Premier bachelier en sciences de l'ingénieur
Année académique 2019-2020

Projet d'introduction à l'analyse numérique : Etude du cycle carbone-climat pour une planète hypothétique

Travail réalisé par Louis Hogge, Hugo Jacobs et Antoine Martin

1 Projet

1.1 Méthodes de la sécante et de la bissection

Ces deux premiers programmes permettent de calculer une racine de fonctions linéaires ou non-linéaires. Ils nous seront utiles pour la suite du projet. Les arguments d'entrée et de sortie des fonctions sécante et bissection nous sont donnés. En entrée, elles prennent la fonction dont on doit trouver la racine, deux points initiaux ainsi qu'une tolérance et en sortie, renvoient la racine trouvée et un statut.

Dans les deux cas, nous avons implémenté une boucle appliquant l'algorithme et se terminant, si la fonction converge, lorsque l'image de la valeur calculée s'est suffisamment rapprochée de la racine. La dernière image calculée sera donc celle de la racine et, si celle-ci est trouvée, la fonction renvoie un statut de 0. Le nombre d'itérations et le temps d'exécution dépendent donc fortement de la tolérance entrée en argument. Afin de réduire le temps d'exécution au minimum, nous ne faisons appel à la fonction entrée en argument qu'une seule fois par boucle.

Il y a deux cas pour lesquels les algorithmes ne peuvent pas fonctionner. Le premier est le cas où la fonction entrée ne possède aucune racine. Pour détecter ce cas, nous comptons le nombre d'itérations de la boucle. Lorsque celui-ci devient trop élevé, nous en concluons que l'algorithme ne converge pas vers une racine et le programme s'arrête en renvoyant un message d'erreur et un statut de -1. Nous avons choisi de fixer ce nombre à 60, car il permet d'atteindre une précision largement suffisante (de l'ordre de 2^{-60} pour la bissection, plus précis encore pour la sécante). Le deuxième cas est celui pour lequel les points entrés en arguments ne permettent pas de trouver la racine. On entend par là, pour la bissection, des points n'ayant pas des images de signes contraires et, pour la sécante des points ayant des images identiques. Ces cas sont testés dès le début du code et, s'ils sont détectés, stoppent immédiatement l'exécution du programme et renvoient un message d'erreur ainsi qu'un statut de 1.

En plus de ces cas, nous avons également envisagé une erreur supplémentaire possible pour la sécante. En effet, l'algorithme de la sécante comprenant un quotient, il faut envisager le cas où le dénominateur de celui-ci vaut zéro et donc où le quotient n'est pas un nombre fini. Dans ce cas, le programme s'arrête à nouveau et renvoie un message d'erreur ainsi qu'un statut de -1.

Nous avons donc suivi une démarche quasiment identique pour la bissection et la sécante à savoir l'implémentation de l'algorithme pour le cas général, puis l'ajout de conditions pour les différentes erreurs qui peuvent survenir.

1.2 Mise en équation du modèle climatique

Afin de mettre en équation le modèle climatique, nous implémentons une fonction intitulée `odefun` prenant en entrée le temps t et un vecteur de cinq composantes contenant les variables T (la température en $°C$), C_a (le carbone dans l'atmosphère en Gt), C_s (le carbone dans le sol en Gt), W_a (l'eau dans l'atmosphère en g/cm^2), W_s (l'eau dans le sol en g/cm^2) et permettant de calculer la valeur des dérivées de ces dernières qui sont retournées dans le vecteur de sortie. Cette fonction définit le modèle climatique d'une planète hypothétique sous forme d'un système de cinq équations différentielles ordinaires à partir de trois bilans : le bilan d'énergie de la Terre (influençant la température), le bilan du carbone (système couplé atmosphère-sol) et le bilan de l'eau (système couplé atmosphère-sol). En résolvant ces équations, on obtient les dérivées souhaitées.

Nous avons décidé de séparer cette mise en équation en plusieurs fonctions et donc de faire les calculs des données nécessaires aux équations différentielles dans des fonctions différentes de la fonction

principale `odefun`. Cela a permis de faciliter notre travail, d'améliorer la lisibilité du code et de tester les fonctions au fur et à mesure que nous les créions.

