# Avalanches en milieu alpin avec Netlogo

Mathieu Petit, Thomas Le Bras

25 mars 2008

**Résumé**

Ce TD fait suite aux premières heures d'initiation à la simulation par agent en Netlogo. La dernière fois, vous avez pu découvrir le logiciel, son interface, et, dans le troisième tutorial, le langage de programmation qui lui est associé. Aujourd'hui, nous simulerons un environnement géographique de type alpin dans lequel nous déclencherons des avalanches pour déterminer quelles sont les zones à risques et quelles est le meilleur emplacement pour un refuge. En fin de TD, des alpinistes tenterons de rejoindre le refuge en évitant d'être emporté par les avalanches.

Le TD se déroule en 3 temps :

- dessin du terrain, de l'isométrique 0° et des chutes de neige,
- déclenchement d'avalanches et analyse des zones de risque,
- ajout d'agents "alpinistes" sur le terrain et déplacements.

## Table des matières

# 1 Contenu de l'archive .zip

Les données de démarrage du TD sont fournies dans le fichier `avalanche.zip` que vous trouverez sur l'Intranet de cours. Le fichier `avalanche_debut.nlogo` vous servira de base d'interface et de code. Les fichiers `*.pgm` sont des modèles numériques de terrains carrées de 257 unités où les zones élevés tirent vers le blanc. Les fichiers `*.data` sont la correspondance lisible par Netlogo des modèles de terrains. `pgm2data` et `pgm2plot` sont des scripts Unix de conversion depuis PGM vers les formats Netlogo et Gnuplot, respectivement. `color_scale.data` contient la palette de 256 couleurs sous forme de triplets RGB qui à chaque niveau de gris du modèle numérique associe la couleur de représentation sur la carte.

1. désarchivez `avalanche.zip` dans votre répertoire personnel,
2. démarrez Netlogo et ouvrez le fichier `avalanche_debut.nlogo` (Fig. 1).

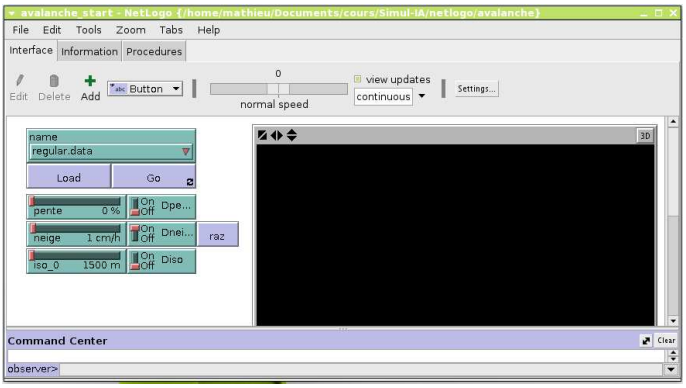



FIG. 1 – l'interface de `avalanche_debut.nlogo`

Un terrain s'affiche sur une grille de $257 * 257$ patches. Nous considérons qu'une unité de grille vaut 10 mètres dans la réalité. Une variation d'un niveau de gris dans le modèle numérique représentera une dénivellation de $10/3$ mètres. Entre deux patches de hauteurs différentes, on mesure au minimum une variation de $3,33m/10m$ soit une pente de 30%. Pour résumer :

| | terrain réel | modèle numérique |
|---|---|---|
| taille sur X | 2,57 km | 257 patches |
| taille sur Y | 2,57 km | 257 patches |
| dénivelé couvert | 853m | 256 unités |

La partie haute de l'interface permet de choisir le terrain, d'initialiser la carte et de lancer la simulation. La force des chutes de neige, la hauteur de la courbe isométrique 0° et la pente minimale de déclenchement d'une avalanche sont réglables par les sliders. Les boutons On/Off permettent de visualiser ces paramètres sur le terrain.

# 2 Initialisation du terrain et chutes de neige

Un appui sur le bouton "Load" lance la procédure `to load` chargée d'initialiser le terrain. Pour le moment rien de visible ne se produit.

1. dans `to load`, initialiser la hauteur réelle du terrain `height` à $1500+10/3*$ `elevation`[1],
2. vérifiez en inspectant quelques patches que les hauteurs sont bien initialisées.

