Rapport du projet de programmation

Projet II : Simulation d'un feu de forêt

Dans le cadre du cours de :
Méthode de calcul et de programmation en physique (SPHYB107)
2022-2023

D'Addamio Diego
BAC 1
20221950

MATLAB®

Université de Namur
Faculté des sciences
Département de Physique

UNIVERSITÉ DE NAMUR

Table des matières

Le feu est par définition un dégagement d'énergie thermique et lumineuse par suite d'une combustion, elle est engendrée par une source de chaleur, un combustible et un comburant. Il a la capacité de faire une réaction en chaine rapide et peu contrôlable. Sa propagation est multidirectionnelle via les combustibles proches, tant que les conditions de combustions soient respectées.

La combustion spontanée est un procédé d'auto-échauffement d'une matière ayant une température d'inflammation assez basse, qui finit par s'enflammer dû à l'oxydation en présence d'humidité et de l'énergie thermique ne pouvant se dégager. Elle serait alors une source de chaleur mère, entraînant la propagation d'un feu. De plus les matériaux facilitant ce principe sont d'origines végétales tel que les tas de foins, charbon, etc…

Donc ce phénomène pourrait donc être une cause d'un feu de forêt si ceux-ci se forment. Alors essayons de trouver un système qui pourrait nous aider à déterminer si une forêt serait à surveiller dans le cas où, une combustion spontanée se produirait ou même déterminer le seuille de densité limite d'une forêt entrainant la propagation totale d'un feu de forêt. Cependant, ce fait n'est pas à confondre avec la combustion instantanée[1].

Nous pouvons alors avec un système de programmation cellulaire, simuler un feu de forêt grâce à ces conditions, à partir d'un arbre aléatoirement mis en feu par cette combustion spontanée et à une certaine densité d'arbre sur une surface choisie.

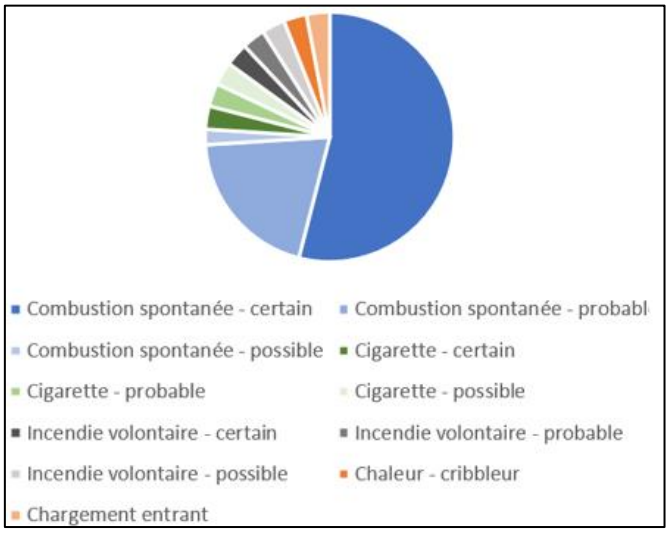

Fig. 1

Figure 1 : Analyse de cause de déclanchement de feu par suite d'un compostage – Rynk & Buggeln (2009)

[1] Combustion d'une personne vivante sans raison apparente.

Pour commencer, replaçons les différents points importants qui sont plus qu'utile à la résolution de la problématique annoncée dans l'introduction d'un point de vue statistique. Le point de vue visuel de la propagation d'un feu de forêt ne sera pas traité ici car celui-ci ne nous sera d'une grande utilité dans notre problématique.

1. Détail de densité

Le détail de densité est la subdivision du pourcentage maximum de la densité (100%), le principe est de « découper » autant de fois le pourcentage maximum pour percevoir le maximum de détail possible. Ainsi, nous allons simuler tous les découpages pour le plus de précision.

Donc, le cadre statistique est basé sur le détail de densité voulu par l'utilisateur par la variable *Max_Cut_Density*, il sert à voir, avec le découpage de la densité en pourcent, l'augmentation du rapport d'arbre brulé et d'arbre initial en fonction de ce découpage. En voici deux comme exemple :

