_g$  à l'aide du mode 3 en changeant la valeur de  $1 \text{ m s}^{-1}$  à  $3 \text{ m s}^{-1}$  avec un pas de 0.5. (((EXPLIQUER 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  $\text{H}_2\text{O}$  et  $\text{CH}_4$  de 3 à 0.3. (((EXPLIQUER GRAPHIQUES)))

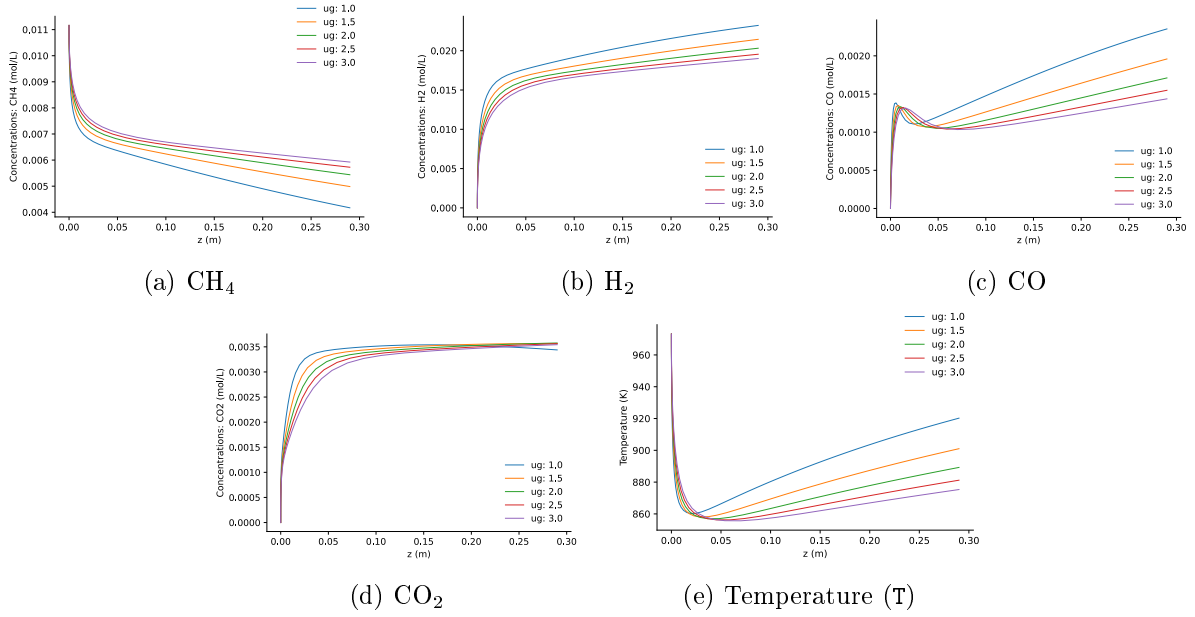


FIGURE 3.6 – Concentrations et températures (T) des gaz

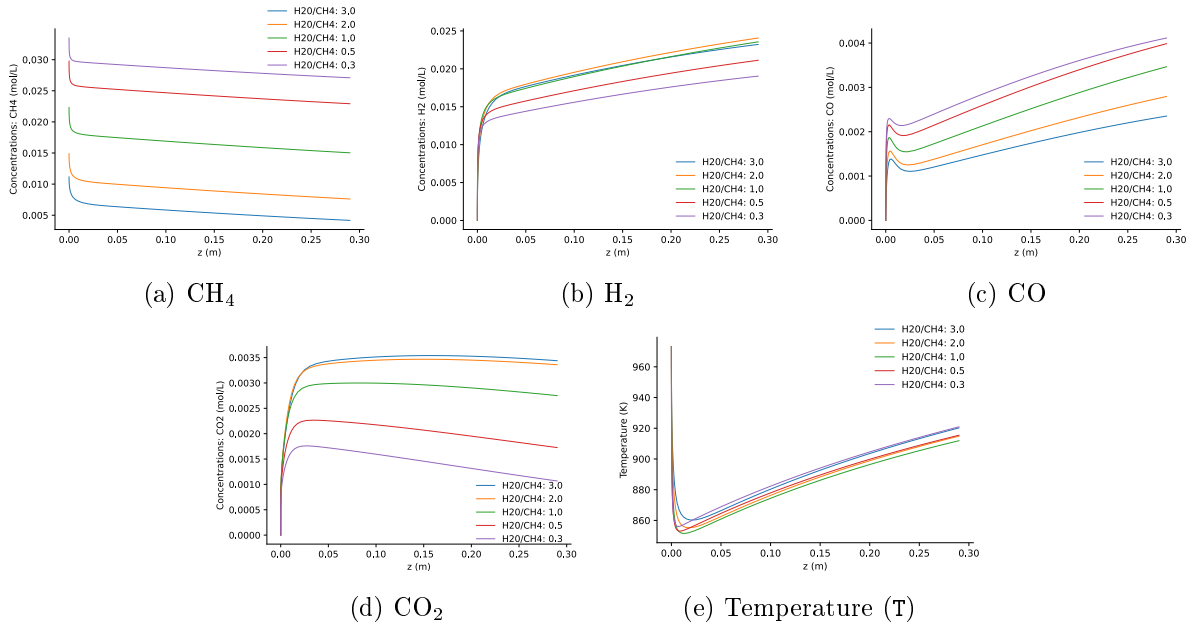


FIGURE 3.7 – Concentrations et températures (T) des gaz



# Question 4

## 4.1 Taux de CO<sub>2</sub> optimal par rapport à $u_s$

`optimise_us` crée la fonction du pourcentage de CO<sub>2</sub> en sortie par rapport à  $u_s$ , ce qui nous permet de déterminer un  $u_s$  optimal pour une concentration donnée. Cette fonction prend 5 arguments en entrée :

- $Y$ , le pourcentage de CO<sub>2</sub> souhaité à la sortie du réacteur
- $u_s$ , la valeur de  $u_s$
- $CO$ , 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<sub>2</sub> divisé par la somme des concentration finale de CH<sub>4</sub>, H<sub>2</sub>, CO et de CO<sub>2</sub>. Le tout est ensuite soustrait par  $Y$  pour corriger le décalage vertical.

