université PARIS-SACLAY

Modèle stochastique d'une population de *Zea mays L*

UE Projet Bio-mathématiques

Elora Vigo, Margot Hully, Laura Fuentes Vicente

Année 2021/2022

Table des matières

1 Introduction **3**

2 Développement du maïs **3**

3 Modèle **4**

3.1 Simplification du développement du maïs 4

3.2 Hypothèses 4

3.3 Paramètres biologiques 5

3.3.1 Coefficients de transition 5

3.3.2 Probabilité de former une panicule 5

3.4 Méthode de programmation 5

4 Résultats **6**

4.1 Résultats méthodologiques 6

4.2 Résultats biologiques 7

5 Conclusion **8**

6 Glossaire **9**

7 Annexes **11**

7.1 Annexe 1 : Inflorescence du maïs 11

7.2 Annexe 2 : Paramètre de la loi de Poisson 11

7.3 Annexe 3 : Figure 5a de l'article 12

7.4 Annexe 4 : Code Python 13

1 Introduction

La production de maïs a un impact économique important dans de nombreux pays d'Amérique. Comme toutes les plantes, le maïs est soumis, au cours de sa croissance, à différentes contraintes abiotiques et biotiques. Parmi ces stress, on retrouve par exemple toutes les maladies causées par des micro-organismes externes. Pour minimiser les pertes, un des enjeux majeurs est donc le contrôle de la propagation des agents pathogènes. Les tissus de revêtement de la plante constituent la première lignée de défense contre ceux-ci [1]. Les agents pathogènes peuvent franchir cette première barrière par différentes voies, notamment grâce aux dégâts foliaires causés par les ravageurs. Parmi les principaux ravageurs du maïs, on trouve plusieurs insectes, dont les Coléoptères, Lépidoptères, Diptères, Hémiptères ainsi que des Arachnides [2]. Selon le stade de développement et la surface foliaire des plantes, ces derniers ne les colonisent pas de la même manière.Ces deux paramètres dépendent grandement des conditions du milieu dans lequel elles vivent (comme la température ou l'humidité). Comprendre et modéliser la dynamique de croissance de populations de maïs pourrait donc servir de base pour étudier la dynamique de propagation des vecteurs et donc a posteriori celle des agents pathogènes. Néanmoins, pour aboutir à une telle modélisation, il faudrait disposer d'un modèle d'étude de l'évolution de la surface foliaire. Comme les auteurs ne citent pas un tel modèle compatible avec le leur, nous avons décidé de nous focaliser sur l'effet de la température sur la croissance d'une population de maïs et plus précisément sur l'effet de l'augmentation de la température due au réchauffement climatique sur cette croissance.

2 Développement du maïs

Le maïs est une plante herbacée annuelle de 1 à 3 mètres. L'inflorescence mâle, ou panicule, se situe à l'apex de la tige. Elle est composée d'épillet eux-mêmes composés de fleurons (ou fleurs) qui contiennent les étamines (et donc les sacs polliniques). Les inflorescence femelles, ou épis, sont situés à l'aisselle des feuilles de la plante [3]. Un épi peut contenir 1000 ovules, mais seulement 4 à 500 seront fécondés [4]. Chaque ovule de l'épi développe une soie (les styles), ce sont les "fils" visibles à l'extérieur de l'épi [Annexe 1].
Le développement du maïs passe par différentes phases : végétative, reproductive et la phase de développement et maturation de la graine [5]. Quand la graine est mature, toutes les activités métaboliques cessent. Elles ne reprennent que lorsque les conditions sont favorables, on a alors germination. La racine et la tige s'allongent, les feuilles s'initient et se développent. L'initiation de primordia (ébauche d'une nouvelle feuille) se fait à intervalles de temps réguliers nommés plastochrone [A]. Lors de la phase de reproduction, les organes reproducteurs sont produits. L'apex de la tige se différencie en panicule, puis quelques jours après les épis sont prêts pour la fécondation [6]. Les étamines des fleurons libèrent du pollen qui est transporté par le vent jusqu'aux soies des épis. A ce moment-là, le pollen germe et produit un tube pollinique à l'intérieur de la soie, dans le but de fertiliser l'ovule se trouvant à son extrémité. Lors de la dernière phase, les graines se gorgent d'eau et de produits provenant de la photosynthèse. Au bout d'un moment, une couche de cellules au niveau du point d'attachement de la graine à l'épi devient noire. Cela crée ainsi une barrière empêchant le mouvement de sucres de l'épi vers la graine [7].

3 Modèle

3.1 Simplification du développement du maïs

Pour modéliser le développement d'une population de maïs, les auteurs ont décidé de diviser la population en 5 grandes classes correspondant aux principaux stades de développement du maïs : graine (S), jeune plant (P), développement des fleurs (F), développement de la graine (G) et stade de la couche noire (BL). Les individus de la population sont chacun dans une classe donnée à un instant t et ils peuvent en changer suite à divers événements caractérisés par un coefficient ω : émergence ω_{emr} (S $\rightarrow$ P), initiation du panicule ω_{tir} (P $\rightarrow$ F), pollinisation et fécondation ω_{pfr} (F $\rightarrow$ G) et maturation de la graine ω_{mtr} (G $\rightarrow$ BL). Pour quatre de ces classes s'ajoutent des sous-catégories, notées P_k, F_k, G_k, BL_k où k indique le nombre de feuilles ($k \geq 6$ car une plante émerge avec 6 feuilles). Le modèle prend donc en compte une cinquième transition qui n'a lieu qu'au stade P : l'initiation de primordia ω_{lir} ($P_k \rightarrow P_{k+1}$). Une plante au stade P peut donc ou bien former une nouvelle feuille, ou bien la panicule. Pour déterminer quelle transition a lieu, on applique la "probabilité de former une panicule en ayant k feuilles".

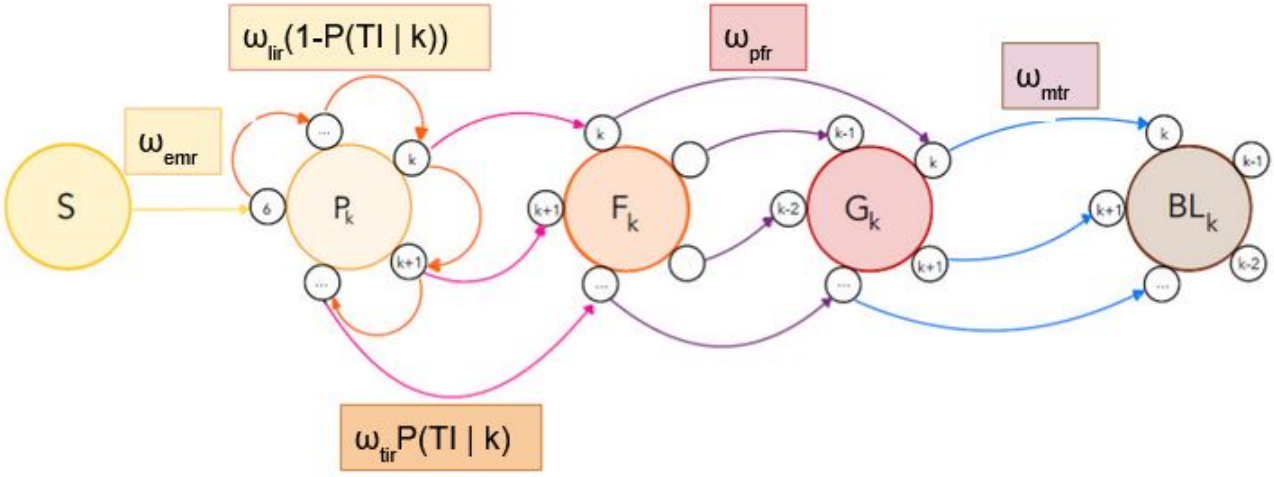


Figure 1: Schéma du modèle du développement du maïs (stades et transitions)
S: stade semis, P_k: initiation de nouvelles feuilles, F_k: développement des inflorescences, G_k: développement des graines, BL_k: maturation des graines
Les encadrés correspondent aux taux de transition entre deux états, avec ω des coefficients et TI l'événement "initialisation de la panicule".

3.2 Hypothèses

Afin de simplifier la réalité, les auteurs ont posé diverses hypothèses. Ils ont tout d'abord choisi de ne prendre en compte que la température comme facteur environnemental. C'est en effet un paramètre qui influence le développement du maïs de manière importante et qui est peu contrôlable dans les cultures. De plus, ils ont supposé qu'il était possible de réduire suffisamment les intervalles de temps pour qu'un seul événement se déroule à la fois (négligence des événements multiples). Par ailleurs, on a une propriété d'absence de mémoire concernant le temps qui sépare deux événements. On considère également que le modèle suit un processus markovien (l'état du système ne dépend que de l'état précédent). De surcroît, dans le modèle présenté dans l'article, les plantes de maïs ne meurent jamais (que ce soit par vieillesse, pathologie ou herbivorie). Enfin, on ne modélise aucune variabilité au sein de la population en termes de capacités de croissance (les coefficients de transition sont communs à toute la population). Prendre des valeurs moyennes semble suffisant dans le cadre de ce modèle puisqu'on considère une population dans son ensemble sans s'intéresser au devenir de chaque individu en particulier.