Lors du calcul d'une des données, à savoir la productivité P, il est nécessaire de construire des fonctions g_T et g_W par interpolation à partir de données fournies. Pour cela, nous avons décidé d'utiliser la fonction `interp1` implémentée dans MATLAB, car celle-ci permet de renvoyer directement l'image d'une valeur entrée sans devoir passer par des coefficients, comme c'est le cas avec la plupart des autres fonctions permettant d'interpoler. En dehors de l'intervalle connu, nous avons fixé les valeurs à 0. Pour choisir quel type d'interpolation utiliser, nous avions principalement deux options : l'interpolation polynomiale et celle par splines cubiques. La première possédant un mauvais comportement aux bornes de l'intervalle et se comportant mal lorsque le nombre de points donnés est important, nous avons décidé de plutôt utiliser la deuxième option qui permet d'éviter ces problèmes et d'amener une bonne précision.

On illustre l'interpolation de la fonction g_T sous forme de graphique :

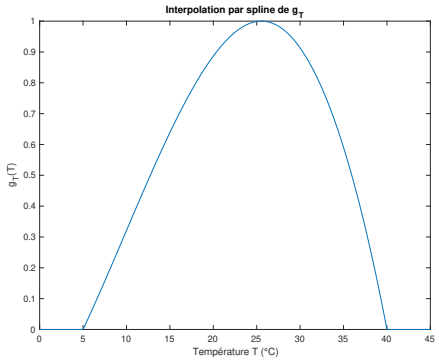

FIGURE 1 - Interpolation de g_T

Ci-dessous, un tableau reprenant les résultats obtenus pour différentes conditions, le temps entré en argument n'ayant pas encore d'influence à ce stade du projet. Les conditions initiales sont exprimées sous la forme $[T\ C_a\ C_s\ W_a\ W_s]$ et les valeurs des dérivées trouvées sous la forme $[dT\ dC_a\ dC_s\ dW_a\ dW_s]$.

Conditions initiales	Valeurs des dérivées
[20 1000 500 10 10]	[0.4676 -117.0770 117.0770 43.2405 -43.2405]
[15 250 250 5 5]	[-2.2655 -71.9201 71.9201 16.8474 -16.8474]
[15 10000 5000 10]	[3.3180 334.0061 -334.0061 37.6695 -37.6695]

Dans ces résultats, on remarque que les dérivées de la quantité de carbone contenu dans l'atmosphère et de la quantité de carbone contenu dans le sol sont opposées ; il en est de même pour l'eau. On peut donc en conclure que pour une certaine augmentation de la quantité de carbone ou d'eau contenue dans l'air, leur quantité contenue dans le sol diminue de la même façon. De même, pour une diminution de leur quantité contenue dans l'air, leur quantité contenue dans le sol augmente également de la même façon. D'un point de vue physique, ce constat est tout à fait logique : la quantité totale de carbone ou d'eau étant constante.

1.3 Modélisation du climat

Nous résolvons maintenant le système d'équations différentielles mis en équation dans la fonction `odefun` créée précédemment.

1.3.1 Implémentation d'Euler explicite

Pour ce faire, nous commençons par implémenter la méthode d'Euler explicite vue au cours. Nous avons décidé de lui donner une forme identique à celle d'`ode45` en ce qui concerne les arguments d'entrée et de sortie pour pouvoir plus facilement changer de méthode en cas de besoin. La fonction Euler prendra donc en argument la fonction permettant de donner la valeur des dérivées, un intervalle de temps, des conditions initiales ainsi qu'une tolérance et retournera en sortie un tableau contenant toutes les valeurs calculées et un vecteur contenant toutes les valeurs du temps correspondantes.

