Université de Cergy-Pontoise

RAPPORT

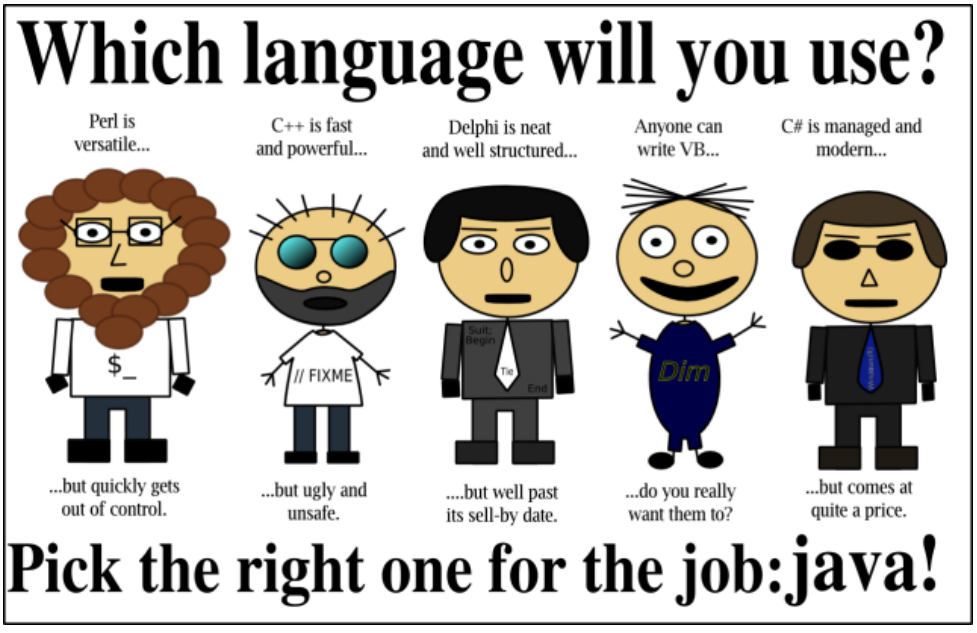
TD 2 IA
Master 1

sur le sujet

Logique floue

rédigé par

Boyan CHEYNET



Décembre 2024

Table des matières

Table des figures

Liste des tableaux

1 Introduction

Ce projet vise à développer un contrôleur pour un robot mobile e-puck, simulé dans l'environnement Webots, en utilisant la logique floue. L'objectif principal est de permettre au robot de suivre un mur tout en ajustant de manière dynamique sa vitesse et son orientation, en fonction des informations provenant de ses capteurs de distance situés à gauche et à droite.

La logique floue est choisie pour sa capacité à gérer les incertitudes et les imprécisions inhérentes aux systèmes réels. Contrairement aux méthodes classiques qui reposent sur des seuils stricts pour prendre des décisions, la logique floue permet de traiter des informations incomplètes ou imprécises en fournissant des réponses graduelles. Cette approche est particulièrement adaptée aux environnements dynamiques, où les conditions peuvent varier rapidement et où une solution rigide pourrait ne pas suffire pour assurer un suivi stable et précis.

Dans le cadre de ce projet, la logique floue est utilisée pour définir un ensemble de règles permettant de moduler la vitesse et l'orientation du robot en fonction de la proximité des obstacles détectés. Par exemple, lorsque le robot se rapproche trop près du mur, il peut ajuster son orientation pour éviter les collisions, tout en maintenant une vitesse appropriée pour naviguer en toute sécurité.

Les capteurs de distance du robot jouent un rôle clé dans ce processus, fournissant des données qui sont ensuite fuzzifiées, inférées selon les règles définies, puis défuzzifiées pour générer des commandes précises de vitesse et d'orientation. Ce projet explore l'application de la logique floue dans un contexte de navigation autonome, avec pour objectif d'optimiser les performances du robot tout en garantissant sa stabilité dans des environnements complexes.