3.3 Paramètres biologiques

3.3.1 Coefficients de transition

Pour modéliser la croissance d'une population de maïs, nous devons donc dans un premier temps déterminer les coefficients des diverses transitions (ω_i). Pour ce faire, les auteurs de l'article ont utilisé un second modèle : le modèle thermodynamique de Sharpe et DeMichele [8] adapté aux modèles biologiques par Schoofield et Sharpe [9]. Ce modèle part du principe qu'un processus biologique (comme l'émergence) n'est qu'une succession de réactions biochimiques plus ou moins complexes catalysées par diverses enzymes. Pour simplifier la réalité, on considère que la vitesse du processus biologique ne dépend que de la vitesse d'une réaction, dite "réaction d'engagement". L'enzyme qui catalyse cette réaction est appelée "enzyme contrôle" du processus. On estime ensuite que la vitesse du processus est proportionnelle à la concentration d'enzyme "contrôle" active et à sa processivité. Enfin, on suppose que l'activité de l'enzyme ne dépend que de la température et que celle-ci s'inactive en deçà et au-delà de deux températures d'inactivation. A partir de ces trois hypothèses et grâce à quelques simplifications supplémentaires, Sharpe et Schoofield ont abouti à l'équation (1) qui permet de calculer la vitesse instantanée d'un processus pour un individu donné en fonction de la température. Les différentes constantes présentes dans l'équation sont estimées par une méthode de régression non-linéaire à partir d'observations, on retrouve leurs valeurs dans le tableau 2 de l'article [A].

$$\omega(T) = \frac{\rho_{(25°C)} \frac{T}{298[K]} e^{\frac{\Delta H_A}{R}(\frac{1}{298[K]} - \frac{1}{T})}}{1 + e^{\frac{\Delta H_L}{R}(\frac{1}{T_{1/2_L}} - \frac{1}{T})} + e^{\frac{\Delta H_H}{R}(\frac{1}{T_{1/2_H}} - \frac{1}{T})}} \tag{1}$$

3.3.2 Probabilité de former une panicule

Le choix entre "pousse de feuilles" et "formation de la panicule" dépend du hasard. Soit N la variable aléatoire comptant le nombre de feuilles à l'âge adulte, et TI l'événement "formation de la panicule". Les auteurs proposent trois sous-modèles distincts pour calculer la probabilité de l'événement TI pour une plante à k feuilles : $P(TI\Big|k)$. Le sous-modèle A considère que tous les plants ont le même nombre de feuilles à l'âge adulte ($N_{max} = 15$). On a donc $P(TI\Big|k) = 1$ si $k = N_{max} = 15$ et 0 sinon.
Pour les sous-modèles B et C, on utilise les probabilités conditionnelles :

$$P(TI|k) = \frac{P(\text{formation panicule} \cap \text{k feuilles})}{P(\text{k feuilles})} = \frac{\text{"proba d'initiation du panicule et d'avoir k feuilles"}}{\text{"proba d'avoir k feuilles"}} = \frac{P(N=K)}{P(N \geq k)}.$$

Pour le sous-modèle B, on recense le nombre final de feuilles dans différentes cultures de maïs et on approche la distribution obtenue par une loi normale de paramètres $\mu = 15$ et $\sigma^2 = 4$. Tandis que dans le sous-modèle C, on approche N par une loi de Poisson de paramètre $\lambda = \frac{w_{lir}}{w_{tir}}$ qui correspond à l'espérance de la loi, c'est-à-dire, au nombre moyen de feuilles à l'âge adulte [Annexe 2].

3.4 Méthode de programmation

Pour programmer le modèle, la première étape consiste à "choisir" quel événement se déroule en premier. Ce choix dépend du temps que met un événement à se produire et comme il s'agit d'un modèle stochastique, cette durée dépend du hasard. Dans notre programme, nous avons utilisé l'algorithme de Gillespie. Il s'agit de tirer aléatoirement, selon des lois exponentielles de paramètres bien choisis, le temps qui s'écoule jusqu'à la prochaine transition de chaque type. On détermine alors quel événement a lieu le plus rapidement, on applique ses effets sur la population et on ajoute sa durée au temps écoulé. On réitère la même opération jusqu'à atteindre un temps final choisi à l'avance.
Revenons aux paramètres des lois exponentielles, on sait que si X suit une loi Exp(λ), on a $\mathbb{E}(X)=\frac{1}{\lambda}$. Dans notre cas, la loi exponentielle modélise le temps qui s'écoule entre deux événements du même type, son espérance correspond donc au temps moyen entre ces deux transitions. Ainsi, λ est le "nombre moyen d'événements par unité de temps", autrement dit à la vitesse instantanée du processus (à l'échelle de la population) que l'on peut trouver dans le tableau 1 de l'article [A].

4 Résultats

4.1 Résultats méthodologiques

A partir de l'équation (1), nous avons représenté les valeurs des coefficients de transition en fonction de la température (Fig.2). Les courbes obtenues semblent assez similaires aux courbes de l'article [A] et elles semblent approcher de manière cohérente les valeurs retrouvées dans la bibliographie de l'article [10].

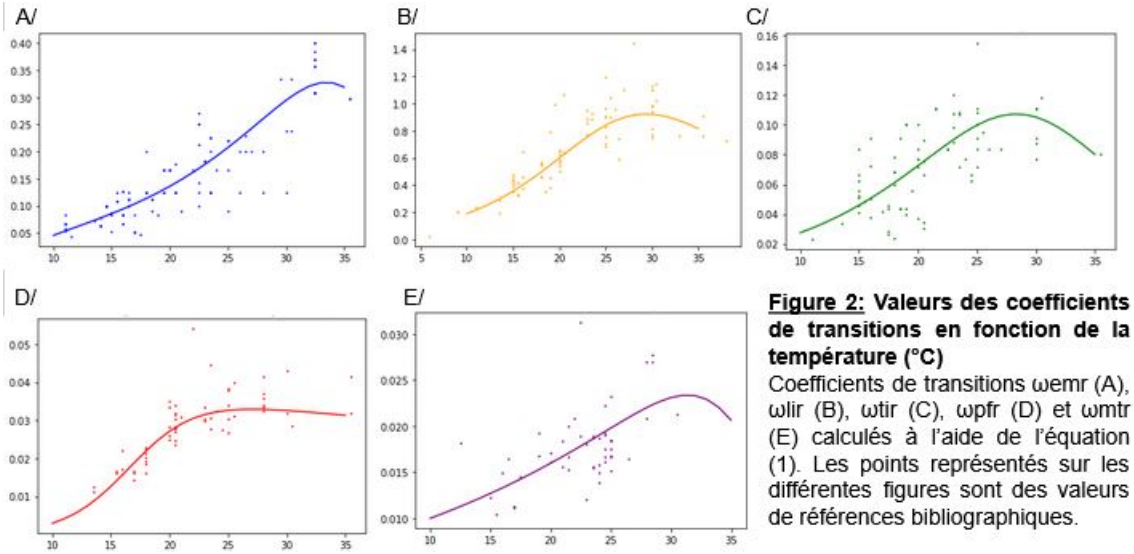


Figure 2: Valeurs des coefficients de transitions en fonction de la température (°C)
Coefficients de transitions ωemr (A), ωlir (B), ωtir (C), ωpfr (D) et ωmtr (E) calculés à l'aide de l'équation (1). Les points représentés sur les différentes figures sont des valeurs de références bibliographiques.

Par la suite, nous avons modélisé et tracé l'évolution de la croissance d'une population de maïs pour une température fixée à 16°C (Fig. 3). En comparant la figure 3 et l'annexe 3 (figure 5a de l'article), on remarque que les dynamiques des deux populations présentent des allures semblables. En effet, on observe qu'une diminution du nombre d'individus dans une sous-population entraîne une augmentation dans la sous-population suivante. Cependant, on remarque tout de même des différences non négligeables notamment au niveau de la sous-population P. On constate d'une part que le pic associé à la population P se situe aux alentours de 25 jours pour notre figure, alors qu'il a lieu plus tôt dans l'article. D'autre part, on voit que la décroissance de la courbe associée à la population P est moins accentuée sur nos résultats. Cette différence se répercute sur les stades suivants puisqu'on observe un décalage et un étalement dans le temps. En outre, on remarque une différence entre les courbes obtenues avec les sous-modèles A et C dans notre figure, alors qu'il n'y a pas de grande différence sur l'article. Avec la modélisation C, le retard des courbes est d'autant plus important. L'écart peut être dû au fait que pour le C, les plantes peuvent avoir plus que 15 feuilles et donc restent potentiellement plus longtemps au stade P. De plus, la grande différence observée par rapport à l'article peut être une conséquence d'une mauvaise estimation des ω_{lir} et ω_{tir} que l'on utilise dans la loi de Poisson.

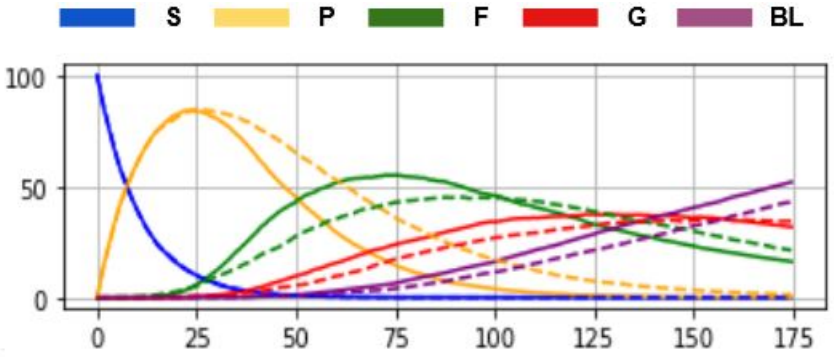


Figure 3: Moyenne sur 50 simulations du nombre de plant de maïs dans chaque stade en fonction du temps (en jours)
Les courbes pleines représente le modèle A, celles en pointillées le modèle C. Les 50 simulations pour chaque modèles sont réalisées avec une population initiale de 100 individus, à 16°C.

4.2 Résultats biologiques

Grâce à notre version du modèle de l'article, nous avons ensuite voulu modéliser la croissance du maïs dans des lieux et à des périodes de l'année pendant lesquelles se font réellement les cultures de maïs. Pour cela, nous avons choisi deux régions productrices de maïs : la Bretagne [11] en France et la "Corn belt" aux États-Unis [12] pour lesquelles nous avons relevé les températures mensuelles des années 1945 et 2020, en nous basant sur les températures de deux villes de ces régions arbitrairement choisies : Rennes [13] et Chicago[14]. Pour notre modélisation, nous avons supposé que les graines de maïs étaient semées le 25 avril et nous avons supposé que les températures de chaque mois étaient constantes et égales aux moyennes mensuelles. Cela nous permet de modéliser la croissance du maïs à Rennes et Chicago en 1945, 2020 et avec des températures augmentées de 1,5°C par rapport à 2020, pour simuler l'effet du réchauffement climatique sur la température. Sur la figure 4 ainsi obtenue, on observe des dynamiques de populations différentes selon les villes et les années. Si on compare Chicago et Rennes, on voit que, quelle que soit l'année, le développement du maïs est plus rapide à Chicago : les pics de chaque stade sont atteints plus précocément. Cette différence est due aux différences de températures entre les deux villes et notamment aux températures en moyenne assez élevées à Chicago en juin-juillet-août (>18°C en 1945 et >23°C en 2020). Si on s'intéresse maintenant pour chaque ville aux différences de dynamiques de croissance, on remarque aux deux endroits que le développement du maïs est plus rapide en cas d'une augmentation de 1.5°C qu'en 2020 et plus rapide en 2020 qu'en 1945. Comme on sait que la température augmente globalement depuis le XIXe siècle, on peut dire qu'une augmentation globale de la température accélère la croissance du maïs. C'est assez logique dans les gammes de températures considérées, car les réactions biologiques se font souvent plus vite à des températures plus élevées, on le voit d'ailleurs bien sur les valeurs des coefficients de transitions (Fig. 2).

