GUILBERT--LY Anne-Soline Numéro SCIPER : 302376

a) Phase d'analyse

La fonction main se décompose en 3 phases pour les 3 différents niveaux.

Dans chaque niveau, on ré-utilise la fonction construire_passage qui cherche dans la grille libre, un (ou plusieurs) passage(s) du nord au sud pour les mettre dans la grille passage. Pour le premier niveau, on utilise une fonction récursive cherche_chemin qui va parcourir libre. Cependant, dans les niveaux « b » et « c », on modifie légèrement construire_passage : la fonction construire_un_seul_passage s'arrête dès qu'elle trouve un chemin qui va du nord vers le sud, pour optimiser le temps de calcul.

Nous avons donc trouvé une solution au sous-problème qui est de construire passage. On peut ensuite calculer p' = (nombre de grilles traversables)/(nombres de terrains) dans les niveaux « b » et « c ».

Dans le niveau « b », la fonction calcul_pprime utilise d'autres fonctions comme init_libre_hasard qui est similaire à l'initialisation de la grille libre au niveau « a », mais avec une probabilité p et un générateur de hasard. Ensuite on appelle construire_un_seul_passage comme mentionnée ci-dessus, puis terrain_traversable qui envoie un booléen dès qu'elle trouve une case true sur la dernière ligne de la grille passage. Enfin on initialise passage à false pour que sa prochaine construction se déroule correctement (on fait les mêmes opérations pour nbt terrains). Ainsi, le sous-problème du calcul de p' est résolu.

Enfin, dans le dernier niveau, on utilise calcul_pprime pour pourvoir calculer max', min', et p' qu'on rentre dans un tableau, tout comme p. On affiche finalement p et p'. L'approche dichotomique est réalisée comme ci-dessous.

C'est ainsi qu'on utilise le principe d'abstraction et de ré-utilisation tout au long du programme.

b) Pseudocode

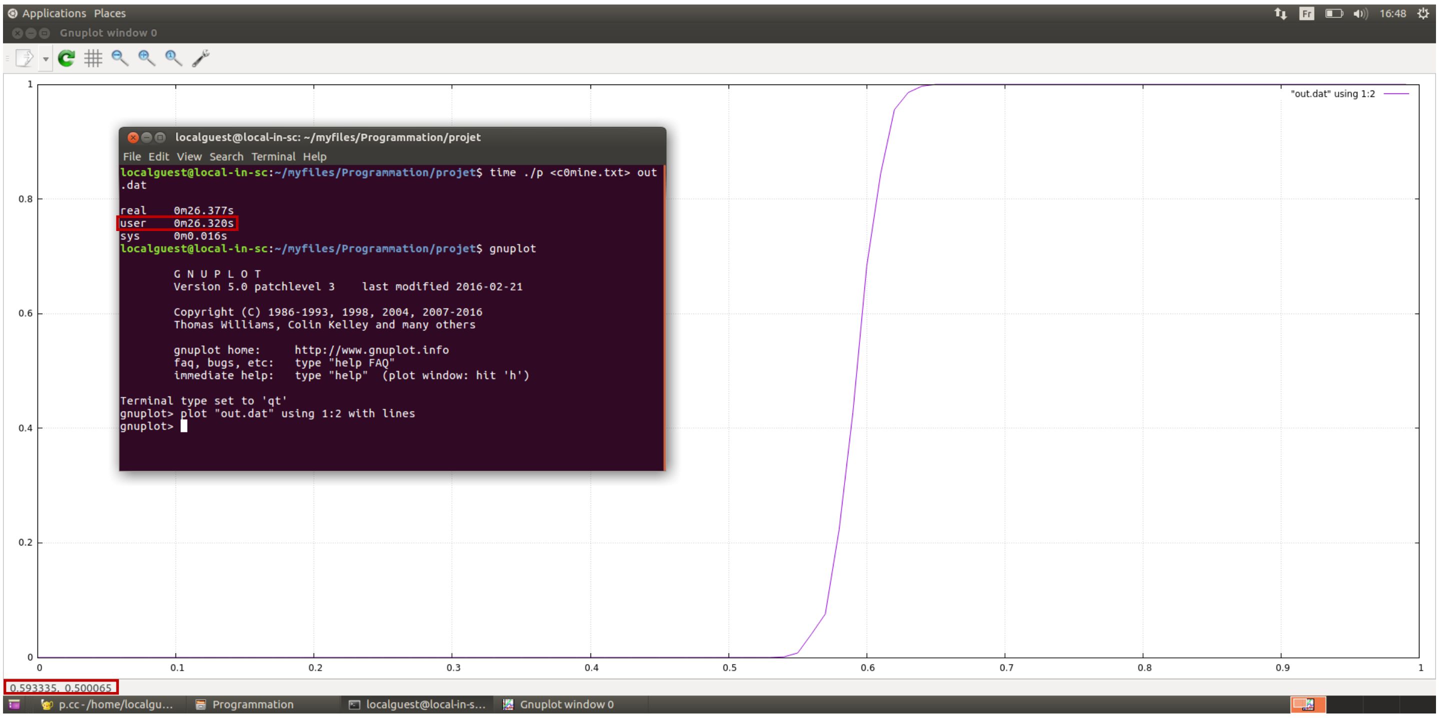
```
dichotomie
entrée : min, max, minprime, maxprime, error, p, pprime, n, libre, passage, e , nbterrains, courbe
sortie : aucune
Faire
    p ← (max+min)/2
    pprime ← calcul_pprime(libre, n, passage, e, p, nbterrains)
    error ← pprime – (maxprime + minprime)/2
    Ajouter p à courbe.p *
    Ajouter pprime à courbe.pprime **
    Si (error>0) :
        max ← p
        maxprime ← calcul_pprime(libre, n, passage, e, max, nbterrains)
    Si (error<0) :
        Min ← p
        Minprime ← calcul_pprime(libre, n, passage, e, min, nbterrains)
Tant Que ((max-min)>MIN_DELTA_P et |error| > MAX_ERROR)

Trier : courbe.p *
Trier : courbe.pprime **
Afficher : p et pprime
```

