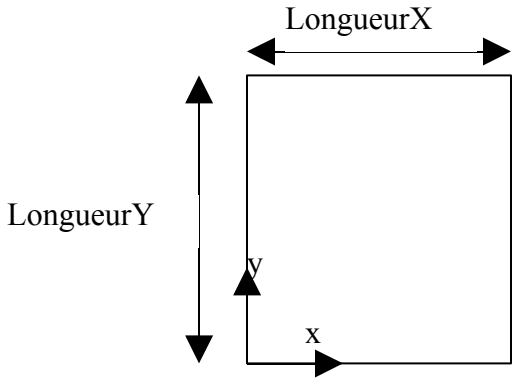
Calcul de Position Robot à partir de 3 balises

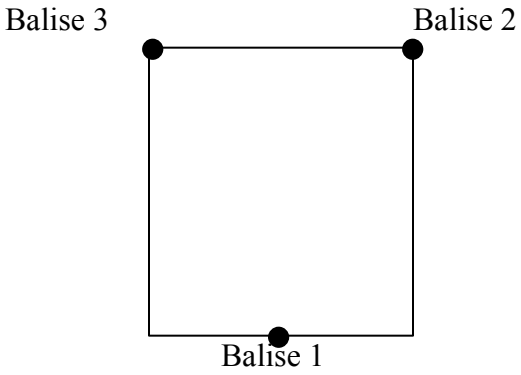
1. Mise en place

a) On considère un terrain de largeur **LongueurX** et de hauteur **LongueurY**. Un repère est cartésien est placé sur le terrain avec pour origine le point inférieur gauche.



b) Sur ce terrain, sont disposé, 3 balises aux coordonnées suivantes :

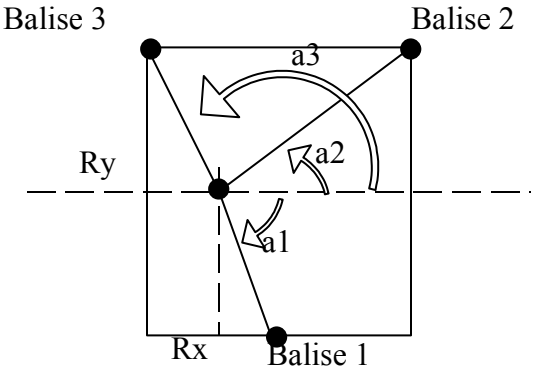
- Balise 1 : B1x = LongueurX /2 ; B1y = 0 ;
- Balise 2 : B2x = LongueurX ; B2y = LongueurY ;
- Balise 3 : B3x = 0 ; B3y = LongueurY ;



c) Sur ce terrain, on dispose un robot dont les coordonnées seront notées (Rx, Ry). On assume que :

- Rx appartient à [0, LongueurX]
- Ry appartient à [0, LongueurY]

d) On notera respectivement a1, a2 et a3 les angles formés par l'axe des abscisses et la droite rejoignant le robot respectivement avec les balises 1, 2 et 3.



Remarque 1: *Le choix du calcul des angles par rapport à l'horizontale est dans un souci de simplicité des calculs. L'abscisse est le repère naturel des calculs trigonométrique. Une référence sur l'axe vertical pourrait induire des erreurs d'interprétation fatale aux calculs. Si le choix de l'axe vertical est essentiel, les calculs ci-dessous pourront être utilisés en rapportant les angles mesurés depuis l'axe vertical sur l'axe horizontal (il suffit pour cela d'ajouter PI/2 aux angles mesurés).*

Remarque 2 (Importante) : *les angles sont signés, ce qu'il signifie que leur direction (sens trigonométrique ou sens anti-trigonométrique) a une importance capitale. Dans les calculs qui vont suivre, ne jamais inversé les termes d'une différence ou ne jamais tirer la valeur absolu d'une valeur. De même, la mesure des angles doit être signée.*

Remarque 3 (Importante) *: les calculs ci-dessous tiennent compte de mesures en Radian car il s'agit de l'unité naturelle de la trigonométrie et car la plupart des fonctions de calcul de divers langage de programmation (C, Java, Fortran,…) effectuent leurs calculs en Radians. Rappelons que :*

- *180° = PI radians*
- *90° = PI/2 radians*
- *360° = 2*PI radians*

On notera alors que :

- L'angle a1 appartient à [- PI ; 0] soit [-180° ; 0°]
- L'angle a2 appartient à [0 ; PI/2] soit [0° ; 90°]
- L'angle a3 appartient à [PI/2 ; PI] soit [90° ; 180°]

