Projet en modélisation: évolution d'une population de moustiques tigres en France

LBRTI2102
Prof. Emmanuel Hanert

Brieuc Ryelandt

Janvier 2018

Table des matières

1 Introduction

Le moustique tigre, *Aedes albopictus*, est originaire d'Asie. Il a été observé la première fois en France en 2004 près de Nice (moustique tigre.info, 5/11/2017) et est classé comme espèce envahissante.
L'étude de ce moustique est intéressante car il est vecteur de la dengue et du chikungunya qui sont des maladies initialement tropicales qui commencent à s'installer dans nos régions avec la venue de celui-ci. Il est également vecteur du virus Zika.

Pour ces raisons, il est très étudié et mon but dans ce travail est de modéliser son évolution en France. Pour cela, j'utilise un modèle de population préexistant (Lacour, 2013). Mon but a été d'implémenter ce modèle dans Matlab et de l'adapter pour pouvoir le tester sur la France. Ensuite, il était prévu de le faire tourner sur base des projections climatiques pourrait donner une idée de l'évolution de l'espèce avec le changement climatique.

1.1 Écologie du moustique tigre

Un concept important dans le modèle de population que j'utilise est la diapause. C'est un paramètre de la réponse hivernale des insectes (Lacour, 2016). Celui-ci synchronise le cycle de vie des insectes aux saisons. Lorsque la photopériode passe sous un certain seuil, les œufs passent en diapause et n'éclosent pas même si les conditions climatiques sont favorables.

1.2 Changement climatique et impact sur les populations de moustiques

Les populations de moustiques évoluant avec les températures, on peut s'attendre à voir une migration de celles-ci vers le nord dans un contexte de changement climatique. Par contre, la diapause dépend de la photopériode. Il serait donc intéressant de voir s'il existe une latitude au dessus de laquelle le moustique ne peut plus se développer même si les conditions climatiques sont favorables. L'évolution des précipitations dans le contexte du changement climatique est encore peu connue et elle constitue donc une limite à l'utilisation du modèle.
Je n'ai pas testé dans mon modèle cette réponse au changement climatique pour pouvoir me concentrer sur les autres étapes de modélisation mais je décris dans la suite de ce rapport la méthode que j'aurais utilisée.

2 Système et données

2.1 Modèle de population

Le modèle de (Lacour, 2013) décrit le cycle de vie du moustique tigre comme 10 stades de développement différents et donc 10 réservoirs (figure 1). Ces 10 réservoirs sont liés entre eux par des fonctions de transitions et un taux de mortalité est également appliqué à chaque stade de développement.

2.2 Modèle géographique

Le passage au stade géographique a été fait en utilisant une fonction de diffusion - réaction (Équation 1). Le coefficient de diffusion tenant compte de la vitesse de déplacement d'un moustique et ne s'appliquant qu'aux moustiques adultes. Le modèle de population décrit précédemment est utilisé comme équation de réaction.

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(K\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + q \tag{1}$$

2.3 Données

Les données utilisées dans mon modèle sont d'une part des archives météorologiques de météo france (26/12/2017) pour le modèle de population et d'autre part, pour le coefficient de diffusion du modèle *diffusion-réaction* utilisé pour la deuxième partie a été calculé sur base de la distance parcourue en une journée par un moustique (Lacour, 2016) et la taille de la maille. J'émets de grosses réserves sur cette estimation car l'équation n'est pas correcte d'un point de vue dimensionnel. J'ai donné la priorité à la logique (si la maille est plus grande, le coefficient de diffusion doit être plus petit). La calibration et la validation du modèle devraient permettre d'affiner cette valeurs. D'autre part, beaucoup de modèle de diffusion pour les insectes sont proposés dans la littérature. Une analyse de celle-ci pourrait permettre d'affiner mon modèle si celui-ci s'avère trop peu précis à l'étape de validation.

Calcul du coefficient de diffusion:

$$K = 1/(d.v) \qquad \text{où v la vitesse du moustique et d la distance d'une maille}$$

$$K_x = \frac{1}{78.85 * 0.1} = 0.0013 j/km^2$$

$$K_y = \frac{1}{111 * 0.1} = 0.001 j/km^2$$

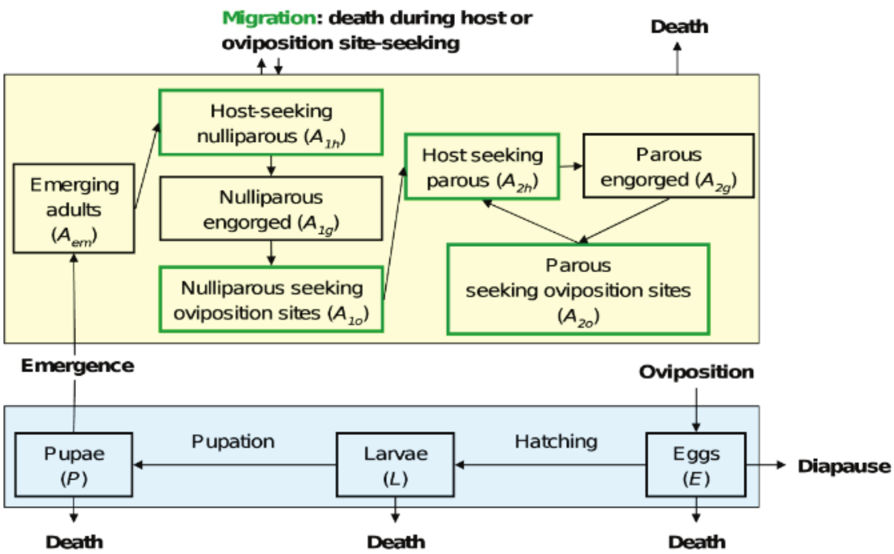

Figure 1: Stades de développement du moustique tigre. En bleu, les stades aquatiques et en jaune les stades aériens (uniquement les femelles) (Lacour, 2013)

3 Modèle et hypothèses

3.1 Évolution d'une population en fonction de la température

Pour modéliser une population de moustique tigre, j'ai utilisé un modèle existant (Lacour, 2013). Celui-ci est composé de 10 équations différentielles ordinaires, 9 fonctions de transition et mortalité (Table 2) et 20 paramètres (Table 1).

$$
\begin{cases}
\dot{E} & = \gamma_{Ao}(\beta_1 A_{1o} + \beta_2 A_{2o}) - (\mu_E + z * f_E)E \\
\dot{L} & = z * f_E E - [m_L(1 + L/k_L) + f_L]L \\
\dot{P} & = f_L - [m_P + f_P]P \\
\dot{A}_{em} & = f_P P \sigma exp[-\mu_{em}(1 + P/k_P)] - [m_A + \gamma_{Aem}]A_{em} \\
\dot{A}_{1h} & = \gamma_{Aem} A_{em} - (m_A + \mu_r + \gamma_{Ah})A_{1h} \\
\dot{A}_{1g} & = \gamma_{Ah} A_{1h} - (m_A + f_{Ag})A_{1h} \\
\dot{A}_{1o} & = f_{Ag} A_{1g} - (m_A + \mu_r + \gamma_{Ao})A_{1o} \\
\dot{A}_{2h} & = \gamma_{Ao}(A_{1o} + A2o) - (m_A + \mu_r + \gamma_{Ah})A_{2h} \\
\dot{A}_{2g} & = \gamma_{2h} A_{2h} - (m_A + f_{Ag})A_{2g} \\
\dot{A}_{2o} & = f_{Ag} A_{2g} - (m_A + \mu_r + \gamma_{Ao})A_{2o}
\end{cases}
$$