## 2.1 Pente du terrain

Nous commencerons par initialiser la valeur de la pente $\Delta$ sur un patch $p$. Elle correspond à l'écart type des hauteurs $h_1$ à $h_8$ des 8 voisins de $p$ :

$$\Delta_p = \sqrt{\sum_{i=1}^{8} \left( h_i - \frac{\sum_{j=1}^{8} h_j}{8} \right)^2} \tag{1}$$

Nous avons fixé précédemment qu'une d'unité de longueur de 10m vaut 1px : il faut diviser $\Delta_p$ par 10 pour retrouver la pente réelle selon l'échelle du terrain.

1. complétez la procédure `to load` en utilisant la fonction `variance` pour initialiser les valeurs de pente `slope` des patches[2],
2. vérifiez que les pentes des patches sont bien initialisées.

## 2.2 Dessin de la carte

Après chargement de la carte, des élévations et calcul de la pente, `to load` lance la procédure `color_patch` sur chaque patch. La liste `colors` contient 256 valeurs de type RGB indicées de 0 à 255. On désigne un élément d'une liste par `item <indice> <liste>`

1. dans `color_patch`, initialisez la couleur du patch courant à `colors`[`elevation`][3],
2. vérifiez que le coloriage de la carte fonctionne correctement en chargeant un terrain.

## 2.3 Affichage de la courbe isométrique 0°

La courbe isométrique 0° désigne la limite pluie/neige : au dessus, il neige, et en dessous, il pleut. Elle est paramétrable sur le dénivelé maximal représentable, soit 853m. La variable `iso_0` contient l'altitude de l'iso 0°.

1. dans `color_patch`, écrivez la condition portant sur la valeur de la variable `Diso` qui décide de l'affichage de la courbe,
2. dans cette condition, coloriez le patch courant en rouge si l'élévation du terrain est comprise entre les altitudes de l'iso $-5$ et iso $+5$,
3. avec le terrain "`regular.data`" chargé et démarré, jouez sur la hauteur de l'iso pour vérifier qu'elle varie bien sur tout le dénivelé du terrain. Corrigez si nécessaire votre condition d'affichage[4].

[1] `set height (10 / 3 * elevation + 1500)`

[2] Dans l'environnement patches : `set slope (sqrt (variance [height] of neighbors)) / 10`

[3] `set pcolor item elevation colors`

[4] `if diso = true [ if height > iso_0 - 5 and height < iso_0 + 5 [ blend-color red .7 ]]`

## 2.4 Affichage des pentes dangereuse

On considère en général qu'une avalanche peut se déclencher sur des pentes supérieures à 60%, soit 30°. toutefois, cette pente minimale dépend beaucoup des conditions d'enneigement, de la météo et de l'orientation des versants. Pour simplifier le problème, nous laissons l'utilisateur apprécier la valeur de la pente à risque dans l'intervalle [0%,150%] (soit [0°,56°]). La variable initialisée par le slider est `pente`.

1. dans `color_patch`, codez l'affichage des pentes dangereuses en transparence par un appel a `blend_color` (`blend_color violet .5`) à la condition que `Dpente` soit vrai et que `pente` soit inférieure à la pente du patch courant[5],
2. toujours sur le terrain "`regular.data`", vérifiez que l'ensemble du terrain devient dangereux pour une pente de déclenchement inférieure à 30%.

## 2.5 Affichage de la neige

La neige s'affiche en transparence de blanc sur chaque patch. On affiche de la neige si d'une part `Dneige` est vrai et si, d'autre part, $0,1 <$`snow_level`. Le patch devient opaque blanc quand `snow_level`$> 8$.

1. dans `color_patch` codez la condition d'affichage en transparence de la neige un appel à `blend_color` dans un bloc `if`[6].

Si vous lancez la simulation, le terrain reste brun ... normal car si l'on sait maintenant afficher de la neige, il n'en tombe pas pour autant.

