Université de Lille

2017-2018

Modélisation par automates cellulaires de problèmes physiques et mécaniques

Thomas Drucké
Elian Opdebeeck
Charlène PHAN

Licence 3 SMI

Encadrants :
Stéphanie Mathez-Bagot
Enrico Cazalvarini

Remerciements

Nous adressons nos remerciements aux personnes qui nous ont aidés dans la réalisation de notre projet intégrateur.

En premier lieu, nous remercions Monsieur Enrico Calzavarini, assistant professeur à l'université de Lille et chercheur au laboratoire de mécanique de Lille pour l'aide qu'il a fournie à nous guider dans notre travail et à trouver des solutions pour avancer. Nous le remercions également pour sa disponibilité et la qualité de ses conseils.

Nous remercions aussi Madame Stéphanie Mathez-Bagot, enseignante de Technique d'Expression et de Communication à l'université de Lille 1 pour nous avoir relus et corrigés, et pour ses précieux conseils dans l'élaboration de notre écrit et de notre oral.

Nous sommes également reconnaissants envers Monsieur Emmanuel Leriche, pour son encadrement et son implication dans la réalisation de ce projet ainsi que tous les professeurs responsables de ce module d'enseignement.

Enfin, nous tenons à adresser des remerciements à toutes les personnes de notre entourage pour leur précieuse aide à la relecture et à la correction de notre projet.

Introduction

Avec la démocratisation d'Internet, le numérique a bouleversé nos vies et nos activités, en changeant même notre vision du monde. Celui-ci a permis de faire un pas de géant dans de nombreux domaines et se retrouve aujourd'hui au cœur de grands enjeux sociétaux tels que la gestion de l'énergie et des ressources naturelles, la santé, ou encore la préservation de l'environnement. Cependant, aussi importantes que soient les innovations technologiques, l'Homme n'est pas capable de prévoir tous les phénomènes et catastrophes naturelles. En revanche il est possible, grâce à des logiciels de simulation numérique, d'anticiper certaines de ces situations afin d'agir et éventuellement de les éviter. Le coût colossal de certaines ressources a incité les chercheurs à développer de nouvelles méthodes de travail. C'est pourquoi aujourd'hui la modélisation numérique apparaît comme l'un des meilleurs moyens de simuler des situations réelles à moindre coût, de manière rapide et efficace. De nombreux logiciels sont donc apparus ces dernières années afin de répondre à différentes problématiques et nous allons étudier l'un d'eux, Netlogo.

Dans un premier temps, nous nous familiarisons avec le logiciel Netlogo à travers l'étude d'un programme dont l'objectif est de simuler lors d'un incendie, la propagation du feu à travers une forêt selon différentes variables. Dans un second temps, on s'intéresse à un problème dont l'enjeu est primordial: la fonte des glaces en Arctique, à travers le développement d'un programme réalisé à partir du logiciel Netlogo. Nous nous intéressons finalement aux limites de notre problème et les possibilités d'optimisation ainsi que sur le rôle des automates cellulaires appliqué à des situations physiques complexes.

Sommaire

Familiarisation avec le logiciel Netlogo

1.1 Origine des automates cellulaires

La naissance des automates cellulaires résulte de la collaboration entre le mathématicien et physicien américano-hongrois John von Neumann (1903-1957) et le mathématicien américain Stanislaw Ulam (1909-1984) au début des années 50. Leur motivation était de réaliser des systèmes dynamiques à partir de règles d'évolution simples. Cela leur a permis de modéliser les phénomènes complexes d'autoreproduction que l'on observe dans la nature.

Un automate cellulaire est une grille à maillage carré que l'on peut également représenter par un tableau en 2 dimensions. Chaque case du tableau est appelé cellule et chacune de ces cellules peut être dans un certains état. Partant d'une configuration donnée, l'état de ces cellules varie en fonction de l'état des cellules voisines. C'est donc selon des règles de changement d'état et de voisinage qu'ils ont réussit à générer des figures complexes que l'on peut trouver dans la nature telles que les spirales ou les fractales.

À partir de là, se posait la question suivante : ces mécanismes récursifs peuvent-ils réussir à expliquer des phénomènes physiques complexes ?

1.2 A quoi sert la modélisation ?

Aujourd'hui, même si beaucoup de problèmes peuvent être résolus à l'aide de formules mathématiques, physiques et mécaniques, il existe d'autres problématiques très complexes tels que les tremblements de terre que nous n'avons pas encore réussi à traduire à travers des équations différentielles simples, car les facteurs qui entrent en jeu sont beaucoup trop aléatoires pour qu'on puisse les résoudre de manière analytique. C'est donc là qu'intervient la modélisation numérique.

La modélisation consiste à construire et à utiliser un modèle qui est une représentation simplifiée de la réalité pour montrer les aspects importants du système étudié. Il est impossible de réaliser un modèle qui soit complètement fidèle à la réalité, c'est pourquoi on va sélectionner les facteurs qui nous semblent les plus pertinents afin de modéliser notre problème de manière à ce qu'il se rapproche le plus de la réalité. Il existe donc énormément de façons différentes de représenter un même problème.

Différentes approches sont possibles pour structurer la pensée du modélisateur et la formalisation du problème. Nous pouvons citer en premier les approches mathématiques telles que les équations différentielles ainsi que les approches informatiques (par simulations) comme les automates cellulaires qui sont des systèmes multi-agents. L'utilisation des systèmes multi-agents apparaissent comme la solution la plus adapté du fait de leur proximité avec la réalité et de leur adaptabilité à tous les contextes.

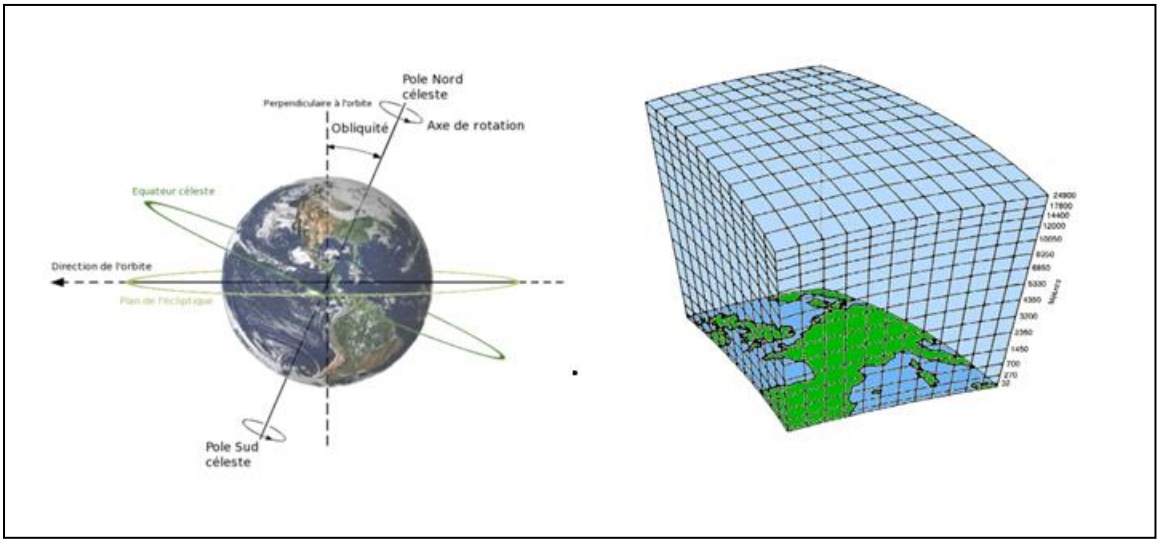

FIGURE 1: REPRESENTATION SIMPFIEE DE LA REALITE

1.3 Fonctionnement du logiciel Netlogo

Netlogo est un environnement de modélisation programmable pour simuler des phénomènes naturels ou sociaux, développé par l'université Northwestern en 1999. C'est un logiciel assez simple pour les étudiants et les enseignants, mais assez avancé pour servir d'outil puissant pour les chercheurs dans de nombreux domaines.

Le premier avantage de Netlogo est sa facilité de prise en main. Cela vient à la fois de son interface graphique et du langage de programmation employé, le langage Logo.