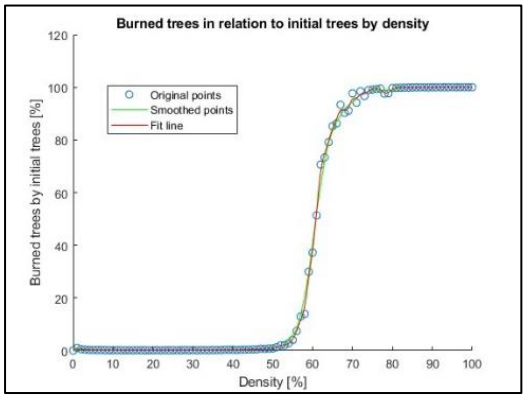

Fig. 2

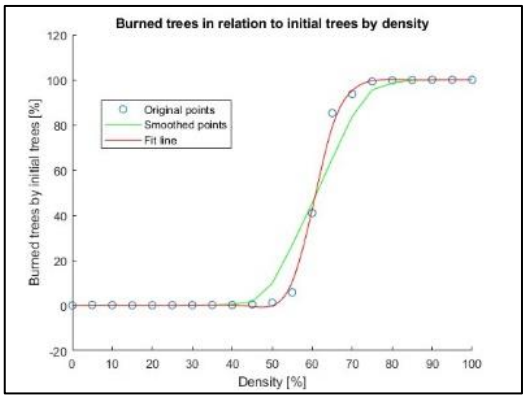

Fig. 3

Figure 2 : *Max_Cut_density* équivaut à 100, c'est-à-dire que chaque pourcent a été simulé.

Figure 3 : *Max_Cut_density* équivaut à 20, c'est-à-dire que chaque 5 pourcents a été simulé.

2. Génération de la forêt et propagation

La génération de la forêt dépend du détail de densité (3.1.) entré précédemment mais aussi d'un caractère aléatoire sur la disposition des arbres et des plaines. Ce qui fait que deux forêts ne seront non similaires sauf dans les cas extrêmes de densité. De plus, le caractère aléatoire ne respecte pas forcément la densité voulue. Ceci a donc toute son importance de le prendre ce critère en compte dans une étude statistique.

De plus, dans l'étude statistique, nous étudierons exclusivement une forêt faite d'arbre et de plaine pour une étude optimale de la problématique. Dans le cadre de visualisation, il sera possible d'y ajouter certains paramètres optionnels.

3. Moyenne de donnée

La génération d'une matrice d'élément nul ou unitaire est formée grâce au détail de densité comme expliqué ci-dessus (3.2.). La génération se faisant aléatoirement, il est donc possible que la simulation ne soit pas représentative pour une certaine densité. C'est à ce moment là que nous devons prévoir une moyenne pouvant approximer la représentation générale d'une situation. Donc, nous allons déterminer le nombre de fois qu'une simulation avec la même densité d'arbre, dans notre matrice, pour la même densité dans le but de faire cette approximation. Ce critère-ci choisi par l'utilisateur, sous la variable *Average_Loop*. L'importance de celui-ci est montrée ci-dessous par les figures 4 et 5.

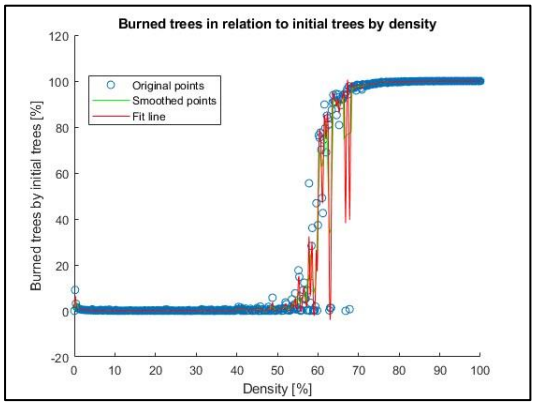

Fig. 4

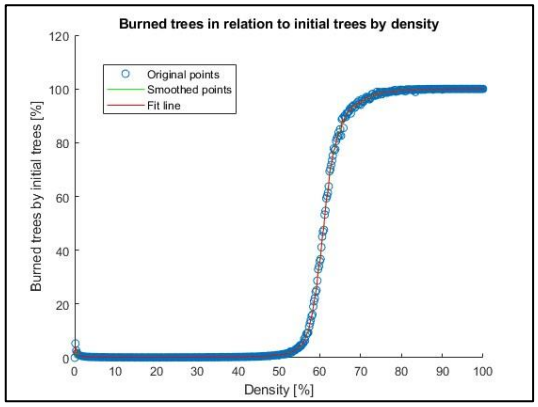

Fig. 5

Figure 4 : *Average_Loop* équivaut à 1, c'est-à-dire que la même densité a été simulée une seule fois

Figure 5 : *Average_Loop* équivaut à 500, c'est-à-dire que la même densité a été simulée 500 fois

1. Données préliminaires
   1.1. Gestion des variables et constantes

Le programme commence par la gestion des variables et constantes, selon la volonté de l'utilisateur. Il nous demandera la taille de la simulation, cette taille, nommée *SizeWorld*, fera référence à l'ordre de la matrice carrée.