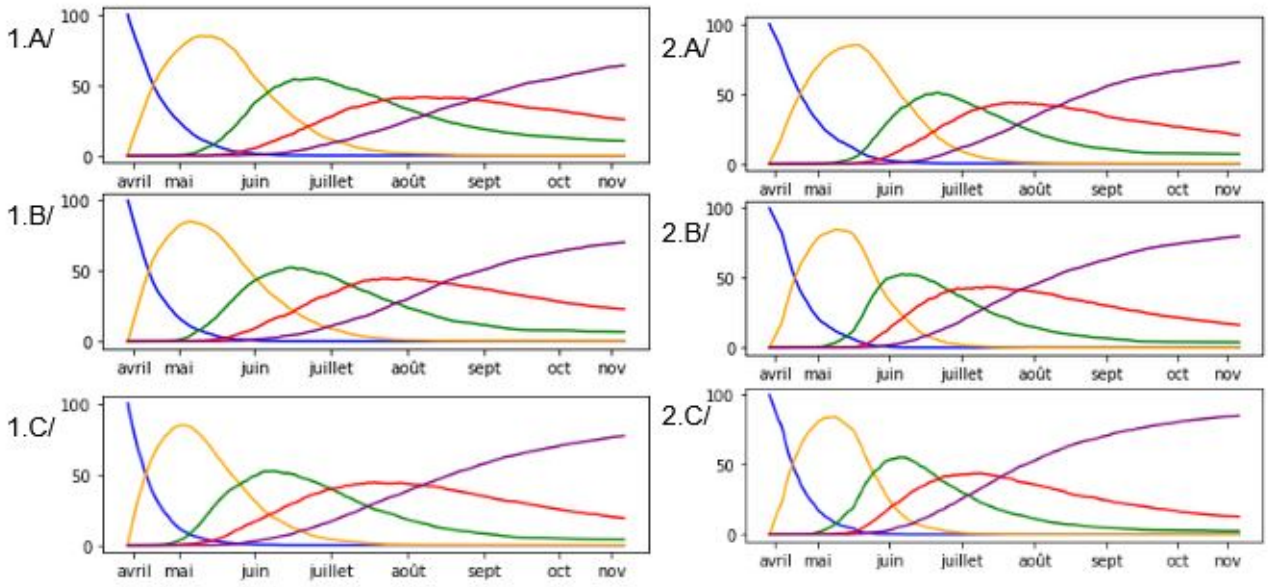


Figure 4: Evolution de la population de maïs en fonction du temps pour trois années différentes à Rennes (1) et Chicago (2)
Les graphiques sont des moyennes pour 20 simulations du modèle A, sur 200 jours pour une population de 100 individus. Températures mensuelles moyennes en 1945 (A), en 2020 (B) et dans quelques années avec le réchauffement climatique (+1.5°C) (C).

5 Conclusion

Pour conclure, le modèle que nous avons programmé à partir de l'article propose une dynamique de population relativement similaire à celle de l'article. Néanmoins, nous avons un problème de retard de développement qui pourrait être dû à des paramètres mal estimés ou au fait que l'article utilise un type d'approximation que nous n'avons pas appliqué. Nous avons tout de même pu modéliser l'effet d'une augmentation de la température due au réchauffement climatique et avons ainsi pu observer que la croissance du maïs est accélérée dans ces conditions. Cependant, au vu des courbes des coefficients de transitions, il est probable que cette accélération ne se poursuive pas pour des températures très élevées. De plus, le dérèglement climatique en cours ne se traduit pas uniquement par une augmentation de température donc il faudrait un modèle plus complexe pour essayer de modéliser les réels effets de ce phénomène sur la dynamique de croissance du maïs.

En début de projet, nous avons eu du mal à comprendre toute la dimension stochastique du modèle. En effet, nous pensions que le hasard n'intervenait qu'au moment de former le panicule (à cause de la probabilité). De ce fait, pour le modèle A , la simulation était toujours la même puisque le nombre maximal de feuilles est fixé dans ce cas (loi de Dirac). Par la suite, nous avons donc dû ajouter une part d'aléatoire au niveau des temps des transitions. Pour cela, les auteurs utilisent une "approche quasi-linéaire" et une "approche multinomiale" (loi de probabilité). Les sources citées dans l'article n'étaient pas assez claires pour que nous réussissions à comprendre de quoi il s'agit et nous avons donc utilisé la méthode de Gillespie. Dans l'article, les auteurs proposent 3 trois sous-modèles pour approcher la probabilité de formation de la panicule. Le sous-modèle B, que nous n'avons pas programmé, approche la distribution du nombre de feuilles maximum par une loi normale. Or, la probabilité d'initiation de la panicule se calcule $\frac{P(N=k)}{P(N\geq k)}$ et comme la loi normale est une loi à densité, P(N=k) devrait valoir 0, la probabilité devrait donc être toujours nulle. Les auteurs n'ont pas précisé s'il fallait utiliser une correction de continuité pour calculer cette probabilité. Enfin, il nous manquait certains paramètres pour calculer les coefficients de transition à partir de l'équation (1). Nous avons donc dû les estimer à partir de la figure 2 de l'article [A]. Ces difficultés pourraient éventuellement expliquer les différences que l'on observe entre nos figures et celles de l'article. En suivant les conseils de notre tutrice, nous avons décidé de ne pas aller plus loin dans l'amélioration du modèle pour rentabiliser le temps et développer les modèles avec variation de température de Chicago et Rennes.

Les principales limites du modèle proposé correspondent aux hypothèses posées. En effet, on considère ici une population fermée de plantes immortelles. Or, comme la vitesse instantanée de passage d'un stade à un autre (paramètre de la loi exponentielle qui calcule le temps de transition) dépend de la taille des sous-populations, une mort importante à un ou plusieurs des stades pourrait modifier la dynamique générale de la population de maïs. De plus, la perte de feuilles, qui aurait un impact sur l'étude de la surface foliaire, est négligée. Pour finir, la température est un paramètre de caractère très variable à plusieurs échelles temporelles (entre autres : jour, semaine, mois). Dans l'article, la température est fixée tout au long de la simulation. Ainsi, les effets des variations de température au cours des saisons sont négligés. Or celle-ci est un paramètre clé dans la détermination des coefficients de transition. La dynamique de population ainsi obtenue est donc assez éloignée de la réalité.

6 Glossaire

Agent pathogène : l'ensemble de bactéries, virus, phytoplasmes et mycètes capables de pénétrer dans une cellule, tissu ou organe végétal et déclencher une maladie

Apex : Pointe ou sommet d'un organe, ici de la plante

Espérance : moyenne des valeurs prises par la variable aléatoire

Étamines : organes floraux mâles transporteurs de sacs de pollen

Fécondation : processus par lequel le tube pollinique pénètre dans l'ovule de la plante pendant le processus de reproduction

Germination : déclenchement du développement d'une graine en une cellule végétative quand les conditions extérieures redeviennent favorables

Herbacées : de dit de plantes non ligneuses, ne produisant pas de bois et dont les parties aériennes meurent après la fructification des fleurs

Inflorescence : disposition des fleurs sur la tige d'une plante à fleur

Ovules : gamète femelle destiné à être fécondé

Plastochrone : Période qui s'écoule entre l'initiation d'une feuille et celle de la feuille suivante

Pollen : libéré et produit par les étamines des plantes à graines, c'est le gamétophyte mâle (ce qui produit les gamètes)

Primordia : premier stade de développement d'une feuille.

Réaction d'engagement : la majorité des réactions d'une voie métabolique sont réversibles, mais généralement, il y a une réaction irréversible qui engage la réaction à se faire dans un certain sens

Sacs polliniques : cavités dans lesquelles sont élaborées les cellules mères des grains de pollen

Styles : organe qui relie l'ovaire au stigmate (organe destiné à recevoir le grain de pollen)

Tube pollinique : extension en forme de tube qui émet les grains de pollen

Références

[1] Kenneth Mason Jonathan Losos Susan Rundell Singer : Peter Raven, George Johnson. Biologie, volume 4e édition, traduction de la 11e édition américaine. De Boeck Supérieur, 2017.

[2] Wikipedia. Ravageurs du maïs et liste des ennemis du maïs.

[3] Préservons la nature. https ://www.preservons-la-nature.fr/flore/taxon/3979.html.

[4] LeBulletin des agriculteurs. https ://youtu.be/FXMgfeLZC.

[5] Gnis pedagogie. Origine et caractéristiques du maïs. https ://www.gnis-pedagogie.org/sujet/mais-origine-caracteristiques/.

[6] LeBulletin des agriculteurs. Contribution à l'étude de l'évolution de la production du maïs. https ://bit.ly/3oODfBT.

[7] Paul R. Carter. Kernel black layer formation in corn : Anatomy, physiology, and causes. https ://www.pioneer.com/us/agronomy/kernel-black-layer-formation.html.

[8] Peter JH Sharpe and Don W DeMichele. Reaction kinetics of poikilotherm development. Journal of theoretical biology, 64(4) :649–670, 1977.

[9] Robert M Schoolfield, PJH Sharpe, and Charles E Magnuson. Non-linear regression of biological temperature-dependent rate models based on absolute reaction-rate theory. Journal of theoretical biology, 88(4) :719–731, 1981.

[10] Arnaud Guyader. Processus markoviens de sauts. http ://www.lpsm.paris/pageperso/guyader/files/teaching/Sauts.pdf.

[11] BASF France Division Agro. https ://on.basf.com/3rZq9Ug.