2 Système de la logique floue

2.1 Variables linguistiques

Pour contrôler le robot, nous avons défini deux types de variables linguistiques : les variables d'entrée, qui correspondent aux distances mesurées par les capteurs, et les variables de sortie, qui déterminent l'orientation et la vitesse du robot. L'image des graphiques associés à ces variables est présentée en annexe.

— Variables d'entrée

— Distance (à droite et à gauche) : Proche, Moyenne, Lointaine. L'image associée à cette variable est en annexe pour visualiser le graphe (1).

$$\text{Lointaine (en rouge)} \ : g(x) = \frac{1}{1 + e^{(x-65)\cdot 0.2}} \tag{1}$$

$$\text{Moyenne (en vert)} \ : f(x) = \frac{1}{0.4\sqrt{2\pi}} e^{-0.0002\left(\frac{x-100}{0.4}\right)^2} \tag{2}$$

$$\text{Proche (en bleue)} \ : t(x) = \frac{1}{1 + e^{(-x+160)\cdot 0.08}} \tag{3}$$

Les distances sont calculées en prenant le maximum des valeurs retournées par les capteurs de chaque côté du robot. L'image de cette méthode est disponible en annexe.

— Variables de sortie

— Orientation : Tourner à gauche, Tout droit, Tourner à droite. L'image du graphe pour cette variable est en annexe (2).

$$\text{Tourner à gauche (en bleue)} \ : g(x) = \frac{1}{1 + e^{(x+20)\cdot 0.1}} \tag{4}$$

$$\text{Tout droit (en rouge)} \ : f(x) = \frac{1}{0.4\sqrt{2\pi}} e^{-0.00018\left(\frac{x}{0.2}\right)^2} \tag{5}$$

$$\text{Tourner à droite (en vert)} \ : t(x) = \frac{1}{1 + e^{(-x+20)\cdot 0.1}} \tag{6}$$

— Vitesse : Lente, Moyenne, Rapide. Le graphe associé à cette variable se trouve aussi en annexe (3).

$$\text{Lente (en orange)} \ : g(x) = \frac{1}{1 + e^{(x-3)\cdot 12}} \tag{7}$$

$$\text{Moyenne (en rouge)} \ : y = \begin{cases} \frac{x-2.5}{0.5} & 2.5 \leq x < 3 \\ 1 & 3 \leq x < 4 \\ \frac{4.5-x}{0.5} & 4 \leq x < 4.5 \end{cases} \tag{8}$$

$$\text{Rapide (en bleue)} \ : f(x) = \frac{1}{0.4\sqrt{2\pi}} e^{-0.1\left(\frac{x-4.5}{0.2}\right)^2} \tag{9}$$

Ces sorties contrôlent directement l'orientation et la vitesse du robot, qui seront ensuite converties en vitesses pour chaque roue.

2.2 Règles floues

Nous avons défini 6 règles principales, dont 3 pour le contrôle de l'orientation du robot et 3 pour la vitesse du robot, en fonction des distances perçues à droite et à gauche. On a pris comme décision que le robot allait longer le mur par la gauche :

— Règle de vitesse

1. **(Courbe orange)** Si la distance est proche des deux côtés, alors la vitesse est lente. Cela nous permet de faire les manœuvres plus aisément.

2. **(Courbe rouge)** Si la distance est moyenne, alors la vitesse est moyenne. Cela nous permet de faire un potentiel changement d'angle sans trop aller vite.
3. **(Courbe bleue)** Si la distance est lointaine, alors la vitesse est rapide. Comme on est loin, on peut se permettre d'aller vite.

— Règle d'angle

1. **(Courbe bleue)** Si on est loin à gauche ou proche à droite, alors on tourne à gauche. On fait cela pour rester collé au mur de gauche.
2. **(Courbe rouge)** Si on est moyen à gauche et loin à droite, alors on va tout droit. Cette règle permet de rester à une bonne distance du mur de gauche tout en le longeant.
3. **(Courbe verte)** Si on est proche à gauche et loin à droite, alors on tourne à droite. On l'utilise pour se décoller du mur de gauche si on est trop proche.