Tout d'abord, il nous a semblé important de commencer par créer un tableau dont la largeur dépend de la taille du vecteur de conditions initiales et la longueur de l'intervalle de temps et du pas que nous choisirons plus tard. Nous avons également créé un vecteur colonne de même longueur. Nous fixons leur taille, car rajouter une case à chaque itération est un processus beaucoup plus coûteux en terme de mémoire et de temps. Ensuite, nous avons implémenté l'algorithme d'Euler dans une boucle. A chaque itération, celle-ci remplit le tableau et complète le vecteur temps. La nombre d'itérations dépend donc de la taille de l'intervalle et du pas. A la fin de la boucle, le tableau et le vecteur sont donc ainsi remplis de valeurs, celles du tableau tendant normalement vers un équilibre. La boucle s'arrête lorsque le temps final est atteint. Pour vérifier que les valeurs tendent bien vers un équilibre, nous regardons si les cinq dernières valeurs de la première ligne du tableau sont bien espacées d'une valeur ne dépassant pas la tolérance entrée en argument. Si aucune tolérance n'est entrée, la tolérance utilisée par défaut est de 0,1. Notre dernière incertitude était donc le choix du pas. Nous avons pris en compte deux critères : la précision et le temps d'exécution. Ci-dessous, un tableau reprenant les temps d'exécution obtenus pour différents pas.

Pas	Temps
0,001	33.236824 secondes
0,01	3.487093 secondes
0,1	0.423966 secondes

Le pas de 0,1 est donc le plus performant en terme de temps et n'entraîne pas de perte de précision trop importante. En effet, si on continue d'augmenter le pas, cette perte devient trop importante et les valeurs ont du mal à tendre vers un équilibre. Pour illustrer ceci, voici, sous forme de graphique, une comparaison de valeurs retournées par Euler à partir d'une même fonction et de conditions initiales identiques pour un pas de 1 et pour un pas de 0,1 :

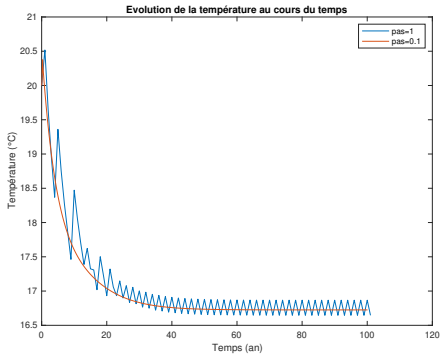

FIGURE 2 - Comparaison de pas

Au vu des résultats de ces tests, nous avons décidé de choisir un pas de 0,1, cela nous semblant idéal

tant au niveau de la précision que du temps d'exécution.

Ci-dessous sont illustrées les résolutions du système d'équations différentielles de `odefun` sous forme de graphiques. Ces résultats sont séparés en trois graphiques afin d'avoir des échelles similaires et de permettre une meilleure lisibilité. On retrouve : la température (en $°C$), la quantité de carbone (C_s et C_a) et la quantité d'eau (W_s et W_a). Au vu de leur signification physique, les variables C_a, C_s, W_a et W_s ne peuvent être négatives. Dès lors, lorsque la résolution amène à des variables négatives, elles sont fixées à 0 une fois entrées dans la fonction `odefun`. Les conditions initiales, l'intervalle de temps (en années) et la tolérance entrés pour obtenir ces résultats sont :

— Température T=20$°C$
— C_a et C_s=1000 Gt
— W_a et W_s=15 g/cm^2
— Intervalle de temps=[0 100]
— Tolérance=0.001

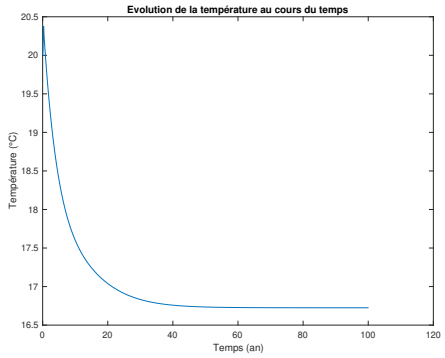

FIGURE 3 - Résultats pour la température (Euler)

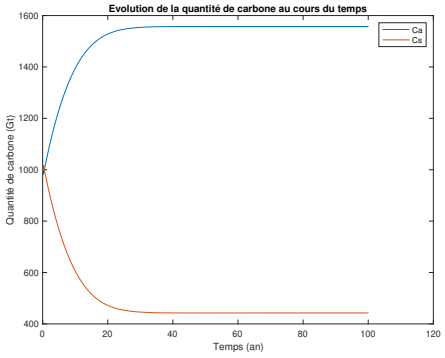

FIGURE 4 - Résultats pour la quantité de carbone (Euler)