Table 1: Paramètres du modèle de population du moustique tigre

Paramètre	Définition	Valeur
β_1	Nombre d'œufs par femelle (1)	95
β_2	Nombre d'œufs par femelle (2)	75
k_L	Capacité de l'environnement en larves (larves/ha)	250 000
k_P	Capacité de l'environnement en nymphes (nymphes/ha)	250 000
σ	Sex-ratio à la naissance	0.5
μ_E	Taux de mortalité des œufs(jour^{-1})	0.05[1]
μ_L	Taux de mortalité minimum des larves (jour^{-1})	0.08
μ_P	Taux de mortalité minimum des nymphes (jour^{-1})	0.03
μ_{em}	Taux de mortalité durant l'émergence (jour^{-1})	0.1
μ_A	Taux de mortalité minimum des adultes (jour^{-1})	0.02
μ_r	Taux de mortalité des adultes en exploration (jour^{-1})	0.08
T_E	Température minimale pour l'éclosion des œufs ($°C$)	10.4
TDD_E	Nombre de "degree-day"[2] pour le développement des œufs ($°C$)	110
γ_{Aem}	Taux de développement des adultes émergents (jour^{-1})	0.4
γ_{Ah}	Taux de transition entre adulte en chasse et adulte engorgé (jour^{-1})	0.2
γ_{Ao}	Taux d'oviposition	0.2
T_{Ag}	Température minimale de maturation des œufs $°C$	10
TDD_{Ag}	Nombre de "degree-day"[2] pour la maturation des œufs ($°C$)	77
t_{start}	début de la saison favorable	10 Mars
t_{end}	fin de la saison favorable	30 Sept

[1] Par soucis de calibration du modèle, j'ai utilisé le taux suivant: 0.02
[2] Degree-day: quantité de chaleur accumulée nécessaire pour le développement

Table 2: Fonctions du modèle de population du moustique tigre

Fonction	Définition	Expression
f_E	Fonction de transition entre œufs et larves	Équation 2
f_L	Fonction de transition entre larves et nymphes	$f_L(t) = -0.0007T^2(t)+0.0392T(t)-0.3911$
f_P	Fonction de transition entre nymphe et adulte émergent	$f_P(t) = T^2(t) - 0.0051T(t) + 0.0319$
f_{Ag}	Fonction de transition pour l'oviposition	Équation 2
m_L	Mortalité des larves (jour^{-1})	$m_L(t) = exp(-T(t)/2) + \mu_L$
m_P	Mortalité des nymphes (jour^{-1})	$m_P(t) = exp(-T(t)/2) + \mu_P$
m_A	Mortalité des adultes (jour^{-1})	$max(\mu_A; 0.04417 + 0.00217.T(t))$
k_L	Capacité de l'environnement en larves (ha^{-1})	Équation (3)
k_P	Capacité de l'environnement en nymphes (ha^{-1})	Équation (3)

$$f_x(t) = \begin{cases} \frac{T(t)-T(x)}{TDD_x} \text{si } T(t) > T_x \; x = \{E; A_g\} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{ailleurs} \end{cases} \tag{2}$$

$$k_x(t) = k_x(P_{norm}(t) + 1 \; x = \{L, P\} \tag{3}$$

Il y a trois variables d'entrée dans le modèle: la température, la somme normalisée pour varier entre 0 et 1 des précipitations des 14 derniers jours et les dates de début et de fin de diapause qui correspondent aux date entre lesquelles la photopériode est inférieure à une certaine valeur. Pour l'implémentation du modèle de population de Lacour (2013), c'est une constante car ce modèle est calibré sur la région de Nice et la diapause a lieu entre le 30 septembre et le 10 mars.

3.1.1 Contribution personnelle à cette partie du projet

L'implémentation d'un modèle existant m'a permis de gagner beaucoup de temps de recherche et de calibration pour me concentrer sur la composante géographique de mon projet. Celle-ci m'a quand même demander un certain travail d'une part dans l'implémentation elle-même dans Matlab (Annexe 7.2) et d'autre part dans la collecte et le traitement des données météo (Annexe 7.1). J'ai également du modifier le taux de mortalité des œufs qui n'était pas cohérent dans la publication de Lacour (2013) car les figures présentant ses résultats ne correspondaient pas à la condition initiale qu'il imposait ainsi qu'à son taux de mortalité. Mon implémentation ne reproduit donc pas exactement les même résultat mais je m'en suis contenté pour pouvoir avancer dans mon projet.

3.1.2 Implémentation

Pour les données météo, j'ai récupéré sur le site de Météo-France des fichiers de données reprenant par mois une grande série de mesures pour un grand nombre de stations. J'ai donc écrit une fonction (Annexe 7.1) pour en extraire les données qui m'intéressaient.
Pour faciliter la suite du projet (càd la collecte des données pour toutes les différentes stations), j'ai rendu cette fonction la plus automatique possible. Elle n'a donc pas d'argument d'entrée. La première chose qu'elle fait est de charger les différentes informations sur les stations via la fonction GetStations.m (Annexe 7.4). Cette dernière utilise un fichier fourni par météo france (26/12/2017) reprenant les noms, numéros et coordonnées géographiques de chaque station de mesure (Figure 2).