Ces règles permettent de moduler la vitesse du robot en fonction de sa proximité avec les obstacles, ce qui est essentiel pour assurer un suivi stable du mur tout en évitant les collisions.

2.3 Justification des choix

Les règles floues ont été conçues pour répondre à deux objectifs principaux : assurer un suivi optimal du mur tout en maintenant une vitesse adaptée aux différentes situations.

Les règles de vitesse permettent au robot d'adapter sa rapidité en fonction de sa proximité aux obstacles. Une vitesse lente est cruciale dans les situations nécessitant des manœuvres précises, tandis qu'une vitesse rapide est justifiée lorsque le robot est loin des obstacles, optimisant ainsi son temps de déplacement.

Les règles d'orientation sont pensées pour maintenir le robot à une distance stable du mur de gauche. Le robot ajuste continuellement sa trajectoire pour rester proche du mur sans s'y heurter, tout en réagissant aux obstacles détectés à droite. Ce système assure un contrôle fluide et efficace, permettant au robot de suivre le mur de manière autonome et sécurisée.

L'ensemble de ces règles garantit une flexibilité dans les réactions du robot face à son environnement, tout en assurant un comportement cohérent et sécurisé, qu'il s'agisse de suivre une trajectoire droite ou de manœuvrer pour éviter les obstacles.

3 Implémentation du Contrôleur Flou

3.1 Présentation succincte du code

Le code implémente un contrôleur flou pour un robot mobile simulé dans Webots. Ce contrôleur utilise les données des capteurs de distance pour ajuster la vitesse et l'orientation du robot en fonction des distances perçues à gauche et à droite.

Le programme suit plusieurs étapes principales :

— Initialisation du robot et des capteurs.
— Fuzzification des valeurs des capteurs, c'est-à-dire la transformation des distances mesurées en variables floues (proche, moyen, loin).
— Inference basée sur des règles floues pour déterminer la vitesse et l'angle d'orientation du robot.
— Défuzzification pour convertir les valeurs floues en valeurs précises de vitesse et d'angle.
— Application des vitesses calculées aux moteurs du robot.

Chaque étape du processus est réalisée dans une boucle principale, qui exécute ces calculs en temps réel à chaque cycle de simulation.

3.2 Choix des algorithmes et méthodes de contrôle

Les algorithmes de fuzzification et défuzzification sont basés sur des fonctions de type sigmoïde et gaussienne, qui permettent de modéliser des transitions douces entre les différentes catégories de distance (proche, moyen, loin). Cela est crucial pour obtenir un comportement fluide et graduel du robot.

Les règles d'inférence sont conçues pour ajuster à la fois la vitesse et l'orientation du robot en fonction des distances perçues :

— Pour la vitesse, les règles déterminent si le robot doit aller lentement, à une vitesse moyenne, ou rapidement, selon la proximité des obstacles.
— Pour l'angle, les règles déterminent si le robot doit tourner à gauche, aller tout droit, ou tourner à droite, en fonction de la distance des obstacles à gauche et à droite.

L'inférence se fait par des opérations logiques sur les variables floues, en sélectionnant les valeurs maximales ou minimales selon les règles définies.

3.3 Explication des choix

Dans notre implémentation, plusieurs choix ont été faits pour optimiser le comportement du robot et assurer une conduite plus fluide et réaliste. Tout d'abord, en ce qui concerne la méthode de défuzzification, nous avons choisi d'appliquer la méthode des centroïdes. Cette méthode permet de récupérer les points de la courbe qui sont les plus proches de l'inférence, en calculant le minimum entre les valeurs de la courbe et les inférences des règles floues. Cela garantit une transition douce entre les différents niveaux de commande, plutôt qu'une approche brutale qui pourrait provoquer des mouvements saccadés du robot.