## 2.6 Chutes de neige et fonte

A chaque tour de simulation, les chutes de neiges se produisent sur `neige` * 200 patches choisis aléatoirement parmis l'ensemble des patches, à la condition que le terrain soit au dessus de l'isométrique 0°. La quantité de neige ajoutée $\Delta_{neige}$ est :

$$\Delta_{neige} = \sqrt{\frac{h_{terrain} - h_{iso\_0°}}{\text{denivele\_max}}}$$
$$\text{avec : } h_{iso\_0°}\text{, l'altitude de l'isométrique } 0°, \qquad (2)$$
$$h_{terrain}\text{, l'altitude du terrain considéré,}$$
$$\text{denivele\_max} = 850\text{, le dénivelé maximal représentable.}$$

1. dans `to snow` codez la sélection de `neige` * 200 patches[7],
2. ajouter de la neige à ces patches s'ils sont au dessus de la l'iso 0°[8],
3. testez les chutes de neiges sur le terrain "`regular.data`". Veillez à ce que la neige soit cantonnée au dessus de la ligne iso,
4. Jouez sur l'intensité de la chute de neige et vérifiez que le terrain se blanchit plus vite s'il neige beaucoup.

[5] `if dpente = true and slope > 0.01 * pente [...]`
[6] `if dneige = true and snow_level > .1 [blend_color white (snow_level / 8)]`
[7] `ask n-of (neige * 200) patches[...]`
[8] `if height > iso_0[set snow_level snow_level + sqrt((height - iso_0) / 850)]`

Tout les terrains en dessous de l'iso 0° sont concernés par la fonte de neige. On applique le principe inverse, c'est à dire que plus le terrain est bas par rapport à la ligne iso, plus la neige fond vite. On détermine comme réaliste $\Delta_{fonte}$ suivant :

$$\begin{aligned} &\Delta_{fonte} = -\sqrt{\frac{h_{iso\_0^\circ} - h_{terrain}}{\text{denivele\_max} * 10}} \\ &\text{avec : } h_{iso\_0^\circ}\text{, l'altitude de l'isométrique } 0^\circ, \\ &\qquad h_{terrain}\text{, l'altitude du terrain considéré,} \\ &\qquad \text{denivele\_max} = 850\text{, le denivele maximal représentable.} \end{aligned} \tag{3}$$

1. dans `to snow`, demandez à tout les patches situés sous l'iso qui ont `snow_level` $>$ 0 d'ajouter $\Delta_{fonte}$ à leur niveau de neige[9].
2. testez les chutes de neiges sur le terrain "`regular.data`". Vérifiez que la neige en dessous de l'iso fond correctement.

# 3 Déclenchement d'avalanches naturelles

Une pente en avalanche peut être modélise par l'appel récursif depuis un patch vers ses voisins moins élevés en distribuant une quantité de neige du haut vers le bas. Deux fonctions sont nécessaires : le déclencheur `to trigger_avalanche` qui retourne un patch ou se déclenche une avalanche, et la fonction de récursion `to avalanche` qui modelise l'effondrement de la pente.

## 3.1 Déclencheur d'avalanche

La procédure `to go` appelle le déclenchement d'une avalanche sur un patch pris au hasard à chaque tour de simulation. L'avalanche se déclenche si 2 conditions sont réunies sur un patch : le niveau de neige doit être supérieur à 8 et la pente du patch doit être supérieure à la pente de déclenchement minimale.

1. Dans `to trigger_avalanche`, codez la condition de déclenchement[10],
2. dans ce cas, colorez les voisins en bleu, colorez le patch courant en rouge et attendez une seconde[11],
3. vérifiez que les déclenchements fonctionnent en exécutant la simulation.

## 3.2 Modèle de l'avalanche

la procédure récursive `to avalanche` s'appelle avec comme paramètre la liste des patches qui sont en cours d'avalanche. 80% de la neige du patch courant se réparti sur les patches inférieurs, en proportion de la différence de hauteur entre patch courant et inférieur :

$$neige_{inf} -= 0.8 * neige_{courant} * \frac{h_{courant} - h_{inf}}{\sum_{p_{inf}} \left(h_{p_{inf}} - h_{courant}\right)} \tag{4}$$

[9] `[(850 * 10) / ( iso_0 - height) sqrt -`
`if height < iso_0 and snow_level > 0 [ set snow_level snow_level`

[10] `[...] 8 < level_snow and 100 / pente < slope if`

[11] `ask neighbors [set pcolor blue], set pcolor red, wait 1`

$neige_{inf}$ est le niveau de neige d'un patch inférieur, $neige_{courant}$ est le niveau de neige du patch courant, $\sum_{p_{inf}}(...)$ est la somme des différences d'altitude entre les patches inférieurs et le patch courant.