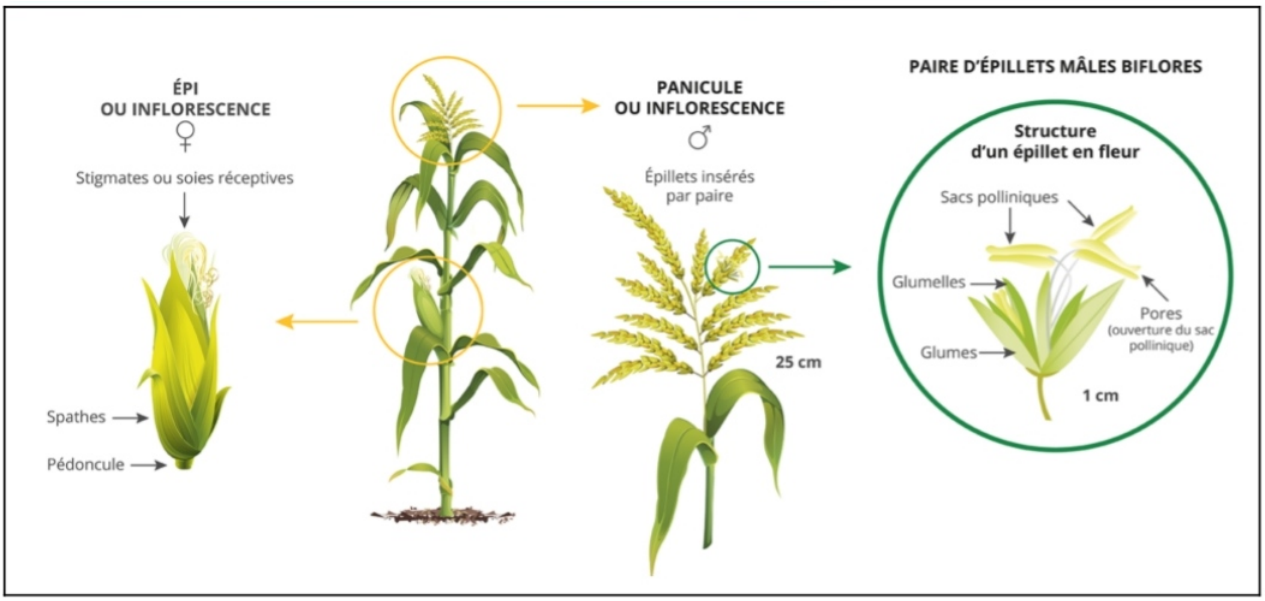
[12] Perspectives Agricoles. https ://bit.ly/3ysaxtF.

[13] Meteo Bretagne. https ://bit.ly/3pUgATN.

[14] National Weather Service USA. https ://bit.ly/31SDH98.

7 Annexes

7.1 Annexe 1 : Inflorescence du maïs



7.2 Annexe 2 : Paramètre de la loi de Poisson

On a posé $\lambda = \frac{\omega_{lir}}{\omega_{tir}}$.

On peut vérifier ce choix de λ de deux manières :

- $\frac{\omega_{lir}}{\omega_{tir}} = \frac{\text{nombre moyen de feuilles initiées par plastochron}}{\text{nombre moyen de panicules initiées par plastochron}} = \frac{\text{nombre moyen de feuilles initiées}}{\text{nombre moyen de panicules initiées}}$

  $=$ nombre moyen de feuilles par panicule $=$ nombre moyen de feuilles à l'âge adulte

- $\omega_{lir} = \text{plastochron}^{-1}$ (jour^{-1})

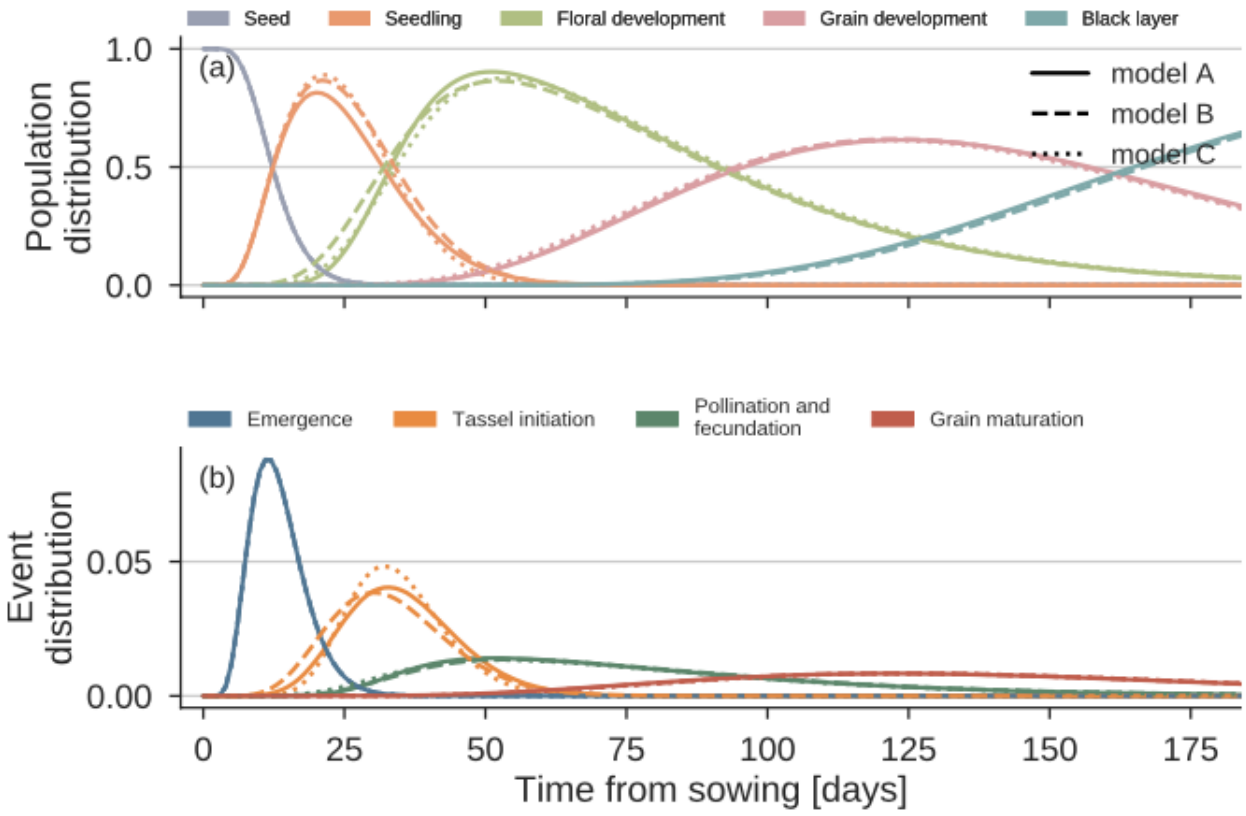
  ω_{tir} = nombre moyen de panicules initiés par unité de temps (jour^{-1})

donc ω_{lir}^{-1} = temps moyen entre le début de la croissance et la formation de la panicule

Le nombre de feuilles avant l'apparition de la panicule peut être estimé par :

$$\frac{\text{durée entre le début de la croissance et l'initiation de la panicule}}{\text{temps d'apparition d'une feuille (plastochrone)}} = \frac{\omega_{tir}^{-1}}{\omega_{lir}^{-1}} = \frac{\omega_{lir}}{\omega_{tir}}$$