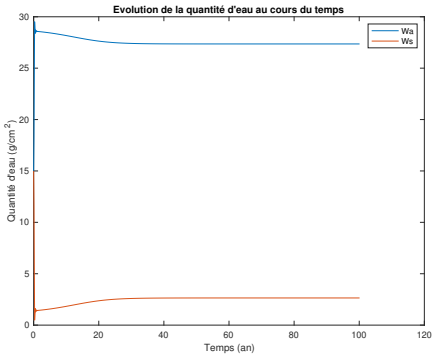

FIGURE 5 - Résultats pour la quantité d'eau (Euler)

Les oscillations pouvant apparaître au début des courbes disparaissent si nous diminuons le pas, mais n'ont pas d'influence sur les résultats à l'équilibre.

Il nous est également demandé de déterminer pour 3 différentes conditions initiales si les points à l'équilibre correspondent à un désert, une planète glaciaire, un climat tempéré,etc. Les conditions sont exprimées sous la forme $[T\ C_a\ C_s\ W_a\ W_s]$.

Conditions initiales	Points à l'équilibre	Climat
[20 10000 10000 1 1]	[39.557 20000 0.001 2 1.097e-13]	Désert
[10 100 100 1 1]	[-51.488 200 1.047e-05 0.007 1.993]	Polaire
[20 10000 10000 20 20]	[16.406 20000 0.001 2.946 37.054]	Tempéré

Pour les premières conditions, la température à l'équilibre avoisine les $40°C$ et de plus, la quantité totale d'eau est très faible. On en conclut donc que ces conditions correspondent à un climat désertique chaud. Les deuxièmes conditions amènent elles aussi à un climat très sec, mais cette fois extrêmement froid. Elles correspondent aux conditions d'un climat polaire et pourraient également nous faire penser à un climat désertique froid. Les dernières conditions amènent à des températures d'environ $16°C$ donc un peu plus élevées que la moyenne belge qui est d'environ $10°C$. Le climat est également fort humide. Cela nous fait penser à un climat tempéré, mais au vu des températures, pourrait également correspondre, par exemple, à un climat méditerranéen.

1.3.2 Utilisation d'`ode45`

La seconde méthode de résolution du système d'équations différentielles est l'utilisation de la fonction `ode45` appliquant la méthode de Runge-Kutta. Cette fonction étant implémentée dans MATLAB, nous ne devons donc pas le faire nous-mêmes. Nous avons illustré sous forme de graphiques les résultats obtenus avec cette fonction pour les mêmes conditions initiales qu'au point précédent. Nous superposons ces résultats avec ceux obtenus grâce à Euler afin de comparer les deux méthodes.

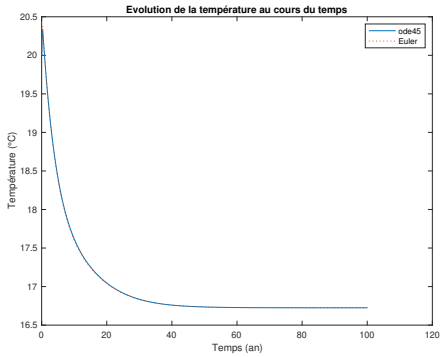

FIGURE 6 - Résultats pour la température (`ode45`)

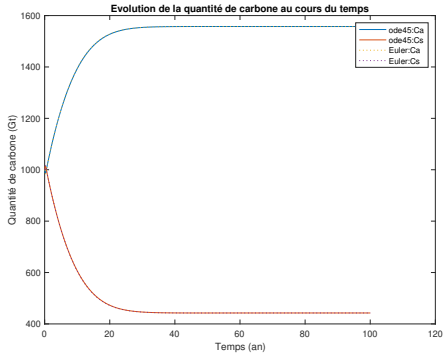

FIGURE 7 - Résultats pour la quantité de carbone (`ode45`)

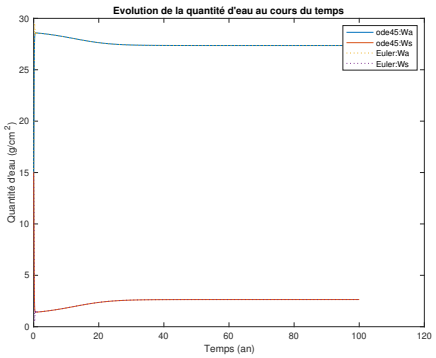

FIGURE 8 - Résultats pour la quantité d'eau (`ode45`)

Nous remarquons que la différence entre les courbes données par Euler et par `ode45` est extrêmement petite et qu'il est très difficile de les différencier sur des graphiques. Les deux méthodes nous renvoient donc bien des résultats identiques.