Ensuite, il initialisera tout les paramètres pouvant être choisi par l'utilisateur, tout ces paramètres sont représenté par des variables associée à un booléen, initialement égal à *false*.

1.2. Choix de l'utilisateur et paramétrage

Il nous demandera ensuite de choisir entre l'étude statistique de la propagation d'un feu de forêt ou l'affichage en temps réel de celui-ci, ce choix est initialisé dans une boucle *while* et ne s'arrête que lorsque l'utilisateur y aura mis une réponse valide. Ce choix va réglé les demandes de paramètrage, via un *switch* par la valeur entrée par l'utilisateur.

Dans le cas de l'étude, il nous demandera les valeurs de *Max_Cut_density* et *Average_Loop* précédemment expliquée dans la partie théorique.

Dans le cas où nous voudrions afficher la simulation, il nous demandera alors les différents paramètres nécéssaires tel que la densité ainsi que le nombre de simulation voulue. Il nous demanderas aussi les paramètres optionnels, tel que la présence de vent, de rivière, de tonnerre ou la création d'un GIF.

2. Génération
   2.1. Arbres

2.1.1. Étude

Dans le cas de l'étude, les densités seront choisies dépendamment de *Max_Cut_density* mais pour pouvoir toute les faire directement l'une à la suite de l'autre, nous mettons deux boucles *for* emboitées de façon qu'une fasse varier la densité et l'autre répétant celle-ci.

La fin de ces deux boucles sont situées dans la fin du programme pour englober tout le processus de génération, de propagation ainsi que la récolte de donnée.

2.1.2. Affichage

Dans le cas de l'affichage, la densité est déterminée dès le début par l'utilisateur. Par la présence des boucles emboitées, nous pouvons alors se servir d'une pour pouvoir simuler, d'affiler, des feux de forêt de même paramétrage.

La génération commence par la création d'une matrice carrée nulle, nommée *World*, d'ordre *SizeWorld+2* afin de créer une bordure remplie de zéros pour prévoir la généralisation de la propagation du feu expliquée plus bas.

Il va ensuite générer une matrice d'ordre *SizeWorld*, de nombre aléatoire entre 0 et 1 et va comparer chaque valeur par rapport à la densité, si l'élément arbitraire a_{ij} est plus petit que la densité générée ou voulue, alors a_{ij} sera de valeur 1, sinon 0.

Il va égaler la matrice résultante ci-dessus à la sous matrice de *World*, *World (2 : SizeWorld+1, 2 : SizeWorld+1)*, donc formation d'une barrière de 0. Nous allons associés une valeur à la variable *Tree_Init*, étant le nombre de 1 dans cette matrice, ce qui désigne le nombre d'arbre dans notre forêt.

2.2. Rivière

Une rivière se forme selon son booléen attitré à la variable *River_ON*.

Pour initialiser la rivière, il va être généré un nombre entier entre 3 et *SizeWorld-1*, désignant le centre de la rivière dont sa largeur est comprise entre 3 et 5. On comprend donc pourquoi son centre est désigner entre 3 et *SizeWorld-1*. La rivière est générée ligne par ligne de haut en bas par une boucle *for* de 2 à *SizeWorld+1*.

Dans cette boucle nous assignons, en fonction du centre, la largeur aléatoire entre 1 et 2 en partant de ce centre, non inclus, à gauche et à droite. Donc la largeur varie au fur et à mesure qu'elle se construit. De plus, le centre peut se déplacer à gauche et à droite en l'incrémentant de 1 ou -1.

Mais son déplacement va être orienté si le centre se trouve aux extrémités de la matrice de simulation, c'est-à-dire à la colonne 3 ou *SizeWorld-1*. Attention, tous les mouvement pris par la rivière sont dû à un facteur aléatoire. Dans la figure 6, nous pouvons y voir la génération d'une rivière à déplacement et embranchement aléatoire.