La plate-forme Netlogo correspond à une approche de simulation dite « à pas de temps discret », c'est-à-dire qu'elle fait évoluer un ensemble d'entités, aussi appelés automates cellulaires, par intervalles de temps successifs et de durée identique. Cette approche de modélisation correspondante consiste donc à identifier les entités que l'on souhaite intégrer dans le modèle puis à décrire le comportement de chacune d'entre elles au cours de chaque intervalle de temps. Parmi ces entités on distingue tout d'abord l'environnement : celui-ci est modélisé sous la forme d'un quadrillage constitué de cellules que l'on appelle « patches ». Ensuite nous avons les agents mobiles « turtles ». Ils se déplacent dans l'environnement, et peuvent interagir aussi bien avec l'environnement qu'avec les autres agents mobiles. La dernière entité qui joue le rôle prédominant est l'observateur. C'est lui qui contrôle le déroulement de la simulation en envoyant des instructions à tous les agents du monde simulé.

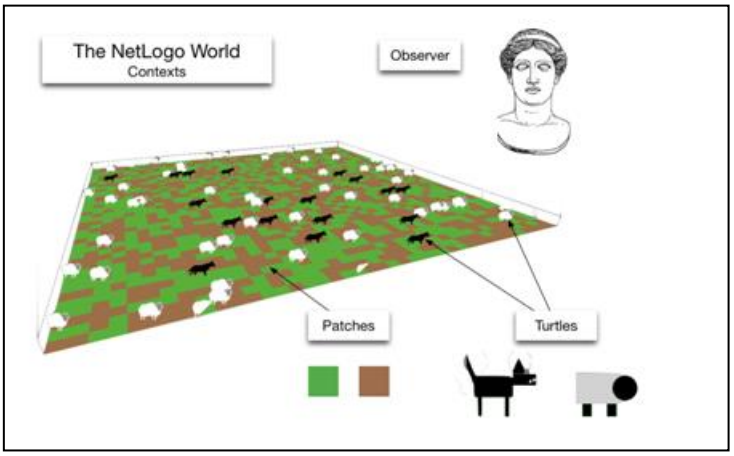

FIGURE 2: PRESENTATION DES AUTOMATES CELLULAIRES

Le logiciel Netlogo permet de faire tourner des modèles déjà disponibles sur la plate-forme mais aussi de faire tourner notre propre modèle qu'on peut ensuite déposer en ligne pour partager avec les autres utilisateurs. Lorsque l'on lance le logiciel, on retrouve trois fonctionnalités principales ayant des rôles différents dans la construction du modèle : Interface, Info et Code.

- l'onglet info contient les informations de description du modèle ;
- l'onglet code contient une zone éditable de texte dans laquelle le modélisateur écrit son code netlogo. Il possède notamment un bouton check qui est activé à chaque sauvegarde du fichier et permet de détecter des erreurs syntaxiques ;
- l'onglet interface contient l'interface graphique du simulateur. Lorsqu'un nouveau modèle est créé, elle affiche un environnement par défaut. Elle peut ensuite être facilement personnalisée par le modélisateur. Ce dernier peut ajouter des éléments de contrôle de la simulation (boutons), des éléments permettant de configurer les valeurs d'entrées des paramètres de la simulation (sliders, switch, chooser, etc.) Ou d'afficher des sorties et des indicateurs de la simulation (monitor plot, etc.). En bas de l'onglet interface, on trouve la partie command center, qui contient la console dans laquelle sont affichés les différents messages produits au cours de la simulation.

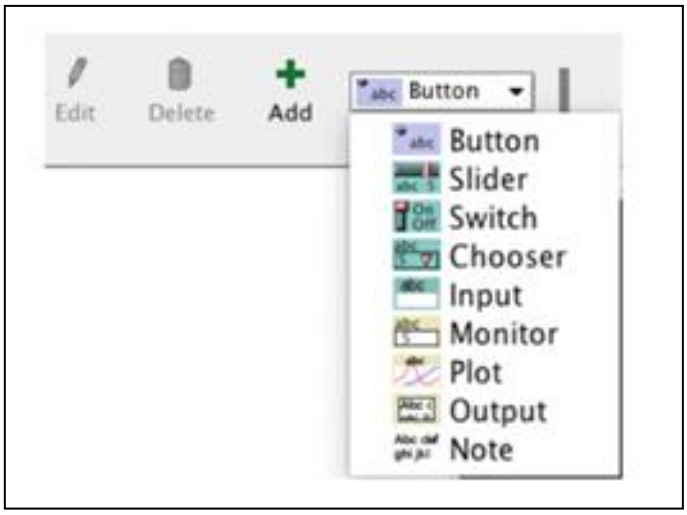

FIGURE 3: ELEMENT DE CONTROLE DE LA SIMULATION

A partir du Command Center, l'utilisateur peut donner à une ou plusieurs « turtles » un certains nombre d'instructions dans le Command Center afin de lui faire changer son déplacement, son orientation sa couleur, etc. L'utilisateur peut y exécuter directement du code Netlogo, ce qui lui permet notamment de tester la simulation en cours. Il peut également choisir si le code est exécuté dans le contexte de l'observer, des turtles ou des patches.

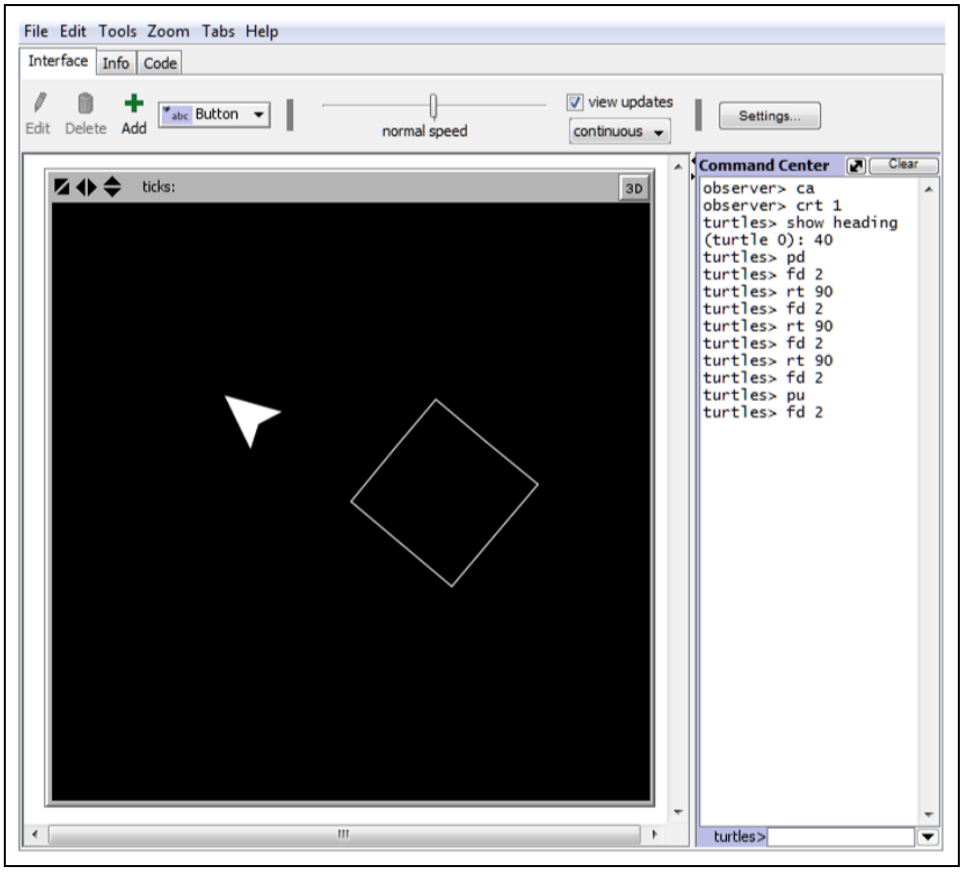

FIGURE 4: DEPLACEMENT D'UNE TURTLE

- observer> crt 1 : cette fonction permet de créer un certain nombre de turtles donné par l'utilisateur qui seront situé au centre des coordonnées mais qui auront une couleur et une orientation aléatoire autour du monde ;
- fd n : (forward) fait bouger la tortue vers l'avant ;
- rt n : (right) fait tourner la tortue vers la droite selon n degrés ;
- pd : (pen-down) laisse une trace lorsque la tortue se déplace ;
- setxy xy : se déplace à la position d'abscisse x et d'ordonnée y.