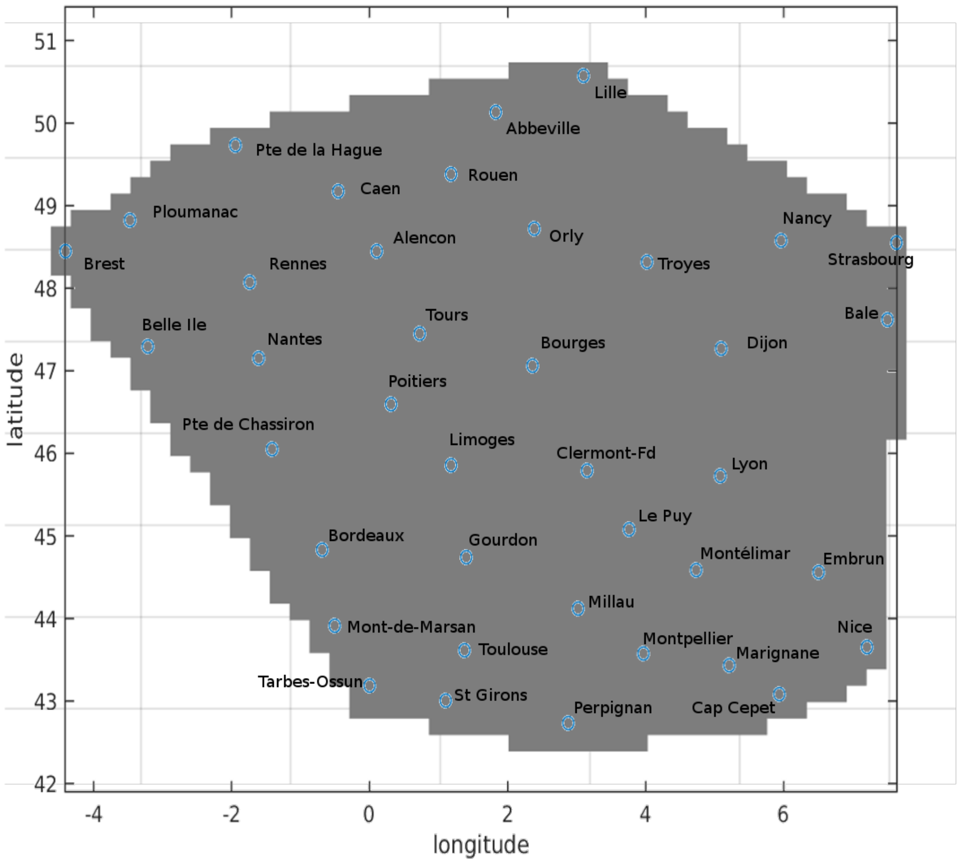

Figure 2: Localisation (approximative) des stations de mesure.

Une boucle va ensuite prendre un à un les 48 tableaux de données. Afin d'optimiser la lecture des données (*xlsread* est la fonction limitante pour les performances de ce programme), les fichiers de données ne sont ouvert qu'une seule fois et on sélectionne ensuite par station les 4 colonnes qui nous intéressent (numéro de station, date, température, précipitations). Ensuite, la fonction élimine les NaN présents dans les mesures de précipitations (et les autres mesures de la même ligne) et sépare les données par station.
À ce stade, plusieurs traitements des données sont effectués:

- Retrait des erreurs de mesures (*NaN*)
- Conversion des dates en format numérique de *Matlab*
- Moyenne des températures par jour [1]
- Somme des précipitations par jour

Les données sont ensuite sauvegardées dans une matrice par station et par mois (dans un fichier pour ne pas surcharger la mémoire vive). Le programme passe ensuite au mois suivant et répète l'opération. Une fois tous les tableaux de données lus, il faut les mettre bout à bout (Annexe 7.1, ligne 51). Pour ce faire, les fichiers enregistrés précédemment sont chargés station par station. Une fois toutes les données d'une station fusionnées, deux opérations sont effectuées:

- Élimination des *NaN* générés par la moyenne des températures
- Somme des précipitations sur 14 jours et normalisation pour les faire varier entre 0 et 1

[1] Cette moyenne génère des *NaN* car il existe des jours sans données associées

Une fois les données utilisables, le modèle de population peut être utilisé. Pour ce faire et toujours dans un soucis de flexibilité entre les différentes fonctions, celui-ci est implémenté dans *mosquito.m* (Annexe 7.2) qui est appelé par la fonction *runPop.m* (Annexe 7.3). La fonction *mosquito.m* reprend simplement les équations du modèle et les applique selon la population d'entrée, la température du jour et la somme normalisée des précipitations des 14 derniers jours. Afin d'observer l'effet de la diapause, j'ai voulu implémenter le calcul de la période de diapause selon la latitude mais les fonctions *datenum* et *datevec* prenaient beaucoup de ressources et j'ai donc simplifié le calcul de la diapause en la considérant constante.
runPop.m utilise un schéma Euler explicit pour faire tourner le modèle de population. Grâce à la fonction *GetStations.m*, celui-ci permet de faire tourner le modèle de population sur l'ensemble des stations pour la période souhaitée avec beaucoup de facilité. On a donc en sortie un graphique par station météo avec la même période de simulation et la même condition initiale. Il arrive que pour certains jours, il n'y ai pas de données météo. Dans ce cas, on prend une valeur d'un jour proche ou, dans le pire des cas, la valeur un an avant.

À ce stade du modèle, on ne tien pas compte de la diffusion géographique des moustiques mais on peut déjà voir beaucoup de choses intéressantes. D'abord, pour la zone où le modèle a été calibré (Figure 3), si on obtient une population stable dont le nombre d'individus varient selon les conditions météorologique (la troisième année, un temps plus sec empêche le développement des larves et des nymphes ce qui résulte en un nombre d'œufs en fin de saison quatre fois plus petit.
À Lyon (Figure 4), une population deux fois plus petite d'adultes arrive à se maintenir mais diminue à la deuxième et à la troisième année. Il serait intéressant de voir si elle se maintiendrait à plus long terme ou augmenterait avec le réchauffement climatique.
À Nancy (Figure 5), la population ne survis qu'à la première année.
Les 40 stations ont été testées et se rapprochent chacune de l'un de ces trois exemples.

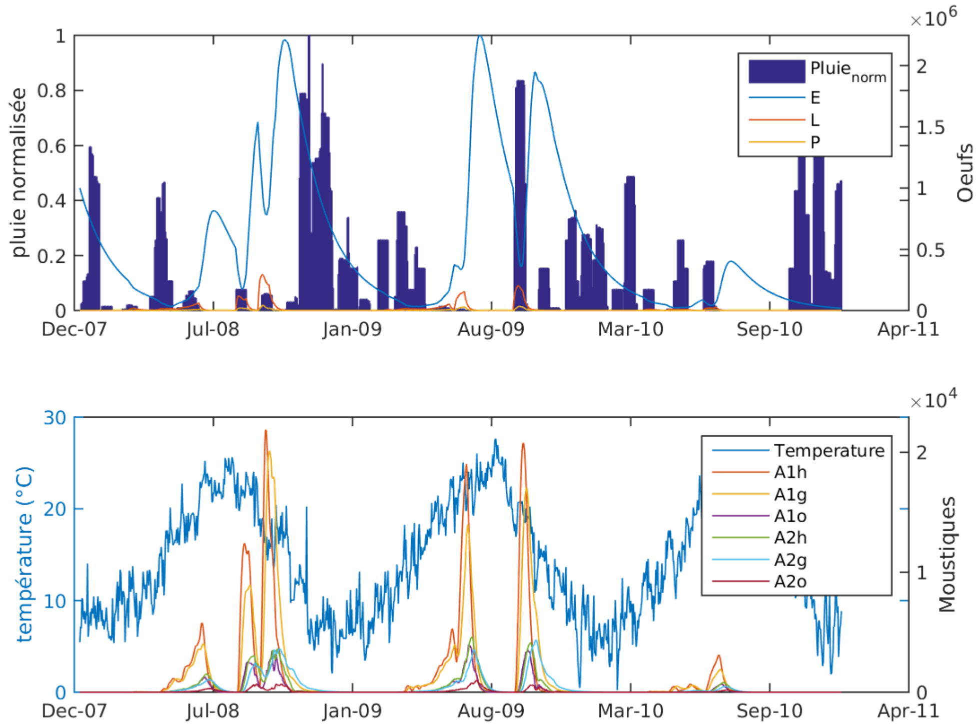

Figure 3: Évolution d'une population de moustiques à Nice entre le 1/1/2008 et le 31/12/2010 (population initiale: 10^6 œufs)