7.3 Annexe 3 : Figure 5a de l'article



7.4 Annexe 4 : Code Python

```
import numpy as np
import matplotlib.pyplot as plt
import random as rd
import pandas as pa

# PARTIE 1 : FIGURE RELATIVE AUX COEFFICIENTS DE TRANSITION

# création des vecteurs "phénomènes" temporaires contenant les paramètres
# connus de l'équation (1) pour chaque phénomène (et des 0 ailleurs)
emr = (0.21, 60, 280, 310, 0, 0)
lir = (1.40, 90, 150, 300, 0, 0)
tir = (0.13, 70, 180, 303, 0, 0)
pfr = (0.56, 240, 250, 290, 0, 0)
mtr = (0.02, 30, 0, 0, 0, 0)

# Codage de l'équation (1)
"""
But : calculer la valeur d'un coefficient de transition

Arguments :
    - T = la température (en K)
    - pheno = vecteur contenant les valeurs des autres paramètres de l'équation pour
        une transition donnée

Sortie : le coefficient de transition
"""

def W(T, pheno) :

    # on récupère la valeur de chacun des paramètres de l'équation
    rho=pheno[0]
```

```
    DHa=pheno[1] * 10**3
    DHh=pheno[2] * 10**3
    Tdemih=pheno[3]
    R= 8.314462
    DHl= pheno[4]*10**3
    Tdemil= pheno[5]

    # équation (1)
    numerateur= rho*(T/298)*np.exp((DHa/R) * ((1/298) -(1/T)))
    denominateur= 1 + np.exp ((DHl/R)*((1/Tdemil)- (1/T)))+ np.exp((DHh/R)*((1/
        Tdemih) -(1/T)))

    return numerateur/denominateur

# Détermination des paramètres manquants de l'équation
# Fonction test n1 (pour emr, lir, tir, pfr)
"""
But : déterminer la valeur de DHl et TdemiL

Arguments :
    - erreur = valeur d'erreur acceptée entre la valeur exacte et celle qui est
        calculée
    - Tmin, Tmax, Dmin, Dmax = valeurs extrêmes de chacun des paramètres
    - pas_T, pas_D = écart entre les différentes valeurs de DHl et TdemiL testées
    - pheno = vecteur contenant les valeurs des autres paramètres de l'équation pour
         une transition donnée
    - solpheno = "solutions exacte" pour 25, 30, 15, 35C lues sur la figure 2 de l'
        article'

Sortie : un tableau avec toutes les valeurs de DHL et TdemiL pour lesquelles
l'erreur n'est pas dépassée
"""

def test_omega(erreur, Tmin, Tmax, pas_T, Dmin, Dmax, pas_D, pheno, solpheno) :
```

```
61
62      # on crée des vecteurs contenant toutes les valeurs de DHl et TdemiL àtester
63      T = np.arange(Tmin,Tmax, pas_T)
64      D = np.arange(Dmin, Dmax, pas_D)
65
66      # on crée un vecteur solution vide
67      solutions = np.zeros((1,2))
68
69      for i in T :
70          for j in D :
71
72              # on crée un vecteur contenant les valeurs des paramètres de l'équation
                    pour un évènement donné
73              test = (pheno[0], pheno[1], pheno[2], pheno[3], j, i)
74
75              # on calcule la différence entre la "valeur exacte" et la valeur calculée
76              # (grâce àl'équation) pour un couple de valeurs DHl et TdemiL
77              if ((np.abs(W(298, test) - solpheno[0]) <erreur) & (np.abs(W(303, test) -
                     solpheno[1]) < erreur) & (np.abs(W(288, test) -solpheno[2])<erreur)
                    & (np.abs(W(308, test) - solpheno[3]) < erreur)):
78
79                  # si pour chaque point exact, la différence est inférieure àl'erreur
80                  # souhaitée, on ajoute les valeurs testées dans le vecteur solution
81                  newrow = np.array([i, j])
82                  solutions = np.vstack([solutions, newrow])
83
84      return solutions
85
86  # création des vecteurs "solutions exactes"
87  solemr = (0.21, 0.3, 0.08, 0.36)
88  sollir = (0.8, 1, 0.35, 0.75)
89  soltir = (0.1, 0.107, 0.045, 0.075)
90  solpfr =(0.035, 0.034, 0.012, 0.03)
91
```

```
92 # vecteurs obtenus après multiples tests avec la fonction
93 # pour mtr on utilise une fonction légèrement modifiée puisqu'il y a 4 paramètres à
       trouver)
94 emr = (0.21, 60, 280, 310, -298.8, 279.6)
95 lir = (1.40, 90, 150, 300, -298.9, 259.2)
96 tir = (0.13, 70, 180, 303, -300.3, 259)
97 pfr = (0.56, 240, 250, 290, -341.8, 251.7)
98 mtr = (0.02, 30, 289, 310, -263, 251)
99
100
101 # Tracé des courbes résultats méthodologiques
102
103 # on récupère les données de coefficients fournis par l'article
104 sortie_emr = pa.read_csv('C:/Users/margo/Desktop/Margot/cours/université/2021-2022/
       S5/projet␣mais/testcoeff_emer.csv', sep=';')
105 sortie_lir = pa.read_csv('C:/Users/margo/Desktop/Margot/cours/université/2021-2022/
       S5/projet␣mais/testcoeff_lir.csv', sep=';')
106 sortie_tir = pa.read_csv('C:/Users/margo/Desktop/Margot/cours/université/2021-2022/
       S5/projet␣mais/testcoeff_tir.csv', sep=';')
107 sortie_pfr = pa.read_csv('C:/Users/margo/Desktop/Margot/cours/université/2021-2022/
       S5/projet␣mais/testcoeff_pfr.csv', sep=';')
108 sortie_mtr = pa.read_csv('C:/Users/margo/Desktop/Margot/cours/université/2021-2022/
       S5/projet␣mais/testcoeff_mtr.csv', sep=';')
109
110 # on crée un vecteur contenant différentes températures
111 temperature = np.arange(10, 36 , 1)
112
113 # on trace sur un même graphe la courbe obtenue avec la fonction W et les points
114 # expérimentaux pour chacune des transitions
115
116 plt.scatter(sortie_emr['temp'], sortie_emr['rate'], s = 1.5, c ='blue')
117 plt.plot(temperature, W(temperature + 273, emr), c ='blue')
118 plt.title('Evolution␣de␣Wemr␣en␣fonction␣de␣la␣température␣(en␣C)')
119 plt.show()
```

```
120
121 plt.scatter(sortie_lir['temp'], sortie_lir['rate'], s = 1.5, c ='orange')
122 plt.plot(temperature, W(temperature + 273, lir), c ='orange')
123 plt.title('Evolution de Wlir en fonction de la température (en C)')
124 plt.show()
125
126 plt.scatter(sortie_tir['temp'], sortie_tir['rate'], s = 1.5, c ='green')
127 plt.plot(temperature, W(temperature + 273, tir), c ='green')
128 plt.title('Evolution de Wtir en fonction de la température (en C)')
129 plt.show()
130
131 plt.scatter(sortie_pfr['temp'], sortie_pfr['rate'], s = 1.5, c ='red')
132 plt.plot(temperature, W(temperature + 273, pfr), c ='red')
133 plt.title('Evolution de Wpfr en fonction de la température (en C)')
134 plt.show()
135
136 plt.scatter(sortie_mtr['temp'], sortie_mtr['rate'], s = 1.5, c ='purple')
137 plt.plot(temperature, W(temperature + 273, mtr), c ='purple')
138 plt.title('Evolution de Wmtr en fonction de la température (en C)')
139 plt.show()
140
141 # PARTIE 2 : REPRODUCTION DE LA FIGURE 2A DE L'ARTICLE
142
143 # Fonctions préliminaires
144
145 # Fonction pour appliquer l'algorithme de Gillespie
146 """
147 But : tirer aléatoirement (selon des lois bien définies) l'évènement qui se
148 déroulera en premier dans la population
149
150 Precision: on etudiera d'une part le temps minimum parmi les populations S, G et M,
    puis d'autre part, le temps minimum pour chacune des sous populations P et F à
    chaque pas de temps. On prendra finalement le temps minimal parmi les trois
    valeurs obtenues.
```

```
151
152 Arguments :
153     - popS, popPk, popF, popG : les vecteurs populations des stades S àG
154     - wemr, wlir, wtir, wpfr, wmtr : les coefficients des transitions
155     - M : le sous-modèle choisi
156
157 Sortie :
158     - indice : un couple (a, b) où a correspond àl'évènement qui a lieu en
159 premier (0 : émergence, 1 : pousse de feuille, 2 : initiation du panicule,
160 3 : fécondation, 4 : maturation de la graine) et b correspond au nombre de
161 feuilles - 6
162     - tpsmin : le temps que prend l'évènement àse dérouler
163 """
164 def algorithmeGP(popS, wemr, popPk , wlir, popF, wtir, popG, wpfr, wmtr, M):
165
166     #Calcul de certaines vitesses instantanées de transition
167     TauxE = popS * wemr
168     TauxF = popF * wpfr
169     TauxM = popG * wmtr
170
171     # on distingue d'abord le cas où il n'y a que des plantes P6 dans la
172     # classe P (pas d'autres sous-catégories)
173
174     if type(popPk) == np.float64 :
175
176         # on indique que la plante a 6 feuilles
177         indicePk1 = 0
178         indicePk2 = 0
179
180         # calcul des vitesses instantées des différentes transitions
181         TauxPk1 = popPk * wlir * (1-proba(0,M,wlir,wtir)) # pousse d'une feuille
182         TauxPk2 = popPk* wtir* proba(0, M, wlir, wtir) #initiation du panicule
183
184         # si la vitesse est nulle, on met un temps très grand pour que
```

```
185         #l'évènement n'ait pas lieu
186         if TauxPk1 == 0 :
187             tpsmin_Pk1 = 10000000
188 
189         # sinon, on tire la durée de déroulement de l'évènement selon une loi
190         # exponentielle de paramètre la vitesse instantanée de l'évènement
191         else :
192             tpsmin_Pk1 = rd.expovariate(TauxPk1)
193 
194         # idem pour l'évènement "initiation" du panicule
195         if TauxPk2 == 0 :
196             tpsmin_Pk2 = 10000000
197         else :
198             tpsmin_Pk2 = rd.expovariate(TauxPk2)
199 
200 
201     # dans le cas où on a plusieurs sous-catégories dans la classe P
202     elif type(popPk)!= int :
203 
204         #on crée deux tableaux de même taille que le vecteur Popk
205         taille_P = len(popPk)
206 
207         tps_Pk1 = np.zeros(taille_P)
208         tps_Pk2 = np.zeros(taille_P)
209 
210         # Boucle calculant àchaque k le taux puis le temps de transition
211         # on applique exactement le même principe qu'au-dessus
212         for i in range (0, taille_P):
213             TauxPk1= popPk[i] * wlir * (1-proba(i,M,wlir,wtir)) #on calcule a chaque
                    fois le taux
214             TauxPk2 = popPk[i] * wtir* proba(i, M, wlir, wtir) #pareil
215 
216             if (TauxPk1 == 0):
217                 tps_Pk1[i] = 10000000
```

```
218             else:
219                 tps_Pk1[i] = rd.expovariate(TauxPk1)
220 
221             if(TauxPk2 == 0) :
222                 tps_Pk2[i] = 10000000
223             else:
224                 tps_Pk2[i] = rd.expovariate(TauxPk2)
225 
226         # on récupère le plus petit temps pour chaque type d'évènement
227         tpsmin_Pk1 = min(tps_Pk1)
228         tpsmin_Pk2 = min(tps_Pk2)
229 
230         # on note pour quel k le temps est minimal
231         indicePk1 = np.where(tps_Pk1 == tpsmin_Pk1)[0][0]
232         indicePk2 = np.where(tps_Pk2 == tpsmin_Pk2)[0][0]
233 
234     # Tableau regroupant les vitesses instantanées desévènements
235     Tabtaux = [TauxE, 0, 0, TauxF, TauxM]
236 
237     # tableau qui regroupera les temps de chaque transition
238     tps_transi = np.zeros(5)
239 
240     # Boucle calculant pour chaque stade le temps de transition et remplissant le
        tableau
241     for i in range (5):
242         Lambda = Tabtaux[i]
243 
244         # on applique toujours le même principe pour calculer le temps
245         if (Lambda==0):
246             tps_transi[i]= 10000000
247         else:
248             tps_transi[i] = rd.expovariate(Lambda)
249 
250     # on ajoute au tableau les valeurs de temps précédemment calculées pour lir &
```

```
        tir
    tps_transi[1] = tpsmin_Pk1
    tps_transi[2] = tpsmin_Pk2

    # on récupère le temps minimal du tableau
    tpsmin = min(tps_transi)

    # on récupère le numéro de la transition qui a le temps minimal
    # le second membre de l'indice est fixé à0 par défaut car on ne
    # prend pas en compte le nombre de feuille pour emr, pfr, mtr
    indice = np.array([np.where(tps_transi == tpsmin)[0][0] , 0])

    # dans le cas de lir & tir, on note le nombre de feuilles
    if indice[0] == 1 :
        indice[1]= indicePk1
    if indice[0] == 2 :
        indice[1]= indicePk2

    return indice, tpsmin



# Fonction qui calcule la probabilité d'initiation du panicule
"""
Arguments :
    - k : le nombre de feuilles - 6
    - M : le sous-modèle choisi
    - Wlir, Wtir : les coefficients de transition de lir et tir

Sortie : P(Ti|k)
"""

def proba(k, M, Wlir, Wtir) :

