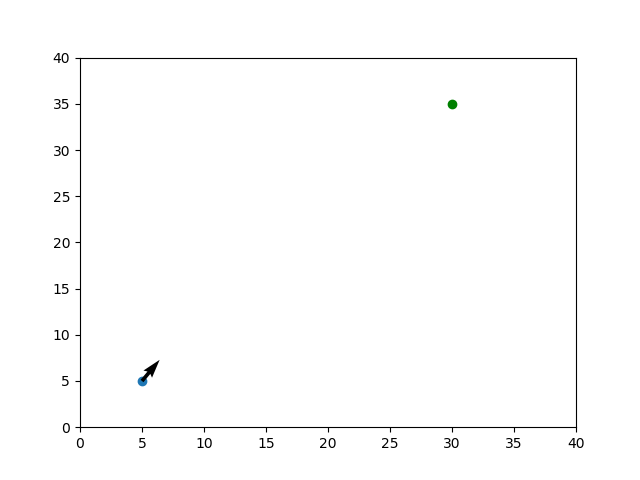
I Suivi d’itinéraire

Le suivi d’itinéraire est un point essentiel de ce projet. Le robot doit être capable de rallier une position demandée de manière autonome. Pour cela, il doit être capable de suivre une consigne donnée par le programme. A tout instant le robot va chercher à aligner sa direction avec un vecteur directeur qui fera office de consigne. Pour l’instant l’asservissement ne se fait que par un correcteur proportionnel.

II Evitement d’obstacle

Lorsque notre robot se déplace d’un point à un autre, il se peut (et dans le cadre du concours c’est même sûr) qu’il y ait un ou des obstacles sur le chemin. C’est pourquoi on cherche à implémenter dans le robot un algorithme d’évitement d’obstacle.

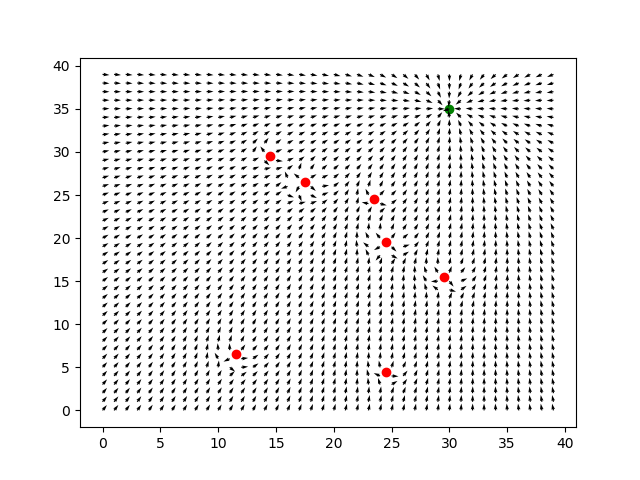
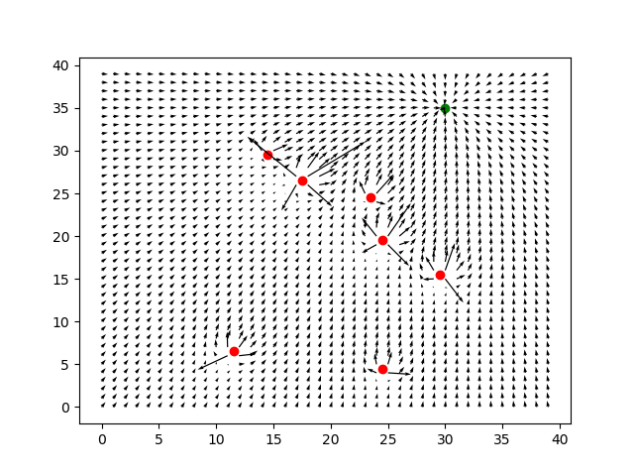
II.1 Les champs de potentiel

 La première technique utilise les champs de potentiel. L’objectif est simple : à chaque point de l’espace on associe un vecteur qui pointe vers le waypoint (en vert ci-dessous). Par exemple au point bleu on associe le vecteur unitaire noir qui pointe vers le point vert. Ce vecteur sera un repère pour asservir le robot (voir Suivi d’itinéraire).

Maintenant on s’intéresse au cas où on détecte un obstacle. Chaque obstacle sera représenté par un point dit « répulsif ». Ils sont ainsi appelés car leur rôle est de faire varier l’inclinaison des vecteurs afin que le robot puisse éviter les obstacles. On veut également que plus le robot soit proche d’n point répulsif, plus l’influence de ce dernier soit forte sur le vecteur directeur. Mathématiquement ce problème se traduit de la manière suivante :