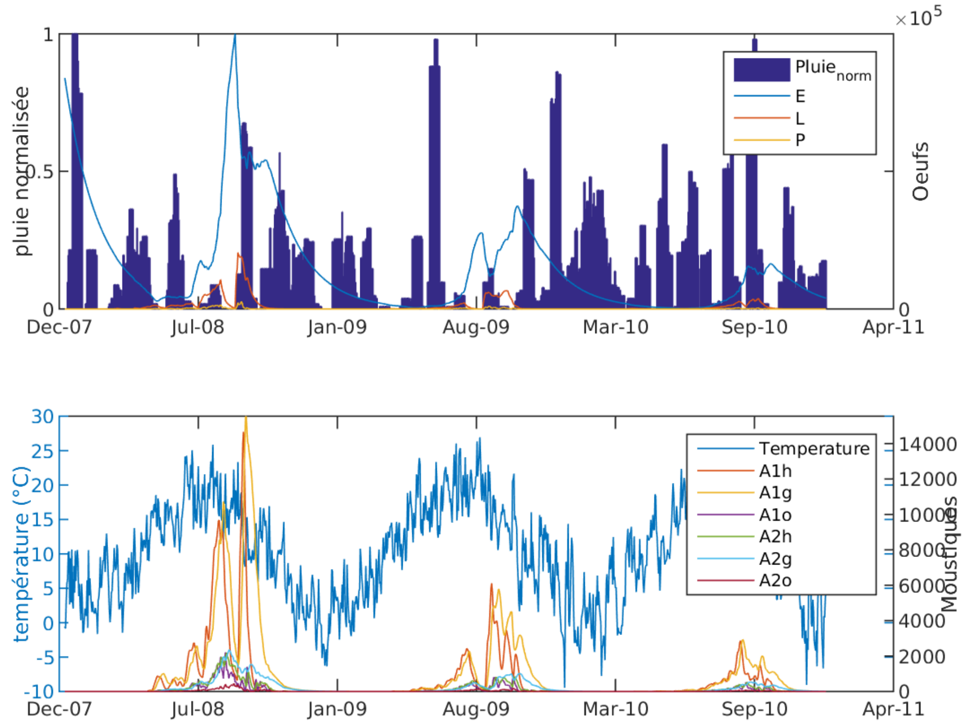

Figure 4: Évolution d'une population de moustiques à Lyon entre le 1/1/2008 et le 31/12/2010 (population initiale: 10^6 œufs)

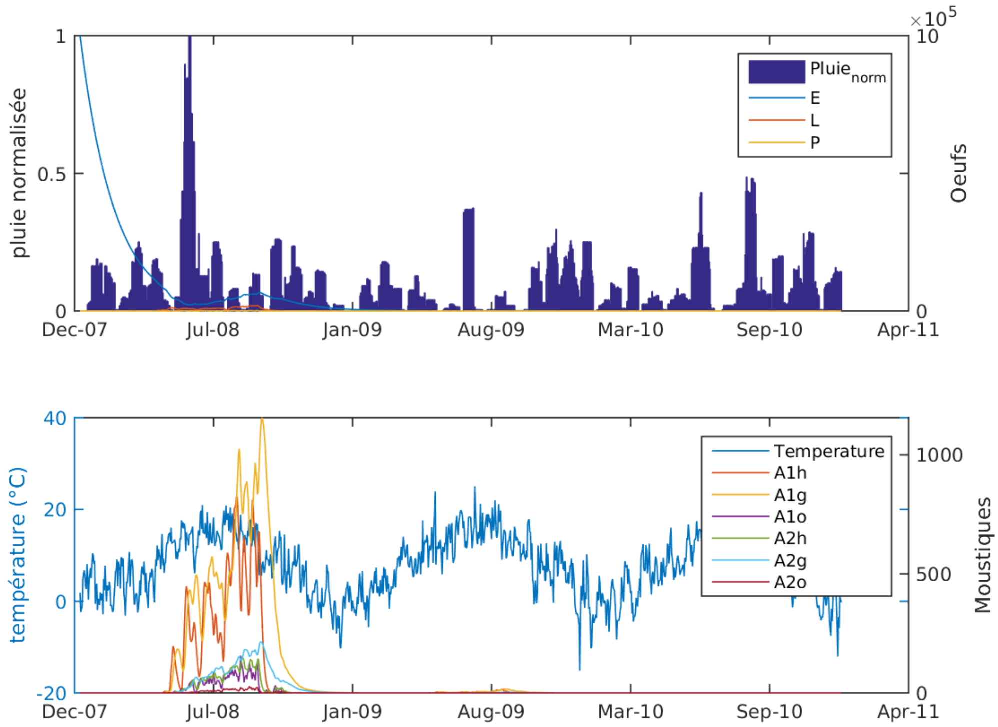

Figure 5: Évolution d'une population de moustiques à Nancy entre le 1/1/2008 et le 31/12/2010 (population initiale: 10^6 œufs)

3.2 Évolution d'une population sur un territoire en deux dimensions spatiales

Le modèle géographique Annexe 7.7) suit une équation de diffusion-réaction (Équation 1). Pour l'implémenter, une grille de la taille de la France a été prise. Pour faciliter l'implémentation, les coordonnées de cette grille sont les coordonnées géographiques adaptées pour que le coté gauche soit la longitude zéro et le côté du bas soit la latitude zéro.
Ce modèle utilise le schéma Adams-Bashforth pour la résolution temporelle. Pour chaque pas de temps, chaque point de la grille calcule l'équation de diffusion sur les stades aériens et l'équation de réaction (*mosquito.m*). Le modèle de population nécessite en entrée la température et la pluviométrie, celles-ci sont stockées dans une matrice générée par les fonctions *interp weather.m* et *get weather.m* (Annexes 7.5 et 7.6).
Le coefficient de diffusion est séparé en une composante selon x et une composante selon y (Calcul section 2.3). On peut modifier la précision de la grille à condition qu'elle soit la même que celle de l'interpolation des données météo (ici, 50).
Comme condition initiale, on place 10^6 œufs de moustiques au environs de Nice (1° à l'ouest).
Les conditions aux frontières sont définie comme tel:

- Pas de flux sur les frontières de la grille
- En dehors de la zone pour laquelle des données météo sont disponibles (pas d'extrapolation de celles-ci), le nombre total de moustique est mis à 0.

Afin de visualiser dans le temps, une vidéo est créée (Annexe 7.8) grâce à la fonction *VideoWriter*.

3.2.1 Optimisation des performances

Le modèle de population contient beaucoup d'équations et il a donc été nécessaire d'alléger les calculs pour réduire le temps de simulation (sans celles-ci, le modèle prenait plusieurs heures pour tourner).