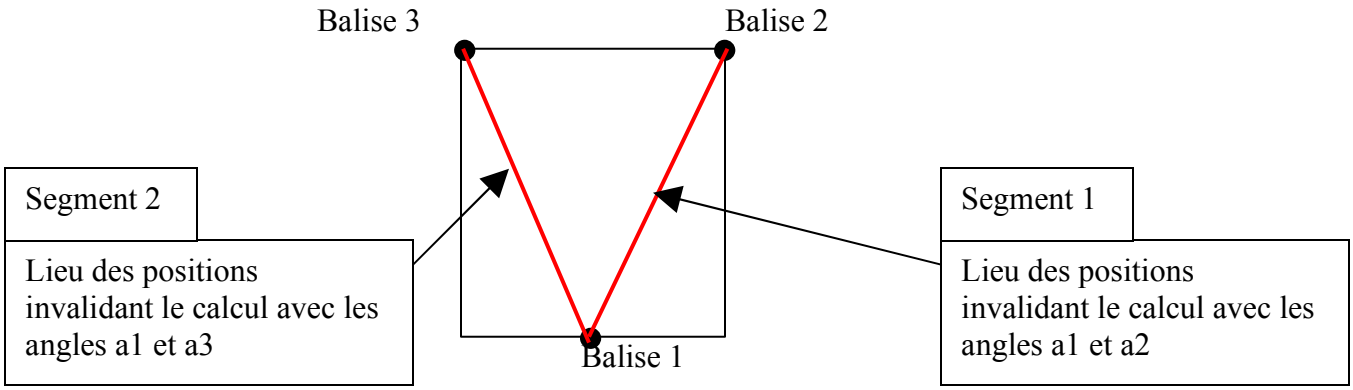
Ces intervalles doivent être considérés lors des mesures : un angle mesuré ne faisant pas parti de l'intervalle correspondant doit être considéré comme invalide.

2. Calcul de la position du robot en fonction des angles

Suivant le schéma ci-dessous, on peut déduire la position du robot à partir de chaque paire d'angle : (a1, a2), (a1, a3) ou (a2, a3).

Les calculs ci-dessous ont des « limites aux bornes ». Ce qui signifie que dans le cas où les coordonnées du robot viennent à prendre des valeurs aux limites (le robot se déplace ou atteint les bords du terrain), certains calculs sont invalidés. Des cas particuliers sont alors à prendre en compte. Cependant, puisque le robot n'a pas une dimension nulle et que le point de mesure devrait se trouver au centre du robot, il ne semble pas possible que le point de mesure atteigne les valeurs aux limites (les bords du terrain)
Ainsi, dans un souci de clarté et de simplification, je mets de coté ici ces cas spéciaux. Si ceux-ci se révèlent nécessaires, n'hésite pas à me demander.

Par contre, le cas où le robot passe sur les segments joignant deux balises est aussi une condition limite que l'on ne peut éviter pour les cas (a1, a2) et (a1, a3). Sur ces segments, les calculs respectifs sont inapplicables (en fait, durant le déroulement du calcul on se retrouve alors devoir diviser par un terme nul, ce qui invalide le calcul).



Ainsi, si le robot arrive sur l'une de ces positions, il faut absolument ne pas tenir compte de la position donnée par le calcul avec les angles correspondant : le résultat sera toujours faux. De même, une position trop proche de ces segments donnera un résultat de plus en plus imprécis au fur et à mesure que l'on se rapproche d'un segment.
Il faut donc déterminer une marge d'erreur invalidant le calcul. Des informations sont données ci-dessous.

Calcul à partir des angles (a1, a2).

D'après les données, on a :

Tan(a1 + PI/2) = (B1x – Rx) / Ry
Tan(a2) = (B2y – Ry) / (B2x –Rx)

On en déduit alors :

Ry * [Tan(a1 + PI/2) + (1/Tan(a2))] = B1x - B2x + [B2y / Tan(a2)]

Rx = B2x – [(B2y – **Ry**) / Tan(a2)]

Invalidation du calcul : ici le calcul devient invalide si

$$\text{Tan(a1 + PI/2) + (1/Tan(a2)) = 0}$$

Ce qui se produit pour chaque point du segment 1. Il faut invalider le calcul si ce terme devient plus petit qu'une valeur donnée. Dans le code Java décrit ci-dessous, j'ai pris la valeur limite 0.001 avec de bons résultats.

Calcul à partir des angles (a1, a3).