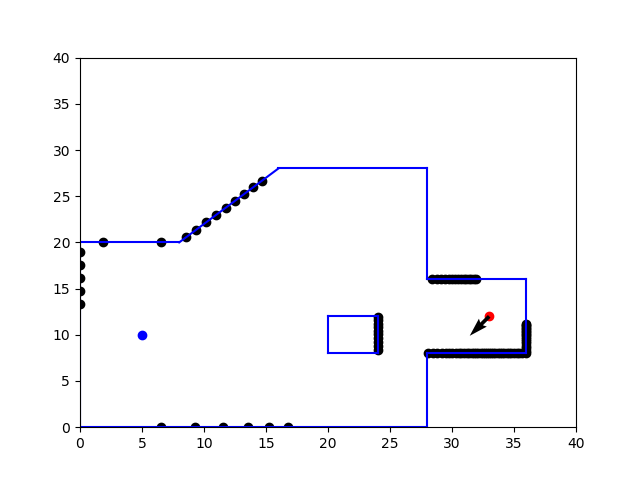
* On crée un waypoint à suivre
* A tout point de l’espace on associe un vecteur unitaire qui pointe vers le waypoint
* A tout point répulsif on associe un champ de potentiel de norme égal à l’inverse de la distance entre le robot et le point répulsif et de direction radiale
* On ajoute les deux champs

A l’aide de cette méthode on obtient bien des vecteurs qui tendent à éviter les obstacles autour des points répulsifs et des vecteurs loin des points répulsifs qui pointent vers le waypoint à atteindre. Ci-dessous le résultat brut avec les points répulsifs en rouge (fig2) et le résultat avec les champs normés pour y voir plus clair (fig3)



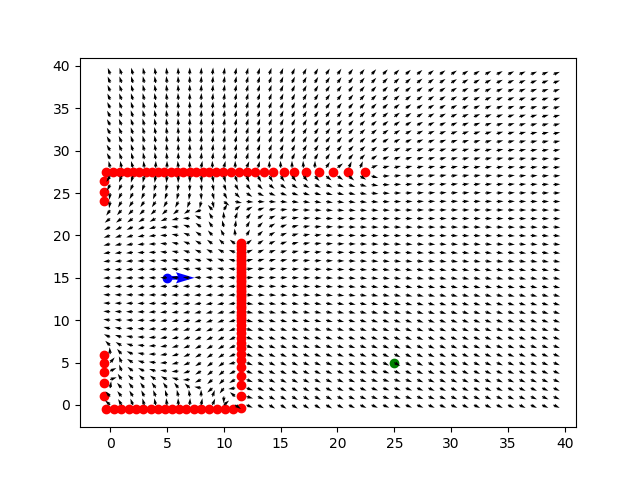
II.2 Simulation d’un Lidar

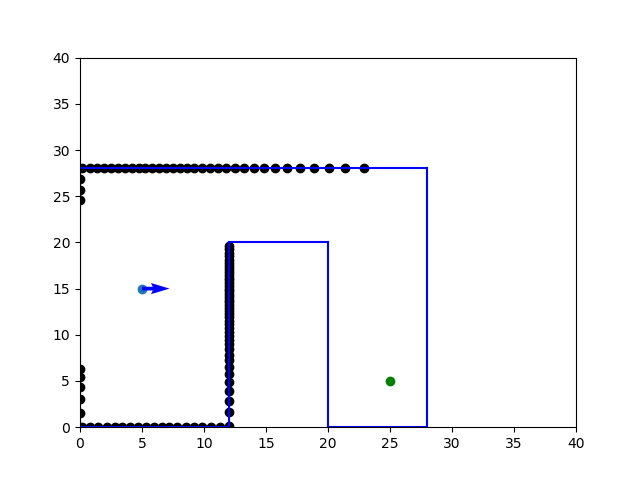
Maintenant qu’une technique d’évitement a été mise au point on va chercher à simuler notre lidar. En effet, il est bien plus aisé de tester un algorithme dans plein de situations générées numériquement que de prendre du temps pour créer un possible environnement physique et voir si le robot réel réagit comme convenu. On simule d’abord une pièce ou un environnement dans lequel évolue le robot. Après un traitement mathématique (Tiré d’un exercice de robotique de Luc JAULIN :<https://www.youtube.com/watch?v=oHbTrxpnOHo&feature=youtu.be>) , on acquiert la position des murs. Voici le résultat obtenu avec 150 points d’acquisition.



On remarque que plus les murs sont loin plus l’espacement entre les points noirs images du mur sont espacés. Cela est tout à fait normal car le lidar prend une valeur pour chaque pas angulaire. En faisant varier ce pas on augmente et on diminue le nombre de points obtenus. En simulation, on ne dépassera pas 200 points pour avoir un temps raisonnable de calcul.

II.3 Mise en évidence des limites des champs de potentiel

 Reprenons notre simulation avec une salle différente, où le vecteur bleu représente l direction du robot. On applique donc notre algorithme et on obtient la fig5.



Le problème est le suivant : le champ de vecteur résultant tend à faire rebrousser le robot en arrière. Or il y a aussi un mur derrière, mais il est situé dans l’angle mort du robot. Il donc va se retrouver à osciller et à faire tout le temps demi-tour.