En ce qui concerne le calcul de l'angle et de la vitesse, un autre choix important a été de prendre la moyenne des valeurs des deux capteurs de distance (gauche et droite) au lieu de simplement prendre le maximum des deux valeurs. Cette approche a été préférée car elle empêche le robot de prendre des décisions trop extrêmes basées sur la lecture d'un seul capteur. En effet, si l'on choisissait uniquement le capteur avec la valeur maximale, cela entraînerait des comportements trop réactifs et moins équilibrés. La moyenne permet de lisser les résultats obtenus des capteurs, offrant ainsi une conduite plus stable et plus fluide, surtout lorsque les valeurs des capteurs sont proches. Cela permet également d'éviter des décisions trop radicales qui pourraient mener le robot à des comportements imprévus ou trop erratiques.

4 Conclusion et Perspectives

4.1 Synthèse

Le projet a permis de développer un contrôleur flou pour un robot mobile simulé dans Webots. Ce contrôleur ajuste la vitesse et l'orientation du robot en fonction des distances perçues par les capteurs de distance situés à gauche et à droite. En utilisant la logique floue, le robot est capable de gérer de manière fluide et graduelle son comportement face aux obstacles, en adaptant sa trajectoire en temps réel.

Les règles floues définies pour la gestion de la vitesse et de l'orientation ont permis de moduler les comportements du robot en fonction des variations des distances mesurées. La méthode de défuzzification par les centroïdes a été appliquée pour convertir les valeurs floues en commandes précises, assurant ainsi une navigation stable.

4.2 Limites

4.2.1 Notre projet

Bien que le projet ait permis de réaliser un contrôleur fonctionnel, certaines limitations sont apparues au cours du développement. Tout d'abord, les capteurs de distance utilisés par le robot présentent une précision limitée, ce qui peut conduire à des erreurs de mesure, notamment dans des situations où la surface des obstacles est irrégulière ou les distances sont faibles. Cette imprécision affecte la capacité du robot à ajuster sa trajectoire de manière optimale, en particulier dans des environnements complexes où une détection fine des obstacles est cruciale.

4.2.2 Logique floue

La logique floue, bien que adaptée à la gestion de l'incertitude, présente aussi certaines limites. La définition des règles et des fonctions de fuzzification nécessite une expertise approfondie pour éviter des comportements non optimaux du robot. Par ailleurs, la méthode de défuzzification par les centroïdes, bien qu'efficace, peut devenir coûteuse en termes de calculs lorsque le nombre de variables et de règles augmente, ce qui limite l'évolutivité du système.

Les règles fixées pour la gestion de la vitesse et de l'orientation, bien que raisonnables dans un cadre simple, peuvent ne pas être suffisantes pour traiter des scénarios plus complexes. Une approche plus dynamique, intégrant des mécanismes d'adaptation en temps réel, pourrait offrir une meilleure flexibilité face aux variations de l'environnement.

4.3 Perspectives à long terme

À long terme, plusieurs pistes d'amélioration peuvent être envisagées pour rendre ce système plus robuste et adaptable. L'intégration de capteurs plus précis, tels que des capteurs LiDAR ou des caméras, permettrait d'améliorer la détection des obstacles et de réduire les erreurs de mesure. Cela ouvrirait également la voie à une meilleure gestion des environnements complexes et à une navigation plus autonome.

Une autre perspective est l'intégration de techniques d'apprentissage automatique, telles que le renforcement, pour permettre au robot d'apprendre et de s'adapter à son environnement au fur et à mesure de son interaction avec celui-ci. Ce type d'approche permettrait au robot d'ajuster ses comportements en fonction des situations rencontrées, sans nécessiter de reconfiguration manuelle des règles floues.

Enfin, l'utilisation d'algorithmes d'optimisation pour ajuster les paramètres de fuzzification et de défuzzification en temps réel pourrait améliorer l'efficacité du contrôleur flou et offrir une meilleure réactivité face à des environnements plus dynamiques.