```

```
    # loi de Dirac de paramètre 9 (15-6) pour le sous-modèle A
    if M == 'A' :
        if k == 9 :
            return 1
        else :
            return 0

    # on applique la formule (2) de l'article dans le cas où Nmax suit
    # une loi de poisson
    if M == 'C' :
        return Poisson_exacte(Wlir/Wtir, k)/Poisson_sup(Wlir/Wtir, k)

# Fonctions pour calculer des probabilités avec une loi de poisson
"""
Arguments :
    - l : paramètre de la loi de poisson
    - k : nombre de feuilles - 6

Sortie : P(Nmax = k)
"""
def Poisson_exacte(l, k) :
    return np.exp(-l) * (l**k) / np.math.factorial(k)

"""
Arguments :
    - l : paramètre de la loi de poisson
    - k : nombre de feuilles - 6

Sortie : P(Nmax >= k)
"""
def Poisson_sup(l,k) :
    result = 0
    for i in range(k) :
        result = result + Poisson_exacte(l, i)
```

```
318     return 1-result
319
320
321 # Fonctions pour appliquer les effets des différents évènements
322
323 """
324 But : appliquer les effets de l'émergence àla population
325
326 Arguments :
327     - Stab, Ptab, Ftab, Gtab, BLtab : les vecteurs populations
328     - l = la longueur du vecteur temps au moment où on applique la fonction
329
330 Sortie : les vecteurs populations modifiés
331 """
332 def Emergence(Stab, Ptab, Ftab, Gtab, BLtab, l) :
333
334     # on ajoute une case de plus àchaque vecteur population
335
336     # l'émergence diminue la population S de 1 individu donc on remplit la
337     # nouvelle case avec la dernière taille de population S - 1
338     Stemp = np.append(Stab, Stab[l-1]-1)
339
340     # F, G et BL ne sont pas affectés par l'émergence donc on remplit
341     # la nouvelle case avec la dernière taille des populations F, G et BL
342     Ftemp = np.append(Ftab, Ftab[l-1])
343     Gtemp = np.append(Gtab, Gtab[l-1])
344     BLtemp = np.append(BLtab, BLtab[l-1])
345
346     # pour la population P c'est un peu plus compliqué car on peut avoir des
347     # sous-classes selon le nombre de feuilles
348
349     # Cas où il n'y a qu'une sous-classe (P6 uniquement)
350     if len(np.shape(Ptab)) == 1:
351
```

```
352         # l'émergence augmente la population P6 de 1 individu donc on remplit
353         # la nouvelle case avec la dernière taille de population P6 + 1
354         Ptemp = np.append(Ptab, Ptab[l-1]+1)
355 
356     # cas où il y a plusieurs sous-classes
357     else :
358 
359         # on crée le vecteur colonne qui sera ajouté àla matrice P
360         # le nombre de ligne de la colonne correspond au nombre de lignes de P
361         newcol = np.zeros((len(Ptab[:,l-1]),1))
362 
363         # on remplit la colonne avec les dernières tailles des populations P6, P7,
                ...
364         newcol[:, 0] = Ptab[:, l-1]
365 
366         # on ajoute 1 individu àla population P6
367         newcol[0,0] = newcol[0,0] + 1
368 
369         # on ajoute la dernière colonne àla matrice P
370         Ptemp = np.hstack([Ptab, newcol])
371 
372     return Stemp, Ptemp, Ftemp, Gtemp, BLtemp
373 
374 
375 """
376 But : appliquer les effets de la pousse d'une feuille àla population
377 
378 Arguments :
379     - Stab, Ptab, Ftab, Gtab, BLtab : les vecteurs populations
380     - l = la longueur du vecteur temps au moment où on applique la fonction
381     - transi : le k pour lequel la transition Pk -> Pk+1 a lieu
382 
383 Sortie : les vecteurs populations modifiés
384 """
```

```
385 def Feuille(Stab, Ptab, Ftab, Gtab, BLtab, l, transi) :
386
387     # on ajoute une case àchaque vecteur
388     # on remplit la dernière case avec la dernière taille de population pour
389     # S, F, G, BL car la pousse d'une feuille n'a pas d'effet sur eux
390     Stemp = np.append(Stab, Stab[l-1])
391     Ftemp = np.append(Ftab, Ftab[l-1])
392     Gtemp = np.append(Gtab, Gtab[l-1])
393     BLtemp = np.append(BLtab, BLtab[l-1])
394
395     # cas où P n'a qu'une sous-classe (P6)
396     if len(np.shape(Ptab))== 1 :
397
398         # on crée une nouvelle ligne (qui correspondra àP7) de la même longueur que
                P6
399         newrow = np.zeros(l)
400         # on ajoute cette ligne àla matrice P
401         Ptemp = np.vstack([Ptab, newrow])
402
403         # on crée une nouvelle colonne (à 2 lignes pour P6 et P7)
404         newcol = np.zeros((2,1))
405         # que l'on remplit avec les dernières tailles des sous-populations P6, P7
406         newcol[:, 0] = Ptemp[:, l-1]
407         # on enlève 1 individu àP6
408         newcol[transi, 0] = Ptab[l-1]-1
409         # on ajoute 1 individu àP7
410         newcol[transi+1, 0] = 1
411         # on ajoute la nouvelle colonne àla matrice P
412         Ptemp = np.hstack([Ptemp, newcol])
413
414     # cas où P a plusieurs sous-classes MAIS n'a pas encore de sous-classe P(k+6)+1
415     elif np.shape(Ptab)[0] < transi + 2 :
416         # on crée une nouvelle ligne (qui correspondra àP(k+6)+1) de la même
417         # longueur que P
```

```
418         newrow = np.zeros(l)
419         # on ajoute cette ligne àla matrice P
420         Ptemp = np.vstack([Ptab, newrow])
421 
422         # on crée une nouvelle colonne (à k+2 lignes pour P6 àP(k+6)+1)
423         newcol = np.zeros((transi+2,1))
424         # que l'on remplit avec les dernières tailles des sous-populations P6, P7,
                ...
425         newcol[:, 0] = Ptemp[:, l-1]
426         # on enlève 1 individu àP(k+6)
427         newcol[transi, 0] = Ptemp[transi, l-1]-1
428         # on ajoute 1 individu àP(k+6)+1
429         newcol[transi+1,0] = Ptemp[transi+1, l-1]+1
430         # on ajoute la nouvelle colonne àla matrice P
431         Ptemp = np.hstack([Ptemp, newcol])
432 
433     # cas où P a plusieurs sous-classes et possède une classe P(k+6)+1
434     else :
435         # on crée une nouvelle colonne (avec autant de lignes que P)
436         newcol = np.zeros((len(Ptab[:,l-1]),1))
437         # que l'on remplit avec les dernières tailles des sous-populations P6, P7,
                ..
438         newcol[:, 0] = Ptab[:, l-1]
439         # on enlève 1 individu àP(k+6)
440         newcol[transi, 0] = Ptab[transi, l-1]-1
441         # on ajoute 1 individu àP(k+6)+1
442         newcol[transi+1, 0] = Ptab[transi+1, l-1]+1
443         # on ajoute la nouvelle colonne àla matrice P
444         Ptemp = np.hstack([Ptab, newcol])
445 
446     return Stemp, Ptemp, Ftemp, Gtemp, BLtemp
447 
448 """
449 But : appliquer les effets de l'initiation d'un tassel àla population
```

```
450
451 Arguments :
452     - Stab, Ptab, Ftab, Gtab, BLtab : les vecteurs populations
453     - l = la longueur du vecteur temps au moment où on applique la fonction
454     - transi : le k pour lequel la transition Pk -> Pk+1 a lieu
455
456 Sortie : les vecteurs populations modifiés
457 """
458 def Tassel(Stab, Ptab, Ftab, Gtab, BLtab, l, transi) :
459
460     # on ajoute une case àchaque vecteur
461     # on remplit la dernière case avec la dernière taille de population pour
462     # S, G, BL car l'initiation d'un tassel n'a pas d'effet sur eux
463     Stemp = np.append(Stab, Stab[l-1])
464
465     # on ajoute 1 individu de plus àla dernière taille de population F
466     Ftemp = np.append(Ftab, Ftab[l-1]+1)
467     Gtemp = np.append(Gtab, Gtab[l-1])
468     BLtemp = np.append(BLtab, BLtab[l-1])
469
470     # cas spécial pour P
471     # on crée une nouvelle colonne
472     newcol = np.zeros((len(Ptab[:,l-1]),1))
473     # que l'on remplit avec les dernières tailles des sous-populations P6, P7, ..
474     newcol[:, 0] = Ptab[:, l-1]
475     # on enlève 1 individu de la population Pk d'où part le plant qui forme une
        panicule
476     newcol[transi, 0] = Ptab[transi, l-1]-1
477     # on ajoute la nouvelle colonne àla matrice P
478     Ptemp = np.hstack([Ptab, newcol])
479
480     return Stemp, Ptemp, Ftemp, Gtemp, BLtemp
481
482 """
```

```
483 But : appliquer les effets de la pollinisation/fécondation àla population
484
485 Arguments :
486     - Stab, Ptab, Ftab, Gtab, BLtab : les vecteurs populations
487     - l = la longueur du vecteur temps au moment où on applique la fonction
488
489 Sortie : les vecteurs populations modifiés
490 """
491 def Fecondation(Stab, Ptab, Ftab, Gtab, BLtab, l) :
492
493     # on ajoute une case àchaque vecteur
494     # on remplit la dernière case avec la dernière taille de population pour
495     # S et BL car la pollinisation/fécondation n'a pas d'effet sur eux
496     Stemp = np.append(Stab, Stab[l-1])
497
498     # on enlève ou on ajoute respectivement 1 individu àF et G
499     Ftemp = np.append(Ftab, Ftab[l-1]-1)
500     Gtemp = np.append(Gtab, Gtab[l-1]+1)
501     BLtemp = np.append(BLtab, BLtab[l-1])
502
503     # cas spécial pour P
504     # on crée une nouvelle colonne
505     newcol = np.zeros((len(Ptab[:,l-1]),1))
506     # que l'on remplit avec les dernières tailles des sous-populations P6, P7, ..
507     newcol[:, 0] = Ptab[:, l-1]
508     # on ajoute la nouvelle colonne àla matrice P
509     Ptemp = np.hstack([Ptab, newcol])
510
511     return Stemp, Ptemp, Ftemp, Gtemp, BLtemp
512
513 """
514 But : appliquer les effets de la maturation àla population
515
516 Arguments :
```

```
517     - Stab, Ptab, Ftab, Gtab, BLtab : les vecteurs populations