1.4 Étude d'un programme simulant un feu de forêt

Pour pouvoir introduire une dynamique au sein de l'environnement, nous allons étudier et modéliser des phénomènes qui s'appliquent au niveau de l'espace plutôt qu'au niveau de l'individu. C'est le cas pour l'exemple de la propagation d'un feu de forêt ou encore la propagation d'une maladie.

Pour cela, il faut associer un comportement pour chaque cellule (patch) qui compose le monde. Chaque patch s'anime alors de manière individuelle en faisant évoluer son état interne et en agissant dans le monde.

Nous avons étudié le modèle « Fire » dont le but est de simuler lors d'un incendie, la propagation d'un feu à travers une forêt. Ce modèle nous a permis de comprendre comment les automates cellulaires interagissent entre eux notamment à partir de règles simples communes à tous les automates. Les règles sont les suivantes :

- le nouvel état de chaque cellule est déterminé à partir de sa position spatiale dans l'univers de l'automate, en examinant les états des cellules voisines ;

- toutes les cellules constituant l'univers de l'automate sont mises à jour de manière simultanée et synchrone ;

- toutes les cellules de l'automate utilisent les mêmes règles de transition pour déterminer leur état suivant ;

- un automate cellulaire se déroule dans le temps de manière discrète, génération après génération, en opposition avec la plupart des phénomènes physiques continus.

Dans la simulation du feu de forêt chaque cellule d'une grille carrée peut se trouver dans l'un des quatre états : vide, arbre, cendre, feu et cet état va varier en fonction de l'état des cellules voisines.

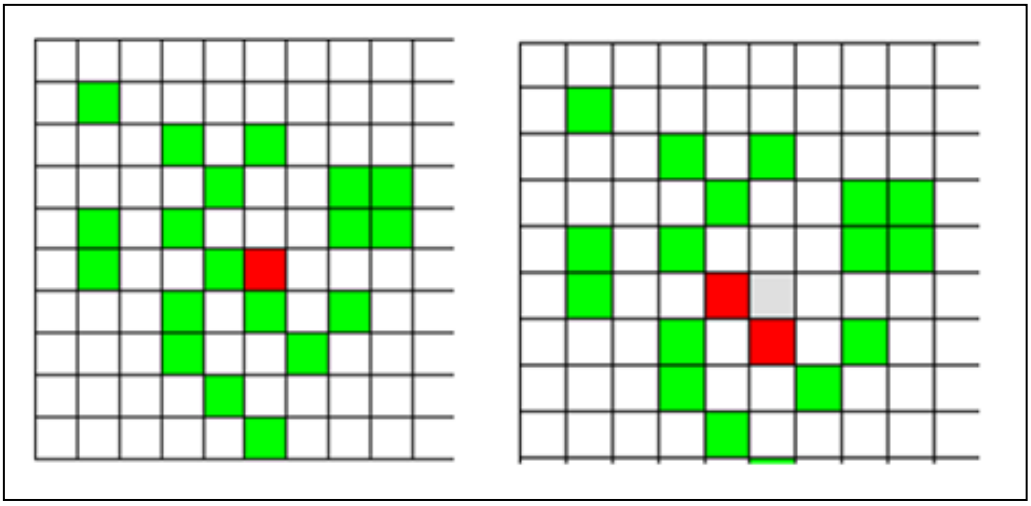

FIGURE 5: DIFFUSION DU FEU A TRAVERS LA GRILLE

C'est selon des règles de changement d'état que le feu va pouvoir se propager. La diffusion du feu à travers la grille se fait de la manière suivante :

Les cellules correspondant à des arbres deviennent en feu si l'une de leur cellule voisine est en feu. Les cellules en feu passeront en cendres à la génération suivante et les cellules en cendres deviendront vides à la génération suivante.

Ce programme nous a également permis de comprendre l'importance des différentes variables telles que la densité ou encore l'effet du vent qui modifie considérablement les chances du feu de brûler intégralement la forêt.

Cas 1 :

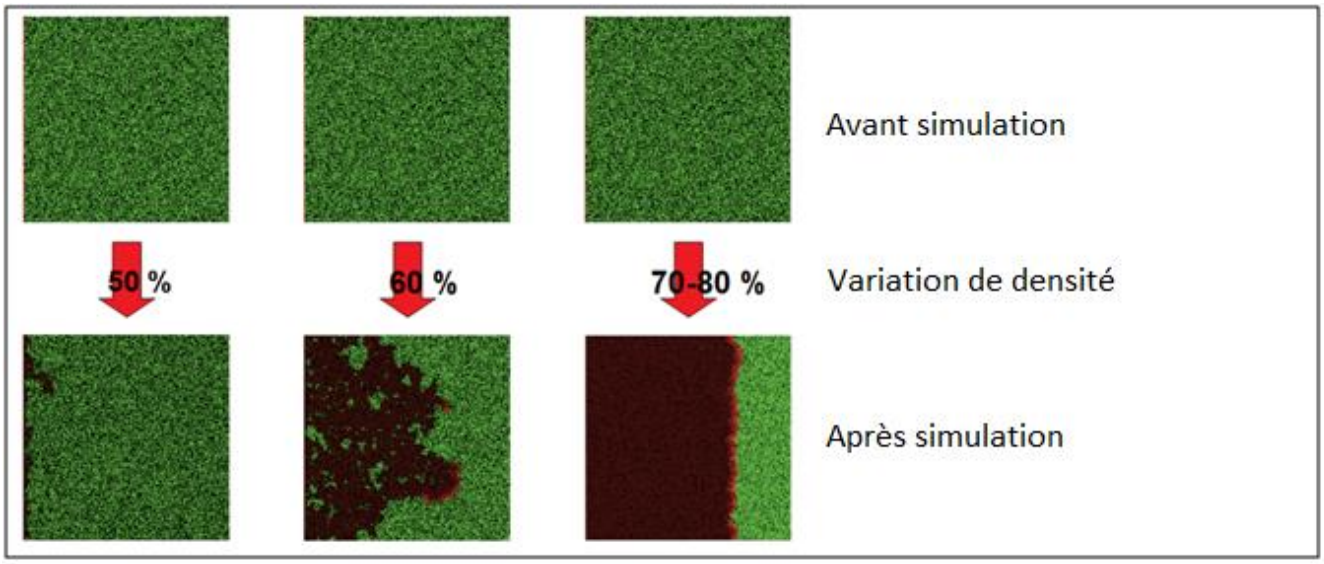

FIGURE 6: SIMULATION EN FONCTION DE LA DENSITE

La densité représente la proximité des arbres les uns avec les autres. Lorsque celle-ci est faible (de l'ordre de 50 %), on remarque que le feu commence à brûler mais n'a aucune chance d'atteindre l'extrémité de la forêt. Avec une densité de l'ordre de 60 % le feu brûle progressivement la forêt en laissant certains arbres non touchés. Et lorsque la densité est de l'ordre de 70-80 %, le feu brûle intégralement la forêt de manière vive.

Cas 2 :