1.3.3 Choix entre Euler et `ode45`

Pour choisir quelle méthode nous allions utiliser dans la suite du projet, nous avons commencé par comparer les temps d'exécution des deux fonctions pour des intervalles de temps entrés différents.

Temps avec Euler(secondes)	Temps avec `ode45`(secondes)
0.788	0.699
0.435	0.336
1.933	2.062
3.748	3.796

Nous remarquons que les deux fonctions ont des temps d'exécution sensiblement identiques. Cependant, nous avons choisi de plutôt utiliser la méthode d'Euler, car celle-ci détecte les cas où l'équilibre n'est pas atteint. Cela nous permet de savoir quand nous devons élargir l'intervalle de temps dans le but d'atteindre cet équilibre.

1.4 Recherche d'un paramètre du modèle pour atteindre un équilibre climatique donné

Dans cette partie du projet, nous déterminons un intervalle de différents paramètres permettant d'atteindre à l'équilibre un intervalle de température donné (15 à 17 $^{\circ}C$). Nous avons dû étudier trois paramètres du modèle : S qui est l'énergie que la terre reçoit du soleil (W/m^2), α_a qui est l'albédo de l'atmosphère supérieure sans nuages ainsi que C_a qui est la quantité de carbone contenue dans l'atmosphère (Gt).

Pour ce faire, nous avons créé trois fonctions de recherche : `rechercheparamS` , `rechercheparamAlpha` et `rechercheparamCa`. Celles-ci prennent en argument les deux températures ($^{\circ}C$) de l'intervalle à atteindre, un vecteur de conditions initiales du modèle climatique étudié, ainsi qu'une valeur minimale et une valeur maximale du paramètre entre lesquelles se trouvent les valeurs recherchées. Nous verrons l'intérêt de ces valeurs plus loin dans ce rapport. Ces fonctions renvoient un intervalle du paramètre. Dans celui-ci, la température à l'équilibre est comprise dans l'intervalle de température voulu. Elle renvoie également un statut afin de vérifier si la fonction s'est terminée correctement.

Avant de créer les fonctions de recherche de paramètre, nous avons d'abord implémenté trois fonctions, une pour chaque recherche. Celles-ci serviront à trouver quand la température à l'équilibre sera égale à une certaine température donnée en fonction d'un des paramètres à chercher. Pour trouver la température à l'équilibre, nous avons utilisé la méthode d'Euler implémentée plus tôt et appliquée aux équations différentielles d'`odefun`. Cependant, comme le système d'équations initial considérait un S et un α_a fixe, nous avons dû modifier la fonction `odefun` afin que S et α_a puissent varier, C_a étant déjà une des conditions initiales au départ. Chacune des fonctions implémentées permet alors de trouver la valeur de la différence entre une température donnée et une température à l'équilibre en fonction d'un des trois paramètres. Par conséquent, lorsque cette fonction renvoie 0, on sait que la température à l'équilibre est égale à celle recherchée et on connaît donc la valeur du paramètre nécessaire pour atteindre celle-ci.

Ensuite, pour les fonctions de recherche de paramètre, nous avons appliqué trois fois le même raisonnement. Nous avons appliqué la fonction de la sécante, créée précédemment, aux fonctions que nous venions de créer. La racine trouvée correspond alors à la valeur du paramètre pour que la température à l'équilibre soit égale à celle recherchée. Ce raisonnement est appliqué une fois pour chaque borne de l'intervalle de température donné. Nous utilisons comme points initiaux, pour la méthode de la sécante, les valeurs minimales et maximales du paramètre entrées en argument. Les deux valeurs ainsi obtenues constituent donc les bornes de l'intervalle du paramètre recherchées. Vu leur signification physique, les paramètres ne peuvent être négatifs. Si la résolution amène à ce cas, ils sont fixés à 0 dans `odefun`. Dans le cas où la sécante rencontre une erreur, un statut de 1 est alors renvoyé par la fonction qui se termine immédiatement.