D'après les données, on a :

Tan(a1 + PI/2) = (B1x – Rx) / Ry
Tan(a3 – PI/2) = (Rx) / (B3y –Ry)

On en déduit alors :

Ry * [Tan(a1 + PI/2) – Tan(a3 - PI/2)] = B1x –B3y * Tan(a3 + PI/2)

Rx = B1x – **Ry** * Tan(a1 + PI/2)

Invalidation du calcul : ici le calcul devient invalide si

$$\text{Tan(a1 + PI/2) – Tan(a3 - PI/2) = 0}$$

Ce qui se produit pour chaque point du segment 2. Il faut invalider le calcul si ce terme devient plus petit qu'une valeur donnée. Dans le code Java décrit ci-dessous, j'ai pris la valeur limite 0.001 avec de bons résultats.

Calcul à partir des angles (a2, a3).

D'après les données, on a :

Tan(a2) = (B2y – Ry) / (B2x –Rx)
Tan(a3 – PI/2) = (Rx) / (B3y –Ry)

On en déduit alors :

Ry [Tan(a3 – PI/2) + (1/Tan(a2))] = B3y * Tan(a3 – PI/2) – B2x + [B2y / Tan(a2)]

Rx = (B3y – **Ry**) * Tan(a3 – PI/2) ;

Invalidation du calcul : ici le calcul devient invalide si

Tan(a3 – PI/2) + (1/Tan(a2))

Normalement, ceci de doit se produire que sur le segment entre la balise 2 et la balise 3, segment que le robot ne peut atteindre.

3. Utilisation du code de test

Le code de test ci-joint permet deux choses :

1/ Vérifier les calculs ci-dessus : Le code demande la taille du terrain et les coordonnées réelle du robot. A partir de ces données, les angles a1, a2 et a3 sont calculés par des formules utilisant uniquement des arcsinus. Ensuite, on utilise ces angles avec les formules ci-dessus pour recalculer les coordonnées du robot. Les résultats sont alors comparables aux données d'origine. Le lancement du code est décrit ci-dessous :

TerrainRobot [-Lx <double>] [-Ly <double>] [-x <double>] [-y <double>]
-Lx : Longueur du terrain (sens des x)
-Ly : Hauteur du terrain (sens des y)
-x : Abscisse du robot
-y : Ordonnée du robot
Valeurs par défaut : Lx=3 Ly=2 x=1.5 y=1

Le calcul tient compte des segments d'invalidation est prévient si on se trouve en situation d'exclusion d'un calcul.

2/ Simuler les déviations de mesure : La mesure de l'angle étant faite par un moteur pas à pas, l'exactitude de la mesure se fait avec une incertitude de (2*PI/pas) où « pas » est le nombre de pas que doit faire le moteur pour effectuer un tour complet.
Le code permet de simuler la position du robot en ajoutant cette incertitude dans le résultat du calcul des angles a1, a2 et a3. La position supposée du robot est alors calculée à partir de ces angles modifiée et la déviation à la position réelle estimée. Cette procédure est répétée autant de fois que demandé et on calcule la moyenne des déviations (en valeur et en pourcentage des mesures du terrain).
Cette simulation prend en compte la taille du robot (estimée à 25cm de diamètre) afin de ne pas faire approcher le point de mesure des angles des bords du terrain à moins de 12.5 cm. On évite ainsi les conditions aux bords pouvant entraîner les erreurs de calculs.
De même, le calcul tient compte des segments d'invalidation est prévient si on se trouve en situation d'exclusion d'un calcul.
Cette simulation permet de connaître le niveau d'incertitude sur la position du robot afin de savoir si elle peut être dangereuse pour le résultat escompté (mettre la balle dans le trou). Le lancement du code est décrit ci-dessous :

TerrainRobot [-Lx <double>] [-Ly <double>] [-simul <int>] [-p <int>]
-Lx : Longueur du terrain (sens des x)
-Ly : Hauteur du terrain (sens des y)
-simul : Nombre de répétition du calcul
-p : Nombre de pas du moteur requis pour une tour complet
Valeurs par défaut : Lx=3 Ly=2 p=100

On remarque ainsi deux choses :

1/ Le calcul de la position du robot en utilisant les angles est précises. Avec des variables « double », l'erreur de calcul se situe toujours en dessous de la quinzième décimale.

2/ La déviation due à une mesure insuffisamment précise des angles (un moteur ayant seulement 100 pas par tour par exemple) peut entraîner une erreur sur la position du robot dépassant le centimètre (et ainsi risquant de faire manquer la cible). Un moteur à 1000 pas par tour apporte une erreur suffisamment faible.