Annexe

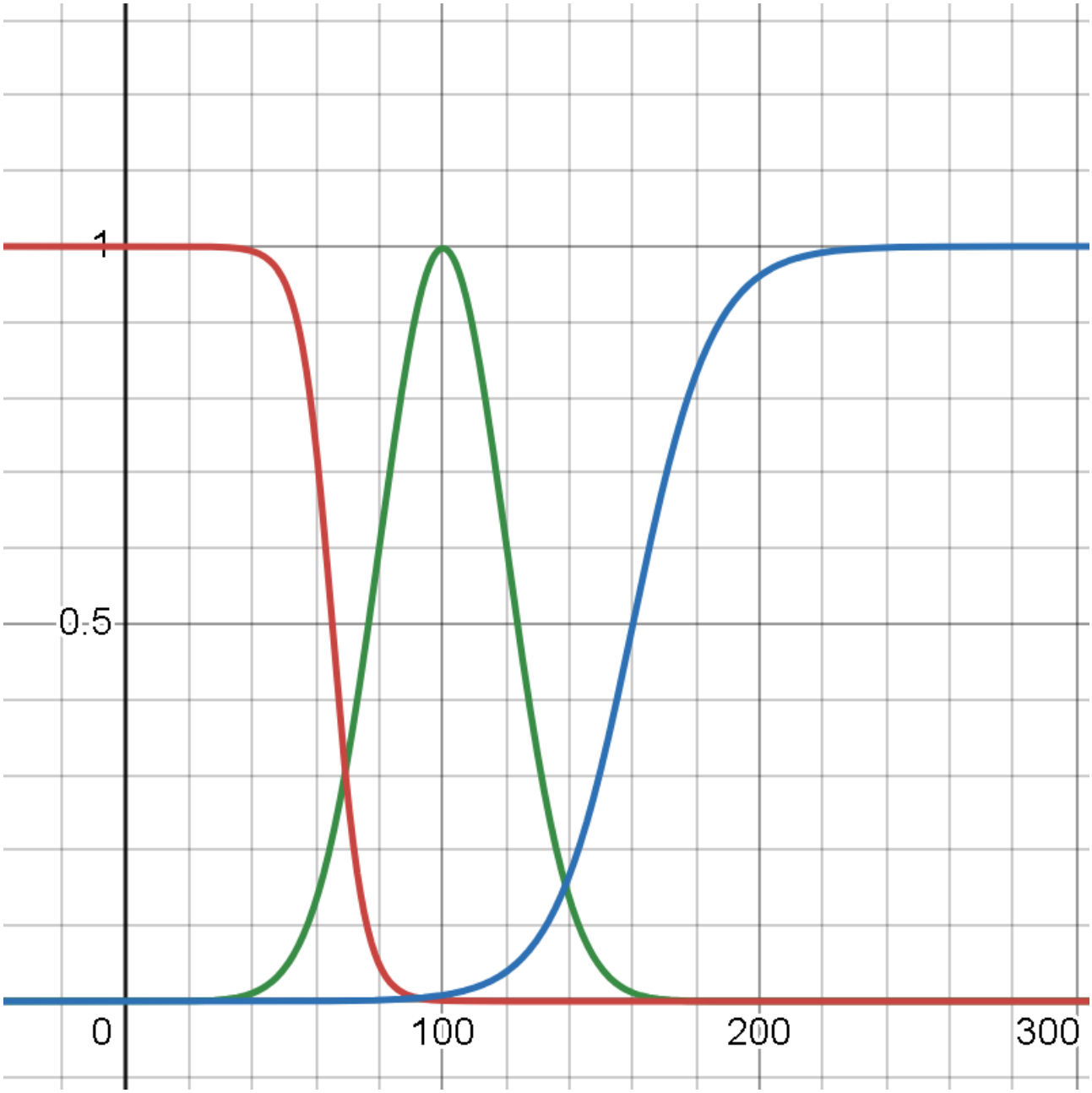


FIGURE 1 – Graphe de la variable "Distance"