- Si la somme des moustiques (tous stades compris) est inférieure à 1, il n'est pas nécessaire de calculer l'évolution de la population.
- Si la somme des moustiques qui diffusent est inférieure à 1, on considère qu'il n'y a pas de diffusion (permet d'éviter les quantités infinitésimales de moustiques qui alourdissent les calculs et n'ont pas de sens physique)
- L'utilisation de certaines fonctions comme *datenum* on été supprimées du modèle de population car trop gourmandes en calcul
- La grille et le pas de temps sont modulables selon les besoin en précision ou en performances

4 Validation du modèle

Le modèle de population étant repris d'une thèse, il a été calibré et validé. Malgré cela, je n'ai pas réussi à reproduire à l'identique les résultats présentées par celle-ci. Il serait donc intéressant de reprendre des résultats expérimentaux pour vérifier s'il tiens encore la route (on pourrait même l'essayer à d'autres endroits du monde). Une réserve toutefois se pose: lorsqu'on le fait tourner dans des régions plus froides, les températures négatives ne sont pas prise en compte par le modèle et sont donc mises à zéro (Annexe 7.3, ligne 43) pour éviter les bugs.

Le modèle géographique n'a également pas été validé. Une manière de le faire serait de chercher des cartes répertoriant les populations de moustique tigre en France et de voir s'il s'en rapproche ou s'il faut le recalibrer.
Toutefois, un comportement du modèle doit avant tout être corrigé. En effet, les moustiques ne meurent pas aussi vite que dans le modèle de population sans l'équation de diffusion. Celle-ci pose donc quelques problèmes. Une piste de solution que je n'ai pas réussit à appliquer est que les moustiques en cours de diffusion ne sont pas pris en compte par le modèle de population et donc échappent au taux de mortalité. Pour corriger ce problème, il faudrait calculer l'un après l'autre la diffusion et la réaction mais cela pose plein de problèmes pour l'intégration dans le temps.

5 Perspectives

Une fois les problèmes cités dans la section précédente résolus, ce modèle pourrait être utilisé pour prédire l'effet du réchauffement climatique sur les populations de moustique tigre. Pour y arriver, il y a plusieurs possibilités. La plus évidente est de prendre les différents scénario du Giec et les appliquer au modèle. Une méthode moins rigoureuse serait de faire varier manuellement les paramètres de température et de pluviométrie pour analyser l'impact de ces deux variables sur le modèle, ce qui permettrait de comprendre quels changements environnementaux seraient favorables ou défavorables aux populations de moustiques (permettant en même temps d'imaginer des moyens de lutte contre la prolifération de ceux-ci).

6 Conclusion

Ce modèle a déjà un petit potentiel et les possibilité d'amélioration sont assez claires. Sa validation et la résolution du problème lié à la diffusion permettraient de l'utiliser à des fins prédictives.

Ce projet a été un vrai plaisir pour moi. Il m'a permis de bien découvrir démarche de modélisation et tous les défis qu'elle comporte. Je tiens particulièrement à remercier Léolo qui malgré un emploi du temps chargé m'a apporté une aide précieuse.

Bibliographie

G. Lacour. A rainfall- and temperature-driven abundance model for aedes albopictus populations. *International Journal of Environmental Research and Public Health*, 2013.

G. Lacour. *Eco-physiological mechanisms and adaptive value of egg diapause in the invasive mosquito Aedes albopictus (Diptera : Culicidae)*. PhD thesis, UCL, faculté des sciences, 2016.

moustique tigre.info. Moustique tigre - portail d'information, 5/11/2017. URL `http://moustique-tigre.info/`.

météo france. portail de données publiques de météo-france, 26/12/2017. URL `https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32`.

7 Annexes

7.1 GetData.m

```
function [] = GetData()
%%Insérer le numéro et le nom de la station%%
Stations = GetStations;

% numSta = 07690;
% nomStation = 'Nice';
N0 = 2008; % année début
NF = 2010; % année fin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = N0:NF
    for j = 1:12
        %%% Insérer le fichier de données %%%
        file = sprintf('row/synop.%d%d.xlsx',i,j);
        data = eval(['xlsread','(file)']);
        %%%%%% Lecture par station %%%%%%
        for l = 1:length(Stations{1,2})
            numSta = Stations{1,2}(l);
            nomStation = sprintf('save/%d%d%d',Stations{1,2}(l),i,j);
            numer_sta = data(:,1);
            date = data(:,2);
            t = data(:,8) - 273;
            precip = data(:,40);

            NaN_hunter = isnan(precip);
            precip(NaN_hunter) = [];
            numer_sta(NaN_hunter) = [];
            date(NaN_hunter) = [];
            t(NaN_hunter) = [];

            station = numer_sta ~= numSta;
            precip(station) = [];
            date(station) = [];
            t(station) = [];
            date = datenum(date_conv(date));
            day = (floor(date(1)):1:floor(date(end)))';
            tday = zeros(length(day),1);
            Pday = zeros(length(day),1);
            Sday = ones(length(day),1) * numSta;
            for k = 1:length(day)
                dayn = date == day(k);
                tday(k) = mean(t(dayn)); % this mean generate NaN because some days there's no temperature data -> removed later
                Pday(k) = sum(precip(dayn));
            end
            save_data = {Sday,day,tday,Pday};

            save(nomStation,'save_data')
        end
```

```
    end
end
%%merge Data%%%
for m = 1:length(Stations{1,2})
    station = cell(1,4);
    for i = N0:NF
        for j = 1:12
            nomStation = sprintf('save/%d%d%d.mat',Stations{1,2}(m),i,j);
            load(nomStation)
            station{1,1} = [station{1,1} ; save_data{1,1}];
            station{1,2} = [station{1,2} ; save_data{1,2}];
            station{1,4} = [station{1,4} ; save_data{1,4}];
            station{1,3} = [station{1,3} ; save_data{1,3}];
        end
    end

    for l = 1:length(station{1,1})
        if isnan(station{1,3}(l))
            if l == length(station{1,1}) || isnan(station{1,3}(l+1))
                if l == 1
                    station{1,3}(l) = station{1,3}(l+2);
                else
                    station{1,3}(l) = station{1,3}(l-1);
                end
            else
                station{1,3}(l) = mean([station{1,3}(l-1),station{1,3}(l+1)]);
            end
        end
    end
    data = struct('numer_sta',station{1,1},'date',station{1,2},'temperature',station{1,3},'pluie_les_3_dernieres_heures',station{1,4});
    %%%%pluvio normalisée%%%%
    pluie_cumulee = data.pluie_les_3_dernieres_heures;
    for k = 1:length(data.pluie_les_3_dernieres_heures)
        if data.date(k) < 15
            pluie_cumulee(k) = sum(data.pluie_les_3_dernieres_heures(1:k));
        else
            date = data.date;
            jours = date >= date(k)-14 & date <= date(k);
            pluie_cumulee(k) = sum(data.pluie_les_3_dernieres_heures(jours));
        end
    end
    data.pluie_les_3_dernieres_heures = pluie_cumulee;
    data.pluie_les_3_dernieres_heures = data.pluie_les_3_dernieres_heures / max(data.pluie_les_3_dernieres_heures);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    nomSta = cell2mat(Stations{2,2}(m));
    nomSta = sprintf('stations/%s',nomSta);
    save(nomSta,'data')
```

```
end
return

function res = date_conv(input)
res = zeros(length(input),6);
res(:,6) = rem(input,100);
input = floor(input/100);
res(:,5) = rem(input,100);
input = floor(input/100);
res(:,4) = rem(input,100);
input = floor(input/100);
res(:,3) = rem(input,100);
input = floor(input/100);
res(:,2) = rem(input,100);
input = floor(input/100);
res(:,1) = input;
return
```