518     - l = la longueur du vecteur temps au moment où on applique la fonction
519
520 Sortie : les vecteurs populations modifiés
521 """
522 def Maturation(Stab, Ptab, Ftab, Gtab, BLtab, l) :
523
524     # on ajoute une case àchaque vecteur
525     # on remplit la dernière case avec la dernière taille de population pour
526     # S et F car la maturation n'a pas d'effet sur eux
527     Stemp = np.append(Stab, Stab[l-1])
528     Ftemp = np.append(Ftab, Ftab[l-1])
529
530     # on enlève ou on ajoute respectivement 1 individu àG et BL
531     Gtemp = np.append(Gtab, Gtab[l-1]-1)
532     BLtemp = np.append(BLtab, BLtab[l-1]+1)
533
534
535     # cas spécial pour P
536     # on crée une nouvelle colonne
537     newcol = np.zeros((len(Ptab[:,l-1]),1))
538     # que l'on remplit avec les dernières tailles des sous-populations P6, P7, ..
539     newcol[:, 0] = Ptab[:, l-1]
540     # on ajoute la nouvelle colonne àla matrice P
541     Ptemp = np.hstack([Ptab, newcol])
542
543     return Stemp, Ptemp, Ftemp, Gtemp, BLtemp
544
545
546
547 # Fonction qui applique le modèle
548 """
549 But : modéliser le nombre d'individus dans chaque classe de la population au
550 cours du temps
```

```
551
552 Arguments :
553     - Tpop : la taille de la population
554     - tmax : la durée maximale sur laquelle on modèlise (en jours)
555     - T : la température (en K)
556     - M : le sous-modèle choisi (pour nous 'A' ou 'C')
557
558 Sortie :
559     - tps : un vecteur contenant les différentes valeurs de temps auxquelles
560 des événements ont lieu
561     - Stab, Ptabf, Ftab, Gtab, BLtab : les vecteurs contenant la taille de
562 chaque classe pour ces valeurs de temps
563 """
564 def Modelisation(Tpop, tmax, T, M) :
565
566     # on crée un vecteur temps
567     tps = np.zeros(1)
568
569     # on crée des vecteurs pour chaque classe de population
570     Stab=np.zeros(1)
571     Ptab=np.zeros(1)
572     Ftab=np.zeros(1)
573     Gtab=np.zeros(1)
574     BLtab=np.zeros(1)
575
576     # calcul coefficients de transition
577     wemr = W(T, emr)
578     wlir = W(T, lir)
579     wtir = W(T, tir)
580     wpfr = W(T, pfr)
581     wmtr = W(T, mtr)
582
583
584     # on initialise tous les vecteurs à0, sauf S qui contient tous les
```

```
585     # individus de la population (on plante uniquement des graines)
586     tps[0] = 0
587     Stab[0] = Tpop
588     Ptab[0] = 0
589     Ftab[0] = 0
590     Gtab[0] = 0
591     BLtab[0] = 0
592
593     # boucle qui tourne jusqu'à ce qu'on atteigne tmax
594     while (tps[len(tps)-1] < tmax) :
595
596         # on récupère la longueur du vecteur temps
597         l = len(tps)
598
599         # 1er cas : si P n'a qu'une sous-classe (P6)
600         if len(np.shape(Ptab)) == 1:
601              # on applique Gillespie pour savoir quel évènement se déroule en
602              # premier et en combien de temps
603              event, t = algorithmeGP(Stab[l-1], wemr, Ptab[l-1] , wlir, Ftab[l-1],
                     wtir, Gtab[l-1], wpfr, wmtr, M)
604
605         # 2e cas : si P a plusieurs sous-classes (P6, P7,...)
606         elif len(np.shape(Ptab))> 1 :
607             # on applique Gillespie pour savoir quel évènement se déroule en
608             # premier et en combien de temps
609             event, t = algorithmeGP(Stab[l-1], wemr, Ptab[:, l-1] , wlir, Ftab[l-1],
                    wtir, Gtab[l-1], wpfr, wmtr, M)
610
611         # on ajoute une nouvelle case au vecteur temps, dont la valeur est celle
612         # du dernier temps enregistré + la durée du nouvel évènement
613         # autrement dit le temps écoulé jusqu'à la fin du nouvel évènement
614         tps = np.append(tps, t + tps[l-1])
615
616         # si l'évènement est 0 on applique les effets de l'émergence
```

```
617         if(event[0] == 0) :
618             Stab, Ptab, Ftab, Gtab, BLtab = Emergence(Stab, Ptab, Ftab, Gtab, BLtab,
                    l)
619
620         # si l'évènement est 1 on applique les effets de la pousse d'une feuille
621         elif (event[0] == 1):
622             Stab, Ptab, Ftab, Gtab, BLtab = Feuille(Stab, Ptab, Ftab, Gtab, BLtab, l,
                     event[1])
623
624         # si l'évènement est 2 on applique les effets de l'initiation d'une panicule
625         elif (event[0] == 2):
626             Stab, Ptab, Ftab, Gtab, BLtab = Tassel(Stab, Ptab, Ftab, Gtab, BLtab, l,
                    event[1])
627
628         # si l'évènement est 3 on applique les effets de la pollinisation/fé
                condation
629         elif (event[0] == 3):
630             Stab, Ptab, Ftab, Gtab, BLtab = Fecondation(Stab, Ptab, Ftab, Gtab, BLtab
                    , l)
631
632         # si l'évènement est 4 on applique les effets de la maturation
633         elif (event[0] == 4):
634             Stab, Ptab, Ftab, Gtab, BLtab = Maturation(Stab, Ptab, Ftab, Gtab, BLtab,
                     l)
635
636
637     # une fois la boucle terminée, on crée un nouveau vecteur P
638     Ptabf = np.zeros(len(tps))
639
640     # on complète ce vecteur avec l'addition de toutes les sous-population k
641     # de P pour chaque temps
642     for i in range(len(tps)) :
643         Ptabf[i] = sum(Ptab[:, i])
644
```

```
645     return tps, Stab, Ptabf, Ftab, Gtab, BLtab
646
647
648 # Fonction qui réalise plusieurs simulations
649 """
650 But : obtenir des valeurs moyennes sur un certain nombre de simulations
651
652 Arguments :
653     - Tpop : la taille de la population
654     - tmax : la durée maximale sur laquelle on modèlise (en jours)
655     - T : la température (en K)
656     - M : le sous-modèle choisi (pour nous 'A' ou 'C')
657     - Nbsim : le nombre de simulations àeffectuer
658     - fonction : nom de la fonction de modélisation àutiliser
659
660 Sortie :
661     - tempscontinu : un vecteur contenant des intervalles de temps réguliers
662 entre 0 et tmax (pas = 0.5)
663     - des vecteurs contenant les tailles moyennes des populations chaque classe
664 pour ces valeurs de temps
665 """
666 def Simulations(Tpop, tmax, T, M, Nbsim, fonction):
667
668     # on crée un vecteur temps avec des intervalles réguliers entre 0 et tmax
669     tempscontinu = np.arange(0, tmax, 0.5)
670
671     # on récupère la longueur de ce vecteur
672     L = len(tempscontinu)
673
674     # on crée des vecteurs qui stockeront les tailles des différentes classes
675     # pour les valeurs de temps choisies
676     Scontinu = np.zeros(L)
677     Pcontinu = np.zeros(L)
678     Fcontinu = np.zeros(L)
```

```
679     Gcontinu = np.zeros(L)
680     BLcontinu = np.zeros(L)
681 
682     # boucle pour réaliser Nbsim simulations
683     for n in range(Nbsim) :
684 
685         # on applique le modèle
686         tps, S, P, F, G, BL = fonction(Tpop, tmax, T, M)
687 
688         # variable d'itération qu'on initialise à1
689         i = 1
690 
691         # pour chaque valeur de temps
692         for t in range(L) :
693 
694             # au temps 0, on incrémente juste les valeurs initiales
695             # des vecteurs résultats avec la valeur initiale obtenue par modélisation
696             if t == 0 :
697                 Scontinu[0] = Scontinu[0] + S[0]
698                 Pcontinu[0] = Pcontinu[0] + P[0]
699                 Fcontinu[0] = Fcontinu[0] + F[0]
700                 Gcontinu[0] = Gcontinu[0] + G[0]
701                 BLcontinu[0] = BLcontinu[0] + BL[0]
702 
703             # pour les autres valeurs de temps
704             else :
705 
706                 # on cherche l'indice pour àpartir duquel la valeur du temps
707                 # de modélisation dépasse la t-ième valeur de temps
708                 while(tps[i]<=tempscontinu[t]) :
709                     i = i + 1
710 
711                 # on incrémente les vecteurs résultats (à l'indice t) avec
712                 # la taille des populations de la modélisation àl'indice i-1
```

```
713                 Scontinu[t] = Scontinu[t] + S[i-1]
714                 Pcontinu[t] = Pcontinu[t] + P[i-1]
715                 Fcontinu[t] = Fcontinu[t] + F[i-1]
716                 Gcontinu[t] = Gcontinu[t] + G[i-1]
717                 BLcontinu[t] = BLcontinu[t] + BL[i-1]
718
719
720     return tempscontinu, Scontinu/Nbsim, Pcontinu/Nbsim, Fcontinu/Nbsim, Gcontinu/
            Nbsim, BLcontinu/Nbsim
721
722 # Tracé des graphiques
723
724 # On réalise 50 simulations pour chaque sous-modèle A et C avec 100 individus
725 # et 175 jours à16C
726 tps_A, S_A, P_A, F_A, G_A, BL_A = Simulations(100, 175, 289, 'A', 50, Modelisation)
727 tps_C, S_C, P_C, F_C, G_C, BL_C = Simulations(100, 175, 289, 'C', 50, Modelisation)
728
729 # On crée la figure vide
730 fig1, (ax1,ax2) = plt.subplots(2)
731 fig1.subplots_adjust(wspace=0.75)
732 ax1.grid(True)
733
734 # on trace les courbes pour les différents stades du sous-modèle A en ligne pleine
735 ax1.plot(tps_A, S_A, color = 'blue')
736 ax1.plot(tps_A, P_A, color = 'orange')
737 ax1.plot(tps_A, F_A, color='green')
738 ax1.plot(tps_A, G_A, color='red')
739 ax1.plot(tps_A, BL_A, color='purple')
740
741 # on trace les courbes pour les différents stades du sous-modèle C en pointillé
742 ax1.plot(tps_C, S_C, color = 'blue')
743 ax1.plot(tps_C, P_C, color = 'orange')
744 ax1.plot(tps_C, F_C, color='green')
745 ax1.plot(tps_C, G_C, color='red')