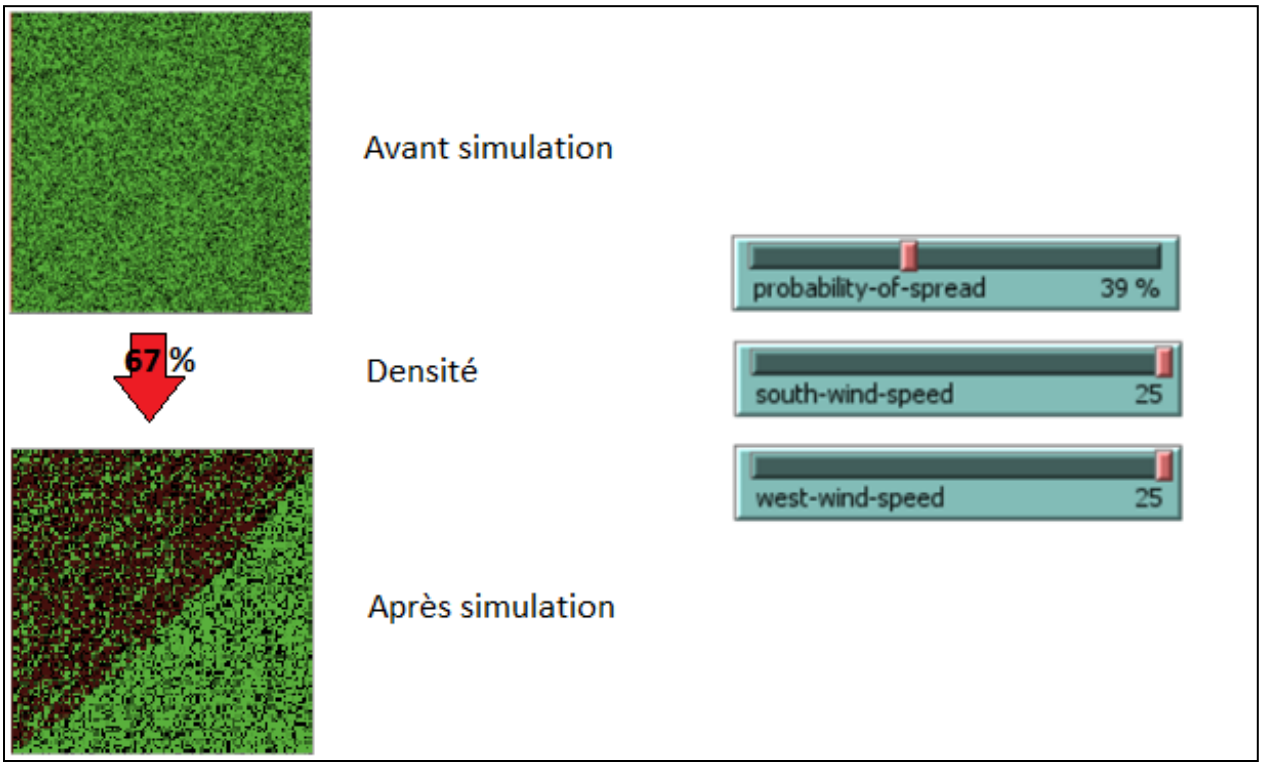

FIGURE 7: INTEGRATION D'UNE NOUVELLE VARIABLE, LE VENT

Dans le cas 2, d'autres variables entrent en jeu telles que la probabilité de propagation du feu, la vitesse du vent venant du sud et celle venant de l'ouest. Ces variables modifient notablement l'orientation de la propagation du feu à travers la forêt.

Utilisation du logiciel Netlogo pour simuler le problème de fonte des glaces

2.1 Historique

Pour modéliser un système, il faut faire en sorte que l'écart entre le modèle et le système à reproduire soit le plus faible possible. Ceci représente un défi de taille pour plusieurs raisons. Il faut, en effet, bien connaître le système à modéliser, trouver les outils adéquats de simulation, créer le modèle avec ces outils en respectant les contraintes du système à imiter.

Un modèle répond toujours à une question scientifique. Pour cela il est nécessaire de bien choisir les facteurs que l'on souhaite prendre en compte et ceux que l'on néglige afin que le modèle soit le plus fiable possible.

FIGURE 8: MARES DE FONTE

Cette photo est très représentative du phénomène de fonte des glaces que nous tentons de modéliser. A mesure que la glace fond, l'eau s'accumule dans les renfoncements de la surface et les approfondie, formant des mares de fonte à la surface du glacier. Ces bassins d'eau douce sont séparés de la mer salée en dessous et autour de celle-ci, jusqu'à ce que des ruptures dans la glace fusionnent les deux.

2.2 Paramètres pris en compte et négligés

Le problème de la fonte des glaces est, comme souvent lors de phénomènes naturels, très compliqué à étudier. En effet, l'étude précise d'une telle situation est difficile car les proportions sont colossales. On étudie notre modèle à l'échelle mésoscopique (échelle intermédiaire entre le macroscopique de notre monde quotidien et le microscopique des atomes et des molécules). La première simplification que nous avons effectuée est que nous considérons un morceau de glace qui a pour objectif de représenter la banquise de manière générale. Nous prenons ainsi en compte un glacier qui flotte sur l'océan que nous observons du dessus.
Les paramètres qui ont un impact sur la fonte de la glace sont très nombreux et il est impossible de tous les considérer. De plus, certains sont négligeables comparés à d'autres. Nous avons donc défini les paramètres qui nous semblaient essentiels à prendre en compte :

- la simulation de la topographie: notre modèle doit contenir différentes hauteurs et épaisseurs de glace afin de rendre le modèle réaliste. L'utilisateur a la possibilité de rendre le relief du glacier plus ou moins lisse afin de tester différentes configurations.

- la fonte de la glace : l'augmentation de la température de l'air et de l'eau fait fondre la glace qui se transforme en eau. Nous prenons aussi en compte le fait que la masse volumique de l'eau n'est pas la même sous sa forme liquide et sa forme solide.

- le déplacement de l'eau : en raison de la gravité, l'eau est un fluide qui se propage vers le point où la hauteur de la glace est la plus basse. On fait donc écouler l'eau issue de la glace vers les points où l'altitude est la moins élevée ce qui va engendrer la formation de mares de fonte.

- la percolation de l'eau dans l'océan : une partie de l'eau formée traverse la glace pour s'écouler dans l'océan, c'est ce qu'on appelle la percolation. Ce paramètre est important à prendre en compte car son influence sur le problème est conséquente.

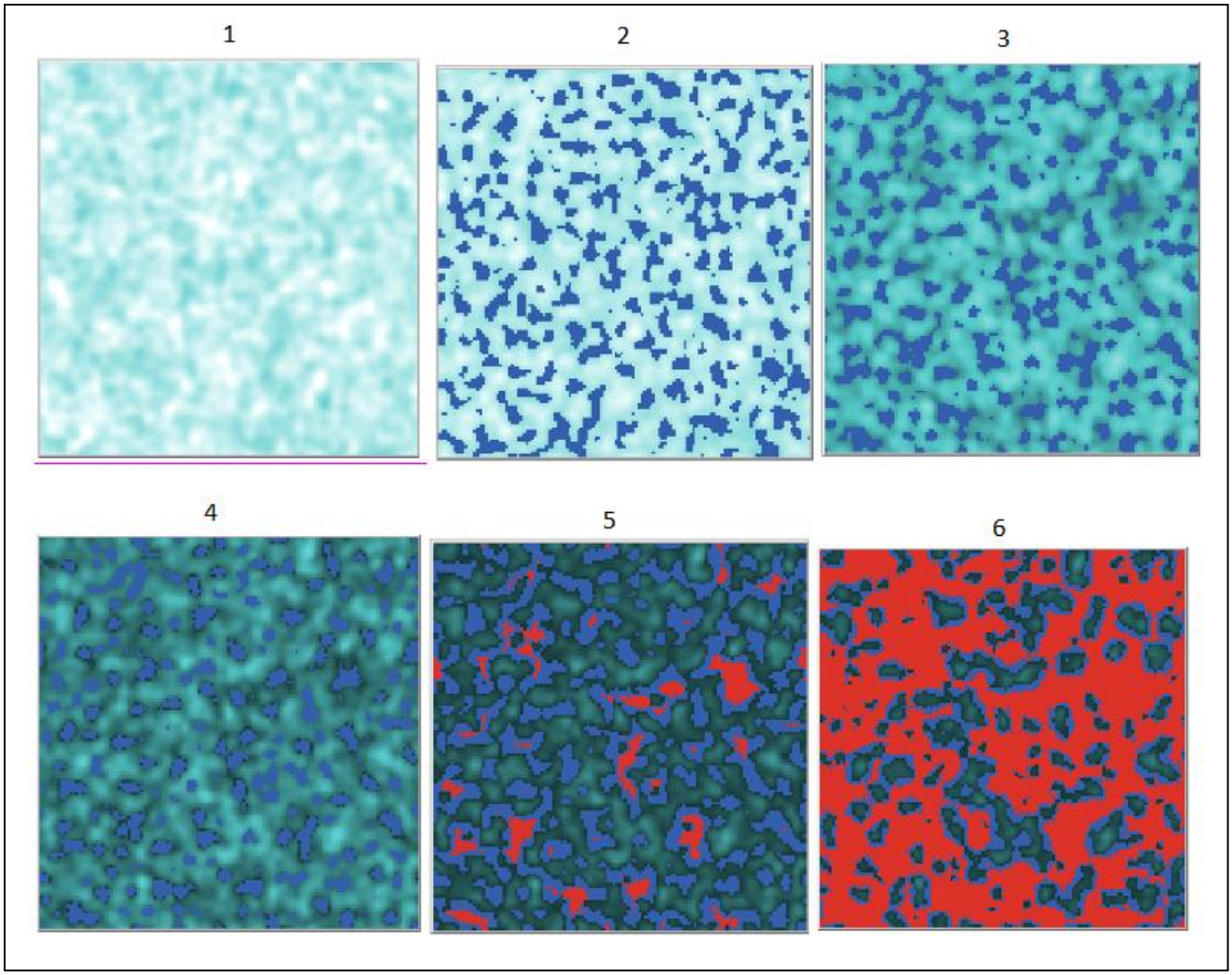

FIGURE 9: SIMULATION DU MODELE DE FONTE DES GLACES

Cette illustration décrit l'évolution de la simulation au cours du temps, découpée ici en 6 étapes essentielles :