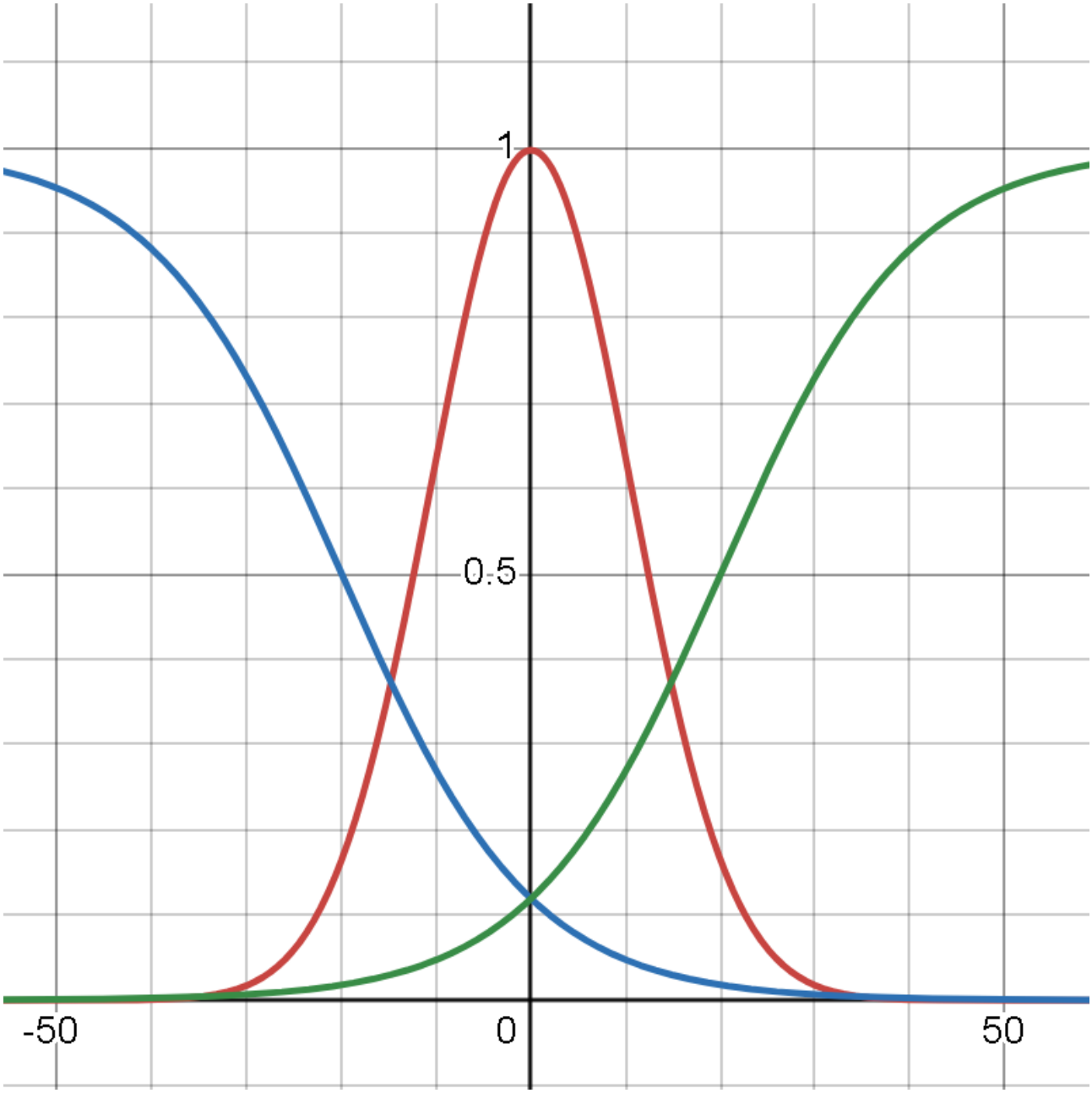


FIGURE 2 – Graphe de la variable "Orientation"