* et ** : courbe.p et courbe.pprime sont des tableaux dans une structure appelée courbe.

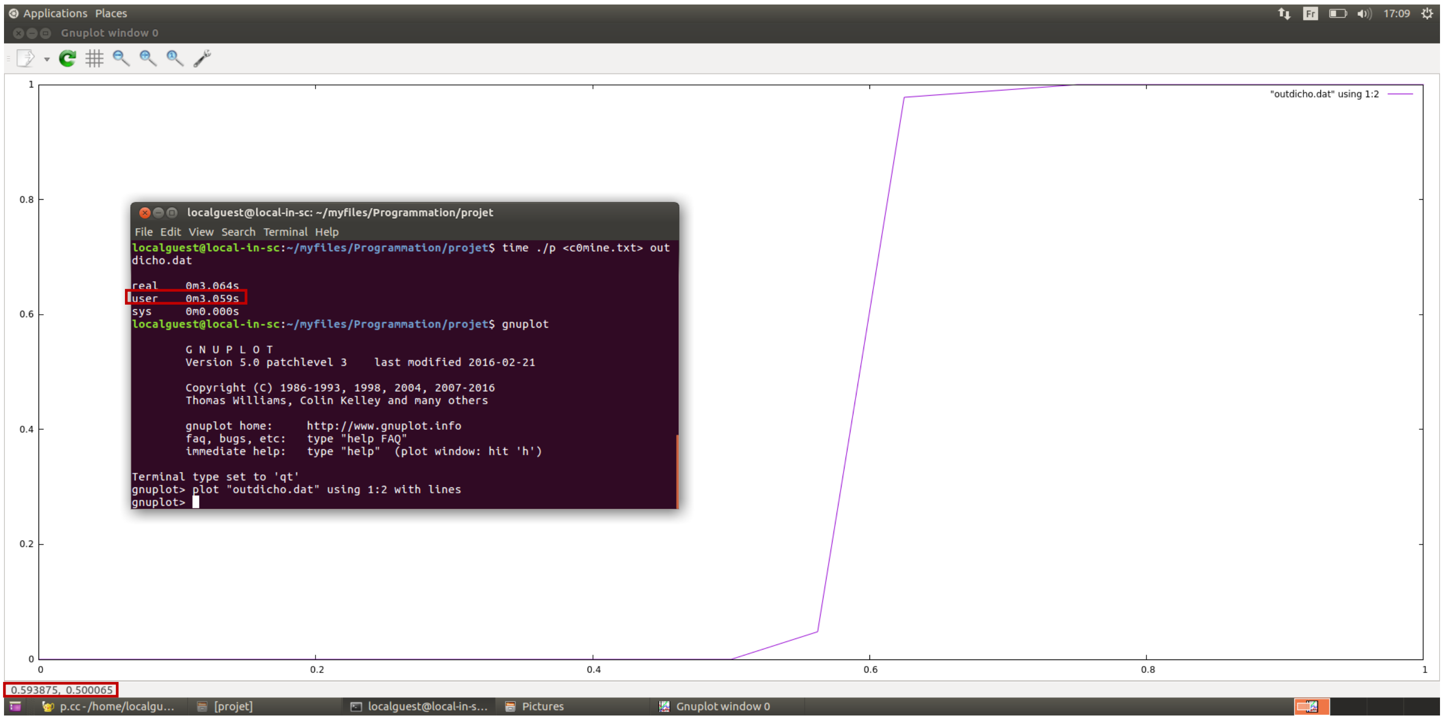
c) <u>Réponses aux questions des sections 3.3.1. et 3.3.2.</u>

Temps calcul « user » pour le <u>3.3.1.</u> : 26,320s.



Et on a pour p' = 0.500065, p= 0.593335

Temps calcul « user » pour le <u>3.3.2.</u> : 4,158s.



Et on a aussi pour p'=0.500065, p=0.593875
Il y a 7 échantillons qui ont été effectué (dont les paires (0,0) et (1,1)), d'où le gain de temps. La précision n'est pas mieux, et on a (environ) le même nombre pour p.