(1) La première image montre la création de la topographie du système étudié. Les dégradés de couleur représentent la différence de hauteur de la glace. Lorsque la cellule est blanche, le relief est élevé alors que si celle-ci tend vers le cyan, alors la hauteur de la glace est basse.

(2) La glace commence à fondre et laisse apparaître de l'eau, qui forme alors des mares de fonte, matérialisées par la couleur bleu.

(3) La fonte continue mais les mares ne s'étendent pas car un équilibre se crée entre la transformation en eau de la glace et la percolation qui provoque l'écoulement d'une partie du liquide dans l'océan à travers la surface du glacier.

(4) À mesure que la glace fond, le contour des mares s'assombrit. Ce changement de couleur symbolise l'albédo de l'eau (c'est-à-dire la capacité de l'eau à absorber la lumière) qui est très bas, ce qui provoque une baisse de la radiation lumineuse reflétée par la glace.

(5) Certaines mares de glace commencent à être percées, l'eau douce présente dans celles-ci déferle alors dans l'océan. Ce phénomène est représenté par l'apparition de tâches rouges.

(6) Les zones rouges s'étendent exponentiellement, ce qui accélère considérablement la fonte de la glace restante. La simulation prend fin lorsque l'intégralité de l'interface est devenue rouge.

2.3 Présentation du code

Dans cette partie, nous expliquons comment nous sommes parvenus à passer d'un problème physique à une simulation numérique. Pour atteindre nos objectifs, nous avons réalisé le codage d'un programme, et nous allons détailler celui-ci afin de donner un aperçu de la syntaxe utilisée par le logiciel Netlogo.
Notre code se compose principalement de commandes qui définissent les actions à exécuter par les « agents ». Celles-ci sont de deux types :

- Primitives : commandes pré-définis dans Netlogo ;
- Procédures : commandes définis par le programmeur

Avant tout il faut introduire certaines variables que l'on souhaite utiliser dans notre programme:

Paramètres variables:

Sur l'écran d'interface, l'utilisateur a également la possibilité de faire varier certains paramètres afin de diversifier les simulations. Voici le rôle de chacun:

- mean-ice-height : désigne la hauteur moyenne de la glace, représentée dans notre problème par la variable des patches appelée « ice » ;
- smooth-cycles : permet de rendre le relief de la glace plus ou moins lisse ;

- time-step : désigne le nombre de jours qui se déroule lors de chaque boucle de la simulation.

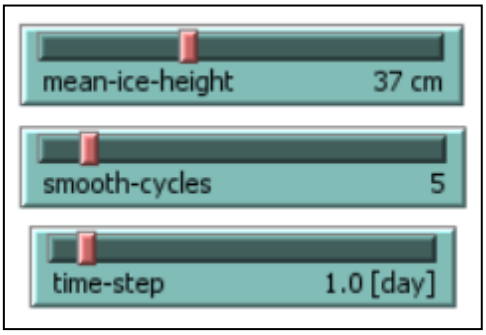

FIGURE 10: PARAMETRES DEFINIS PAR L'UTILISATEUR

```
breed [raindrops raindrop]
```

On utilise le mot-clé « breed » afin de définir un type d'automates mobiles. Le logiciel nécessite d'écrire le nom de l'ensemble d'agents en première entrée puis le nom d'un seul de ces agents que nous appelons ici « raindrop » pour décrire les gouttes d'eau qui déferlent.

```
globals[
  density-ratio
  melt-rate-ice
  coefficient
]
```

Les « globals », tout comme les « breeds » ne peuvent être utilisées qu'au début d'un programme, avant l'entrée des différentes fonctions. Elles définissent de nouvelles variables globales. Ces variables sont dites «globales » car elles sont accessibles à tous les agents et peuvent être utilisées n'importe où dans un modèle. Le plus souvent, les « globals » sont utilisées pour définir des variables ou des constantes qui doivent être utilisées dans de nombreuses parties du programme et il faudra leur attribuer une valeur à un moment donné.

```
patches-own[ice water]

raindrops-own[water-drop]
```

On définit également les variables internes aux automates cellulaires. Les patches (immobiles) ont pour variables interne « ice » et « water », alors que les turtles « raindrops" » ont un paramètre interne appelé « water-drop ».

Une fois que nous avons définis les variables les plus essentielles à prendre en compte pour la simulation de notre problème, nous pouvons nous intéresser au programme principal contenant les différentes fonctions.

```
to draw-map
  ca
   ask patches
  [
  set ice random-float (2 * mean-ice-height)
  set water 0

  set pcolor scale-color cyan (ice + 10) 0 (1.5 * mean-ice-height)
  ]
  smooth-ice
  reset-ticks

  set density-ratio 0.9
  set-default-shape turtles "circle"


end
```

Fonction : Draw-map

Cette fonction a pour objectif de dessiner la topographie de notre morceau de glace. Pour se rapprocher au mieux de la réalité, il est important que le relief soit simulé aléatoirement. On demande ici à chaque patch de s'attribuer aléatoirement une valeur correspondant à la hauteur moyenne de glace. Ensuite la variable « water » est initialisé à 0 sur l'ensemble des automates de la map.
On associe alors une couleur qui varie de cyan à blanc à chaque cellule en fonction de l'épaisseur de glace, c'est-à-dire de la variable « ice » : si la glace est épaisse elle sera de couleur cyan et si l'épaisseur de la glace est fine elle aura une couleur qui se rapproche du blanc.
Finalement on applique la fonction smooth-ice.

```
to smooth-ice
  repeat smooth-cycles [
    ask patches [
      let sum-ice-neighbors sum [ice] of neighbors4
      let mean-ice (sum-ice-neighbors + ice) / 5
      set ice mean-ice
      set pcolor scale-color cyan (ice + 10) 0 (1.5 * mean-ice-height)
    ]
  ]
end
```

Fonction : smooth-ice

Cette fonction sert à régulariser la topographie de notre morceau de glace en réduisant la différence de hauteur de glace entre chaque cellule. En minimisant les écarts on se rapproche ainsi de la réalité.
On introduit alors deux nouvelles variables internes à chaque patch : « sum-ice-neighbors » et « mean-ice ». On réitère la boucle autant de fois que l'utilisateur le souhaite en faisant varier la valeur de « smooth-ice » à l'aide du curseur présent sur l'interface. Pour chaque patch, on donne à la variable "sum-ice-neighbors" la somme des hauteurs de glaces (variable "ice") des quatre patches voisins. Ensuite la variable "mean-ice" prend pour valeur la hauteur moyenne des quatre cellules adjacentes et de celle observée.

Le patch étudié remplace alors sa variable "ice" par "mean-ice" définit précédemment.

Une couleur est finalement associée à chaque agent immobile selon la nouvelle hauteur de glace qui lui est associée. Plus la hauteur de la glace est élevée, plus la couleur du patch sera blanche.

```
to-report compute-mean
  report mean [ice] of patches
end

to-report compute-std
  let avg compute-mean
  let var (mean [ice * ice] of patches)
  report sqrt (var - ( avg * avg ))
end

to-report surface
  let ratio ( count patches with [water > 0 or pcolor = red ] )
  report ratio
end
```

Fonction : to-report

Ces fonctions enregistrent des informations auxquelles on souhaite avoir accès. Celles-ci sont disponibles sur l'interface principale et peuvent varier au cours de la simulation. Elles sont représentées sous forme numérique ou graphique.
"Compute-mean" enregistre la valeur moyenne de la variable "ice" de chaque patch.
"Compute-std" enregistre l'écart type des valeurs moyennes de la variable "ice de chaque patch.