7.2 mosquito.m

```
function [M] = mosquito(m,t,l,P,T)
% This function calculate the evolution of a mosquito population at a
    step
% of time.
% The inputs are:
% m previous population (10x1 vector)
% t the time (format: count in days from the begin of the simulation)
% dt the step of time
% l the latitude
% P the rain
% T the temperature
% The output is M the population (10x1 vector)
%% Parameters
tstart = 70;
tend = 274;

param = struct('beta1',95,'beta2',75,'kl',250000,'kp',250000,...
     'sigma',0.5,'mue',0.02,'mul',0.08,'mup',0.03,'muem',0.1,...
     'mua',0.02,'mur',0.08,'Te',10.4,'TDDe',110,'gammaaem',0.4,...
     'gammaah',0.2,'gammaao',0.4,'Tag',10,'TDDag',77,'dia_st',tstart,...
     'diaend',tend,'temperature',T,'pluie',P);

%% Diapause
m2 = m;
month = t - 733408 +1;
while month > 364
     month = month - 365;
end
if month > param.dia_st && month < param.diaend
     z = 1;
else
     z = 0;
end
% z = get_diapause(l,t);
```

```
%% Model
m(1) = param.gammaao*(param.beta1*m2(7) + param.beta2*m2(10)) - (param.mue + z*fe(param)) *m2(1);
m(2) = z*fe(param)*m2(1) - (ml(param)*(1+m2(2)/kl(param)) + fl(param))*m2(2);
m(3) = fl(param)*m2(2) - (mp(param) + fp(param))*m2(3);
m(4) = fp(param)*m2(3)*param.sigma*exp(-param.muem*(1+m2(3)/kp(param))) - (ma(param)+param.gammaaem)*m2(4);
m(5) = param.gammaaem*m2(4) - (ma(param)+param.mur+param.gammaah)*m2(5);
m(6) = param.gammaah*m2(5) - (ma(param) + fag(param))*m2(6);
m(7) = fag(param)*m2(6) - (ma(param)+param.mur+param.gammaao)*m2(7);
m(8) = param.gammaao*(m2(7)+m2(10)) - (ma(param)+param.mur+param.gammaah)*m2(8);
m(9) = param.gammaah*m2(8) - (ma(param)+fag(param))*m2(9);
m(10) = fag(param)*m2(9) - (ma(param)+param.mur+param.gammaao)*m2(10);
M = m;
return


function res = fe(param)
if param.temperature > param.Te && param.pluie ~= 0
    res = (param.temperature-param.Te)/param.TDDe;
else
    res = 0;
end
return

function res = fl(param)
res = max(-0.0007 * param.temperature^2 + 0.0392 * param.temperature - 0.3911,0);
return

function res = fp(param)
res = max(-0.0008 * param.temperature^2 + 0.051 * param.temperature - 0.0319,0);
return

function res = fag(param)
if param.temperature > param.Tag
    res = (param.temperature-param.Tag)/param.TDDag;
else
    res = 0;
end
return

function res = ml(param)
res = exp(-param.temperature/2) + param.mul;
return

function res = mp(param)
res = exp(-param.temperature/2) + param.mup;
return
```

```
function res = ma(param)
res = max(param.mua,0.04417+0.00217*param.temperature);
return

function res = kl(param)
res = param.kl * (param.pluie + 1);
return

function res = kp(param)
res = param.kp * (param.pluie + 1);
return

function [Z] = get_diapause(l,t)
% This fonction calculate the photoperiod with the date and the latitude
% P = heures de jour  hyp: linéaire avec la latitude et au long de l'année
%
% Au cercle polaire: 66,55°
% P = 24 au solstice été (21-Jun)
% P = 0 au solstice hiver (21-Dec)
% Au tropique cancer: 23.45°
% P = 12 au solstice été
% P = 6.25 au solstice d'hiver
% Au tropique capricorne: -23,45°
% P = 12 au solstice hiver
X = [23.45,66.25];
T = [datenum('21-Jun-0000'),datenum('21-Dec-0000')];
P = [ 0 , 24 ; 6.25 , 12 ];

%%%%%%% Diapause à Nice %%%%%%%%%

lat_Nice = 43.648833;
PP = interp2(X,T,P,lat_Nice,datenum('30-Sep-0000'));

%%%%%% Diapause à l au temps t %%%
time = datevec(t);
t_adjust = datenum([0 time(2) time(3) 0 0 0]);
if t_adjust < datenum('21-Jun-0000')
    t_adjust = t_adjust + 183;
end

if interp2(X,T,P,l,t_adjust) > PP
    Z = 1;
else
    Z = 0;
end
return
```

7.3 runPop.m

```
function [] = runPop()
% Use this function to run the mosquito.m model without the geographic
% model
```

```
close all
%% Parameters & Initialization
dt = 1;
T0 = datenum('1-Jan-2008');
Tend = datenum('31-Dec-2010');
l = 43.648833;
m0 = [1e6 0 0 0 0 0 0 0 0 0];
t = T0:dt:Tend;
dS = 1;

Stations = GetStations();

for s = 1:length(Stations{1,2})
    close all
    nomSta = cell2mat(Stations{2,2}(s));
    nomSta = sprintf('stations/%s.mat',nomSta);
    load(nomSta)
    M = zeros(length(t),length(m0));
    M(1,:) = m0;
    %% Model run with euler explicit
    for k = 2:length(t)
        j = k + 733407;
        flag_day = data.date == j;
        if sum(flag_day)>1
            disp('>1')
        elseif sum(flag_day) == 0
            if sum(data.date == j-1) == 1
                flag_day = data.date == j-1;
                disp('one day missing')
            elseif sum(data.date == j-3) == 1
                flag_day = data.date == j-3;
                disp('two or three days missing')
            else
                disp('error: more than three day missing')
                flag_day = data.date == j-365; % data of last year taken
                datestr(l)
                disp(nomSta)
            end
        end
        T = max(0,data.temperature(flag_day));
        P = data.pluie_les_3_dernieres_heures(flag_day);

        m = mosquito(M(k-1,:),t(k),l,P,T,dS);
        M(k,:) = M(k-1,:) + dt*m;
    end
    %% Plot
    H = figure;
    set(gcf,'units','normalized','outerposition',[0 0 1 1],'name',
        cell2mat(Stations{2,2}(s)));
    subplot(2,1,1)
    ax1 = plotyy(data.date,data.pluie_les_3_dernieres_heures,t,[M(:,1) M
        (:,2) M(:,3)],'bar','plot');
    datetick('x','mmm-yy','keepticks')
```

```
    ylabel(ax1(1), 'pluie normalisée')
    ylabel(ax1(2), 'Oeufs')
    ylim(ax1(2),[0 max(M(:,1))])
    legend('Pluie_{norm}','E','L','P')

    subplot(2,1,2)
    ax2 = plotyy(data.date,data.temperature,t,[M(:,5) M(:,6) M(:,7) M
        (:,8) M(:,9) M(:,10)]);
    datetick('x','mmm-yy','keepticks')
    ylabel(ax2(1),'température (°C)')
    ylabel(ax2(2),'Moustiques')
    ylim(ax2(2),[0 max(max((M(:,4:10))))])

    legend('Temperature','A1h','A1g','A1o','A2h','A2g','A2o')
    nomSta = sprintf('runPop/%s',cell2mat(Stations{2,2}(s)));
    saveas(H,nomSta,'png')
end
return
```