1. dans `to avalanche`, initialisez la liste `inf` avec les patches inférieurs au patch courant[12],
2. initialisez `hlevel` avec $\sum_{p_{inf}}(...)$[13]
3. mettez à jour le niveau de neige des patches inférieurs[14].

La condition de récurence porte, pour un patch inférieur, sur une quantité de neige par rapport à sa pente. si $pente_{inf} * neige_{inf} > 4$, alors, l'avalanche se poursuit sur ce patch, ce qui signifie que l'on l'ajoute à la liste `tmp`

1. écrivez dans `to avalanche` la condition de récurence. Vérifiez que le patch a ajouter ne se trouve pas déjà dans `tmp`[15].
2. ajoutez l'appel à `to avalanche` dans to `trigger_avalanche`[16],
3. vérifiez que les avalanches se déclenchent et parcourent les pentes descendantes.

# 4 Alpinistes et refuge

Dans la suite, vous utiliserez la carte "mountain.data" pour vos simulation.

## 4.1 Dangerosité de la carte

Nous voulons placer un refuge dans un endroit sûr (entendre sans avalanches), et qui soit au plus près des sommets de la carte. Pour cela, il est nécessaire de simuler un nombre important d'avalanches sur le terrain, de mesurer la dangerosité par patch et de prévoir un affichage graphique de cette mesure sur la carte. Un terrain est dangereux s'il a été pris dans une avalanche ou qu'il est dans une pente de déclenchement.

1. Ajoutez une variable interne aux patches qui mesurera le danger de ce patch[17],
2. dans `to avalanche`, à chaque fois qu'une avalanche se produit sur ce patch, incrémentez cette variable[18],
3. ajoutez un bouton on/off à l'interface, nommez le `ddanger`,
4. dans `to color_patch` ajoutez le condition qui permette l'affichage en niveaux de rouges des terrains dangereux (inspirez vous de l'affichage de la neige)[19],
5. vérifiez sur la simulation que l'affichage fonctionne correctement.

[12] `set inf neighbors with [height < [height] of myself]`

[13] `ask inf[set hlevel (hlevel + ([height] of myself - height))]`

[14] `*(([height] of myself - height) / hlevel))`
`set snow_level snow_level + ((0.8 *([snow_level] of myself))`

[15] `[set tmp lput self tmp]`
`if (member? self tmp = false and slope * snow_level > 4)`

[16] `avalanche (lput self [])`

[17] `patches-own[...danger]`

[18] `set danger (danger + 1)`

[19] `blend_color red ((danger + 2) / 10)]]`
`if ddanger = true[ if danger > 1 or slope > 0.01 * pente[`

### 4.2 Placer le refuge

Jouez des conditions de neige forte, avec des pentes de déclenchement de 30°, 60% et une courbe isométrique 0° vers 1600m. au bout d'un certain temps de simulation, vous devez pouvoir déterminer un endroit sûr pour le refuge.

1. en début de code, créez un type de tortue "refuge"[20],
2. à la fin de `to setup`, créez une tortue de type refuge, à la position choisie et avec la forme d'une maison[21],
3. rechargez la scène pour vérifier que le refuge est bien placé.

### 4.3 Alpinistes et déplacements sur la carte

Dans un premier temps, lors de l'initialisation, nous distribuerons aléatoirement une cinquantaine de tortues de type "`alpiniste`". A chaque tour de simulation, ils peuvent se déplacer vers l'avant, dans un angle de 90° et sur une unité.

1. A la fin de `to setup`, ajoutez 50 tortues de type "alpiniste". elles sont placées aléatoirement sur la carte[22],
2. implémentez la méthode to move_alpiniste[23],
3. Les alpinistes doivent être sensibles aux avalanches qui se produisent. Complétez `to avalanche` pour qu'un alpiniste présent sur un patch en effondrement soit aléatoirement déplace vers un patch inférieur (s'il y en a un).