"Surface" enregistre le nombre de cellules sur lesquelles il y a de l'eau. Ainsi nous pouvons obtenir un taux de la surface où la glace est couverte par de l'eau (ou qu'il n'y ait plus de glace du tout).

```
to melt
  ask patches [
    set melt-rate-ice 1.2
    ifelse (water > 0) [
    set coefficient water / 100 ]
    [set coefficient 0]
    if ice > 0 [
      set ice ice - (melt-rate-ice * time-step ) - (coefficient * time-step)
      set water water + (density-ratio * melt-rate-ice * time-step ) + (coefficient * time-step)
    ]
    set pcolor scale-color cyan (ice + 10) 0 (1.5 * mean-ice-height)

    if ice <= 0 [
      set ice 0
      set water 0
      set pcolor red
    ]

  ]

end
```

Fonction : melt

Cette fonction est utilisée pour faire fondre la glace à la surface du glacier, ce qui laisse apparaître de l'eau. Pour y parvenir, nous réalisons différents tests sur chaque patch:
Tout d'abord, nous demandons à chacun d'entre eux si la donnée « water » est supérieure à 0. Si c'est le cas, on affecte à une variable globale (appelée « coefficient ») la valeur de water divisée par 100. Sinon, le coefficient devient nul.
Ensuite, nous demandons à chaque cellule si la hauteur de glace qui lui correspond est supérieure à 0. Si c'est le cas, on donne aux variables internes « water » et « ice » de nouvelles valeurs qui dépendent de différents facteurs : time-step, coefficient et deux paramètres dont on défini les valeurs:

- Melt-rate-ice : C'est le taux de fonte de la glace.
- density-ratio : Cette donnée représente le rapport entre les masses volumiques de l'eau et de la glace.

Une fois les deux premières boucles réalisées, les cellules se colorent à nouveau en fonction de leur dernière valeur de "ice" affectée.
La dernière boucle demande à chaque patch de se colorer en rouge si la valeur "ice" qui lui correspond est inférieure ou égale à 0. Ainsi, il réinitialise les variables "water" et "ice" à 0.

```
to go
  ask patches [
    sprout-raindrops 1 [
    set water-drop water
    set size 2
    set color blue
    ]
    set water 0
  ]
  ask raindrops [ flow ]
end
```

Fonction : go

Cette fonction fait apparaître des automates cellulaires mobiles (les turtles). Sur chaque patch, on utilise l'entité appelée « raindrops » ayant une variable interne « water-drop », à laquelle on donne la valeur "water" du patch. On réinitialise alors cette valeur à 0.
On demande finalement aux turtles de se déplacer selon la fonction « flow ».

```
to flow
  loop[
  let target min-one-of neighbors [ ice + water ]
  ifelse ([ice + water ] of target )  < (ice + water )[
      move-to target
    ][
   set water water + water-drop
   set water-drop 0
   die
   ]
  ]
end
```

Fonction : flow

Cette fonction régie le déplacement des « raindrops ». Le but est alors que les gouttes d'eau déferlent vers les cellules où la hauteur d'eau ajoutée à celle de glace est minimum. Il y a ainsi la création des mares de fontes.
Tout d'abord, on identifie parmi les voisins de celui où se trouve la turtle, la cellule dont la somme "ice + water" est la plus faible. On appelle cette cellule « target ».
Ensuite, si la somme « ice + water » de « target » est inférieure à celle où se trouve « raindrop », alors ce dernier se déplace vers le patch « target ».

Si ce n'est pas le cas, le patch où se trouve « raindrop » ajoute à sa variable "water" le paramètre "water-drop" de l'automate mobile. Ce dernier est finalement détruit. Dans le langage Netlogo on dit qu'il meurt grâce à la fonction « die ».

```
to seepage
  ask patches [
    if water > 0 [
  set water water - 0.8 * time-step ]

  if water < 0 [
    set water 0 ]
]
end
```

Fonction : seepage

Cette fonction représente la percolation de l'eau présente à la surface de la glace. On demande à chaque patch si la valeur de « water » qui lui correspond est supérieure ou inférieure à 0. Dans le premier cas, la valeur "water" est réduite proportionnellement à time-step et à un coefficient que nous avons défini. Dans le second cas, la variable "water" est nulle.

```
to setcolor
  ask patches [
    if water > 0  [
      set pcolor scale-color blue water (water * 2) 0]
  ]
end
```

Fonction : setcolor

Cette fonction colore en bleu toutes les cellules sur lesquelles il y a la présence d'eau. On peut ainsi identifier les mares de fonte sur l'écran d'interface.

```
to simulation
melt
seepage
go
setcolor
tick

  let val sum [ice] of patches
  if val = 0 [stop]
end
```

Fonction : simulation

Cette fonction ordonne toutes les actions réalisées précédemment. Il est possible de déclencher cette simulation en utilisant le bouton « simulation » présent sur l'interface. Tout d'abord se déclenche la fonte de la glace et ainsi l'apparition d'eau. Ensuite s'effectue la percolation de celle-ci, suivie de la transformation de l'eau en automates mobiles qui vont alors déferler vers les zones où l'altitude est minimale. Enfin on affecte la couleur bleu aux mares de fonte, là où il y a la présence d'eau. On répète alors le processus jusqu'à ce que la valeur totale des variables « ice » de chaque patch soit égale à 0 (disparition totale de la glace). Ainsi la totalité de la map sera colorée en rouge ce qui signifie la fin de notre simulation.

2.4 Résultats graphiques et interprétation

Les objectifs de nos simulations sont divers : nous voulons tout d'abord comparer le temps de fusion de l'intégralité de la glace étudiée dans deux situations différentes. Dans le cas 1, nous lissons au maximum la surface du glacier afin d'avoir un relief régulier. Dans le cas 2, nous simulons une topographie composée d'une hauteur de glace très variable.
Nous souhaitons également observer l'évolution de la hauteur moyenne de l'eau et de la glace.
Enfin nous voulons interpréter la progression de la surface que représentent les mares de fontes par rapport à la totalité de l'espace étudié.