```

```
746 ax1.plot(tps_C, BL_C, color='purple')
747
748 # PARTIE 3 : MODÉLISATION AVEC DES VARIATIONS DE TEMPÉRATURE
749
750 # Fonction de modélisation adaptée
751 """
752 But : modéliser le nombre d'individus dans chaque classe de la population au
753 cours du temps en prenant en compte des variations de température
754
755 NB. Il s'agit àpeu de chose près d'une fonction très semblable àla première
756
757 Arguments :
758     - Tpop : la taille de la population
759     - tmax : la durée maximale sur laquelle on modèlise (en jours), 200 ici
760     - T : un vecteur contenant les températures mensuelles (en C) d'avril à
761 décembre sur une année choisie, dans un lieu choisi
762     - M : le sous-modèle choisi (pour nous 'A' ou 'C')
763
764 Sortie :
765     - tps : un vecteur contenant les différentes valeurs de temps auxquelles
766 des événements ont lieu
767     - Stab, Ptabf, Ftab, Gtab, BLtab : les vecteurs contenant la taille de
768 chaque classe pour ces valeurs de temps
769 """
770 def Modelisation2(Tpop, tmax, T, M) :
771
772     # on convertit les températures en Kelvin
773     T = T + 273
774
775     tps = np.zeros(1)
776     Stab=np.zeros(1)
777     Ptab=np.zeros(1)
778     Ftab=np.zeros(1)
779     Gtab=np.zeros(1)
```

```
780     BLtab=np.zeros(1)
781
782     tps[0] = 0
783     Stab[0] = Tpop
784     Ptab[0] = 0
785     Ftab[0] = 0
786     Gtab[0] = 0
787     BLtab[0] = 0
788
789
790     while (tps[len(tps)-1] < tmax) :
791         # on considère que les graines sont plantées le 25 avril
792         # les 5 premiers jours, on prend la température mensuelle moyenne d'avril
793         if tps[len(tps)-1]<=5 :
794             wemr = W(T[0], emr)
795             wlir = W(T[0], lir)
796             wtir = W(T[0], tir)
797             wpfr = W(T[0], pfr)
798             wmtr = W(T[0], mtr)
799
800         # les 31 jours suivants, on prend celle de mai
801         elif tps[len(tps)-1]<=36 :
802             wemr = W(T[1], emr)
803             wlir = W(T[1], lir)
804             wtir = W(T[1], tir)
805             wpfr = W(T[1], pfr)
806             wmtr = W(T[1], mtr)
807
808         # les 30 jours suivants, on prend celle de juin
809         elif tps[len(tps)-1]<=66 :
810             wemr = W(T[2], emr)
811             wlir = W(T[2], lir)
812             wtir = W(T[2], tir)
813             wpfr = W(T[2], pfr)
```

```
814             wmtr = W(T[2], mtr)
815
816         # les 31 jours suivants, on prend celle de juillet
817         elif tps[len(tps)-1]<= 97 :
818             wemr = W(T[3], emr)
819             wlir = W(T[3], lir)
820             wtir = W(T[3], tir)
821             wpfr = W(T[3], pfr)
822             wmtr = W(T[3], mtr)
823
824         # les 31 jours suivants, on prend celle de août
825         elif tps[len(tps)-1]<=128 :
826             wemr = W(T[4], emr)
827             wlir = W(T[4], lir)
828             wtir = W(T[4], tir)
829             wpfr = W(T[4], pfr)
830             wmtr = W(T[4], mtr)
831
832         # les 30 jours suivants, on prend celle de septembre
833         elif tps[len(tps)-1]<=158 :
834             wemr = W(T[5], emr)
835             wlir = W(T[5], lir)
836             wtir = W(T[5], tir)
837             wpfr = W(T[5], pfr)
838             wmtr = W(T[5], mtr)
839
840         # les 31 jours suivants, on prend celle d'octobre
841         elif tps[len(tps)-1]<=189 :
842             wemr = W(T[6], emr)
843             wlir = W(T[6], lir)
844             wtir = W(T[6], tir)
845             wpfr = W(T[6], pfr)
846             wmtr = W(T[6], mtr)
847
```

```
848         # les derniers, on prend celle de novembre
849         elif tps[len(tps)-1]<=200 :
850             wemr = W(T[6], emr)
851             wlir = W(T[6], lir)
852             wtir = W(T[6], tir)
853             wpfr = W(T[6], pfr)
854             wmtr = W(T[6], mtr)
855
856
857         l = len(tps)
858
859         if len(np.shape(Ptab)) == 1:
860              event, t = algorithmeGP(Stab[l-1], wemr, Ptab[l-1] , wlir, Ftab[l-1],
                     wtir, Gtab[l-1], wpfr, wmtr, M)
861
862         elif len(np.shape(Ptab))> 1 :
863             event, t = algorithmeGP(Stab[l-1], wemr, Ptab[:, l-1] , wlir, Ftab[l-1],
                    wtir, Gtab[l-1], wpfr, wmtr, M)
864
865         tps = np.append(tps, t + tps[l-1])
866
867         if(event[0] == 0) :
868             Stab, Ptab, Ftab, Gtab, BLtab = Emergence(Stab, Ptab, Ftab, Gtab, BLtab,
                    l)
869
870         elif (event[0] == 1):
871             Stab, Ptab, Ftab, Gtab, BLtab = Feuille(Stab, Ptab, Ftab, Gtab, BLtab, l,
                     event[1])
872
873         elif (event[0] == 2):
874             Stab, Ptab, Ftab, Gtab, BLtab = Tassel(Stab, Ptab, Ftab, Gtab, BLtab, l,
                    event[1])
875
876         elif (event[0] == 3):
```

```
            Stab, Ptab, Ftab, Gtab, BLtab = Fecondation(Stab, Ptab, Ftab, Gtab, BLtab
                , l)

        elif (event[0] == 4):
            Stab, Ptab, Ftab, Gtab, BLtab = Maturation(Stab, Ptab, Ftab, Gtab, BLtab,
                 l)


    Ptabf = np.zeros(len(tps))
    for i in range(len(tps)) :
        Ptabf[i] = sum(Ptab[:, i])

    return tps, Stab, Ptabf, Ftab, Gtab, BLtab


# On crée des vecteurs contenant les températures moyennes mensuelles (en C)
# àRennes et Chicago, d'avril àmai, en 1945 et 2020
T_Rennes_1945 = np.array([12.4, 14.1, 16.7, 18.4, 17.7, 16.7, 13.2, 7.7, 6.5])
T_Rennes_2020 = np.array([14, 15.7, 16.9, 18.9, 20.1, 17.7, 12.7, 10.7, 7.4])

T_Chicago_1945 = np.array([9.4, 12.2, 18.3, 22.2, 22.2, 17.8, 10.6, 3.9, -5.6])
T_Chicago_2020 = np.array([8.9, 15, 23.3, 26.1, 24.4, 18.9, 10.6, 8.3, 0])

# on crée des vecteurs de températures prévisionnelles en cas de réchauffement
# climatique de 1.5 C dans les deux villes

T_Rennes_RC = T_Rennes_2020 + 1.5
T_Chicago_RC = T_Chicago_2020 + 1.5


# Tracé des graphiques

# on fait 20 simulations pour chaque ville et chaque année, avec le sous-modèle A
# pour 100 individus, 200 jours, semis au 25 avril
```

```
909
910 tps, S_R_1945, P_R_1945, F_R_1945, G_R_1945, BL_R_1945 = Simulations(100, 200,
        T_Rennes_1945, 'A', 20, Modelisation2)
911 tps, S_R_2020, P_R_2020, F_R_2020, G_R_2020, BL_R_2020 = Simulations(100, 200,
        T_Rennes_2020, 'A', 20, Modelisation2)
912 tps, S_R_RC, P_R_RC, F_R_RC, G_R_RC, BL_R_RC = Simulations(100, 200, T_Rennes_RC, 'A
        ', 20, Modelisation2)
913
914 tps, S_C_1945, P_C_1945, F_C_1945, G_C_1945, BL_C_1945 = Simulations(100, 200,
        T_Chicago_1945, 'A', 20, Modelisation2)
915 tps, S_C_2020, P_C_2020, F_C_2020, G_C_2020, BL_C_2020 = Simulations(100, 200,
        T_Chicago_2020, 'A', 20, Modelisation2)
916 tps, S_C_RC, P_C_RC, F_C_RC, G_C_RC, BL_C_RC = Simulations(100, 200, T_Chicago_RC, '
        A', 20, Modelisation2)
917
918 # On crée les vecteurs contenant les positions et noms des étiquettes de l'axe des
        abcisses
919 positions = [2.5, 20.5, 51, 81.5, 112.5, 143, 173.5, 194.5]
920 etiquettes = ['avril', 'mai', 'juin', 'juillet', 'août', 'sept', 'oct', 'nov']
921
922 # On crée la figure vide pour Rennes 1945 et 2020
923 fig2, (ax1,ax2) = plt.subplots(2)
924 fig2.subplots_adjust(wspace=0.75)
925 ax1.set_xticks(positions)
926 ax1.set_xticklabels(etiquettes)
927 ax2.set_xticks(positions)
928 ax2.set_xticklabels(etiquettes)
929
930 # On trace les courbes des différents stades sur chaque graphe
931 ax1.plot(tps, S_R_1945, color = 'blue')
932 ax1.plot(tps, P_R_1945, color = 'orange')
933 ax1.plot(tps, F_R_1945, color='green')
934 ax1.plot(tps, G_R_1945, color='red')
935 ax1.plot(tps, BL_R_1945, color='purple')
```

```
936
937 ax2.plot(tps, S_R_2020, color = 'blue')
938 ax2.plot(tps, P_R_2020, color = 'orange')
939 ax2.plot(tps, F_R_2020, color='green')
940 ax2.plot(tps, G_R_2020, color='red')
941 ax2.plot(tps, BL_R_2020, color='purple')
942
943 # idem pour Chicago 1945 et 2020
944 fig3, (ax1,ax2) = plt.subplots(2)
945 fig3.subplots_adjust(wspace=0.75)
946 ax1.set_xticks(positions)
947 ax1.set_xticklabels(etiquettes)
948 ax2.set_xticks(positions)
949 ax2.set_xticklabels(etiquettes)
950
951
952 ax1.plot(tps, S_C_1945, color = 'blue')
953 ax1.plot(tps, P_C_1945, color = 'orange')
954 ax1.plot(tps, F_C_1945, color='green')
955 ax1.plot(tps, G_C_1945, color='red')
956 ax1.plot(tps, BL_C_1945, color='purple')
957
958 ax2.plot(tps, S_C_2020, color = 'blue')
959 ax2.plot(tps, P_C_2020, color = 'orange')
960 ax2.plot(tps, F_C_2020, color='green')
961 ax2.plot(tps, G_C_2020, color='red')
962 ax2.plot(tps, BL_C_2020, color='purple')
963
964 # idem pour les cas de réchauffement climatique
965 fig4, (ax1,ax2) = plt.subplots(2)
966 fig4.subplots_adjust(wspace=0.75)
967 ax1.set_xticks(positions)
968 ax1.set_xticklabels(etiquettes)
969 ax2.set_xticks(positions)
```

```
970 ax2.set_xticklabels(etiquettes)
971
972 ax1.plot(tps, S_R_RC, color = 'blue')
973 ax1.plot(tps, P_R_RC, color = 'orange')
974 ax1.plot(tps, F_R_RC, color='green')
975 ax1.plot(tps, G_R_RC, color='red')
976 ax1.plot(tps, BL_R_RC, color='purple')
977
978 ax2.plot(tps, S_C_RC, color = 'blue')
979 ax2.plot(tps, P_C_RC, color = 'orange')
980 ax2.plot(tps, F_C_RC, color='green')
981 ax2.plot(tps, G_C_RC, color='red')
982 ax2.plot(tps, BL_C_RC, color='purple')
```