### 4.4 Alpinistes et comportements supplémentaires

Pour terminer le travail de ce TD, vous coderez les propositions suivantes :

1. Les alpinistes "meurent" s'ils sont emportés par une avalanche sur plus de 200m ou s'il chutent de plus de 30m[24],
2. les alpinistes sont un facteur déclencheur supplémentaire d'avalanches,
3. quant deux alpinistes se rencontrent, ils marchent ensemble,
4. les alpinistes reprennent le cap du refuge tout les cinq déplacements,
5. le refuge est détruit s'il est touché par une avalanche.

## 5 Créer un modèle numérique de terrain

*Cette section est optionnelle.* Pour les simulations, il est très simple de créer votre propre modèle de terrain avec un logiciel d'illustration ou de retouche d'image.

Avec Gimp, vous pouvez suivre ces étapes :

1. démarrez Gimp,
2. créez un fichier vide de 257*257 pixels (`fichier` → `nouveau`),

[20] `breed [refuge]`

[21] `set shape "house", set color red ]`
`create-refuge 1 [ setxy -2 -13, set size 6`

[22] `ask alpiniste[right ((random 90) - 45), forward 1 ]`

[23] `ask alpiniste[right ((random 90) - 45), forward 1 ]`

[24] Utilisez les variables des alpinistes `last_height`, `last_xcor`, `last_ycor`

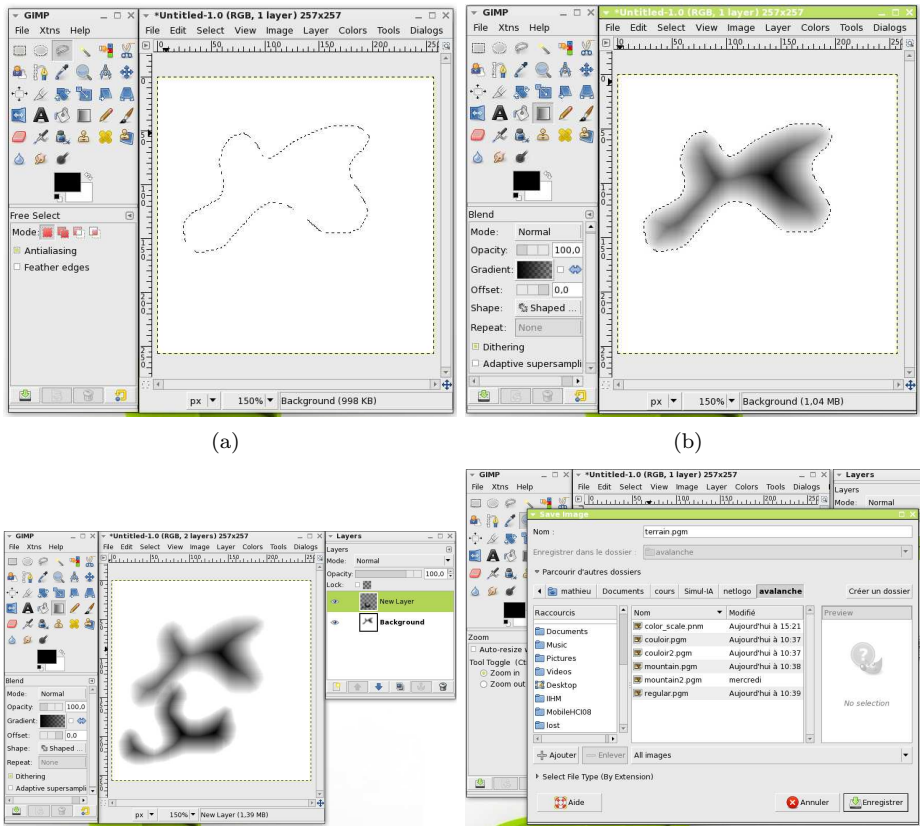



FIG. 2 – Création d'un modèle numérique de terrain dans Gimp

3. avec l'outil de sélection libre, dessinez les frontières d'une montagne (Fig. 2(a)),
4. paramétrez l'outil de remplissage en dégradé et clic-maintenu pour le remplissage depuis le centre de la sélection vers le bord (Fig. 2(b)),
5. créez un nouveau calque transparent et répéter les 2 précédentes opérations pour ajouter une nouvelle montagne (Fig. 2(c)),
6. menu `image` → `aplatir l'image`,
7. inversez l'image (`couleurs` → `inverser`),
8. ajoutez du flou à l'image (`filtre` → `flou`). Répétez cette operation deux ou trois fois,
9. sauvez le fichier avec une extension `.pgm` dans le répertoire de décompression de `avalanche.zip` (Fig. 2(d)). **Choisissez le format de sauvegarde ASCII non compressé**.