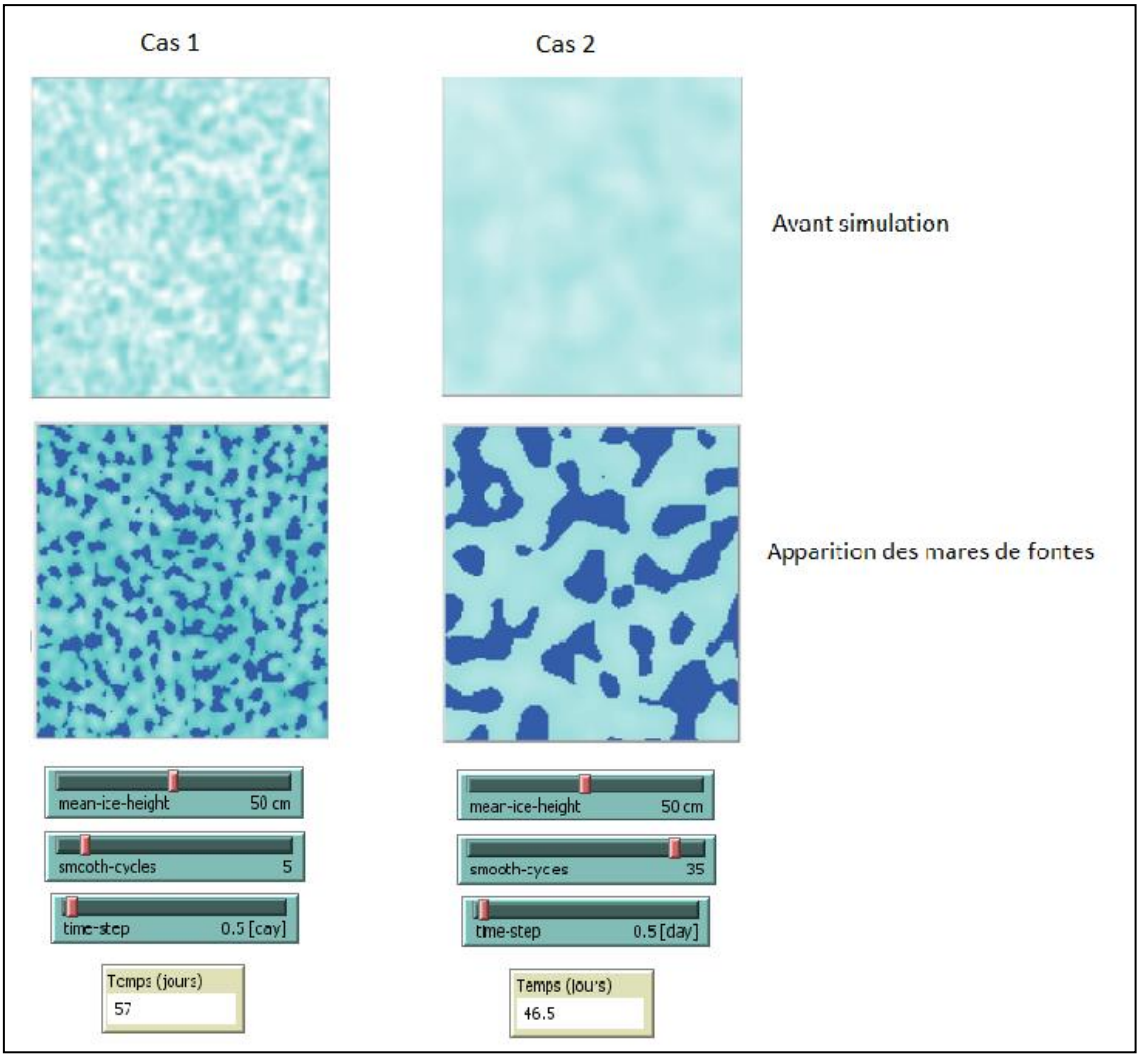

FIGURE 11: COMPARAISON DU TEMPS DE FUSION DE L'INTEGRALITE DE LA GLACE DANS DEUX CAS DISTINCTS

On remarque que lorsque le relief est très irrégulier, le temps de fonte totale du glacier est largement supérieur à celui dans le cas d'une topographie lisse.
Lors de la simulation, on observe que dans le cas 1, de nombreuses mares de fonte de petite taille se forment alors que dans le cas 2, peu de mares se créent mais celles-ci sont très étendues. On suppose alors que la vitesse de fonte de la glace est accélérée par la présence de l'eau.

Nous avons dans un second temps réalisé un graphique et un histogramme correspondant respectivement à la moyenne de la profondeur de l'eau et à l'évolution de la surface recouverte d'eau au cours de la simulation.

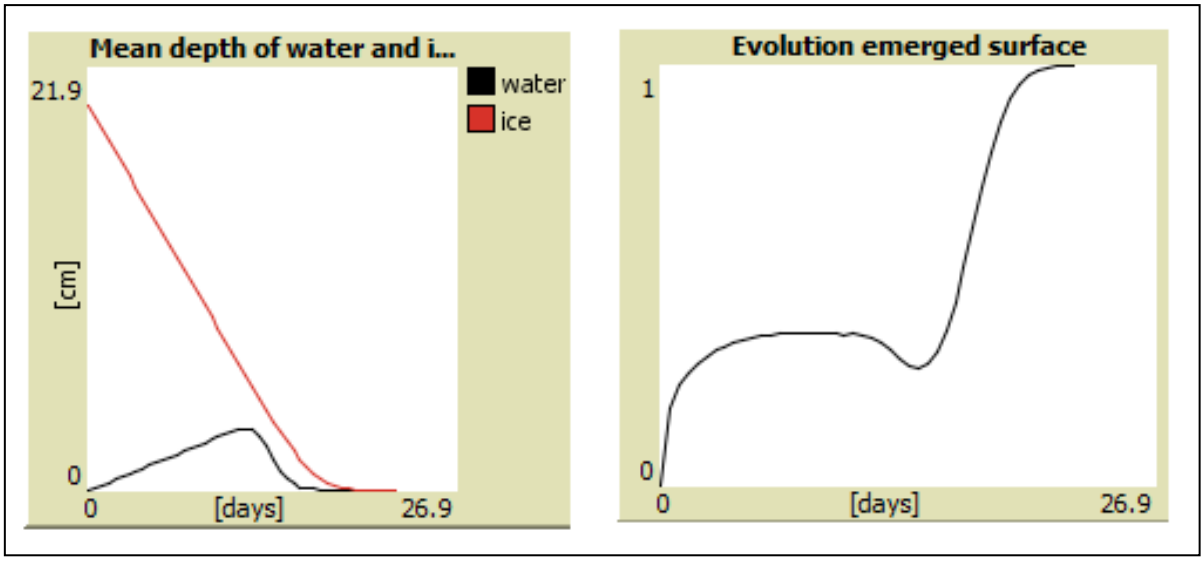

FIGURE 12: GRAPHIQUES PRESENTANT LE COMPORTEMENT DE L'EAU ET DE LA GLACE

Le premier graphique est composé de deux courbes. L'une (noire) décrit l'évolution de la profondeur moyenne de l'eau présente à la surface alors que la deuxième (rouge) dessine l'évolution de la hauteur de la glace. Cette dernière décroît linéairement durant toute la simulation, ce qui est cohérent puisqu'il n'y a pas d'apparition de glace et que celle-ci fond. On précisera également que la hauteur de la glace baisse, mais que l'écart-type reste le même durant toute la modélisation.

Pour l'évolution de l'eau, On observe que dans un premier temps la variable étudiée croît de manière linéaire, avant de chuter jusqu'à devenir nulle. On observe que dans un premier temps la variable étudiée croît de manière linéaire, avant de chuter jusqu'à devenir nulle. La simulation graphique nous indique que le point à partir duquel la profondeur moyenne de l'eau atteint une valeur maximale avant de baisser significativement correspond au moment où certaines mares de fonte sont "percées", ce qui va causer le déferlement immédiat de l'eau dans l'océan. Ce résultat paraît cohérent car les cellules où l'eau est la plus profonde sont celles au centre des mares de fonte. Ainsi lorsque celles-ci se percent et laisser couler l'eau douce dans l'océan, la variable « water » est alors remise à 0, il est donc normal que la hauteur moyenne chute en parallèle de ce phénomène.

Le deuxième graphique montre l'évolution de la surface recouverte d'eau. On observe d'abord une croissance rapide qui correspond à l'eau qui apparaît rapidement au début de la fonte de la glace. Il y a ensuite une stagnation lorsque l'eau représente environ un tiers de la surface totale du profil étudiée. Celle-ci s'explique par le fait que le niveau de l'eau est censé augmenter, mais des phénomènes comme la percolation réduisent sa progression. On a donc ici un équilibre entre les différents paramètres pris en compte. Cet équilibre se

termine sur une légère baisse suivie d'une forte croissance qui prendra fin en même temps que la modélisation. Cette forte augmentation démarre lorsque l'eau des mares de fonte commence à déferler dans l'océan.

Ce dernier point est très important : il nous permet d'affirmer qu'une fois que la glace qui compose le fond des mares de fonte disparaît, alors la fonte de la glace aux alentours est largement accélérée.

Optimisation

3.1 Mesure de l'erreur

La validation de notre modèle est une étape complexe car il faut s'assurer que le modèle simulé est bien représentatif de la réalité. Pour cela il faut pouvoir comparer les valeurs obtenues avec les valeurs expérimentales du monde réel. Ce qui nécessite de plus une expertise humaine, car un expert peut, en effet, comparer les résultats des expérimentations de simulation avec sa connaissance du terrain.

Dans notre programme, il est nécessaire que l'utilisateur attribue des valeurs à certains paramètres physiques (taux de fusion de la glace, densité de l'eau, …). Il est toutefois difficile de définir certains d'eux avec précision et ceci peut expliquer que nos résultats ne modélisent pas la stricte réalité.

3.2 Amélioration possible

Le modèle que nous avons réalisé ne prend en compte que certains paramètres. Pour se rapprocher davantage de la réalité, nous pourrions considérer de nombreuses autres données qui peuvent avoir un impact plus ou moins élevé sur l'évolution de notre système.
Parmi les paramètres que nous n'avons pas pris en compte, on retrouve le vent, les éventuelles chutes de neige, la radiation solaire, etc.
De plus on a considéré que la fusion s'effectue de manière latérale alors que dans la réalité la fusion s'effectue aussi de manière horizontale car il y a des effets de convection thermique.