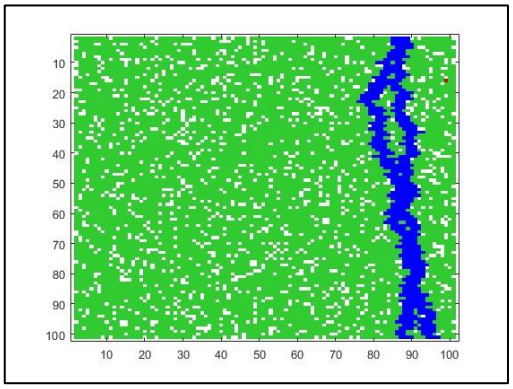

Fig. 6

Figure 6 : Affichage avec une rivière

2.3. Vent

Le vent est désigné par une direction et un sens, ici, sa direction sera définie par un entier aléatoire entre -2 et 2, nommé *Wind_Direction*. Si |*Wind_Direction*|=2 alors la direction sera horizontale, si |*Wind_Direction*|=1 alors sa direction sera verticale. Son sens sera désigné par le signe attitré à celui-ci.

3. Propagation du feu

3.1. Premier arbre mis en feu

Pour qu'un feu commence il faut choisir un arbre qui sera le premier à brûler. Nous allons le choisir aléatoirement en prenant une ligne et une colonne aléatoire dans la matrice *World*. Or le caractère aléatoire ne choisit pas un élément ayant la valeur 1, représentant un arbre. Donc nous disposons d'une boucle *while* qui va s'arrêter uniquement si elle en a trouvé un. Cependant, le programme va utiliser *break* si *Tree_Init* est nul, une sécurité contre une possible boucle infinie.

Une fois que nous avons celui-ci, nous allons attitrer la valeur 3 aux coordonnées trouvées pour annoncer que cet arbre prend feu. Nous pouvons l'observer dans la figure 6 en haut à droite.

Pour la suite, le programme va copier notre matrice *World*, cette copie se nomme *Ghostworld*, elle va représenter les futures modifications.

Dans une boucle *while*, nous mettons comme première condition *Tree_Init* si celui-ci est nul, alors le programme utilise *break*. La seule unique façon d'arrêter cette boucle est qu'il n'y a plus d'arbre en feu, donc que la somme des 3 dans la matrice *World* soit nul. Nous utilisons la fonction externe *Colorworld*, son usage sera expliqué dans le point 6.

Dans celle-ci, il évaluera chaque élément de *World*, sans les bordures, si celui-ci est de valeur 3, alors c'est un arbre en feu, donc il va se propager via 4 conditions tel qu'une représente une 1 directions possibles. Si la direction arbitraire arrive face à un arbre dans *World*, alors la valeur de celui-ci sera modifiée par 3 dans *Ghostworld*, signifiant qu'il va brûler. Une fois que l'évaluation aux 4 directions est finie, la valeur de l'élément regarder sera modifié en 2 dans *Ghostworld*, signifiant qu'il sera brûlé.

Alors, une fois que tous les éléments ont été évalué, le programme va égaler *World* à *Ghostworld*, signifiant que les modifications se sont faites dans le présent.

Dans le cas de l'affichage, nous allons afficher la matrice *World* et encore utilisé *ColorWorld*.

4. Gestion des données

Pour rappel, tout ce procédé est inclus dans deux boucles (2.1.1).

Pour chaque simulation de même densité, il va additionner le rapport entre le nombre d'arbre brûlé et le nombre d'arbre initial et les stocker dans un vecteur nommé *DataForest*. Dont les lignes représentent la densité en fonction du détail de densité.

5. Graphiques représentatifs

Le programme va effectuer la moyenne arithmétique des données en divisant *DataForest* par *Average_Loop*.

Il va premièrement faire un nuage de points des coordonnées, en *x* les valeurs de chaque densité et en *y* les valeurs du vecteur *DataForest*. Il va ensuite lisser, via *smooth*, les données du vecteur *DataForest* pour en relier en vert, les points toujours en relation avec la densité effectuée. Et pour terminer il va un passage à la courbe du nuage de point initial pour former la meilleure courbe possible via la fonction *fit*. La précision peut être interprétée via la superposition des deux courbes.

6. La fonction externe : ColorWorld

Cette fonction permet de déterminé la *colormap* adéquate à la simulation. Cette fonction possède 3 arguments, une valeur logique, une matrice et sa taille. La valeur logique permet de ne pas charger de *colormap* lorsqu'il n'y a pas d'affichage. La matrice référée y est importante, la matrice va être analysée selon le nombre de certains éléments dans celle-ci. Et la taille de la matrice y est mise aussi car le programme la possède dans ses variables, donc au lieu de demander un calcul, nous allons nous référer à quelque chose déjà calculé.