7.4 GetStations.m

```
function [Stations,S] = GetStations()
data = importdata('postesSynop.csv');

NomsStations = cell(size(data.textdata,1)-1,1);
NumStations = zeros(size(data.textdata,1)-1,1);
for k = 2:size(data.textdata,1)
    NomsStations{k-1,1} = data.textdata{k,2};
    NumStations(k-1) = str2double(data.textdata{k,1});
end
Latitude = data.data(:,1);
Longitude = data.data(:,2);

flag_station_France = Latitude > 51.089 | Latitude < 42.333 & Longitude >
    8.23 | Longitude < -4.795;
NumStations(flag_station_France) = [];
NomsStations(flag_station_France) = [];
Latitude(flag_station_France) = [];
Longitude(flag_station_France) = [];

Stations = {'Num',NumStations;'Nom',NomsStations;'Latitude',Latitude;'
   Longitude',Longitude};
P = plot(Longitude,Latitude,'o');
ylabel('latitude')
xlabel('longitude')
title('Stations météo en France')
axis equal
saveas(P,'Stations_météo','png')
return
```

7.5 interp weather.m

```
function [Daily_temp_interp,Daily_rain_interp] = interp_weather(t)
% Input: weather data and location of each station
%        Precision of the grid for geographic model
```

```
%         Time (datenum) already floored to day
% Output: rectangle matrix with interpoled data
% Interpolation method: linear , no extrapolation
%
if nargin ~= 1
    t = datenum('1-Jan-2008');
    disp('insert time!!!')
end
%%%% Load Stations %%%%%%
Stations = GetStations;

%%%%% Correction Lat & Long %%%%%
Stations{3,2} = Stations{3,2} - 42.333;
Stations{4,2} = Stations{4,2} + 4.795;
% plot(Stations{4,2},Stations{3,2},'o')
% xlabel('Longitude')
% ylabel('Latitude')
% title('Weather stations in France')

%%%%%%%% Load Weather Data %%%%%%%%
Daily_weather = zeros(length(Stations{3,2}),3); % first column: station,
   second: temp, third: rain
for i = 1:length(Stations{1,2})
    nomSta = cell2mat(Stations{2,2}(i));
    nomSta = sprintf('stations/%s.mat',nomSta);
    load(nomSta)
    flag_day = data.date == t;
    if sum(flag_day)>1
        disp('>1')
    elseif sum(flag_day) == 0
        if sum(data.date == t-1) == 1
            flag_day = data.date == t-1;
            disp('one day missing')
        elseif sum(data.date == t-3) == 1
            flag_day = data.date == t-3;
            disp('two or three days missing')
        else
            disp('error: more than three day missing')
            flag_day = data.date == t-365; % data of last year taken
            disp(datestr(t))
            disp(nomSta)
        end
    end
    Daily_weather(i,1) = Stations{1,2}(i);
    Daily_weather(i,2) = max(0,data.temperature(flag_day)); % Pour ré
       soudre un bug : pas de temp négative (chercher meilleur solution
       si temps)
    Daily_weather(i,3) = data.pluie_les_3_dernieres_heures(flag_day);
end
Daily_temp_interp = scatteredInterpolant([Stations{4,2},Stations{3,2}],
   Daily_weather(:,2),'linear','none');
Daily_rain_interp = scatteredInterpolant([Stations{4,2},Stations{3,2}],
   Daily_weather(:,3),'linear','none');
```

```
return
```

7.6 get weather.m

```
function [] = get_weather()
p = 50;
LY = 51.089 - 42.333;     % Latitude length
LX = 8.23 + 4.795;        % Longitude length
t = datenum('31-Dec-2010') - datenum('1-Jan-2008') + 1;
X = (0:LX/(p-1):LX)';
Y = (0:LY/(p-1):LY)';
[Xq,Yq] = ndgrid(X,Y);
figure('units','normalized','outerposition',[0 0 1 1])


for i = 1:t
    t = datenum('1-Jan-2008')-1 + i;
    [Daily_temp_interp,Daily_rain_interp] = interp_weather(t); % interpolation

    Daily_temp_data = Daily_temp_interp(Xq,Yq);
    Daily_rain_data = Daily_rain_interp(Xq,Yq);
    daily_temp = sprintf('save/temp%d.mat',i);
    daily_rain = sprintf('save/rain%d.mat',i);
    save(daily_temp,'Daily_temp_data')
    save(daily_rain,'Daily_rain_data')
   % uncomment to plot
    subplot(1,2,1)
    surf(Xq,Yq,Daily_temp_data);
    shading interp
    view(0,90)
    xlabel('Longitude')
    ylabel('Latitude')
    title(['Temperature in France: ',datestr(t)])
    set(gca, 'CLim', [0 ; 30]);
    colormap(flipud(hot))
    colorbar()
    subplot(1,2,2)
    surf(Xq,Yq,Daily_rain_data);
    shading interp
    view(0,90)
    xlabel('Longitude')
    ylabel('Latitude')
    title(['Precipitation in France: ',datestr(t)])
    set(gca, 'CLim', [0, 1]);
    colormap(flipud(hot))
    colorbar()
   T(i) = getframe(gcf);
end

movie2avi(T,'weather.avi') % /!\ utilise bcp de mémoire (changer en videowriter si + longue durée)

return
```