## 4.2 Recherche d'un $u_s$ optimal

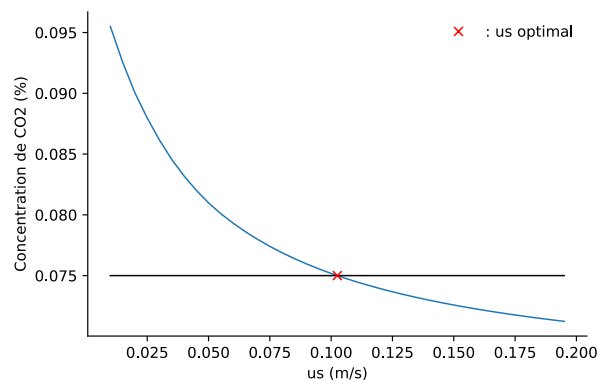


FIGURE 4.8 – Valeur optimale de  $u_s$

Dans ce cas, nous devons trouver la valeur optimale de  $u_s$  pour une concentration en CO<sub>2</sub> de 7,5%. On peut voir la fonction `optimise_us` ainsi que la valeur optimale sur la figure 4.8 qui représente  $u_s$  en fonction du pourcentage de CO<sub>2</sub> parmi les gaz secs en sortie. Notre fonction `secant` en mode `hybrid` nous renvoie une racine située en 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  $x_0$  et  $x_1$  égales, alors on s'inspire de la méthode de la bisection 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 à la bisection. 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 à utiliser.

### 4.3 Effet de la température sur le $u_s$ optimal

La figure 4.9 illustre la variation du  $u_s$  optimal en fonction de la température ( $T$ ). Il est clair que  $u_s$  dépend de la température. Nous le voyons décroître 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  $\text{CO}_2$  donné est petite.

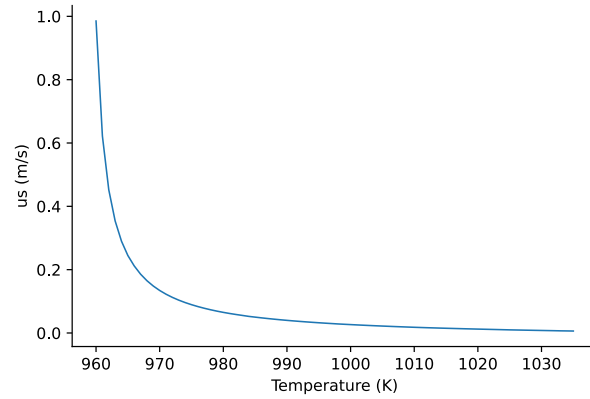


FIGURE 4.9 – L'effet de la température ( $T$ ) sur le  $u_s$  optimal