3.3 Limites du problème

Pour modéliser notre problème, nous avons choisi de ne prendre en compte qu'un échantillon de ce que nous voulons étudier. Ainsi, même si nous prenons en compte un maximum d'informations, il est probable que la complexité et l'ampleur du système nous impose de garder une marge d'erreur.

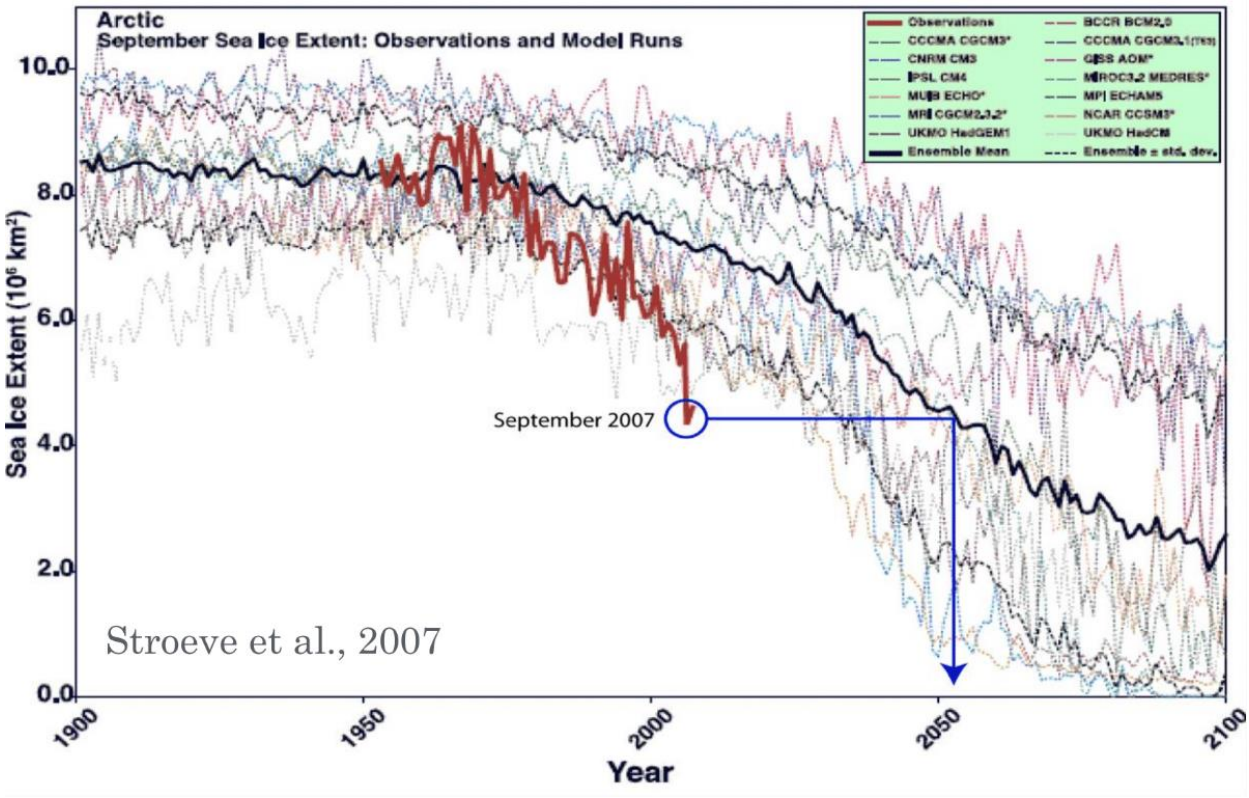

FIGURE 13: GRAPHIQUE COMPARANT LES DIFFERENTES PREVISIONS

On observe sur ce graphique que de nombreuses recherches ont été réalisées durant le siècle dernier afin d'anticiper l'évolution de la fonte des glaces. On remarque que la différence entre les prévisions des scientifiques et la réalité est très large. La surface totale de glace fondue en 2007 n'était attendue que pour 2050. Cela démontre comme il peut-être complexe de simuler avec précision un tel phénomène physique. Mais en associant les différentes précisions, cela nous permet de réaliser une moyenne de ce qui pourrait arriver dans quelques années.

3.4 Domaines d'application des automates cellulaires

Les automates cellulaires sont des outils très puissants. Ils permettent de représenter un problème lorsqu'il est difficile d'y répondre par des équations physiques. On laisse alors ces automates agir selon des règles que l'on crée afin de modéliser de la manière la plus réaliste possible une situation donnée.

Les automates cellulaires sont utilisables dans de nombreux domaines, c'est ce qui en fait un outil très puissant. En effet, nous pouvons les employer afin de résoudre des problèmes complexes comme la compréhension de phénomènes tels que les tremblements de terre, mais il est également possible de les utiliser dans des secteurs comme la physique fondamentale, la biologie, la robotique, la dynamique des fluides, la sociologie, etc. (Propagation de virus, étude comportementale d'une population, …).

Conclusion

Dans ce projet nous avons tout d'abord tenté de nous familiariser avec le logiciel Netlogo à travers l'étude d'un programme qui simule la propagation d'un feu de forêt. Netlogo est un logiciel de modélisation qui nous permet d'ajouter autant de variables et de facteurs qu'on le souhaite afin de rendre notre simulation la plus réaliste possible.

Par la suite on a tenté de créer notre propre modèle, en choisissant une problématique qui préoccupe et alarme de plus en plus les scientifiques: la fonte des glaces en Arctique. Pour cela, nous avons écrit un code qui décrit la fusion de la glace et le déferlement de l'eau douce dans l'océan. Les graphiques obtenus nous ont permis de mesurer l'ampleur du phénomène, difficilement observable à notre échelle.

La différence entre la réalité et le modèle réalisé repose sur le fait que nous avons négligés de nombreuses variables à prendre en compte lors de la fonte des glaces, comme le rayonnement solaire, les chutes de neige ou encore la convection thermique.

Nous pourrions donc intégrer tous ces paramètres à notre simulation. Cependant, la complexité de la réalité fait qu'il est impossible de la simuler de façon complètement fiable. Voila pourquoi il faut se baser sur différents modèles ainsi que sur des calculs théoriques et des expérimentations pour prédire au mieux ce qu'il se passera dans l'avenir.
Une étude complète est souvent impossible à réaliser dans un temps acceptable compte tenu du nombre de paramètres à explorer et de la nature de ces derniers.

La modélisation à base d'agents est une pratique souple, intuitive et proche aussi bien des données que des théories ce qui en fait l'une des méthodes les plus appréciées dans la plupart des communautés scientifiques. Etudier un modèle par simulation nécessite un nombre important d'exécutions, d'hypothèses et de données de terrain afin de produire les résultats escomptés sous différentes formes (tableaux, graphiques, base de données). Mais es enjeux et les ambitions des modélisateurs sont en perpétuelle évolution si bien que la modélisation numérique représente un gain considérable de temps, d'argent et de ressources matérielles.

Bibliographie

Ouvrages:

Netlogo, «A modeling tool» de Juan Carlos Garcia Vasquez et Fernndo Sancho Caparrini

Luthje et Al: Modeling sea ice melt ponds, 2006

Journal of geophysical research, Scott Feltham, 2010

Cours de Enrico Cazalvarini

Modélisation et simulation économiques - Murat Yıldızoğlu

A continuum model for the coupled evolution of sea ice thickness and melt pond distribution - Andrea Scagliarini Institute for Applied Mathematics “Mauro Picone” (IAC), Rome, Italy - Italian National Research Council (CNR)

Transition in the fractal geometry of Arctic melt ponds - The Cryosphere

Images :

http://www.liberation.fr/planete/2017/02/14/geler-artificiellement-l-arctique-pour-pallier-la-fonte-des-glaces_1548208

https://www.curioctopus.fr/read/6523/les-surprenants-motifs-geometriques-de-la-nature-vont-vous-hypnotiser

http://www.geog.leeds.ac.uk/courses/level3/geog3150/practicals/practical1/3_ask.php

Table des illustrations