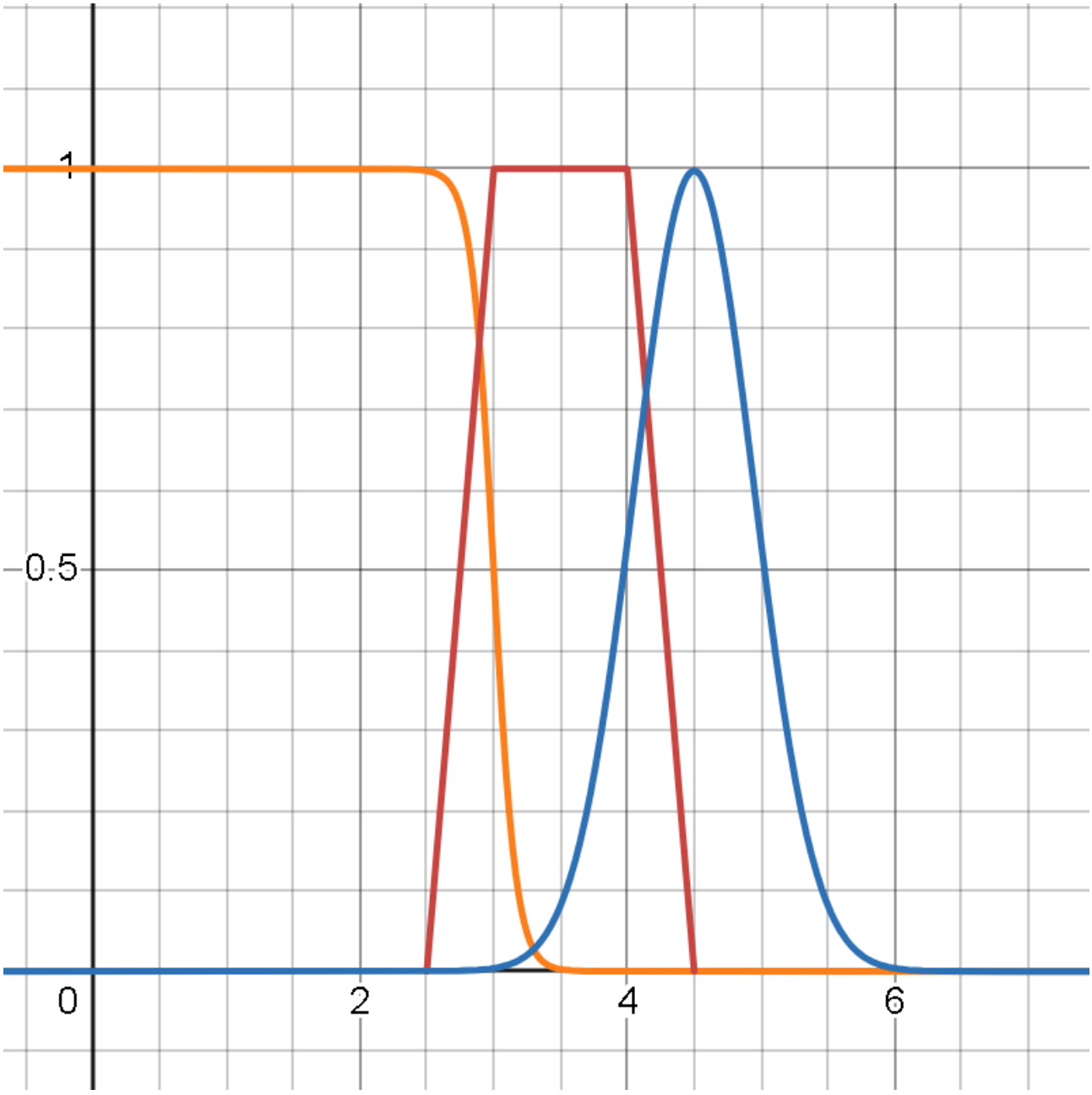


FIGURE 3 – Graphe de la variable "Vitesse"