Cela peut survenir s'il n'existe pas de valeur du paramètre pour laquelle la température donnée est atteinte ou si la méthode d'Euler n'atteint pas un équilibre, l'intervalle de temps étant trop petit par exemple.

Nous avons choisi d'utiliser la méthode de la sécante et non celle de la bissection également vue précédemment, car la sécante admet moins d'hypothèses sur les deux points initiaux qu'elle prend en argument. Elle est donc plus robuste. En effet, pour la bissection, il est nécessaire de connaître deux points dont les images sont de signes opposés, ce qui n'est pas toujours évident dans le cas qui nous intéresse.

Ci-dessous, un tableau reprenant les résultats pour différentes conditions initiales, mais toujours pour un intervalle de température compris entre 15 et 17$°C$:

Conditions initiales	S	α_a	C_a
[15 2000 2000 12 12]	[309.701 318.219]	[0.1276 0.1385]	Ne converge pas
[20 1000 1000 15 15]	[332.068 341.283]	[0.0984 0.1101]	[9180.09 16769.07]
[10 5000 5000 5 5]	[322.340 326.087]	Ne converge pas	[8645.50 18219.15]

On remarque donc bien que pour certaines conditions initiales, il n'est pas possible de trouver un paramètre permettant d'atteindre l'objectif de température.

Nous avons également trouvé intéressant d'observer, sous forme de graphiques, l'effet de ces différents paramètres sur la température, pour chacun avec les mêmes conditions initiales ([20 1000 1000 15 15]).

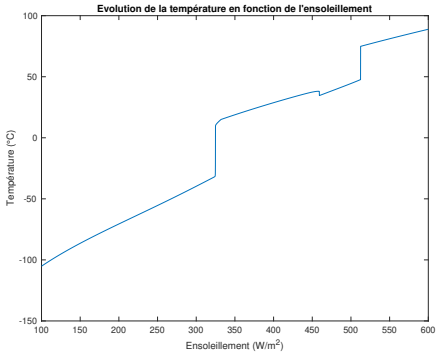

FIGURE 9 - Effet de l'ensoleillement sur la température

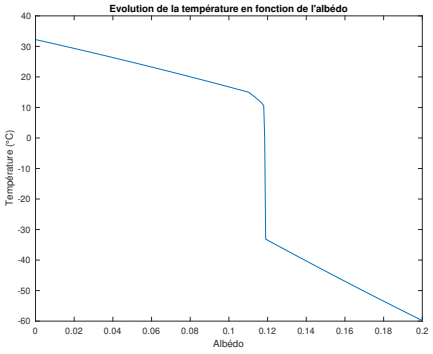

FIGURE 10 - Effet de l'albédo sur la température

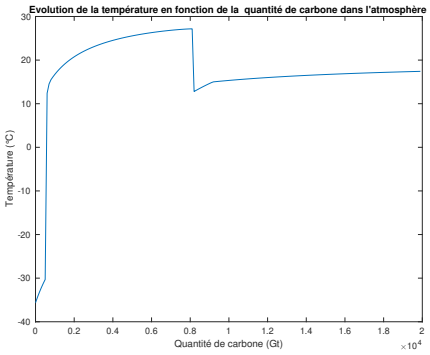

FIGURE 11 - Effet de la quantité de carbone dans l'atmosphère sur la température

On remarque pour le dernier graphique que deux intervalles peuvent correspondre à une température comprise enntre 15 et 17 $°C$. On trouve l'un ou l'autre en fonction de l'intervalle de valeurs mnimum ou maximum entré en argument.

En pratique, ces fonctions pourraient donc mesurer la variation de paramètres à appliquer à un système afin d'en réduire la température, par exemple dans le but de lutter contre le réchauffement climatique. En effet, nous savons qu'actuellement de nombreuses solutions modifiant l'ensoleillement, l'albédo ou la quantité de carbone dans l'atmosphère sont étudiées.