7. La ToolBox externe : GIF

Cette ToolBox créée par Chad Greene, permet de capturer les séquences voulue de l'affichage par la fonction *gif* afin de générer un GIF à la fin d'une simulation enregistrer par la fonction initialement entrée, *gif('myfile.gif')*.

Ces fonctions sont appliquées si et seulement si l'utilisateur en verra son usage.

8. La ToolBox externe : Curve Fiting ToolBox

Cette ToolBox, issue de MathWorks, permet de lisser la courbe passant par nos points initiaux par *fit* ainsi que de relier les points eux même lisser par *smooth*.

Par exemple dans Fig. 3, la courbe rouge représente le lissage de la courbe passant par les points initiaux et la verte représente les points ajuster pour une courbe.

Le projet avait donc comme but de déterminer le seuille critique de densité par laquelle un feu de forêt serait dévastateur. Initialement, j'eu l'idée qu'à partir d'une densité de 40% l'entièreté de la forêt serait brulée. Or d'après nos simulations, il serait de l'ordre de 70-75%. Mais effectivement plus la densité d'arbre augmente et plus le rapport d'arbre brulé et d'arbre initial augmente. Donc les données sont bien réalistes en gardant à l'esprit qu'il ne s'agit que d'une simulation.

Cependant, n'oublions pas que ceci est une simulation et que le programme possède des avantages et inconvénients.

Les points faibles seraient qu'il nous faut choisir entre l'affichage et l'étude, il aurait été plus apprécié si nous avions pu constater si la présence d'eau ou de vent change drastiquement les données finales. De plus, nous sommes obligés de choisir une génération d'un monde, carré, aléatoire dans le cadre de l'affichage.

Pour ce qui est des points forts, je pense que ce code à deux usages possède une bonne optimisation et ne demande pas beaucoup de ressources et variables. Le fait de pouvoir choisir les paramètres, détails et précisions sont aussi des éléments non négligeables dans ses points forts.

Conclusion

En conclusion, par les données récoltées par ce programme, les forêts possédants une densité supérieur à 70% seraient plus apte à causer des dégâts majeurs pour un arbre mis en feu. Cet arbre supposé en feu, dû à la combustion spontanée des éléments qui l'entoures, est donc à éviter. Un feu de forêt peu donc commencer à partir d'un simple échauffement provoquant la décimation de la faune et la flore locale. Donc les amas de foins ou de déchets organiques ne devraient pas être placé dans une forêt qui de plus est humide. Vu que l'humidité joue un rôle majeur dans ce principe, la surveillance d'une forêt sèche est de haute importance mais ce n'est pas parce qu'une forêt est humide qu'elle va ne jamais bruler.

Pour finir, ces quelques pensées finales sont à notées comme améliorations envisageables de ce programme

Création d'une application linéaire s'occupant de cette propagation, évitant le balayage continu des valeurs de la matrice principale, donc augmentant la vitesse de celui-ci.

Augmenter le nombre de paramètres qui peuvent rentrer en jeu tel que la différence d'arbre, ainsi que les rejets émis par la combustion de ceux-ci.

Capture de valeurs réalistes par l'analyse d'une image satellite d'une vraie forêt pour se rendre compte de l'importance d'un feu dans celle-ci.

Remerciements

Je voudrais dédier ces remerciements à HENRY François, pour sa grande aide à la rédaction de ce rapport ainsi qu'à la motivation donnée pour la finalisation et l'optimisation de ce programme.

Nicolas, R. (2022, août 16). *Les combustions spontanées de biosolides : le cas du compost*. ATOSSA. https://atossa.fr/2020/05/25/les-combustions-spontanees-de-biosolides-le-cas-du-compost/

Wikipedia contributors. (2022, 10 mars). *Combustion spontanée physique*. https://fr.wikipedia.org/wiki/Combustion_spontan%C3%A9e_physique

Wikipedia contributors. (2022, décembre 4). *Combustion humaine spontanée*. https://fr.wikipedia.org/wiki/Combustion_humaine_spontan%C3%A9e

Chad, G. (2021, 30 janvier). gif - File Exchange - MATLAB CentralFile Exchange - MATLAB Central. https://nl.mathworks.com/matlabcentral/fileexchange/63239-gif?s_tid=srchtitle_gif_2

MathWorks. *Curve Fitting Toolbox*. (s. d.). MATLAB. https://nl.mathworks.com/products/curvefitting.html

MATLAB Logo : valor, história, PNG. (2022, 18 février). https://logosmarcas.net/matlab-logo/