Le fichier crée contient une liste de 257*257 valeurs entre 0 et 255. Nous transformons cette liste en un fichier `.data` lisible par Netlogo. Dans la suite, nous considérons que ce fichier s'appelle `terrain.pgm`

Sous Linux

1. démarrez une console,
2. placez vous dans le répertoire ou vous avez désarchivé `avalanche.zip`,
3. exécutez le script pgm2data :
   `$ ./pgm2data terrain.pgm > terrain.data`

Sous Windows

1. ouvrez le fichier crée avec votre éditeur de texte favori,
2. supprimez les 4 premières lignes du fichier,
3. ajoutez `[` en première ligne,
4. ajoutez `]` en dernière ligne,
5. sauvegardez.

Dans Netlogo, avec le modèle `avalanche.nlogo` ouvert

1. éditez la liste des noms de fichier et ajoutez `''terrain.data''` (Fig. 3)

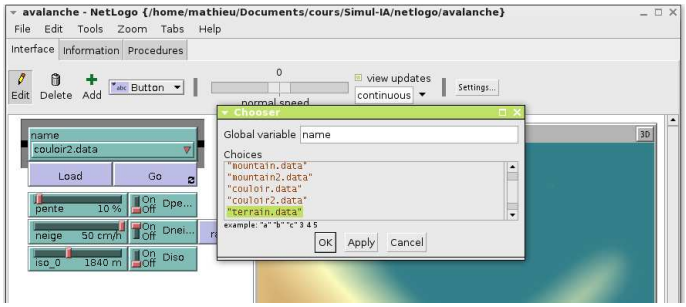



FIG. 3 – Inclusion du modèle crée pour la simulation dans netlogo

# 6 Dessin du terrain avec Gnuplot

*Cette section est optionnelle, elle vous permet une première prise en main du logiciel gnuplot qui sera nécessaire lors du TD suivant sur le perceptron.* Si vous avez installé le logiciel de tracé Gnuplot (Ver. > 4.0), suivez ces étapes pour générer une vue du relief du terrain "mountain.pgm" (Fig. 4(a)).

Sous linux (si vous êtes sous Windows, passez cette étape)

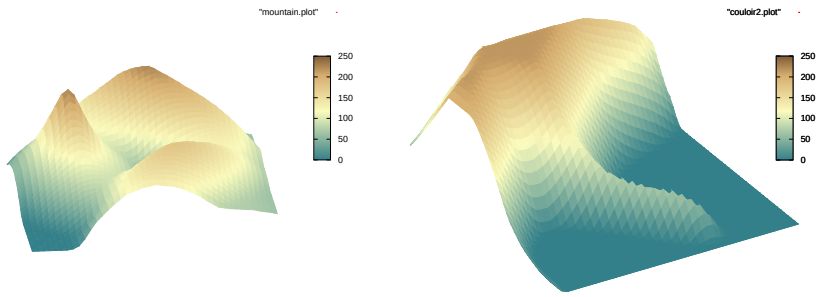
1. démarrez une console,
2. placez vous dans le répertoire ou vous avez désarchivé `avalanche.zip`,
3. transformez la carte de niveau en fichier de données Gnuplot :
   `$ ./pgm2plot 6 mountain.pgm > mountain.plot`
   "6" représente la résolution de la grille. Plus ce nombre est petit, plus le rendu sera précis (et lent).

Sous Windows et Linux

1. démarrez gnuplot depuis le répertoire de décompression de `avalanche.zip`,

2. exécuter la série de commandes suivantes :

```
gnuplot> set hidden3d
gnuplot> set pm3d
gnuplot> set palette file 'color_scale.data' using ($1/255) :($2/255) :($3/255)
gnuplot> set border 0
gnuplot> unset ytics ; unset ztics ; unset xtics
gnuplot> splot ``mountain.plot'' with dots
```




(a) mountain.plot (b) couloir2.plot

FIG. 4 – rendu de différents terrains

Vous pouvez manipuler l'orientation et le niveau de zoom avec la souris.