1.5 Impact de la réduction d'émission sur le climat

Dans cette dernière partie de notre projet, nous créons une fonction permettant de connaître le pourcentage de réduction d'émission de carbone anthropique à appliquer de 2020 à 2100 par rapport à un scénario donné pour que la température ait baissé de $2°C$ en 2100.

Pour cette fonction, nous appliquons le même principe que pour les fonctions de recherche du point précédent. Cette fonction prend donc en argument une température correspondant à la valeur de la diminution voulue en 2100 et un vecteur de conditions initiales du modèle climatique. Elle renvoie le pourcentage recherché ainsi qu'un statut qui permet de vérifier si la fonction s'est terminée correctement.

Tout d'abord, avant de commencer il était nécessaire d'encoder dans `odefun` les valeurs données dans le scénario. Nous avions à notre disposition les émissions de l'année 1860 à 3000 et nous avons donc dû interpoler ces valeurs afin d'obtenir une fonction continue pour tout temps entré. Pour les valeurs de 2020 à 2100, nous avons également rendu possible qu'un pourcentage de diminution soit entré dans `odefun` et appliqué à ce scénario de base pour cet intervalle de temps. Nous avons ensuite, comme au point précédent, créé une fonction qui permet de savoir si la température à l'équilibre en fonction d'un paramètre est égale à celle que l'on cherche à atteindre. Cependant, ici, la température que l'on cherche à atteindre est la température en 2100 du scénario de base, calculée avec Euler, dont on retire la valeur de la diminution donnée en entrée. La fonction fait alors la différence entre cette température et une seconde température calculée, toujours en 2100, par Euler, avec un pourcentage de diminution donné en entrée. Donc, si cette fonction renvoie 0, on sait qu'on a atteint la diminution de température souhaitée.

La fonction permettant de renvoyer le pourcentage recherché est donc une simple application de la méthode de la sécante à la fonction implémentée plus tôt. Les deux points initiaux sont 0 et 1, car un pourcentage est toujours compris entre ces deux valeurs. La racine renvoyée est donc le pourcentage nécessaire pour la diminution de température voulue. Elle sera renvoyée par la fonction avec un statut de 0. Si par

contre la fonction sécante rencontre un problème, le statut renvoyé est 1. Il est également possible que la baisse de température soit trop importante et que la fonction pourcentage de réduction nécessaire soit supérieur à 100%. Dans ce cas, on renvoie un pourcentage de 100% et un message d'erreur.

Ci-dessous, un tableau reprenant les résultats pour différentes conditions, mais toujours pour une réduction de température de $2°C$:

Conditions initiales	% de réduction nécessaire
[20 1000 1000 15 15]	73.4640
[20 1000 1000 5 5]	62.6157
[15 250 250 10 10]	40.7077

En pratique, de telles réductions d'émission nous semblent difficiles à mettre en application. En effet, selon ses résultats, il faudrait réduire chaque année les émissions au minimum de 40% et jusqu'à 75% pour certaines conditions par rapport au scénario prévu. Sachant que la plupart des accords climatiques demandent une réduction d'environ 5% par an et ne sont généralement pas respectés par les pays, des réductions de 40% à 75% ne nous semblent donc pas concevables.

2 Conclusion

En conclusion, le modèle climatique que nous avons implémenté permet d'obtenir une estimation de l'impact de certains paramètres sur le climat. Il pourrait être utilisé pour calculer l'efficacité de certaines solutions, ainsi que leurs effets sur le réchauffement climatique. Par exemple, la capture de CO_2, la modification de l'albédo ou de l'énergie solaire incidente peuvent être étudiées avec les fonctions de recherche de paramètre du point 1.4. Il en est de même pour l'étude de la réduction des émissions anthropiques souvent abordées dans les décisions politiques traitant du réchauffement climatique avec le point 1.5. Cependant, le modèle créé reste très imprécis et ne permet pas d'obtenir des résultats exacts pour des planètes aussi complexes que la Terre. En effet, celle-ci possède un nombre important de climats et donc de conditions différentes que nous ne pourrons pas évaluer avec ce modèle.

Il nous montre cependant que les ingénieurs peuvent avoir un rôle important à jouer dans la mise en place de nouveaux moyens de lutte contre le réchauffement climatique, notamment par le développement de simulations numériques comme celle-ci.