Question 2: Reprenez votre TD précédent et implémentez le déplacement de votre robot en ligne droite et son arrêt devant le premier obstacle rencontré.

Fait. On utilise le réseau de contrôle braitenberg en modifiant les valeurs et les borne de vitesse. Le réseau fonctionne ainsi : tous les sensors ont un poids de 0 à l’exception des 2 capteur de devant et celui de derrière. Ceux de devant demande au robot d’avancer et celui de derrière de reculer avec un poids plus faible. Lorsque le robot rencontre un obstacle, le poids des capteurs est réduit et le robot ralentit jusqu’à que la commande d’avancer s’équilibre avec la commande de reculer. Pour avoir un arrêt stable, on empêche le robot de reculer en réduisant sa borne de vitesse à 0.

const double khepera3\_matrix[9][2] =

{{0,0}, //1

{0, 0}, //2

{0, 0}, //3

{40000,40000}, //4

{40000, 40000}, //5

{0, 0}, //6

{0, 0}, //7

{0, 0}, //8

{-70000, -70000}}; //9

speed[i] = BOUND(speed[i], 0, max\_speed); // on change le bound pour que le véhicule ne peut pas reculer et s’arrête lorsqu’il rencontre un obstacle.

Les capteurs à l’avant sont 4 et 5.

Question 3: Avec le simulateur Webots, en utilisant le robot khepera III, implémentez cet algorithme en utilisant tous les prosimètres du robot. Choisissez les bons poids pour que le robot est un comportement satisfaisant.

Le poids des sensor a pour but d’être équilibré si les distances sensors situé sur les côtés droit et gauche ne voit pas d’obstacle. En revanche, si un obstacle est détecté, alors l’équilibre est rompu. Les poids sont donner tels que si un objet est détecté à droite, le robot va tourner à gauche pour éviter l’obstacle. Il existe également une asymétrie à gauche et à droite dans les valeurs de sorte que le robot privilégie l’évitement par la droite pour éviter les blocages.

Voici les valeurs :

{{-5000, -5000}, //1 {-20000, 40000}, //2 {-30000, 50000}, //3 {-70000, 70000}, //4 {70000, -60000},//5 {50000, -40000}, //6 {40000, -20000}, //7 {-5000, -5000}, //8 {-10000, -10000}}; //9

La matrice des poids est multipliée par la valeur des capteurs de proximité. La valeur ainsi obtenue est alors additionnée à la vitesse actuelle pour chaque moteur du robot.

Question 4: Comment implémenteriez-vous un réseau de neurones pour l’évitement d’obstacles selon braintenberg dans le cas de l’alphabot2 ?

Le réseau neuronal est implémenté avec une matrice de poids mappant la valeur des capteurs de proximité qui est multiplier par la valeur de la matrice vers une vitesse pour les roues droite et gauche. La vitesse obtenue est additionnée à la vitesse actuelle sans dépasser les borne maximal et minimal de vitesse.

-----------

Question 5: Si vous deviez fournir un algorithme d’évitement d’obstacle pour l’Alphabot2, sous forme d’un automate à état fini, quel serait-il (1) ? Implémentez et testez.

Tout droit, si obstacle à droite, tourner à gauche et inversement. Si obstacle trop proche, demi-tour.

Obstacle gauche

aucun obstacle

pas obstacle droite

obstacle deux cotés

obstacle deux cotés

pas obstacle gauche

aucun obstacle

Obstacle droite

---------------

Suivie de contours d’obstacle :

Question 6: Quel algorithme mettriez-vous en place pour un suivi de contours d’obstacles par la droite sur le Khepera III ? Implémentez et testez.

Je ferai en sorte que le robot avance, s’il détecte un obstacle devant il tournerai à droite puis suis la bordure à droite.

const double khepera3\_matrix[9][2] =

{{-0,0}, //1

{-0, 0}, //2

{-0, 0}, //3

{-20000,30000}, //4

{30000, -20000}, //5

{0, -00}, //6

{0, -0}, //7

{-0, -0}, //8

{0, 0}}; //9

-------------

Suivie de ligne au sol :

Question 7: Si vous voulez implémenter un suivi de ligne grâce aux capteurs du robot Alphabot2, que pro- poseriez-vous comme algorithme ? Implémentez et testez.

Il faudrait un capteur infrarouge pour voir la ligne blanche, tant que ce capteur voit cette ligne, l’équation est à l’équilibre, si la ligne n’est pas visible l’équation ferait tourner le véhicule pour essayer de revoir cette ligne. ------------