7.7 geographic model.m

```
function [] = geographic_model()
%
% The goal is to compute a diffusion-reaction model in two geographic
% dimensions.
% The diffusion coefficient is based on the daily flight distance and the
    reaction equation use the
% population model of Aedes albopictus (Lacour,2013)
% The simulation take place in France with the weather data of "météo
% france" (linear interpolation)
% Initial conditions: 1e6 mosquito in Nice area
%
% This code is inspired by the Emmanuel Hanert's code in the LBRTI2102
% course
close all
%% Model parameters

p = 50;   % precision of the grid
Kx = 0.0013; % diffusion coefficient
Ky = 0.001;
T0 = datenum('1-Jan-2008');
Tend = datenum('31-Dec-2010');

[x,y] = get_grid(p);
Dx = size(x,1)/p;
Dy = size(y,1)/p;
Ds = round((Dx*78.85 * Dy*111)/100)*100; % surface par rectangle [km^2]
dS = Ds * 100; %surface en ha
dt =  1;%0.1* min([(Dx^2)/(2*Kx),(Dy^2)/(2*Ky)]);
time = T0:dt:Tend;

fprintf('Mosquito in 2D\n')
fprintf('Time step = %f\n',dt);

%% Initialization

C = get_initial_solution(x,y);          % initial condition

%for movie
flag_frame = 1;
old_t = time(1);
mov = VideoWriter('mosquito2.avi');
mov.FrameRate = 24;
figure('units','normalized','outerposition',[0 0 1 1])
open(mov);

France = imread('France_background2.png');

frame = plot_sol(x,y,C,old_t,France,Ds);
writeVideo(mov,frame);

%% Integration of the equation

```

```
r0 = zeros(size(C));    % right-hand-side at time step n
r1 = zeros(size(C));    % right-hand-side at time step n-1
r2 = zeros(size(C));    % right-hand-side at time step n-2
difx_ij = zeros(10,1);
dify_ij = zeros(10,1);

for k=1:length(time)
    t = time(k);
    r2   = r1;
    r1   = r0;
    % -> computes the solution inside the domain
    for i=1:length(x)
        for j=1:length(y)
            Cij = C(i,j,:)  ;
            % values around Cij + zero flux condition on the boundaries
            if i>1
                Cl = C(i-1,j,:);   % left value
            else
                Cl = Cij;
            end
            if i<length(x)
                Cr = C(i+1,j,:);    % right value
            else
                Cr = Cij;
            end
            if j>1


                Cd = C(i,j-1,:);   % "down" value
            else
                Cd = Cij;
            end
            if j<length(y)
                Cu = C(i,j+1,:);    % "up" value
            else
                Cu = Cij;
            end

            % diffusion pour les 7 phases adultes
            for l = 4:10
                difx_ij(l) = Kx*(Cr(l)-2*Cij(l)+Cl(l))/(Dx*Dx);
                dify_ij(l) = Ky*(Cu(l)-2*Cij(l)+Cd(l))/(Dy*Dy);
            end

            if sum(Cij) < 1 % optimisation du modèle
                if difx_ij + dify_ij < 1 % optimisation
                    r0(i,j,:) = 0;
                else
                    r0(i,j,:) = difx_ij + dify_ij;
                end
            else
                % réaction : modèle de population
                tim = datevec(t);
```

```
                    tim = datenum([0 tim(2) tim(3) 0 0 0]);
                    daily_temp = sprintf('save/temp%d.mat',tim);
                    daily_rain = sprintf('save/rain%d.mat',tim);

                    load(daily_temp) % matrice données interpolées
                    Temp = Daily_temp_data(i,j);

                    load(daily_rain) % matrice données interpolées
                    Pluie = Daily_rain_data(i,j);

                    if isnan(Temp) || isnan(Pluie) % hors des données météo
                        r0(i,j,:) = 0;
                    else
                        q = mosquito(Cij,t,i,Pluie,Temp); % modèle de population
                        for m = 1:10
                            r0(i,j,m) = difx_ij(m) + dify_ij(m) + q(m);
                        end
                    end
                end
            end
        end
        % -> updates the solution
        if k==1
            % -> Euler explicit at 1st timestep
            C = C + dt.*r0;
        elseif k==2
            % -> 2nd order Adams-Bashforth at 2nd timestep
            C = C + dt*((3/2)*r0 - (1/2)*r1);
        else
            % -> 3rd order Adams-Bashforth for all other timesteps
            C = C + dt*((23/12)*r0 - (16/12)*r1 + (5/12)*r2);
        end
            C = max(C,0);
        % -> plots intermediate solutions
        if t > old_t
            flag_frame = flag_frame + 1;
            frame = plot_sol(x,y,C,old_t,France,Ds);
            writeVideo(mov,frame);
            old_t = t;
        elseif k == length(time)
            flag_frame = flag_frame + 1;
            frame = plot_sol(x,y,C,old_t,France,Ds);
            writeVideo(mov,frame);
        end
end
close(mov);
return

function [x,y] = get_grid(p)
LY = 51.089 - 42.333;     % Latitude length
LX = 8.23 + 4.795;        % Longitude length

```

```
[x,y] = ndgrid(0:LX/(p-1):LX,0:LY/(p-1):LY);
return

% === FUNCTION TO COMPUTE THE INITIAL SOLUTION === %
function Cinit = get_initial_solution(x,y)
% la condition initiale place 10^6 oeufs de moustique sur Nice
Cx = 7.209 + 4.795 -1;
Cy = 43.648833 - 42.333;
sigma = 0.05;
Einit = 1e6 * exp(-((x-Cx).^2+(y-Cy).^2) / (2*sigma^2));
Cinit = zeros(size(x,1),size(y,2),10);
Cinit(:,:,1) = Einit;
return

% === FUNCTION TO PLOT SOLUTION === %
function [E] = plot_sol(x,y,C,t,France,Ds)

set(gcf,'Unit','normal','Color',[1 1 1],'Name','Mosquito 2D')

subplot(1,2,1)
    imagesc([x(1) x(end)],[y(1) y(end)],flipud(France)); hold on
    Cplot = log10(C(:,:,1));
    surf(x,y,Cplot); shading interp; caxis([0 8]);
    set(gca,'ydir','normal');
    view(0,90)
    xlabel('Longitude')
    ylabel('Latitude')
    dm = sprintf('log(Oeufs) / %d km^2',Ds);
    axis([0 13 0 8.9])
    pbaspect([1 1 1])
    title('Oeufs de moustiques')
    colormap(flipud(hot))
    c1 = colorbar;
    c1.Label.String = dm;

subplot(1,2,2)
    imagesc([x(1) x(end)],[y(1) y(end)],flipud(France)); hold on
    Cplot = log10(C(:,:,4) + C(:,:,5) + C(:,:,6) + C(:,:,7) + C(:,:,8) +
        C(:,:,9) + C(:,:,10));
    surf(x,y,Cplot); shading interp; caxis([0 8]);
    set(gca,'ydir','normal');
    view(0,90)
    xlabel('Longitude')
    ylabel('Latitude')
    dm = sprintf('log(Moustiques) / %d km^2',Ds);
    axis([0 13 0 9])
    pbaspect([1 1 1])
    title('Moustiques')
    colormap(flipud(hot))
    c2 = colorbar;
    c2.Label.String = dm;

suptitle(['Population de moustiques en France: ',datestr(t)]);
```

```
E = getframe(gcf);
return
```

7.8 Clé USB

- Vidéo du modèle géographique
- Vidéo des données météo interpolées
- Graphiques de population pour chaque station
- Scripts